Operator's Manual

Cavro® Omni Robot

Version 4.0

March 2014

30085582-A

This page has been left intentionally blank.

Operator's Manual

Cavro® Omni Robot

Version 4.0

Tecan Systems, Inc.
2450 Zanker Road
San Jose, CA 95131 USA
T 1 408 953 3100, Toll Free 1 800 231 0711
F 1 408 953 3101
E-mail: helpdesk-sy@tecan.com
Web Site: www.tecansystems.com

30085582-A

Part Number 30085582-A

Copyright and Trademark Information

Product Warranty Information

Tecan Systems warrants that robots manufactured and sold by Tecan Systems will be free from defects in materials and workmanship for a period of twenty-four (24) months from the date of shipment to customer. Tecan Systems' liability for the breach of the foregoing warranty is limited to the repair or replacement of the products found to be other than warranted. Such products will be accepted for return only if the customer returns them to Tecan Systems' factory or repair depot within thirty (30) days from the time of discovery of the alleged defect, and prior to return, fills out a Certificate of Decontamination (document P/N 20730171), obtains a return authorization number from Tecan Systems, provides Tecan Systems with the serial number of each robot to be returned, and prepays freight charges to the factory or a designated Tecan Systems repair depot. No warranty is expressed or implied for:

- Breakage
- Maltreatment
- Unauthorized service
- Units not returned in original or adequate packaging
- Units which are "life-cycled"
- Syringes
- Syringe seals
- Tubing and tubing connections
- Cavro® valves
- Cavro® probes

The foregoing warranties and limitations are customer's exclusive remedies and are in lieu of all other warranties, express or implied, including without limitation any warranty of merchantability or fitness for a particular purpose.

Product Documentation Warranty Information

The information contained in this document is subject to change without notice. Tecan Systems makes no warranty of any kind with regard to this material, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose.

Tecan Systems shall not be held liable for errors contained in this document or for incidental or consequential damages in connection with the furnishing, performance, or use of this material.

Document Status Sheet

Title:	Cavro® Omni Robot Operator's Manual, Version 4.0	
ID:	30085582-A, en	
Date	**Revision**	**Major changes**
March 2014	-A	Added Omni Embedded command language.

This page has been left intentionally blank.

Table of Contents

3 Mechanical Integration

4 Electrical Integration

5 Integration with Omni Configurator and Cavro Fusion

6 Operating in Omni Embedded Mode

A Maintenance Schedule

B Maintenance Tasks

C Specifications

D Abbreviations and Symbols

E Command Processor Command and Error Quick Reference Guide

F Embedded Command and Error Quick Reference Guide

G Conversion Guide: Command Processor to Embedded Commands

Index

List of Figures

List of Tables

This page has been left intentionally blank.

1 Overview

This chapter introduces the Cavro® Omni Robot and provides an overview of its functionality. Reading this chapter provides the context that is required to make the technical information in later chapters meaningful.

1.1 About This Manual

This section describes this manual and provides sources of additional information.

1.1.1 Introduction

This manual describes the installation, operation, and maintenance of the Cavro® Omni Robot. It includes instructions for the physical and electrical installation of the module, and how to integrate the Omni Robot and its accessories through a software application using either the Omni Embedded or the Command Processor operating modes.

1.1.2 Audience

This manual is for original equipment manufacturer (OEM) engineers and OEM users who need to incorporate the Omni Robot into their hardware and/or software product(s). It provides information for OEM user installation and testing, and information that the OEM must pass along to their users. Any references in this manual to “the end user” refers to the eventual customer of the OEM user’s product.

This manual assumes that the reader is technically proficient in electrical, mechanical, or software engineering disciplines, depending on the tasks to be performed. Drawings and specifications necessary to integrate the Omni Robot are provided in this manual.

1.1.3 Manual Organization

This reference contains the following chapters:

- Chapter 1, “Overview”
- Chapter 2, “Safety”
- Chapter 3, “Mechanical Integration”
- Chapter 4, “Electrical Integration”
- Chapter 5, “Integration with Omni Configurator and Cavro Fusion”

- Chapter 6, “Operating in Omni Embedded Mode”
- Chapter 7, “Operating in Command Processor Mode”
- Chapter 8, “Cavro Air Displacement Pipettor Mounting and Integration”
- Chapter 9, “Gripper Mounting and Integration”

This manual also contains the following appendices, which include reference information about the Cavro® Omni Robot:

- Appendix A, “Maintenance Schedule”
- Appendix B, “Maintenance Tasks”
- Appendix C, “Specifications”
- Appendix D, “Abbreviations and Symbols”
- Appendix E, “Command Processor Command and Error Quick Reference Guide”
- Appendix F, “Embedded Command and Error Quick Reference Guide”
- Appendix G, “Conversion Guide: Command Processor to Embedded Commands”

1.1.4 Terminology

The Cavro® Omni Robot and its component parts may be referred to in this manual by their full names or by shortened forms of their names. Some examples include:

- Cavro® Omni Robot: “Omni Robot” and “Omni” are acceptable short forms.
- Universal Z Axis: “Universal Z,” is an acceptable short form.

1.1.5 Regulatory Considerations

As a component designed for incorporation into larger systems that require independent testing and certification, the Cavro® Omni Robot does not carry its own CE mark.

The single and dual arm Omni, as well as the Universal Z, Dual Z, 8-channel head, and gripper options are UL recognized components. The UL file number is E164638.

The Omni as supplied is not finished equipment and is not an IVD device.

1.2 Product Overview

The Cavro® Omni Robot consists of OEM modular robotic components that can be configured to support a wide variety of customer applications. The modules are UL-recognized components that are optimized for liquid handling operations. The

Omni is typically integrated as the main robotic component within a larger OEM laboratory instrument. Providing individual linear motion along three axes (X axis, Y axis, and Z axis), the Omni is offered as individual axes of various lengths and orientations that can be combined to form specific configurations as described in this chapter.

Customers who order a high annual volume of a custom configuration can request a unique part number for their configuration to make repeated orders easier.

The Omni has a modular architecture; it can be configured with:

- A single axis (X, Y, or Z)
- Two axes (XY, YZ, or XZ)
- A single arm (three axes: X, Y, and Z)
- Two arms (six axes: X1, Y1, Z1, X2, Y2, and Z2)
- A fourth axis for a single arm (X, Y, Z, and Z'), using the Dual Z axis
- A seventh and eighth axis for two arms (X1, Y1, Z1, Z1', X2, Y2, Z2, and Z2') using the Dual Z axis
- Other configurations consisting of up to six axes (eight with Dual Z) are also possible

The Z axis is available in the following configurations:

- Standard Z (single pipettor)
- Universal Z (higher payload capacity and mount compatible with multiple different end effectors)
- Universal Z with ADP
- Dual Z (dual pipettor 9 mm or 18 mm spacing)

End effectors are attachments to the Z axis that are chosen based on the task to be performed, such as the Gripper or the 8-channel fixed-probe head.

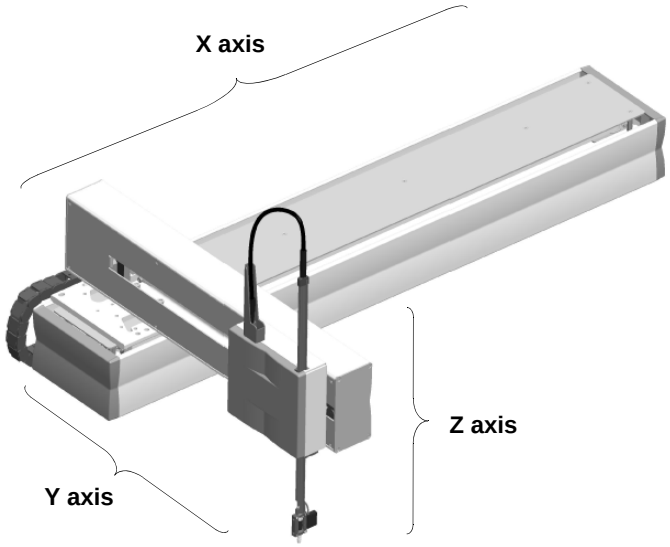


Figure 1-1 *Cavro Omni Robot, single arm configuration*

Figure 1-1 shows a single arm configuration with left-oriented Y and Z axes. The single arm model can also be ordered with right-oriented Y and Z axes. (The figure shows a Standard Z axis; orientation applies to the Universal Z and Dual Z axes as well.)

Note: *Throughout this manual, many figures and examples assume an X, Y, Z configuration. If your system does not have all three axes, your configuration will differ somewhat in appearance and function from the examples. Only the portions of the figures and examples that describe the features you have will apply.*

Note: *Throughout this manual, behavior or specifications particular to the Standard Z axis, Universal Z axis, Universal Z axis with ADP, or Dual Z axis will be specifically identified. Where behavior or specifications are the same, the term "Z axis" will be assumed to refer to all four models.*

In dual arm configurations, the Z axis module can be mounted on either side of the Y axis arm by the factory, providing four possible configurations with different accessible areas. Figure 1-2 shows three examples of dual arm configurations:

- (top) Z axis modules are mounted toward the outside of the robot; the Y axis on the left is left-oriented, the Y axis on the right is right-oriented. This configuration provides the maximum accessible area, but the minimum shared area between two Z axes.
- (middle) Z axis modules are mounted toward the inside of the robot; the Y axis on the left is right-oriented, the Y axis on the right is left-oriented. This configuration provides the minimum accessible area, but the maximum shared area.
- (bottom) Z axis modules are mounted on the same side (left) of their respective Y axis arm; both Y axes are left-oriented. Both Z axes could also be mounted right-oriented.

For a dual arm configuration, programming the Omni with the Command Processor (see Chapter 7, "Operating in Command Processor Mode") provides built-in collision avoidance features to help avoid accidental collision of the arms in an initialized system. Currently, the Omni operated in Embedded Mode (see Chapter 6, "Operating in Omni Embedded Mode") does not provide a collision avoidance feature.

The X axis of the Omni Robot can be mounted to a workspace using the threaded mounting screw holes found at both ends of the axis, or alternatively using the grooves on the underside of the axis.

The Universal Z axis provides a flexible, robust alternative to the Standard Z axis. The Universal Z axis includes the capability to attach and use a variety of options for liquid handling or moving lab samples.

The Omni Robot is available in several standard lengths for each axis; the axis dimensions are listed in Appendix C, "Specifications". Custom lengths are also offered (subject to NRE and minimum volume requirements). If you require an axis of a length that is not listed in the appendix, contact your Tecan representative for more information about custom lengths.

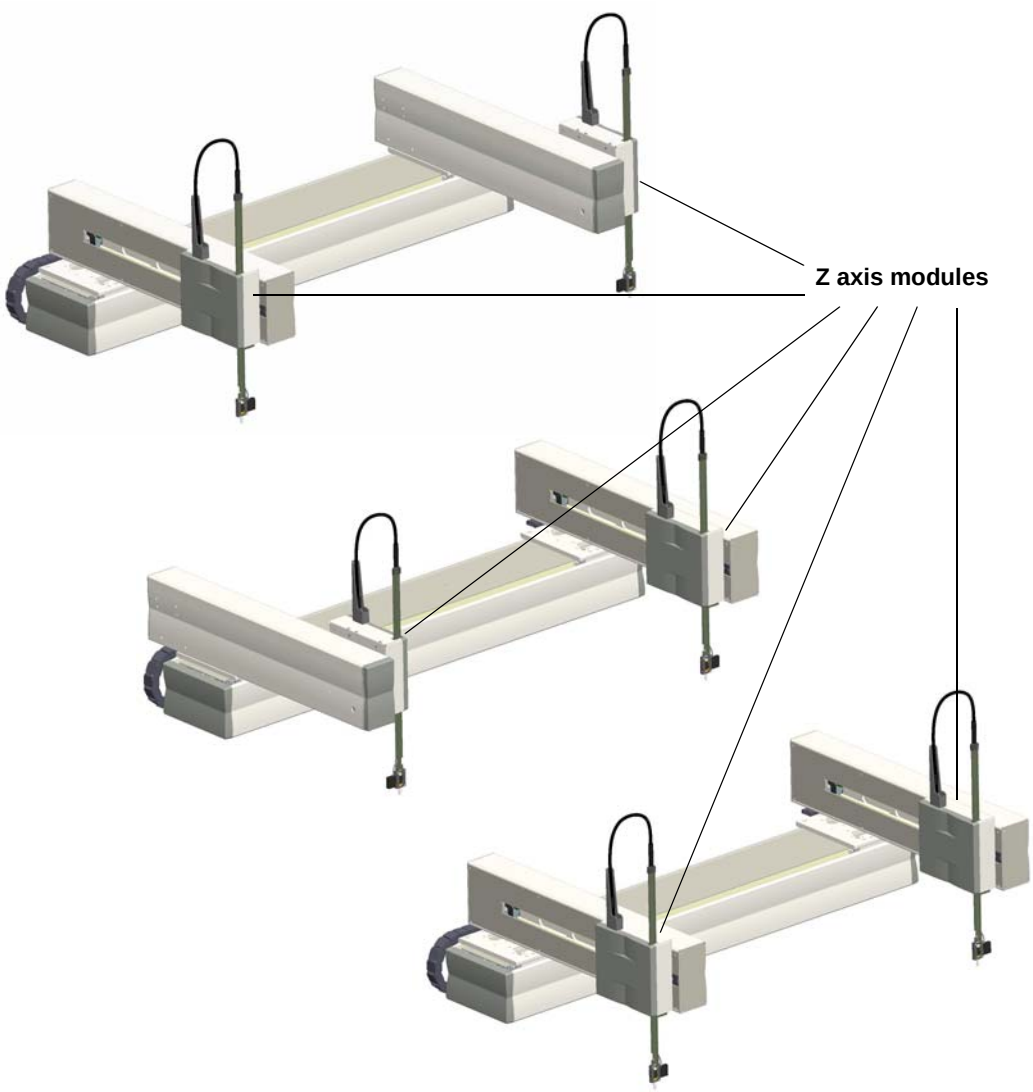


Figure 1-2 *Cavro Omni Robot, Standard Z dual arm configuration examples*

The Omni Robot is compatible with most Tecan diluters (pumps), probes, and Disposable Tip (DiTi) options. Cavro CAN devices are supported by the Command Processor (see Chapter 7, “Operating in Command Processor Mode”), and Omni Embedded Mode (see Chapter 6, “Operating in Omni Embedded Mode”). RS-485 devices are supported by the Command Processor only.

The Omni Robot is RoHS compliant. RoHS compliant modules are built with components that meet the requirements set by the European Union’s Directive 2002/95/EC on the Restriction of Hazardous Substances (“RoHS” Directive).

In Command Processor mode, the Omni Robot allows a software developer to incorporate the operation of the Omni Robot with an existing software system on a Windows platform. Commands are transmitted via easy-to-understand scripts transmitted using the Transmission Control Protocol/Internet Protocol (TCP/IP).

In Embedded mode, command strings can be sent directly to the Omni Robot.

Table 1-1 *Support for Omni features by software version number*

Omni Feature	**Software Version**					
	1.0	**2.0**	**2.2**	**2.2.2**	**3.0/3.1**	**4.0**
Omni Single Arm	X	X	X	X	X	X
Cavro OEM Serial (RS-485) devices	X	X	X	X	X	X
Omni API	X	X	X	X	X	X
Cavro Omni Hub	X	X	X	X	X	X
Omni Dual Arms		X	X	X	X	X
Omni API V2		X	X	X	X	X
Omni Universal Z Axis			X	X	X	X
8-Channel Pipetting			X	X	X	X
Gripper			X	X	X	X
ADP Support				X	X	X
Cavro CAN devices				X	X	X
Omni Dual Z Axes					X	X
Embedded CIB Commands						X

The Omni Robot is shipped with the latest software version. Table 1-2 provides a summary of the software versions shipped to date and the features that are supported by each version. Software and firmware component version numbers are recorded to a log file during startup.

Table 1-2 Omni Robot v4.0 software and firmware components

Component	Version #	Description
NrcCmdProc.dll	4.0.1	Command processor and axis drivers for X, Y, and Z. This is a Microsoft .NET-based Dynamic Link Library (.dll) file that allows the OEM software to communicate with and control the Omni Robot.
AxisDriver.dll	4.0.1	Axes drivers
XAxis_0001_0501.sw	05010001	X axis control program
YAxis_0002_0501.sw	05010002	Y axis control program
ZAxis_0003_0501.sw	05010003	Z axis control program
UnivZ_0008_0301.sw	03010008	Universal Z axis control program
Gripper_0009_0301.sw	03010009	Universal Z axis control program for Gripper
CibApp.mot	4.0.1	CIB firmware

1.3 Features of Operation

The Omni offers high reliability for applications that require the module to perform repetitive motions for an extended period of time. The linear motion slides provide low friction movements with high accuracy and longer life expectancy compared to other bearing designs.

The Omni is especially suitable as a liquid handling component in instrumentation that performs highly repetitive tasks. Some examples include tasks used in clinical labs that perform around-the-clock tests on medical samples, and tasks used by biopharmaceutical companies that perform high-throughput screening for therapeutic drugs.

Tecan offers standard lengths for each axis, and can also create custom axis lengths to fit specific applications. In Command Processor mode, the Omni Robot uses a set of commands that are transmitted using its TCP/IP communications interface, offering easy integration with an OEM software system or other Cavro devices. In Embedded mode, command strings are transmitted using either the TCP/IP or RS-232 communications interface.

A system integrator can mix and match the Omni components to quickly assemble a functional liquid handling system for an instrument, providing shorter development time and improving instrument time to market. Because the Omni Robot is a UL-recognized component, it facilitates the UL approval process.

1.4 Principles of Operation

The following sections contain more in-depth discussion of the concepts that make the Cavro® Omni Robot unique in the marketplace and able to meet a wide variety of OEM needs.

1.4.1 Omni Robot Command Sets

The modularity offered by the Omni extends beyond its physical layout. The Omni robotic modules do not require specific, proprietary software to operate. Starting with version 1.0, the Omni Robot software package has included a command set that can be used by any OEM software that uses TCP/IP. To issue a command to the Omni Robot, all the OEM software needs to do is send a properly formatted line of text to the Command Processor. As long as the command is valid and properly formatted, the Omni Robot will carry out the command, regardless of the OEM software used to submit that command to the Command Processor.

Use of the Command Processor command set requires a Windows-based host PC.

Starting with Version 4.0, an Embedded command set is provided as an alternative to the Command Processor command set. You can now control the Omni using either the Omni Command Processor mode or the Omni Embedded mode. The Embedded commands integrate the Command Processor functionality into the electronics within the Cavro® Omni Robot's Communication Interface Board (CIB). This allows customers to who do not use an external host PC in their system architecture to programmatically control an Omni Robot through a host controller of their own design.

For more information about the Command Processor commands, see Chapter 7, "Operating in Command Processor Mode". For more information about the Embedded commands, see Chapter 6, "Operating in Omni Embedded Mode".

1.4.2 Cavro Omni Robot Coordinate System

The Cavro® Omni Robot allows the user (application developer) to define the coordinate system used by the Omni Robot relative to a position of interest, such as the coordinates of a specific location on the worktable or a test fixture. The Omni Robot then uses this relative coordinate system for all its functions, transparently mapping those coordinates to the physical coordinates of the robot.

The reference position can be set using any arbitrary values for the coordinates, independent of the physical coordinates, based on the application requirements. This is done by performing a calibration using the Omni Configurator software, or programmatically through the Omni Command Processor command set or Omni Embedded command set.

Coordinates and calibration are discussed in greater detail in Section 7.2, ”Omni Robot Coordinate System” on page 7-4 (for the Omni Command Processor) and Section 6.2, ”Omni Robot Coordinate System” on page 6-3 (for Omni Embedded).

1.4.3 Liquid Level Detection

Most Omni Robot Z axes include an integrated capacitive liquid level detection (cLLD) system, allowing quick and accurate liquid detection when using the Omni Robot to move liquids. The cLLD monitors the capacitance between the probe or disposable tip attached to the Z axis in reference to the Omni Robot worktable (or corresponding carrier on the worktable). When the probe or disposable tip touches the liquid surface, the change in capacitance triggers a detection signal.

The Cavro® Air Displacement Pipettor (ADP) performs pressure liquid level detection (pLLD) based on sensing a change of pressure at the probe.

When the Omni and ADP are integrated to work as a system, you can use either one of these methods to perform liquid level sensing, or a hybrid variant (hLLD) that combines the functionality of both cLLD and pLLD.

Liquid level detection evaluates both the liquid detection signal (when the probe moves into the sample liquid) and the exit signal (when the probe retracts). The amount and conductivity of the liquid and the labware type influences the detectability.

The Omni Robot can reliably detect ionic solutions in a microplate. The detectability can be changed by adjusting the cLLD sensitivity, in order to accommodate different liquid samples. Because a high sensitivity to the liquid level also makes the module sensitive to outside interference, the sensitivity can be reduced to decrease the likelihood of interference.

The advantages of this type of liquid level detection are:

- **Minimum submerge depth** of the probe
- **Minimal probe contamination** and, accordingly, only limited need for probe washing
- Appropriate **message if no liquid or not enough liquid is available** for sampling
- Exit signal used for **blockages**, or **significantly short aspiration volumes**

In addition to detecting the liquid level, the Omni Robot can detect whether a probe or disposable tip is attached. The Omni Robot detects when a disposable tip is lost unexpectedly.

1.4.4 Axis Control Board (ACB)

Each Omni Robot axis has an Axis Control Board (ACB) that provides real-time control of its axis. Each ACB operates independently of the other ACBs in the hardware system, allowing users to send commands to multiple axes simultaneously.

The ACB stores the scripts that operate the axis (for example, moving the axis or detecting the liquid level). These scripts are factory-installed, along with the ACB firmware.

Note: *Axis IDs are determined by jumpers set at the factory during system assembly. Typically, the first arm axes are given IDs 1, 2, and 3 for X, Y, and Z respectively. For dual arm configuration, the second arm axes are given IDs 4, 5, and 6 for X, Y, and Z respectively. For a Dual Z axis, the additional axes are given IDs 7 and 8 for the first and second axis respectively.*

1.4.5 Axis Termination Board (ATB)

In a typical three-axis configuration, the encoder signals from the X and Y axes plug into the Y and Z ACBs respectively, and route the signal back to the X and Y axis ACBs. When an Omni is configured without a Y or Z axis on an arm, an Axis Termination Board (ATB) is necessary to connect and route encoder signals in place of the absent ACB.

The ATB also provides an additional 12-pin header for power and communication with other external attachments, including more Omni axes. The additional header on the ATB comes with a blank 12-pin connector housing installed by default which must be removed prior to use. The following arm configurations (1st or 2nd arm) currently require the use of an ATB:

- There is an X axis but no Y (X or XZ)
- There is a Y axis but no Z

It is the responsibility of the OEM customer to mount this board in their finished product in a manner that meets regulatory compliance. This includes but is not limited to meeting safety (UL/CSA), Regional (CE), and EMC standards as applicable.

1.4.6 Communication Interface Board (CIB)

Because each axis contains its own ACB that handles commands for its specific axis, the Omni Robot needs a central point to handle commands coming from the OEM software, route the commands to the proper ACB, and ensure that the responses from each axis are properly sent back to the software. This central point is the Communication Interface Board (CIB). All communication from the OEM application to the Omni Robot axes or other attached Tecan modules passes through the CIB, which communicates with the OEM software.

In Command Processor mode, the Command Processor receives a command from the OEM application and passes it to the CIB; the CIB then passes it to the recipient using either RS-485 or the Controller Area Network (CAN) protocol (depending on the recipient). The CIB also handles responses from the modules (successes, failures, and events) and passes them to the Command Processor. For more information, including a graphic that illustrates how commands are passed between the Command Processor and the CIB, see Section 7.1.1, ”Communication Flow” on page 7-2.

In Embedded mode, the command string is handled by the CIB, then passes it to the recipient using the Controller Area Network (CAN) protocol. The CIB also handles responses from the modules (successes, failures, and events). For more information, including a graphic that illustrates how commands are passed between the host controller and the CIB, see Section 6.1, ”Communication and Connectivity” on page 6-1.

The CIB is shipped with a set of firmware installed, along with a default IP address and listener port. The OEM user has the option of changing the default IP address and configuring the CIB and OEM application to communicate over the same port number. This process is described in Section 7.1.3, ”Setting the CIB IP Address” on page 7-3.

For robots with an X axis, the CIB is mounted inside the X axis, and CIB connectors are accessible from the exterior underside of the Omni Robot. For robots without an X axis, the CIB is located in an optional standalone module called the Cavro Omni Hub, and hub connectors are on the front of the hub.

1.4.7 Cavro Configurator Software

The Omni Robot also comes with the Configurator software. The Configurator enables the user to configure each axis in the Cavro® Omni Robot software system, set defaults for many of the parameters used in the Cavro® Omni Robot command set, and group axes into arms.

The configuration settings made through the Configurator change the power-on default settings for the Omni Robot. However, it is recommended that as a best practice, any parameter settings related to specific applications, including robot calibration, be performed by the OEM application software as part of the system initialization.

Caution! *The Configurator should be run any time an axis is added to or removed from the component.*

The Configurator allows the user to view Cavro® Omni Robot system information for diagnostic purposes, such as to check the firmware version of the CIB and, if necessary, update the CIB and ACB firmware. The Configurator also provides a command line interface that enables the user to send any command to one of the Cavro® Omni Robot components.

The Configurator documentation is included with the software installation CD.

1.4.8 Cavro Fusion

Cavro Fusion is a stand-alone application that gives users the ability to operate Tecan Systems components connected to a Windows-based PC.

While it is capable of operating an Omni Robot with a scripting interface or a GUI, Fusion is strictly intended to be used for testing and demonstration use only.

1.4.9 Cavro Integration Kit

The Cavro Integration Kit - Ethernet provides all the necessary items to enable the user to begin the evaluation of Tecan Cavro components. The Cavro Integration Kit for Ethernet supports a single or dual arm Cavro® Omni Robot with 1-2 pumps. The kit includes the following components:

- Power supply — 24V/5A
- Ethernet cable
- Two-pump cable (to connect pumps to the Cavro® Omni Robot)
- Cavro Fusion software

1.4.10 Cavro Omni Hub

As described in Section 1.4.6, ”Communication Interface Board (CIB)”, the Omni CIB is the central point for handling commands from the OEM software, routing commands to the proper ACB, and ensuring that responses from each axis are properly sent back to the OEM software. This functionality is integrated into the X axis. Configurations without an X axis require a Cavro Omni Hub which contains a CIB, to serve as the central point for communications.

The Cavro Omni Hub encloses the CIB in a sheet metal box and provides a single axis cable for connectivity to axes. A second cable is optional and is available as a spare part (see Table B-2). Installation of a second cable requires opening the Cavro Omni Hub to remove, open, and install the strain relief grommet 12-12.5 in. from the end of the axis cable.

1.5 Additional Information

This section contains additional information that you could find useful after you begin using the Omni Robot.

1.5.1 Related References

The Omni Unpacking Instructions may help with integrating the module into your system. They are included with the unit in the shipping box.

The following manuals are provided in PDF format on the CD that accompanies the unit:

- Cavro Configurator software documentation
- Cavro® Omni Robot Operator's Manual

Contact your Tecan customer support representative for a list of all current and future options.

1.5.2 Contacting Tecan

Direct all questions about the Cavro® Omni Robot, including questions about this document, available modules, upgrades, and options, to:

Tecan Systems, Inc.
2450 Zanker Road
San Jose, CA 95131 USA
1-800-231-0711
helpdesk-sy@tecan.com

This page has been left intentionally blank.

2 Safety

This chapter is the central repository for safety-related information. It includes potential hazards associated with using the Omni Robot of which all users should be aware, whether end users or OEM users.

2.1 Safety Headings and Icons

This manual uses three types of informational notices: warnings, cautions, and notes. These notices highlight important information or potentially dangerous situations.

Notices with the heading ***Note*** give helpful information about the Omni Robot.

Notices with the heading ***Caution!*** indicate a possibility of equipment damage or malfunction. They are marked with one of the following images:

Table 2-1 Caution images

	Indicates a possibility of equipment damage, malfunctions, or incorrect process results if instructions are not followed.
	Disturbance of functions by electromagnetic RF waves. Do not use a wireless device.

Notices with the heading ***WARNING!*** indicate a possibility of severe injury, loss of life, or equipment damage. They are marked with one of the following images:

Table 2-2 Warning images

	Indicates a possibility of severe personal injury, loss of life, or equipment damage if instructions are not followed.
	Fire hazard.
	Electrical hazard.
	Pinch point or mechanical hazards.

2.2 OEM Safety Instructions

WARNING! *In the electrical wiring, the amperage before the current reaches the fuses is potentially high on the power supply side, which could cause serious injury. After the current leaves the fuses, the amperage is within limited energy circuit requirements.*

WARNING! *Unintended motion can occur if the wrong version of the firmware is loaded to the Axis Control Board (ACB) or if the firmware in the module's ACBs are mismatched.*

WARNING! *The possibility of run-away (inherent in all servo systems) exists if the encoder signal is lost. This can occur if the encoder cable is broken, plugged in incorrectly, or if the encoder fails.*

WARNING! *Static electricity can damage the Omni Robot if proper precautions are not taken.*

- *Ferrites and installation instructions may be shipped with the module; the OEM customer is responsible for installing the ferrites according to the installation instructions. These ferrites protect the module against electrostatic discharge (ESD).*

WARNING! *If you use a networked computer to operate the Omni Robot (that is, a computer that is connected both to the Omni Robot and to an external network), you could expose the Omni Robot to security risks, such as allowing unintended control to another user on the network. To eliminate this risk, either operate the Omni Robot using a standalone computer, or take the appropriate safety precautions when implementing an OEM software solution.*

WARNING! *If the configuration of the robot is overwritten with incorrect configuration settings, this could cause injury and/or system damage. Be sure to verify the configuration settings and recalibrate the robot before operating the robot.*

WARNING! *Take care to ensure that the values of any parameters you set are within the valid range for your robot hardware (such as axis dimensions and travel length).*

WARNING! *The timing of the execution of Omni Robot commands may be affected by the host PC performance. Tecan recommends that you verify the execution timing of your Cavro® Omni Robot with the host PC setup in its working configuration.*

WARNING! *In the event of power loss, the Universal Z contains a brake to limit the uncontrolled dropping of the axis to approximately 30 mm. To ensure that the brake remains effective, execute the BrakeTest command regularly.*

WARNING! *Omni Version 4.0 has a hardware-triggered stop that can use the input from pin Int5, but the hardware-triggered stop will not function if the int5 input circuitry or switch is damaged. Users are responsible for testing this functionality.*

2.3 End User Safety Instructions

This section contains the safety rules and warnings that apply to the end user during the use of the Omni Robot. The OEM user is responsible for either mitigating against these possibilities or passing these warnings on to the end user.

WARNING! *Legal regulations, such as local, state, and federal laws, that prescribe the use or application, as well as the handling, of dangerous materials in connection with the Cavro® Omni Robot must be strictly followed.*

WARNING! *If the Cavro® Omni Robot is used in a manner not specified by the manufacturer, the protection provided by the Cavro® Omni Robot may be impaired.*

WARNING! *The end user is responsible for ensuring that the Cavro® Omni Robot is operated in proper condition only, and that maintenance, service, and repair jobs are performed with care and on schedule by authorized personnel only. If covers are present, do not remove them during operation. Failure to conduct proper and timely maintenance could potentially cause safety issues and mechanical failure.*

WARNING! *Use only Tecan consumables and spare parts for maintenance and repair to assure good system performance and reliability.*

WARNING! *Do not remove the covers installed on the Omni Robot during operation. If such elements are removed (for example, to perform maintenance tasks), resume operation only after all have been completely installed and checked.*

WARNING! *The Omni Robot can be used as a stand-alone robot or inside of a protective enclosure. When integrated as a stand-alone robot, OEM users should assess their final design and apply the appropriate labels to inform their customers of potential pinch points.*

WARNING! *Static electricity can damage the Omni Robot if proper precautions are not taken.*

- *Operate the Omni Robot in a statically controlled environment.*
- *Before touching a tip or probe that is attached to the Z axis, make sure that you are properly grounded, or discharge any static charge against a grounded surface.*
- *Ensure the Omni robot chassis has a low impedance ground path for optimal performance.*

WARNING! *The Omni Robot contains pointed tips and other sharp-edged elements, which might cause injuries when you reach into the working area.*

- *Always be aware of mechanical hazards. Do not move your head or hands into the path of moving parts during deployment; a risk of injury, blunt force trauma, or piercing exists.*
- *Wear proper laboratory apparel (such as rubber gloves and safety goggles) as appropriate for your deployment.*
- *Personnel who are not operating the Omni Robot module should take extra care when standing near the module, as an unanticipated movement by one or more axes could cause injury to a bystander. This is especially important if the Omni Robot is being operated remotely through the Ethernet connection.*

WARNING! *The Omni Robot does not support hot plug-in or hot swapping. You must unplug the Omni Robot before adding or removing additional modules.*

Caution! *Do not operate electronic devices near the Omni Robot module. The electromagnetic RF waves from a wireless device could affect liquid level detection or other functions.*

2.4 End User Qualifications

Operators of this module must:

- Be trained to use the OEM hardware system into which the Omni Robot will be integrated.
- Have thorough knowledge of the software application that will be used to run the hardware system.
- Be familiar with good laboratory practice guidelines.

2.5 Component Decontamination

The end user is responsible for performing decontamination when required. The decontamination method must be adapted to the respective system in which it runs and the substances associated with the system. The user takes full responsibility for the appropriate decontamination of the Omni Robot and all associated parts.

The exterior painted surfaces of the Omni Robot can be gently cleaned with 70% isopropyl alcohol in water solution, Sporicidin brand spray disinfectant, 70% ethanol, or 10% bleach in water solution. Moisten a soft cloth with one of these solutions and wipe surfaces gently. Hard rubbing or scrubbing of painted surfaces is not recommended. After using a cleaning agent or disinfectant, the surface should be wiped with clean water.

Caution! *Make sure to not let any cleaning agent or disinfectant touch the electrical components of the module. Do not spray disinfectant into the module; always direct the spray away from the module's inner workings.*

In addition to performing regular decontamination, the user must thoroughly decontaminate the module according to standard laboratory regulations in the following cases:

- Before any maintenance or service work is performed on the module.
- In case of accidents (for example, crash, spilled substances, etc.).
- Before a Tecan field service engineer (FSE) performs any on-site work on the module.
- Before the module or any of its parts are returned to Tecan for repair.
- Prior to storage of the module.
- Prior to disposal of the module or any of its parts.
- Generally, before the module or its parts leave the user's site.

Note: *Before a robot is returned to Tecan, the owner of the robot must confirm in writing that the decontamination has been performed properly and in accordance with good laboratory practice guidelines. A Decontamination Form is included as part of the Return Materials Authorization (RMA) process. Contact Tecan service if an RMA is necessary.*

This page has been left intentionally blank.

3 Mechanical Integration

This chapter describes the tasks involved in the physical assembly of the Omni Robot. The mechanical integration must be completed before the electrical integration can be attempted.

This chapter is organized into the following sections:

3.1 Introduction

The mechanical integration of the Cavro® Omni Robot consists of the following steps:

- Assembling the module
- Mounting the module to the work surface
- Leveling the module to the work surface
- Adding additional modules to the system
- Verifying that the integration is complete

3.2 Possible Cavro Omni Robot Configurations

The Cavro® Omni Robot can be configured in many different ways, depending on your system's needs and the modules available for purchase. Variables to be considered include:

- Length of X and Y axes
- Number of axes
- Orientation of the arms (right or left)
- Z axis and end effector options
- The presence of axis covers
- E-chain size

3.2.1 Length of X, Y, and Z Axes

The Omni Robot can be ordered with different axis travel lengths, configured at the factory.

- The X axis is available in 500 mm, 750 mm, and 1250 mm travel lengths. For dual arm configuration travel lengths, see Table C-1, "X axis dimensions," on page C-11.
- The Y axis is available in 150 mm and 300 mm travel lengths.
- The Z axis is available in 210 mm travel length only.

3.2.2 Single or Dual Arm

The Omni Robot can be configured with a single arm (up to three axes: X, Y, and Z), dual arms (up to six axes: X, Y1, Z1, X2, Y2, and Z2). The Dual Z axis allows for a fourth axis for a single arm (X, Y, Z, and Z') or a seventh and eighth axis for two arms (X1, Y1, Z1, Z1', X2, Y2, Z2, and Z2').

See Section 1.2, "Product Overview" on page 1-2 for a more complete explanation of the dual arm configurations.

3.2.3 Z Axis Orientation to Y Axis

The Z axis can be mounted on either side of the Y axis. In the following list, *left-oriented* means the Z axis is mounted to the left side of the Y axis when viewed from the front of the robot.

The orientation of the Z axis (mounted on the left or right side of the Y axis) is determined at the time the robot is assembled at the factory. The user must specify the Z axis location when placing the order for the robot.

Single Arm Orientation Possibilities

- Z axis left
- Z axis right

Dual Arm Orientation Possibilities

- Z1 axis left of Y1, Z2 axis left of Y2 (both left-oriented)
- Z1 axis right of Y1, Z2 axis right of Y2 (both right-oriented)
- Z1 axis left of Y1, Z2 axis right of Y2 (one left, one right-oriented)
- Z1 axis right of Y1, Z2 axis left of Y2 (one right, one left-oriented)

3.2.4 Z Axis Options

Tecan is constantly developing new options to provide greater possibilities for lab automation. For more information about the features available in different versions of the Omni Robot software, see Table 1-1, “Support for Omni features by software version number,” on page 1-6.

The following list shows the options that can be configured with the Standard Z axis, Dual Z axis, and the Universal Z axis at this time.

- Standard Z axis with single probe or DiTi adapter, includes cLLD capability
- Dual Z axes with single probe or DiTi adapter, includes cLLD capability
- Universal Z axis with 8-channel pipetting head, includes cLLD capability
- Universal Z axis with ADP, includes cLLD capability
- Universal Z axis with gripper, excludes cLLD capability

Note: *The Z axis options can be mounted on single arm or dual arm configurations, in several combinations. For example, users may order custom dual arm configurations with a Universal Z axis and 8-channel pipetting head on one arm and Standard Z axis with single probe on the other.*

3.3 Unpacking the Module

Note: *These instructions assume a three-axis X, Y, Z Omni configuration, but are also applicable to other configurations. Skip instructions that apply to axes not present in a given Omni configuration. If there is no Y axis or Z axis, the Omni will ship with an axis termination PCBA; refer to section Section 1.4.5, ”Axis Termination Board (ATB)” for more information on handling this PCBA.*

This section and Section 3.4, ”Attaching the Z Axis to the X/Y Axis Assembly” describe how to unpack and assemble the Omni Robot. If the Omni or any of its parts are missing or damaged, contact Tecan immediately as described in Section 1.5.2, ”Contacting Tecan” on page 1-13.

A three-axis Omni Robot is shipped in two pieces: a fully assembled and integrated X/Y axis assembly, and a Z axis.

1. Remove inserts from the shipping box as needed to unpack the Omni.
2. Remove the X and Y axes from the shipping box. If the Omni does not have an X axis, a Cavro® Omni Hub will be connected to the Y axis and must be removed at the same time to avoid damaging the Omni.
3. Remove the Z axis from the shipping box (if a Universal Z or Dual Z was ordered, it will be in a separate box).
4. Remove the protective antistatic wrap from the Omni.
5. Remove the accessories box and unpack the bag of assembly hardware.

3.4 Attaching the Z Axis to the X/Y Axis Assembly

The orientation of the Z axis (mounted on the left or right side of the Y axis) is determined at the time the robot is assembled at the factory. The user must specify the Z axis location when placing the order for the robot.

If there are two left-oriented or two right-oriented Z axes, they are not interchangeable. Their addresses are preset based on which arm they were configured to go on.

Equipment required:

- 2.5 mm hex key or driver
- 3 mm hex key or driver

Note: *These assembly instructions only address Universal Z or Dual Z axes if they differ significantly from the Standard Z axis.*

1 Remove shipping lock down screws from the X and Y axes as shown in Figure 3-1. If the Omni does not have a Y axis, there will be a generic bracket connecting the e-chain to the X carriage for shipping.

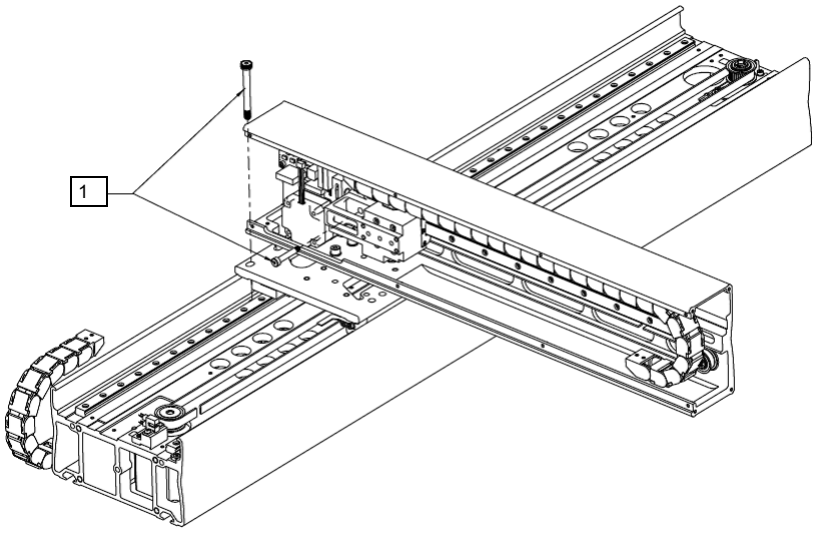


Figure 3-1 *Removing shipping lock down screws*

2 Remove the cover from the Z axis as shown in Figure 3-2.

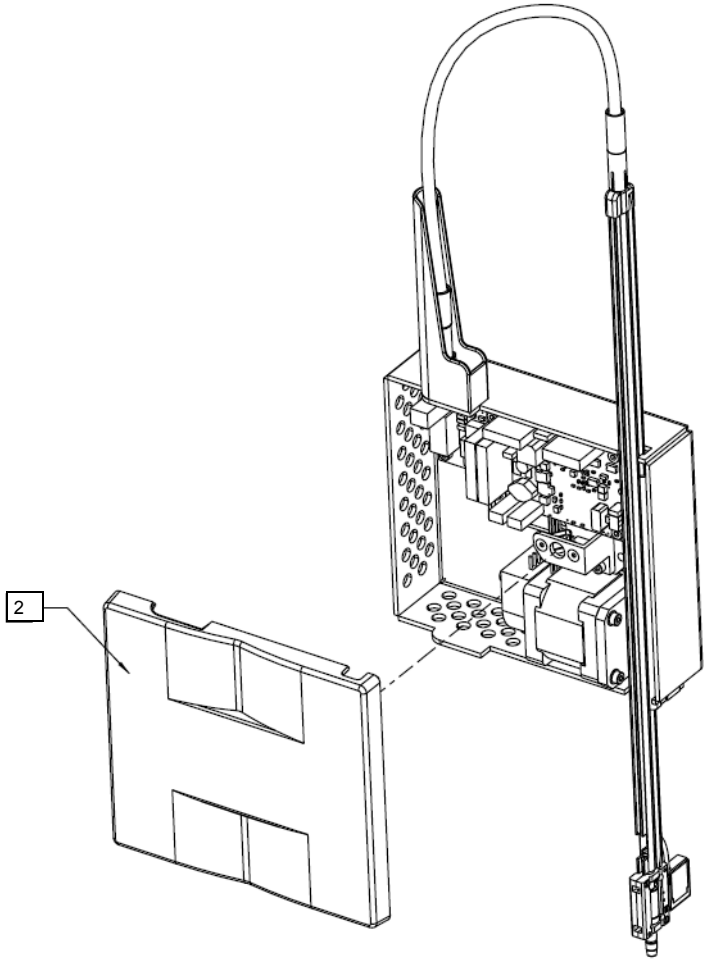


Figure 3-2 *Removing the cover from the Z axis*

3 Be sure to feed any cables and tubing through this opening in the Z axis as shown in Figure 3-3.

4 Mount the Z axis to the rest of the Omni using the four M4x16 screws as shown in Figure 3-3. Make sure the screws are securely fastened (if available, set torque wrench to 22 in-lbs). Z axes with ACB address 3 MUST be mounted on the first arm (Y axis on the left) and Z axes with ACB address 6 mount on the second arm.

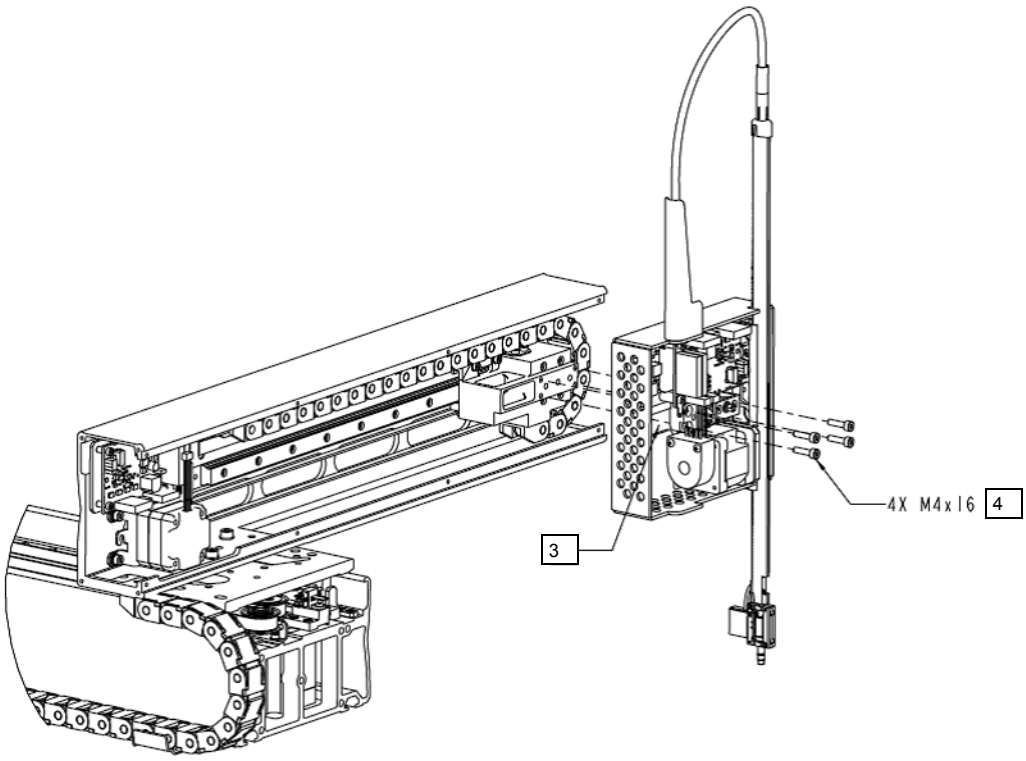


Figure 3-3 *Mounting Z axis to Omni X/Y axis assembly*

5 Snap the ferrite onto the axis cable where shown in Figure 3-4. The ferrite is necessary in order to meet ESD immunity.

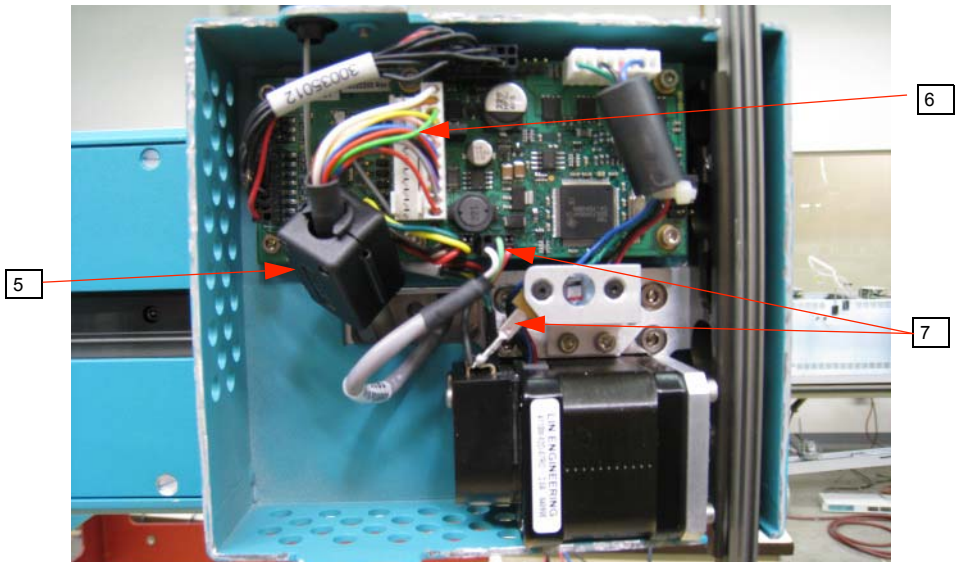


Figure 3-4 *Attaching cables from the Y axis*

6 Plug the axis cable connector from Y axis into the header shown in Figure 3-5. For a Dual Z, the axis cable will plug into this header on the PCBA at the bottom of the unit.

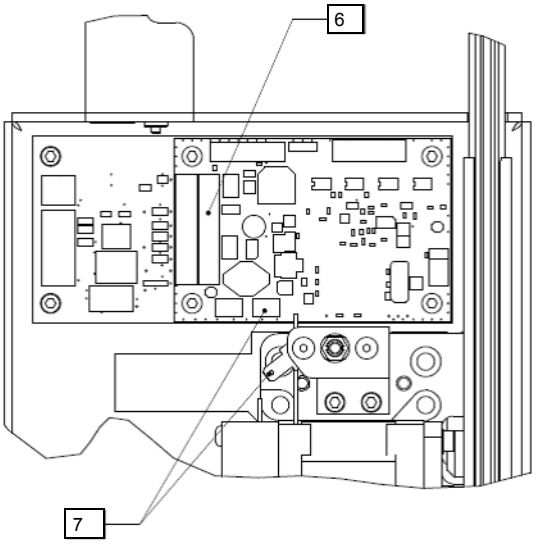


Figure 3-5 *Z axis cable attachment schematic*

7 Plug the magnetic sensor cable (Y encoder cable) from the Y axis into the header and the ground wire onto the tab (located at the cover mounting block as shown in Figure 3-5 for Standard Z and Universal Z axes, and located as shown in as shown in Figure 3-6 for Dual Z axes).

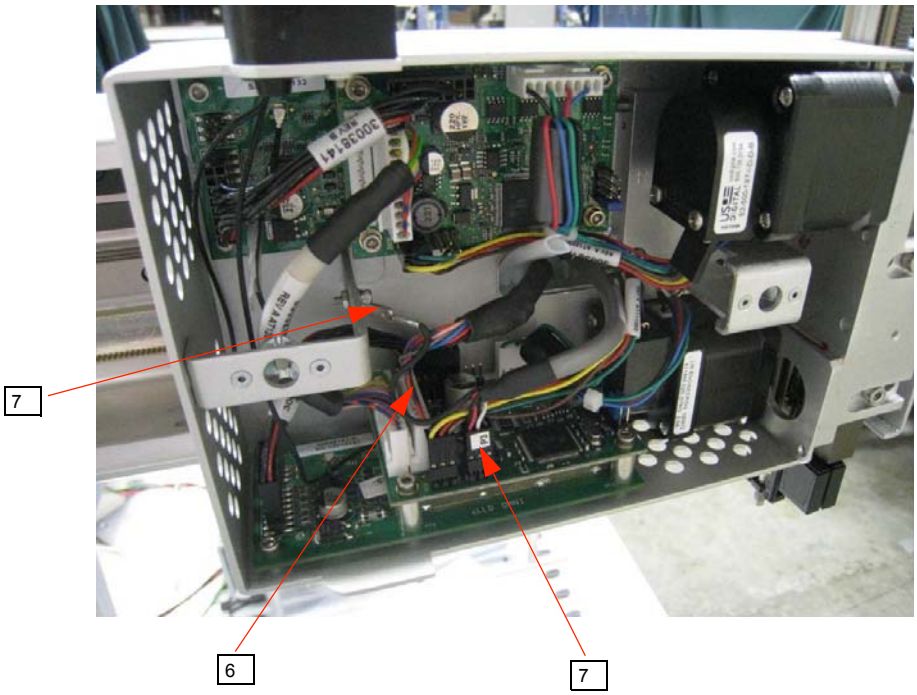


Figure 3-6 *Attaching cables to the Dual Z axis*

8 Route the wires for the relevant axes approximately as shown in Figure 3-6 and replace Z axis cover in the same way as it was removed.

9 Route the cLLD coax down through the tubing/cable guide as shown in Figure 3-7 if the unit has a Universal Z with cLLD capability (includes Universal Z with ADP).

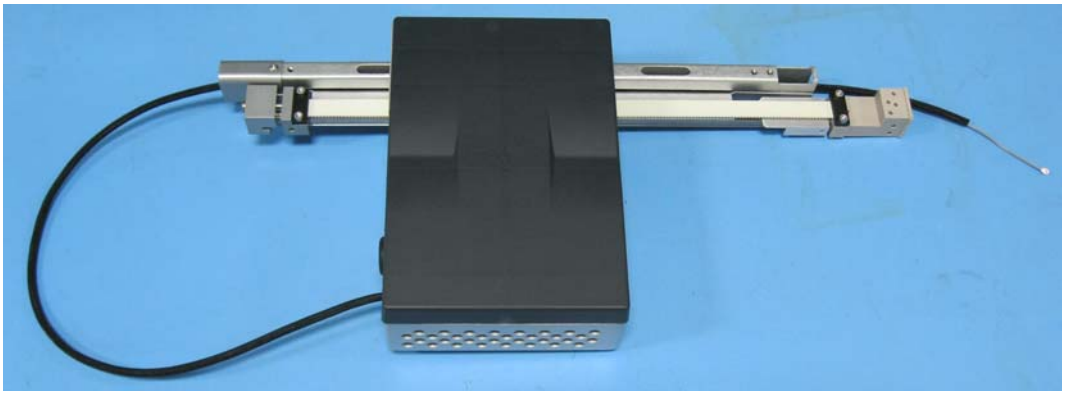


Figure 3-7 *Routing coax cable for Universal Z axis with cLLD capability*

10 Refer to section 7.3, Omni System Imager, in the Cavro® Omni Robot Configurator User Guide for instructions on creating and storing a system image for recovery purposes. The system image can help with system troubleshooting and is needed for configuration of certain spares like the axis control board.

Caution! *Before making any changes to the system power-on defaults, including performing a calibration, it is strongly recommended that you record the configuration details (view and print or save) and create a system restore image with the Omni System Imager tool. Saving a system restore image allows for easy restoration of the entire system configuration, or any portion of the system configuration, through the Omni System Imager. Without a saved image, the system must be reconfigured step by step.*

3.5 Mounting the Module

3.5.1 Mounting to a Chassis or Stand

After mounting the Z axis to the X/Y axis assembly, secure the X axis to the desired mounting surface (chassis or stand).

Before you mount the Omni Robot, ensure that the mounting surface is large enough to accommodate the Omni Robot. See Section C.2, ”Mechanical Specifications” on page C-11 for the dimensions of the robot.

In addition, the mounting surface must be level to the ground and stable, and must include a proper grounding plane.

You can integrate the Omni Robot with an existing system or mount it as a separate unit. In either case, the procedure is the same.

The X axis of the Omni Robot allows two methods of mounting:

- Four threaded holes on each end of the X axis, with three dowel pin holes for optional alignment:
  - Two dowel pin holes for fixed alignment and leveling of the Omni Robot,

    or
  - One dowel pin hole to allow rotational freedom and manual alignment of the Omni Robot to the work surface
- Two mounting grooves (T-nut slots) on the underside of the X axis.

3.5.2 Mounting Using the Sides of the X Axis

Figure 3-8 shows a side view of the X axis. The numbers in the figure refer to the following mounting holes and grooves:

1-4: mounting holes

5-6: mounting grooves

7-8: leveling dowel pin holes used for fixed alignment

9: dowel pin hole used for rotational freedom and manual alignment

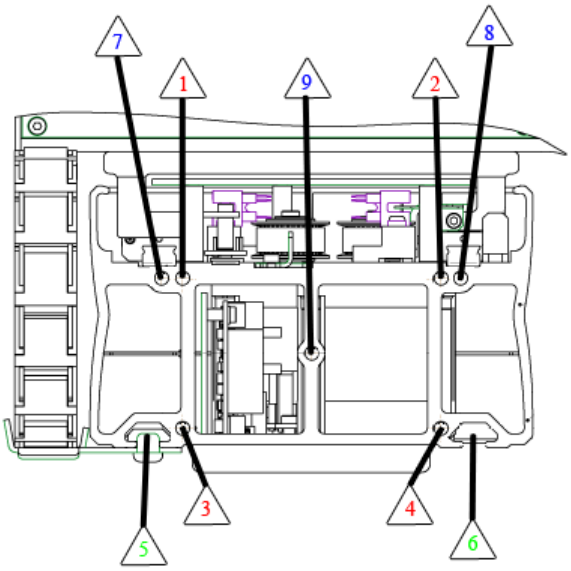


Figure 3-8 *Side view of X axis, with mounting holes marked*

For the most accurate positioning of the module, mount the module using the mounting holes on the X axis. (See Figure C-1, “X axis mounting hole dimensions” on page C-1 for more detail.) Tecan recommends that you use four M6x1.0 screws (with between 6 and 10 mm of thread engagement) on each end to secure the module. Screws should include some form of thread locker or lock washer.

3.5.3 Mounting Using the Mounting Grooves of the X Axis

The mounting grooves are primarily intended for use in mounting additional modules to the Omni Robot. They can be used to mount the X axis to a workspace; however, if you use the mounting grooves to mount the robot, you will need to ensure that the Omni Robot remains properly aligned to the workspace during operation.

If you are mounting the module using the mounting grooves, Tecan recommends that you use T-nuts (available from Tecan); see Section B.7, ”Field Replaceable Parts” on page B-21.

Figure 3-9 shows an example of how the Cavro® Omni Robot can be deployed and integrated with an additional Cavro® module.

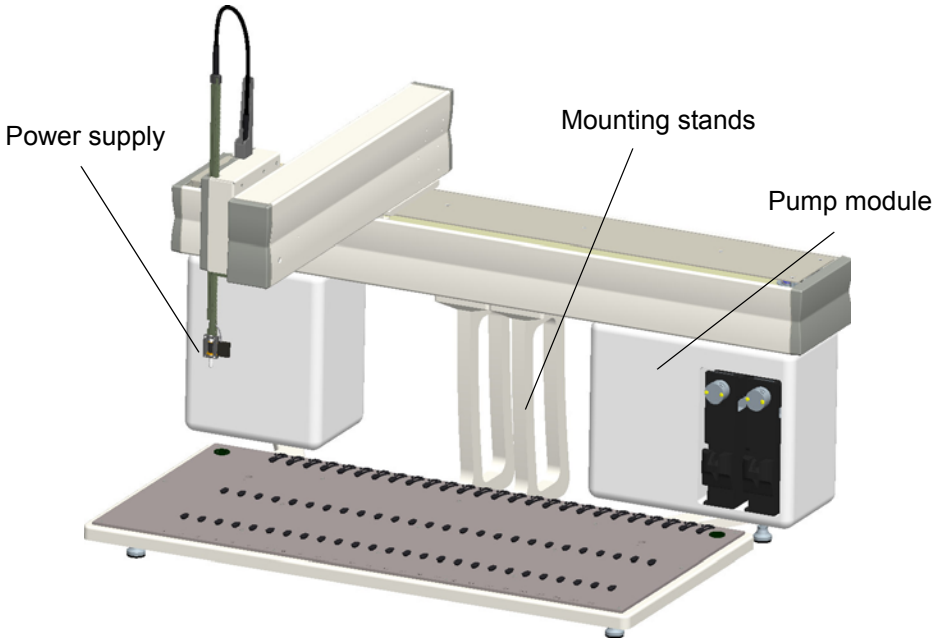


Figure 3-9 *Sample mechanical deployment*

In Figure 3-9, the Omni Robot is mounted over the workspace with a set of stands, which are attached to the mounting grooves on the underside of the X axis. On the right side of the robot, a Cavro® pump module is mounted to the mounting grooves. The power source for the robot is on the left side.

3.5.4 Mounting the Y Axis

Figure 3-10 shows a bottom view of a left-oriented Y axis. A right-oriented Y-axis is the mirror image of Figure 3-10. (See Figure C-3, “Left-oriented Y axis mounting hole dimensions” on page C-3 for more detail.) The numbers in the figure refer to the following mounting holes:

1-3: mounting holes

4-5: dowel pins for alignment

6-7: M3x0.5 tapped holes to connect an e-chain or other small bracket

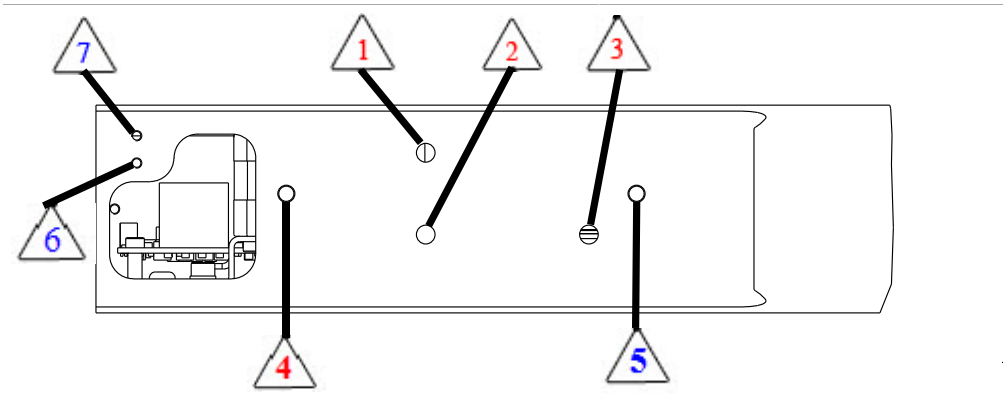


Figure 3-10 *Bottom view of Y axis, with mounting holes marked*

Tecan recommends that you use three M5x0.8 screws (16mm long for at least 7mm of thread engagement) with flat washers installed from inside the axis to secure the module. Screws should include some form of thread locker.

3.5.5 Mounting the Z Axis

Figure 3-11 shows a side view of a left-oriented Standard Z axis. A right-oriented Z-axis is the mirror image of Figure 3-11. (See Figure C-5, “Left-oriented Z axis mounting hole dimensions” on page C-4 for more detail.) The mounting holes and dowel pins are the same for all Z axis types.

The numbers in the figure refer to the following mounting holes:

1-4: mounting holes

5-6: dowel pins for alignment

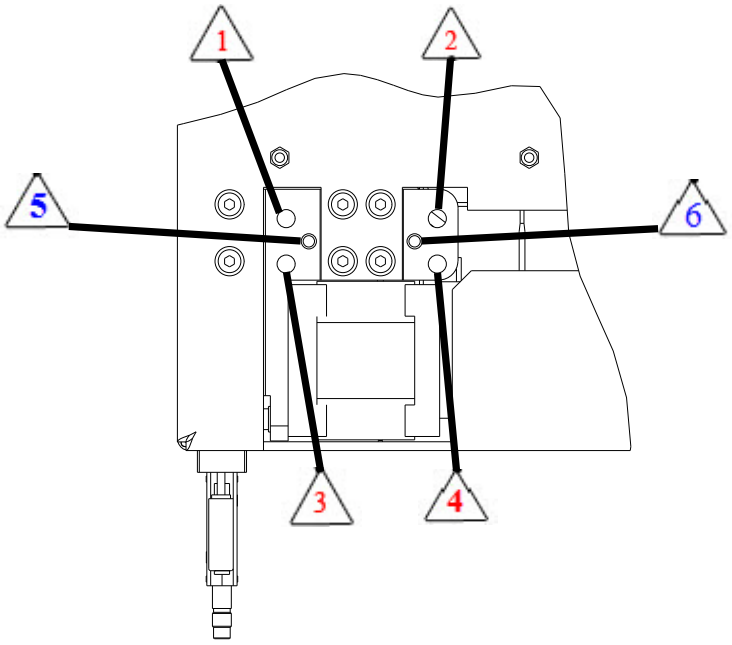


Figure 3-11 *Side view of Z axis, with mounting holes marked*

Tecan recommends that you use four M4x0.7 screws (16mm long for at least 7mm of thread engagement) installed from inside the Z axis to secure the module. Screws should include some form of thread locker.

3.5.6 Mounting a Dual Arm Cavro Omni Robot

The procedure for mounting a dual arm model is the same as for a single arm model.

3.6 Leveling

In addition to ensuring that the mounting surface is level before you mount the Omni Robot, the OEM user must ensure that the mounted Omni Robot is level in relation to the work surface.

Note: *If the Omni Robot is not level to the work surface, the cLLD system will not accurately read the liquid level of your samples.*

When the Y axis and Z axis are integrated with the X axis, they do not need to be leveled separately.

To level the X axis:

1 Locate the dowel pin holes on either side of the X axis. The dowel pin holes are numbered 7 through 9 in Figure 3-8.

2 For fixed alignment to the work surface, the mounting surface should include two dowel pins inserted into the upper dowel pin holes (holes 7 and 8).

3 To allow rotation for manual alignment to the work surface, the mounting surface should include a single dowel pin inserted into the lower, central dowel pin hole (hole 9).

4 Calibrate the module to the work surface according to the instructions in the Configurator documentation (provided as part of the Omni Robot software installation CD).

To level the Y and Z axes:

1 Locate the dowel pins on the axis. The Y axis dowel pins are numbered 4 and 5 in Figure 3-10; the Z axis dowel pins are numbered 5 and 6 in Figure 3-11.

2 For alignment to the work surface, the mounting surface should include two correctly sized and spaced dowel pin holes.

3 Calibrate the module to the work surface according to the instructions in the Configurator documentation (provided as part of the Omni Robot software installation CD).

3.7 Mounting Additional Modules

Note: The Omni Robot is capable of being reconfigured for different options (see Section 3.2.4, ”Z Axis Options” on page 3-3 for some possibilities). However, the reconfiguration may require loading different ACB and CIB software, and/or different cabling and hardware connections. Reconfiguring the Omni Robot should not be performed by OEM *users. Upgrades must be performed only at the factory or by trained field service engineers.*

Mounting different Tecan pumps and diluters, however, can be done by OEM or end users. The Omni Robot is compatible with most Tecan pumps and diluters. Contact Tecan Sales for specific information about your configuration. A mounting groove is provided on the underside of the X axis; these grooves can be used to mount modules (such as Tecan pumps) with T-nuts to the Omni Robot. Figure 3-9 shows an example of a deployment with an example pump module.

3.8 Installing Liquid Tubing

Liquid tubing must be installed for use with the liquid system.

3.8.1 Standard and Dual Z Axis Tubing

To install the liquid tubing:

1 Remove the cover from the Z axis by pulling it straight off as shown in Figure 3-12.

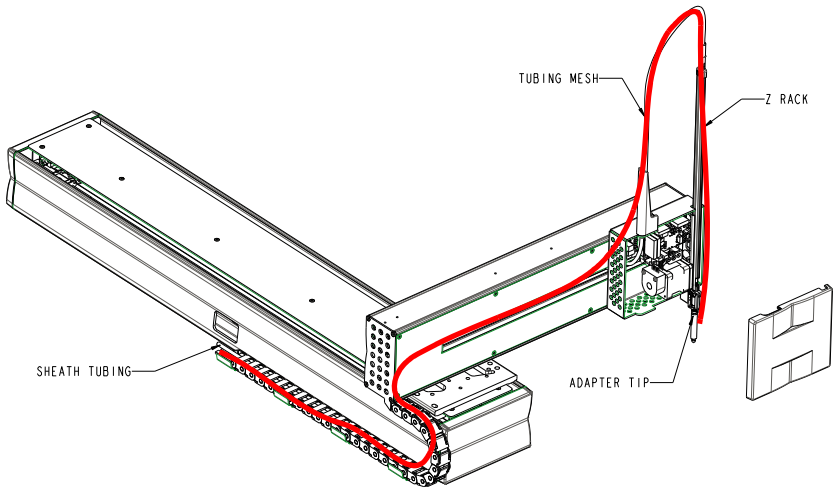


Figure 3-12 *Installation of liquid tubing showing tubing path, Standard Z axis*

2 Feed the tubing into the sheath tubing and through the robot until it reaches the Z axis.

3 Feed the tubing up the hole into the mesh sleeve with the coax cable and down through the Z rack.

Note: *In the Dual Z, the mesh closest to the cover connects to the front-most probe or DiTi adapter.*

4 Using a small screwdriver or Allen wrench, direct the tubing into the tip adapter.

5 Feed the tubing out the tip adapter.

6 Connect the tubing onto the probe or DiTi adapter.

– Be careful not to kink the tubing.
– Do not cut the tubing at an angle.

Wrap the tube near its end with a small piece of emery cloth to have a better grip on the tube.

Note: *Do not use emery cloth on probes, as it would abrade the delicate probe coating. Use a dry piece of emery cloth for an improved grip on tubing only; wet sandpaper could leave tiny particles and thus clog inside the tubing and/or probe.*

7 While holding the tube end wrapped in emery cloth, insert the blank, conical end of the probe 6 to 8 mm into the tubing along its axis.

8 Install the probe or DiTi adapter to the tip adapter and pull the excess tubing out.

9 Replace the cover on the Z axis.

3.8.2 Universal Z Axis Tubing with 8-Channel Head

To install the liquid tubing for the 8-channel pipetting head used with the Universal Z axis, follow these steps.

1 Remove the cover from the Universal Z axis by pulling it straight off as shown in Figure 3-13.

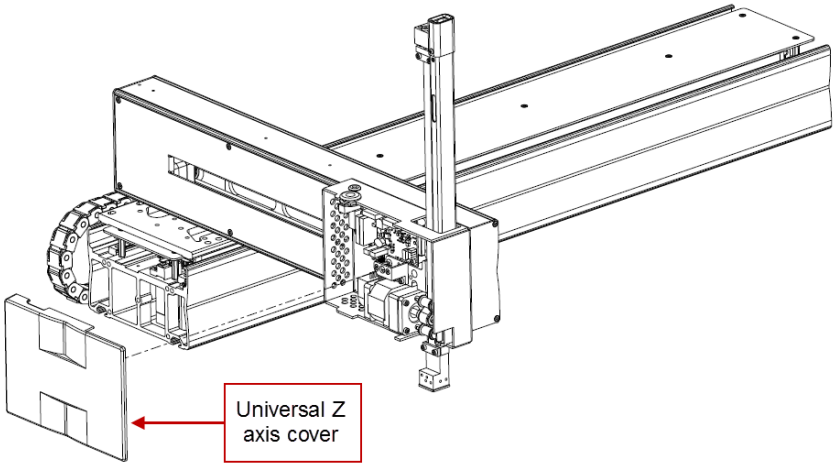


Figure 3-13 *Removing Universal Z axis cover*

2 Mount the 8-channel pipetting head tubing guide spacer, tubing ramp, and 8-channel tubing guide to the Universal Z axis tubing guide using two M3x12 FHCS as shown in Figure 3-14.

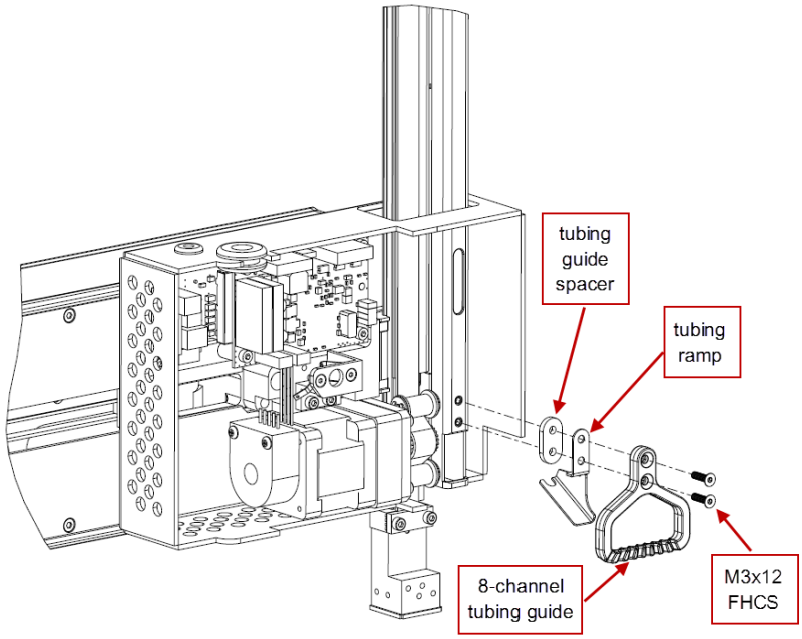


Figure 3-14 *Mounting tubing guide components*

Note: *When installing the tubing ramp component, be sure to align the cLLD coax and braided sleeve within the Z axis tubing guide so it can be captured within the U-shaped cut-out.*

Note: *The 8-channel head tubing guide slots are not symmetrical about the mounting holes. The 8-channel head tubing guide needs to be installed with the guide slots oriented toward the rear of the robot as shown in Figure 3-15.*

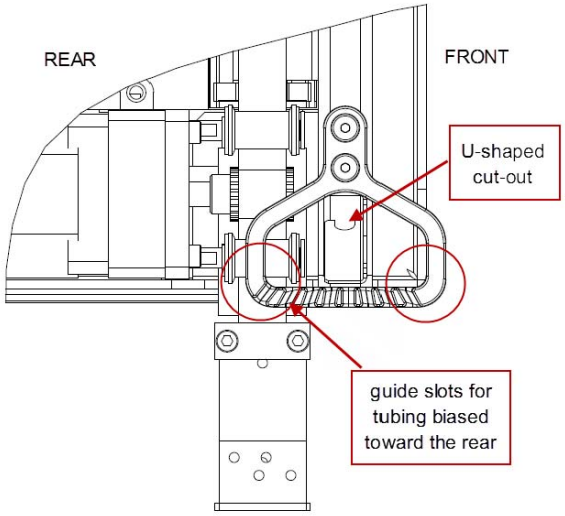


Figure 3-15 *8-channel head tubing guide orientation*

3 Bundle the tubing together with a small rubber band at the very end and feed into the e-chain from behind the X axis. (Tubing can be fed individually, but it will take longer and is more likely to snag during installation.)

4 Move the arm back and forth as tubing reaches the e-chain bend, to facilitate tubing installation.

5 Install the tubing guide exit component at the rear of the axis as shown in Figure 3-16 (make sure it is aligned to the rear of the Y axis).

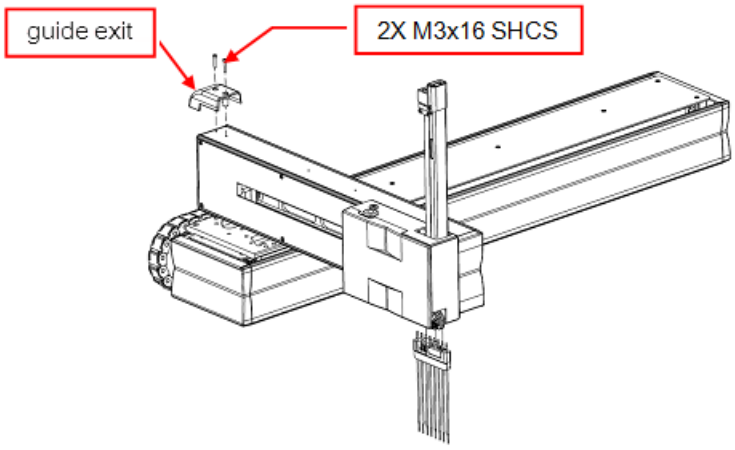


Figure 3-16 *Installing the tubing guide exit component*

6 Route tubing from behind the robot up and through the tubing guide exit component.

7 Install the tubing cover over the tubes, but under the molded piece to conceal them from view and hold them in place as the arm moves as shown in Figure 3-17.

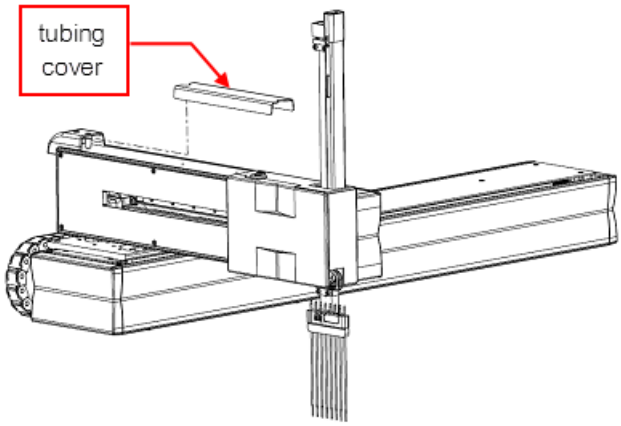


Figure 3-17 *Installing the tubing cover*

8 Feed tubing through the opening in the tubing guide entrance component. The tubing should feed from the underside (side with bosses) and out the top.

9 Install the tubing guide entrance component over the tubing cover and secure to the Y axis as shown in Figure 3-18.

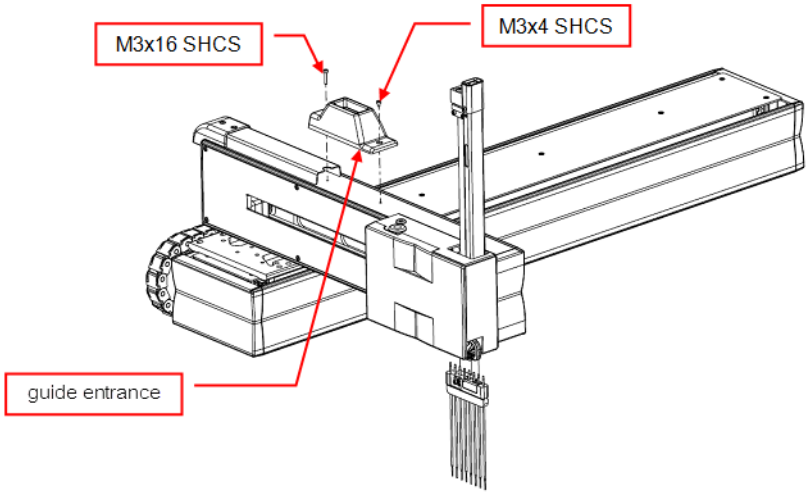


Figure 3-18 *Installing tubing guide entrance component*

10 Continue to feed the tubing down the Z axis tubing guide attached to the linear rail. Be careful not to tangle the tubing with the cLLD coax.

11 Ramp will direct tubes out the front of the guide where they can be separated, and guided to the probes.

12 Install the 8-channel head at this time if it has not already been installed. For installation instructions, see Section 3.9.2, ”8-Channel Pipetting Head”.

13 Connect tubing to probes and route tubing into the guide slots of the 8-channel tubing guide. Refer to Figure 3-15 for location. Wrap the tube near its end with a small piece of emery cloth to have a better grip on the tube.

Note: *Do not use emery cloth on probes, as it would abrade the delicate probe coating. Use a dry piece of emery cloth for an improved grip on tubing only; wet sandpaper could leave tiny particles and thus clog inside the tubing and/or probe.*

14 While holding the tube end wrapped in emery cloth, insert the blank, conical end of the probe 6 to 8 mm into the tubing along its axis.

15 Ensure that the tubing routed from the Y axis into the Z axis tubing guide is not tangled or routed in such a way that the tubing will rub on the linear rail or other abrasive surfaces. Refer to Figure 3-19 for recommended routing and Figure 3-20 for examples of what to avoid.

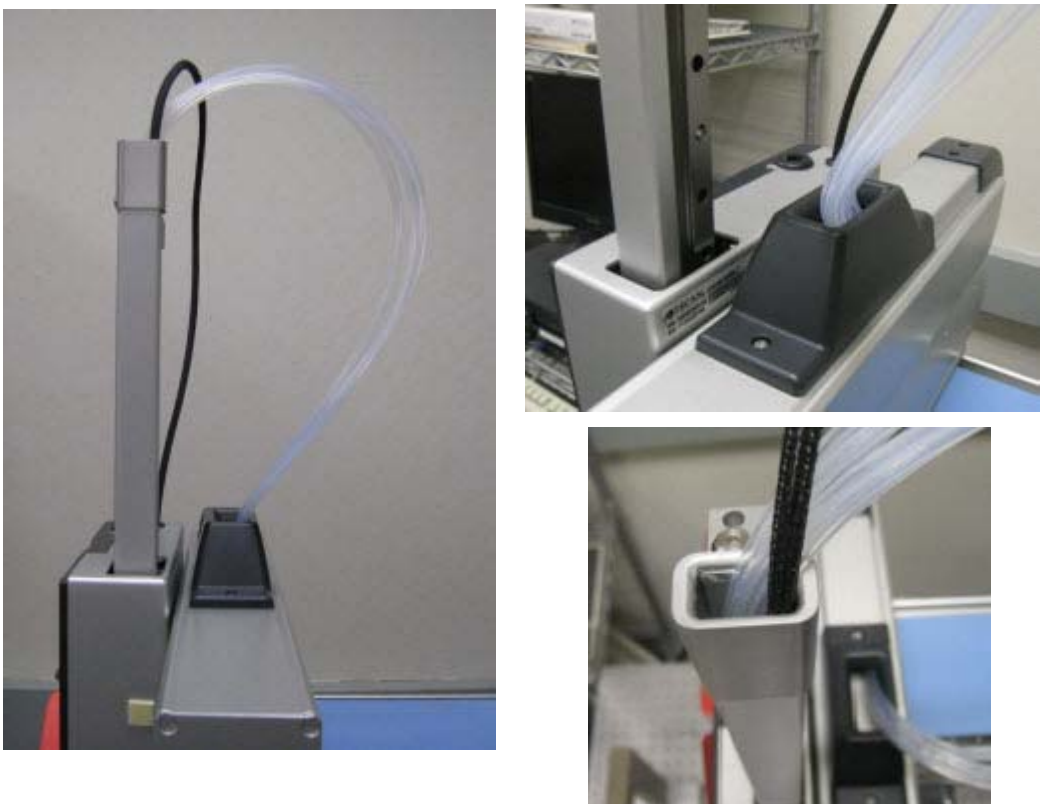


Figure 3-19 *Proper routing of the tubing without tangles or rubbing*

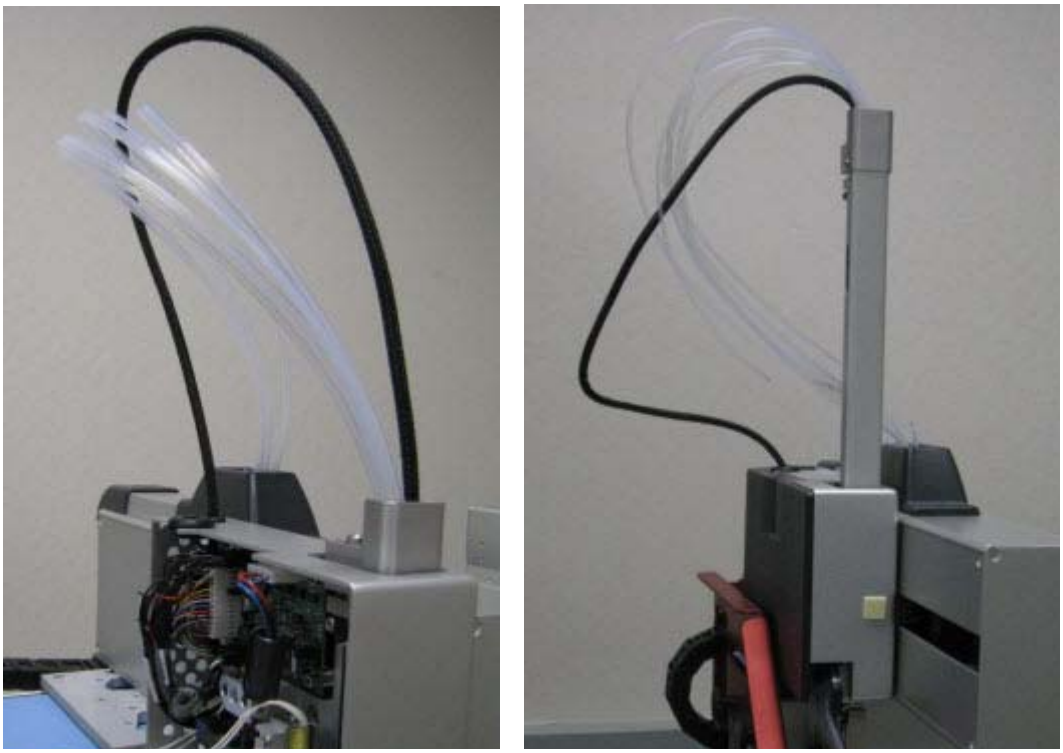


Figure 3-20 *Examples of problems with routing the tubing*

16 Tubing can rub lightly on the e-chain behind the Y axis (see Figure 3-21). It is recommended that the routing be checked at this point by moving the arm back and forth along the X axis and securing the tubes or adjusting the routing so they will not rub on the e-chain.

Figure 3-21 *The correct position of tubing and e-chain behind the Y axis.*

17 Replace the cover on the Universal Z axis.

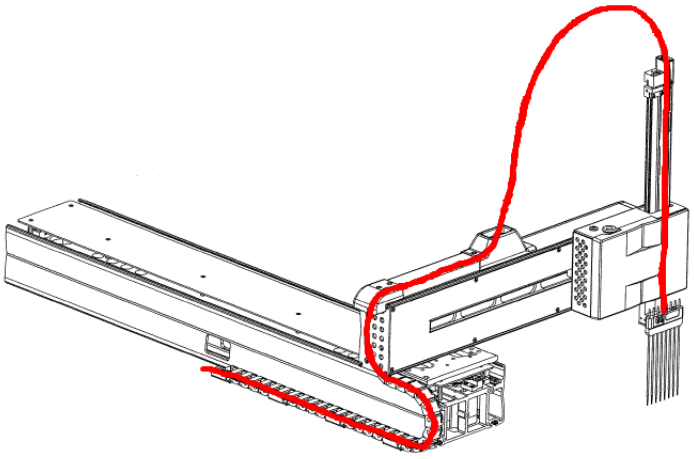


Figure 3-22 *Universal Z axis 8-channel head tubing path*

3.9 Attaching Probes and Disposable Tips

3.9.1 Single Probe or DiTi Cone

Instructions for attaching a single probe or DiTi cone to the Omni Robot can be found in Appendix B, “Maintenance Tasks” — specifically in Section B.2, ”Probes” on page B-2 and Section B.3, ”Disposable Tip (DiTi) Pickup Mechanism and Tips” on page B-7.

3.9.2 8-Channel Pipetting Head

1 Be sure the tubing has already been installed into the system. See Section 3.8.2, ”Universal Z Axis Tubing with 8-Channel Head” on page 3-17.

2 The Z axis cover should remain unattached after tubing installation. If the cover has been replaced, remove it (see Step 1 of Section , ”To install the liquid tubing for the 8-channel pipetting head used with the Universal Z axis, follow these steps.” on page 3-17).

3 Remove sheet metal PCBA cover as shown in Figure 3-23.

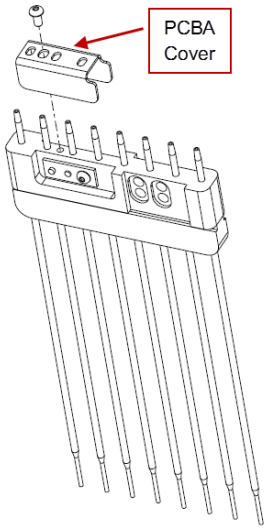


Figure 3-23 *Removing 8-channel head PCBA cover*

4 Verify that the 8-channel head PCBA is located on the same side of the 8-channel pipetting head as the universal mount as shown in Figure 3-24. If it is not, dismount it and remount on the opposite side of the block.

5 Place 8-channel pipetting head attachment against the universal mount with the PCBA in front of the universal mount. Align head to the back surface and bottom lips as shown in Figure 3-24.

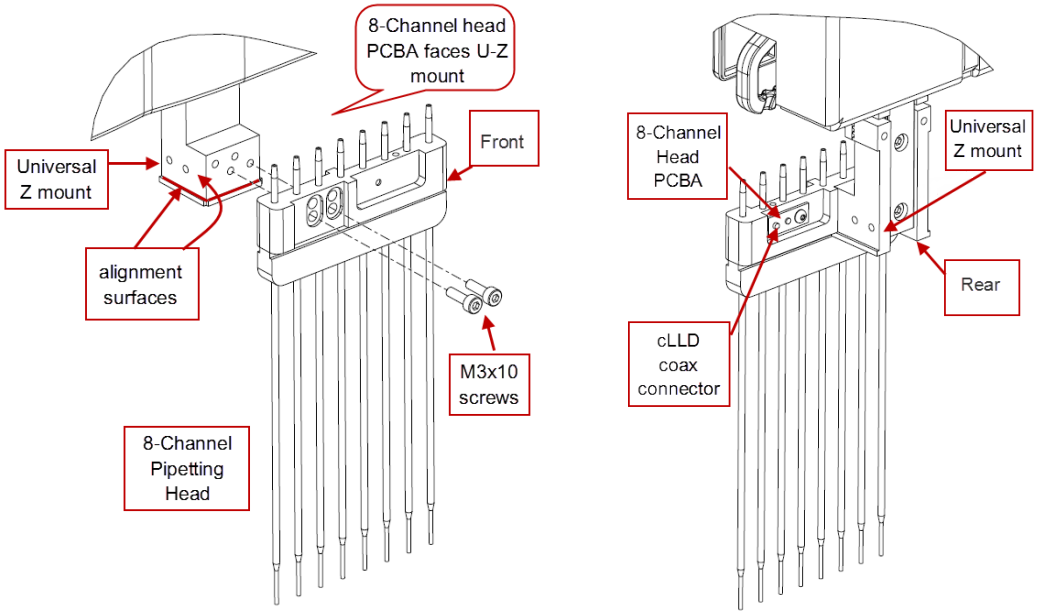


Figure 3-24 *Aligning 8-channel head with Universal Z (two views)*

6 Mount using the two M3 x10 screws provided.

7 Connect the ground wire from the PCBA to the metal axis home block above the plastic mounting block using the hardware provided (star washer, followed by lug, followed by mounting screw) as shown in Figure 3-25.

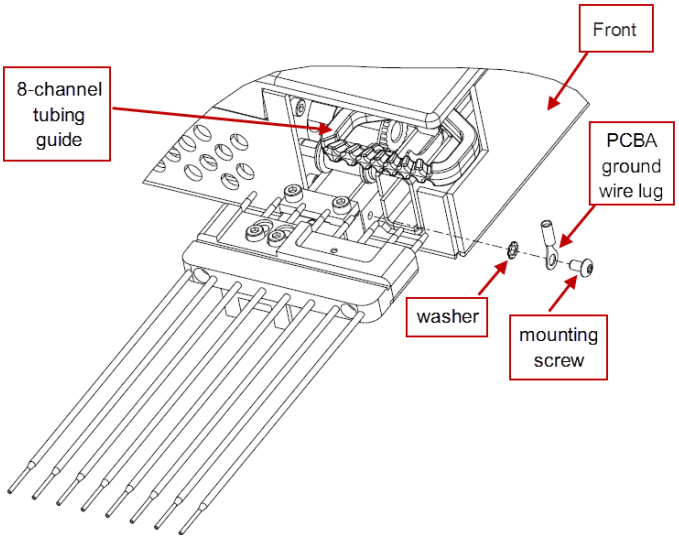


Figure 3-25 *Connecting 8-channel head PCBA to axis home block*

8 Connect the cLLD coaxial cable to the small PCBA and route the wires below the screw head.

9 Reinstall the PCBA cover, making sure to feed the wires through the small opening at the top. Align it against the screw head.

10 Connect tubing to the probes and re-attach the Universal Z axis cover. Wrap the tube near its end with a small piece of emery cloth to have a better grip on the tube.

Note: *Do not use emery cloth on probes, as it would abrade the delicate probe coating. Use a dry piece of emery cloth for an improved grip on tubing only; wet sandpaper could leave tiny particles and thus clog inside the tubing and/or probe.*

11 While holding the tube end wrapped in emery cloth, insert the blank, conical end of the probe 6 to 8 mm into the tubing along its axis.

3.10 Verifying the Assembly

Verify that the mechanical integration is complete by:

1 Checking the stability of the work surface.

2 Checking the Omni Robot for visible damage.

3 Verifying that the Omni Robot is level.

If the mechanical integration is successfully completed, you can proceed to the electrical integration described in Section 4, ”Electrical Integration” on page 4-1.

4 Electrical Integration

The Omni Robot is designed to make electrical hookup as easy as possible. The main power and communication ports on the CIB are easily accessible from the exterior of the Omni Robot, without the need to remove any covers. The electrical setup tasks are the same regardless of the OEM system or mechanical layout.

This chapter assumes that the mechanical installation described in Chapter 3, “Mechanical Integration” has been completed.

Caution! *Always remove power before establishing or removing any connections.*

This chapter is organized into the following sections:

4.1 Introduction

The CIB contains five externally accessible ports, as shown in Figure 4-1:

- A 24V power port
- A standard Ethernet port
- A DA15 port (compatible with a DA15 cable, also called a Cavro® Bus cable)
- An RS-232 port (supported when operating in Omni Embedded mode)
- An auxiliary CAN port (reserved for future support)

The electrical integration consists of the following tasks:

- Embedded Mode:
  - Connect host controller to the CIB Ethernet port

  or
  - Connect host controller to the CIB RS-232 port
- Command Processor Mode:
  - Connect a PC to the CIB Ethernet port
- Connect a power supply to the 24V power port.
- Ground the Omni Robot to a grounding plane.

- Optionally, connect additional modules to the DA15 port.

To perform the electrical setup, you will need the following tools and equipment:

- A standard Category 5 patch cable
- A suitable power supply, based on the electrical specifications listed in Appendix C, “Specifications”

If any additional Tecan modules will be connected to the Omni Robot, you will also need a cable to connect the module to the DA15 port.

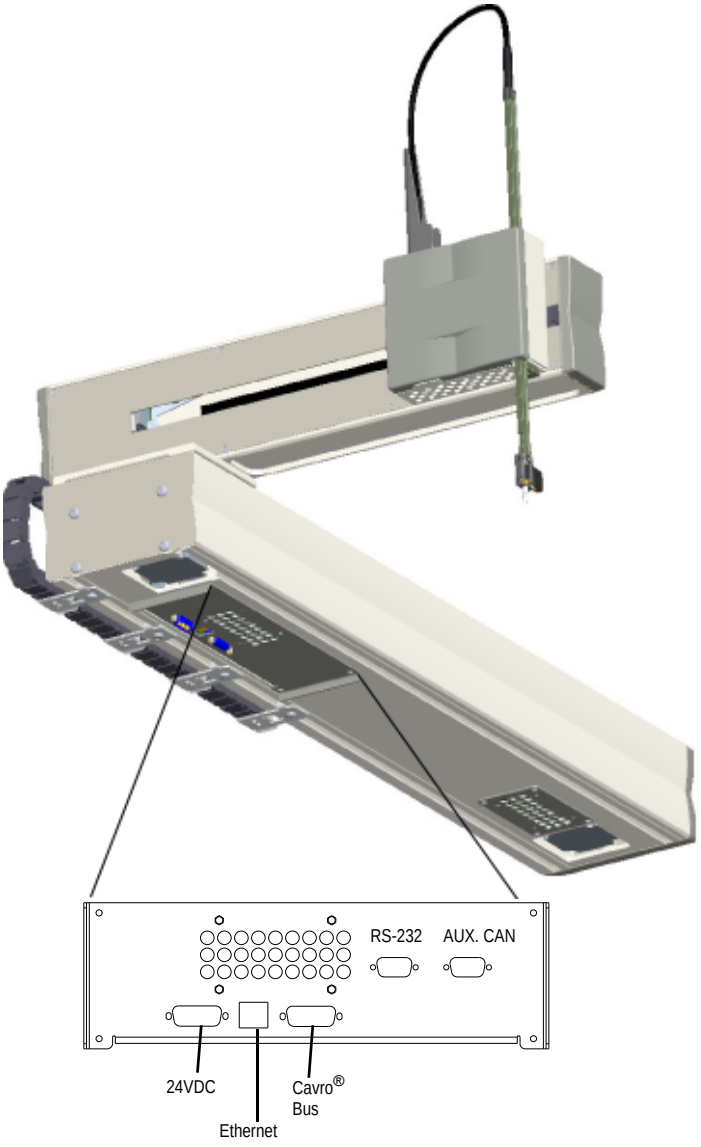


Figure 4-1 *Location of Cavro Omni Robot power and communication ports*

4.2 Connecting the PC or Other Proprietary Host Controller

In Omni Command Processor mode, connect the CIB directly to a dedicated client PC running Windows XP, Vista, or later with a standard Category 5 patch cable. A dedicated client PC is recommended for the best performance.

In Omni Embedded mode, connect the CIB directly to a dedicated client PC or proprietary client controller with a standard Category 5 patch cable. Connection can also be made with a straight DE9 cable. The PC or proprietary controller must be capable of sending Omni Embedded Command Set (OECS) commands using the OEM protocol.

WARNING! *The timing of the execution of Omni Robot commands may be affected by the host PC or controller performance. Tecan recommends that you verify the execution timing of your Cavro® Omni Robot with the host PC or controller setup in its working configuration.*

4.3 Connecting the Power Supply

The OEM user is responsible for providing a power supply for the Omni Robot. The requirements for the power supply are listed in Section C.8, ”Electrical Specifications” on page C-22.

4.4 Grounding

In order to ensure the proper operation of the cLLD system, the Omni Robot must be properly grounded to a grounding plane. This can be accomplished by mounting the module to metal supports, which in turn are mounted to a metal work surface. However, the supports for the Omni Robot module can be made of nonconductive material, but an electrical connection from the module must be made to the metal work surface. Figure 4-2, “Grounding diagram for the Cavro Omni Robot” on page 4-4 illustrates how grounding should be accomplished.

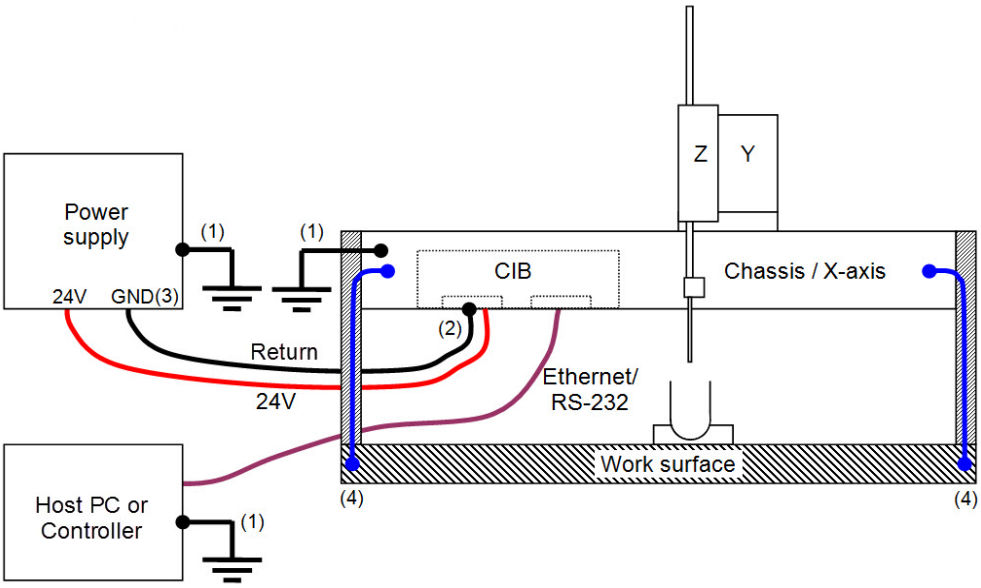


Figure 4-2 *Grounding diagram for the Cavro Omni Robot*

The following notes are referenced in Figure 4-2:

Note (1): All earth ground connections of the whole system should be established from one AC power outlet group to avoid ground loops.

Note (2): GND (24V return) is internally connected to the axis group of the Omni Robot through the CIB board. See Section C.9, ”Connector Pin Out Diagrams” in Appendix C.

Note (3): To avoid ground loops, GND (24V return) at the power supply should be floating (not connected to the earth ground).

Note (4): The work surface needs to be firmly connected to the Omni axis group to ensure proper function of the liquid level detection.

Caution! *Failure to properly ground the module could result in the module failing to communicate properly with the host PC or controller.*

4.5 Connecting Additional Modules

Connect additional Tecan components, such as pumps, to the Omni Robot with a DA15 cable. The cable passes commands to the Tecan components via CAN in Embedded Mode and CAN or RS-485 in Command Processor Mode, and sends responses or events from the modules to the CIB.

To connect an additional module, connect one end of a DA15 cable to the DA15 port on the CIB, and connect the desired Tecan component(s) to the other end of

the DA15 cable. A pin out diagram of the DA15 port is available in Section C.9, ”Connector Pin Out Diagrams” on page C-31.

Caution! *Any cable that is connected to the DA15 port must not be connected to pins 5 and 6. Using pins 5 and 6 could cause the robot or module to malfunction.*

Caution! *Any Tecan CAN component connected to the DA15 port on the CIB must have its CAN baud rate set to 500K.*

4.6 Operating Multiple Omni Robots

If an Omni Robot is operating in close proximity to another Omni Robot, the two cLLD modules could interfere with one another because they are operating at the same frequency, leading to false liquid detection.

You can prevent interference between two cLLD modules in one of two ways:

- Maintain a minimum tip-to-tip distance between the two modules.

  The minimum distance varies depending on the value of the *LLDSensitivity* parameter; as the sensitivity increases, the cLLD system becomes more vulnerable to interference, and the minimum distance increases.
- Place a grounded piece of sheet metal between the two modules, to act as a shield and prevent interference.

For specific values regarding the allowable distance between two modules or the size of the metal shield, see Section C.10, ”Preventing Interference Between Two Robots” on page C-38.

4.7 Verifying the Assembly

After completing the electrical configuration, turn on the host PC or controller and plug in the Omni Robot power source. When the Omni Robot powers up successfully, the configuration has been completed successfully and the module is ready for operation. When the host PC or controller and the Omni are connected and powered up successfully, the Ethernet port on the CIB will blink if the Ethernet connection method is chosen. If the Ethernet port does not blink, then the module is not communicating with the host.

4.8 Power Interruptions and Initialization

This section describes what happens when the Cavro® Omni Robot stops and restarts due to an interruption in electric power.

4.8.1 Safe Failure

When power is interrupted for any reason, the brake is activated on the Cavro® Omni Robot's Universal Z axis to decelerate and stop the Universal Z axis with a payload within specification.

Note: *The Z axis can travel 11 mm before the brake begins to decelerate the system.*

Caution! There is a potential crash danger at power loss. Based on user speed and acceleration settings, overall stopping distance could be greater than the programmed Omni motion control profile.

4.8.2 Automatic Initialization

When power is restored after any electrical power interruption, the brake on the Universal Z disengages when the axis enables as part of the initialization command.

At the axis level, the initialization of each axis is determined by the values of the parameters previously set for that axis, such as speed, acceleration, init mode, etc. See Section 7.7.1, "Arm Motion Commands (Device Type A)" (Command Processor mode) or Section 6.9, "Axis Action Commands" (Embedded mode) for information about the initialization command sequence for each specific type of axis.

In Command Processor mode, at the arm level, the axes are initialized together as a group. You can control the sequence of axis initialization using the travelpos parameter. See CavroInit in Chapter 7 (Command Processor mode) for more information about setting the travelpos parameter.

In Embedded mode, each axis is initialized individually. See Initialize Axis in Chapter 6 (Embedded mode) for more information.

Note: *Double initialization is the default setting for all Z axes*

4.8.3 Manual Initialization

When power is restored after any electrical power interruption, the brake on the Universal Z disengages when the axis enables as part of the initialization command. Then the end user can manually initialize each axis in the sequence that fits their needs or hardware configuration, For example, if the gripper is attached to a Universal Z axis, the following is one possible initialization sequence:

1. Bring the X and/or Y axis to an empty space within the work area where the gripper can move freely without colliding with any objects:
   - Command Processor mode: use the PreInitMoveRel command (see page 7-67)
   - Embedded mode: use the Move Axis to Relative Position Before Initialization command
2. Initialize the gripper with the GripperInit command.
3. Initialize the Universal Z axis with the CavroInit command to bring it to a safe travel position.
4. Initialize the X and Y axes with the CavroInit command.

This page has been left intentionally blank.

5 Integration with Omni Configurator and Cavro Fusion

Omni Configurator

The Configurator is a Windows application that allows the user to reconfigure the power-on default settings of the Omni Robot via Ethernet when needed.

Changes to the default (power-on) configuration settings are not generally recommended. Any configuration changes required by an OEM application should be made programmatically within the application, rather than through the Configurator. This avoids potential problems inherent in making assumptions about the default settings of the robot.

The Configurator is described in a separate document included with the Omni Robot.

Cavro® Fusion

You can also operate the Omni Robot with the Cavro® Fusion software. Fusion is Windows-based software that allows users to manipulate the Omni Robot and other attached devices. This software is intended for system evaluation and demonstration purposes only. The Cavro® Fusion software is described in the *Cavro® Fusion Software User Guide*. Contact your Tecan representative for more information about the Cavro® Fusion software.

This chapter is organized into the following sections:

5.1 Verifying the Software Configuration

WARNING! *If the configuration of the robot is overwritten with incorrect configuration settings, this could cause injury and/or system damage. Be sure to verify the configuration settings and recalibrate the robot before operating the robot.*

Note: *When the Command Processor is started, it will check the version number of all other Omni components installed. A warning message will be displayed if incorrect version numbers exist in the system.*

After installing and configuring the software, the OEM user should test the integration by using the Configurator to manipulate the Omni Robot. If all three axes respond to commands, then the integration was successful.

The following procedure uses these assumptions:

- The module consists of one arm and three axes: one X axis, one Y axis, and one Standard Z axis.
- The addresses of the axes are 1 (for the X axis), 2 (for the Y axis), and 3 (for the Z axis).
- The address of the CIB is 0.

Note: *Axis IDs are determined by jumpers set at the factory during system assembly. Typically, the first arm (axis group) is given ID 0 (A0), its axes are given IDs 1, 2, and 3 respectively. For a dual arm configuration, the second arm is typically given arm ID 1 (A1) and the axes are given IDs 4, 5, and 6. For a Dual Z axis, the additional axes are given IDs 7 for A0 and 8 for A1. If the Omni Robot axis addresses differ from this, use the addresses that have been set for those components.*

Note: *Be sure to rerun the Configurator if an axis is added to or removed from the system.*

Perform the following tasks to test the software configuration:

1 Log in to the Configurator, as described in Step 5 on page 7-4.

Note: *If you changed the IP address of the CIB, you must connect using the new address. The Configurator software automatically tries to connect using the default address; when the connection fails, enter the correct IP address and click* ***Connect*** *to initiate the connection.*

2 Verify the device IDs in the OMNI Configuration window.

3 From the Omni shortcut bar on the right, click **Command Line Tool**.
The Configurator Command Line window appears.

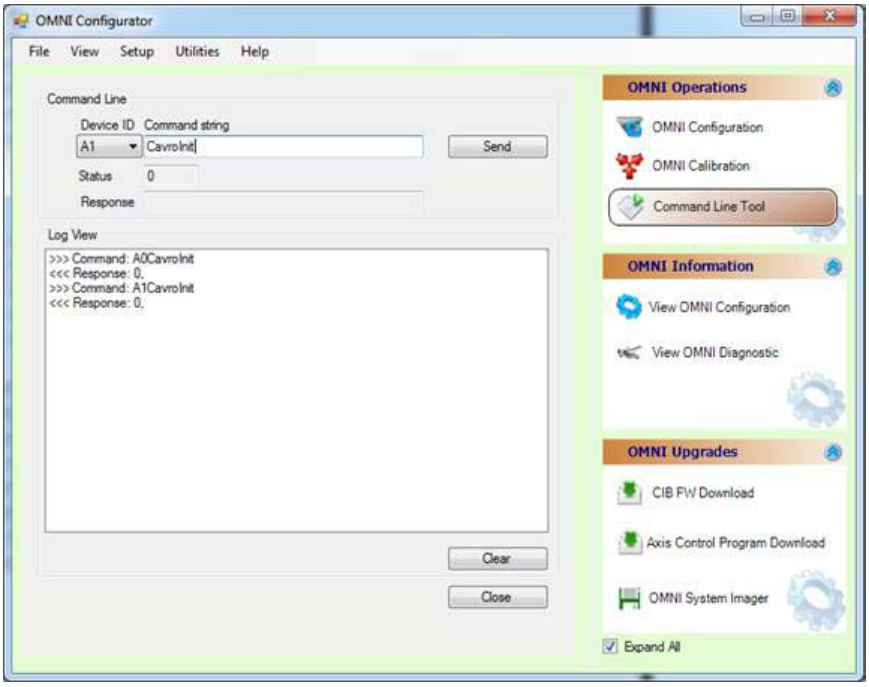


Figure 5-1 *Configurator Command Line window*

4 In the **Device ID** menu, select the device ID for the Omni Robot arm (**A0**).

5 In the **Command** string field, enter the command **CavroInit**, and click the **Send** button.

Note: *The Configurator Command Line Tool automatically adds the delimiter "/".*

6 After the response appears, enter the command **MoveAbs,100,100,100** and click **Send**.

The arm moves to the given coordinates, and the results of the command appear in the Log View pane.

5.1.1 Troubleshooting

If the Configurator software cannot connect to the Cavro® Omni Robot:

- Are the IP address and port number set properly on the CIB, in the software, and on the PC? Ping the Omni Robot from the host to determine if it is reachable.
- Is there power to the Cavro® Omni Robot? The green LED on the CIB should be blinking (remove cover on the CIB to access).

If an axis does not move to the proper position, check the following:

- Are all cables securely fastened?

5.2 Omni Robot System Diagnostics

Cavro® Omni Robot diagnostics (such as the total axis move counts, overall distance travelled by the arm, and the ACB on-board temperature reading) can be viewed using the Cavro® Configurator software.

To view diagnostic information:

1. Run the Configurator.
2. From the **Utilities** menu, select **Omni Information > View Omni Diagnostic**. (You can also find **View Omni Diagnostic** under the **Omni Information** shortcut bar at the right of the Configurator display.

   The Diagnostic information can be viewed onscreen as shown in Figure 5-2, or sent to the system default printer.

   See the Cavro® Configurator Software documentation for more details.

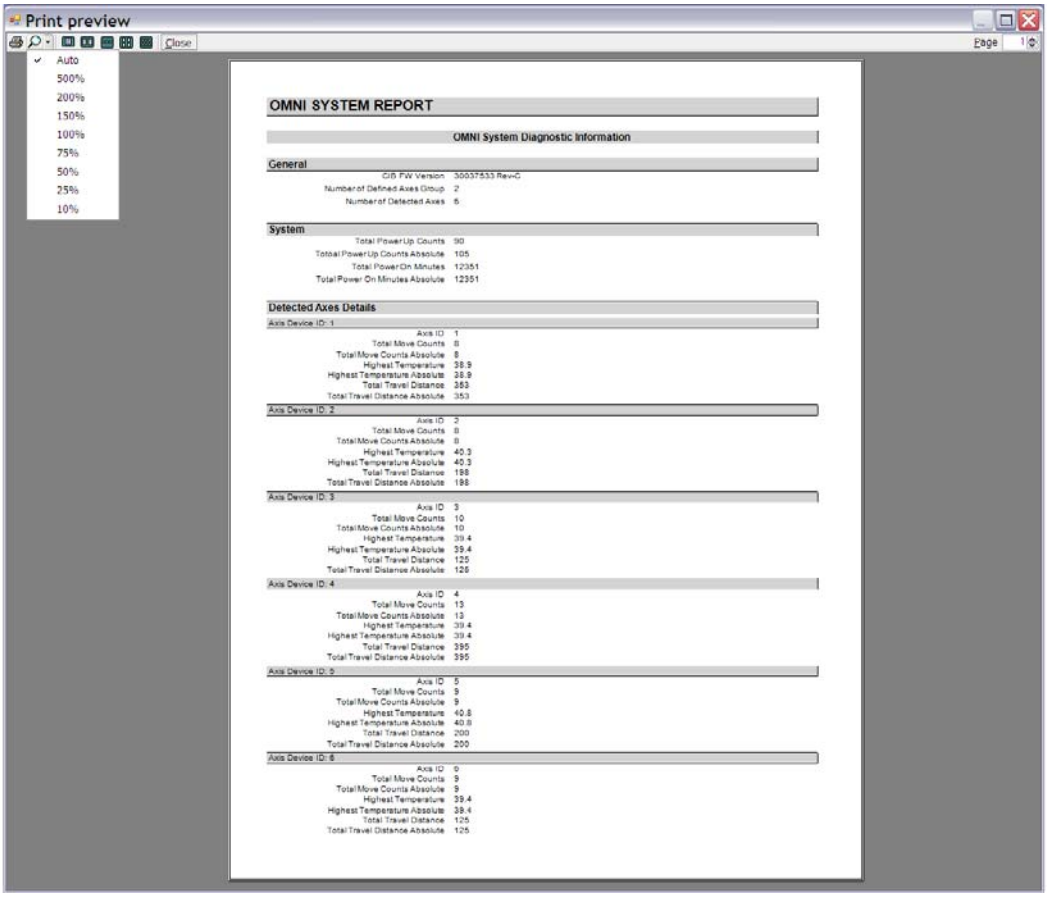


Figure 5-2 *System Diagnostics Window*

5.2.1 Omni Robot Log Files

Log files for the Cavro® Omni Robot are found in the location specified in `CmdProcMain`. For details of log file locations, see Table 7-16, CmdProcMain Class Methods.

This directory holds up to 10 log files, each with a maximum size of 10 MB. As each file is filled, a new one is created, until 10 files exist. Thereafter, as each new log file is created, the oldest file is deleted.

This page has been left intentionally blank.

6 Operating in Omni Embedded Mode

This chapter provides reference information for the Omni embedded control methods and command set. For an example of how to control the Omni with embedded commands in the Windows OS environment, see Section 6.13, ”Controlling Omni Hardware” on page 6-138.

This chapter is organized into the following sections:

6.1 Communication and Connectivity

When operating in the Omni Embedded mode, you can choose only one communication method. The communication method is determined when one of the following events occur:

- Ethernet communication is chosen after a Ethernet connection is established via the embedded connection port number (default: 9897).
- RS-232 is chosen when a valid OEM command string packet is received on the RS-232 port.

To reset the communication method, perform a power cycle on the Omni, and then the next event (Ethernet or RS-232) determines the new communication method.

6.1.1 System Architecture

Figure 6-1 illustrates the Cavro® Omni system architecture.

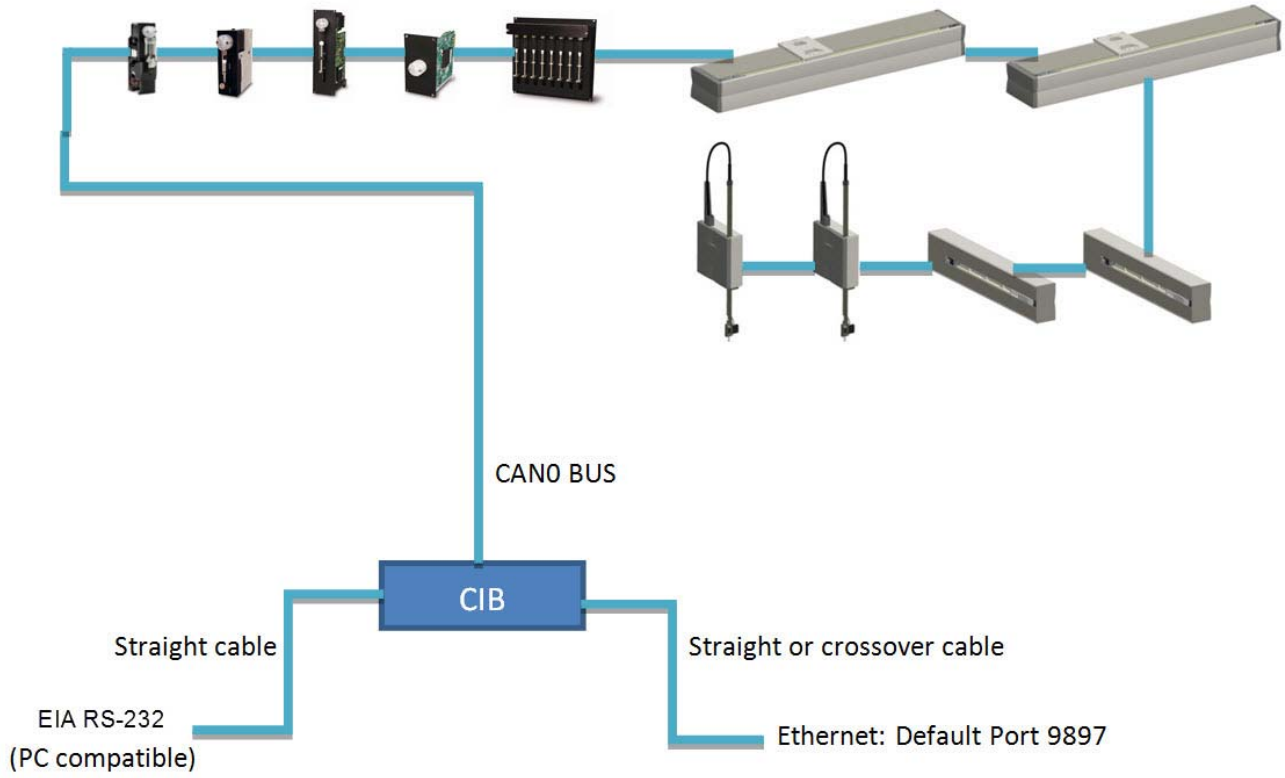


Figure 6-1 *Communication architecture*

Serial RS-232 Communication

This is an EIA RS-232 (PC compatible) port. Connection must be established using a straight RS-232 cable.

The Omni factory default configuration is set to 38400,n,8,1. Baud rates: 9600,n,8,1 and 38400,n,8,1 are supported. To change the baud rate, use the Set RS-232 Baud Rate command. To restore the setting to the factory default value, place J13 on CIB board and perform a power cycle.

TCP/IP Communication

The TCP/IP connection can be made using either a straight or crossover Category 5 patch cable.

MAC Address

Each CIB board is assigned with a unique MAC address which is labeled on the CIB board. To get the value of the MAC address, use the Get MAC Address command.

TCP/IP Address Setting

The CIB factory default IP address is 192.168.0.198. To change the IP address, use the Set IP Address command. To restore the setting to the factory default value, place a jumper on header J13 on CIB board and perform a power cycle.

TCP/IP Listening Port

The CIB factory default TCP/IP listening port number is 9897 for Omni Embedded control. To change the port number, use the Set Port Number for Embedded Connection command. To restore the setting to the factory default value, place J13 on CIB board and perform a power cycle.

6.2 Omni Robot Coordinate System

The following images show the Omni Robot axis positions and parameters. For more information about these parameters, see Section 6.8, "Axis Parameter Commands". Figure 6-2 shows the X and Z axis parameters for a single arm model.

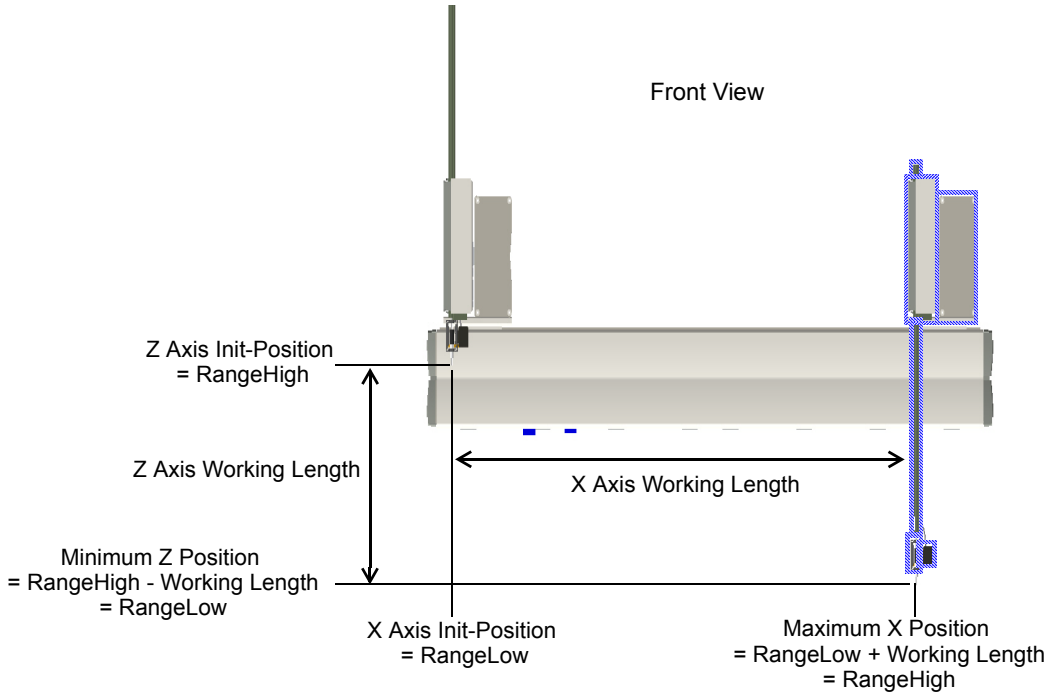


Figure 6-2 *Single arm Cavro Omni Robot axis positions (X and Z axes)*

Figure 6-3 shows the Y axis positions and parameters.

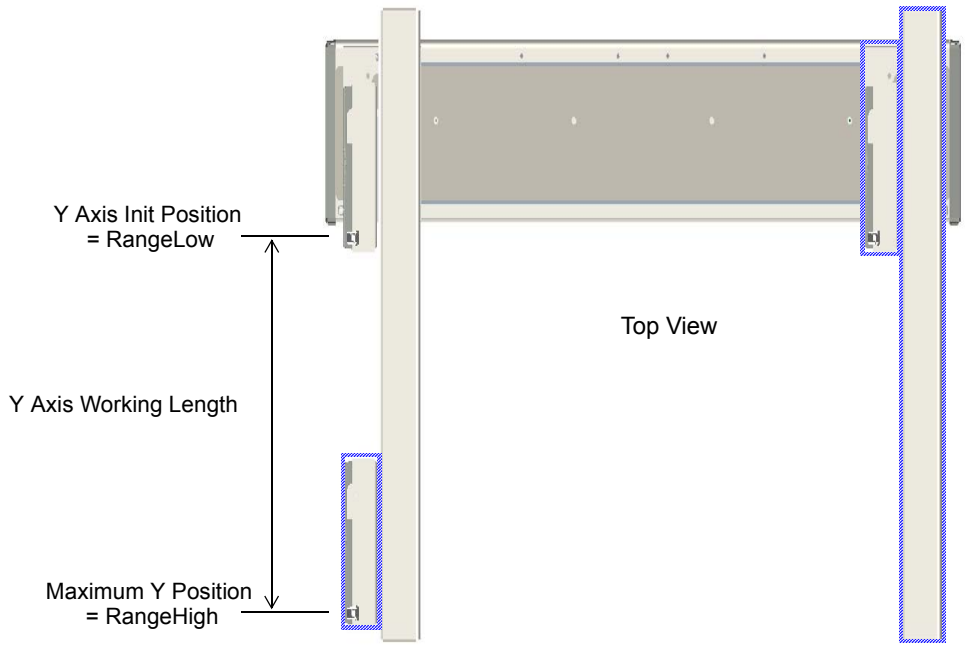


Figure 6-3 *Cavro Omni Robot Y axis positions*

Note: *Unlike X and Y axes, the Axis Initial Position (home) for Z axes defines RangeHigh; RangeLow is at the bottom, farthest from the home position.*

The Cavro® Omni Robot coordinate system provides significant flexibility in that it allows the user (application developer) to define the coordinate system to be used by the Omni Robot, relative to a position of interest to the user (for example, the X, Y, and Z coordinates of a specific location on the worktable or a test fixture). The Omni Robot will then use this relative coordinate system for all its functions, transparently mapping those coordinates to the physical coordinates of the robot.

The Cavro® Omni Robot coordinate system can use coordinates in the range of -10,000 to 10,000 mm.

Coordinate mapping is accomplished by moving the robot axes to a specific physical location, and then setting the coordinate values that are to be used to represent that location. The reference position can be set using any arbitrary values for the coordinates, completely independently of the physical coordinates, based on whatever makes sense in terms of the application. This is done by performing a calibration using the Omni Configurator software, or programmatically using the Omni command set.

For example, suppose you have a fixture for an array of test tubes, and want to use the top left tube as your reference point, call that location by coordinates 100, 100, 100. However, in physical coordinates, that tube is actually at X = 215 mm, Y = 149 mm, and the Z axis height is 128 mm. You can use the calibration

process to specify that the upper left test tube coordinates shall be called 100, 100, 100. Thereafter, all positional references in the application will be in terms of a coordinate system where 100, 100, 100 is at the designated test tube.

This is explained in more detail below. For simplicity, the following discussion will focus on the X axis position, but the principles apply to the Y and Z axes as well.

The X axis parameters shown in Figure 6-2 are:

- X axis Initial Position: the home physical position of the X axis.
- X axis Maximum position: the farthest physical location away from the axis home position.
- Working Length: the distance between the axis home and the maximum X position.
- *RangeLow*: the lowest coordinate address valid for the axis, relative to the reference position.
- *RangeHigh*: the highest coordinate address valid for the axis, relative to the reference position.

There is also an additional parameter, *AxisOffset*, which is the Axis home position coordinate (relative to the reference position), and is equal to *RangeLow* for the X or Y axis.

Figure 6-4, “X axis coordinate mappings relative to user-specified reference point” on page 6-6 shows the use of a reference position, and how it affects all the other parameters.

In terms of *physical* units, the initial (home) position of the X axis is at coordinate zero. For this example, assume the reference point is 215 physical units (mm) from the home position. As part of the calibration process, the user can manually move the robot arm to position 215, and then set this position to be 100 (via the Configurator calibration function to retain calibrated values, or by using the Set Pos command to programmatically alter values, with an argument of 100.

When the reference point is set, other parameters such as the *AxisOffset*, *RangeLow*, and *RangeHigh*, are calculated relative to this reference point. For subsequent commands, such as move commands, the coordinates are assumed to be relative to the new coordinate range.

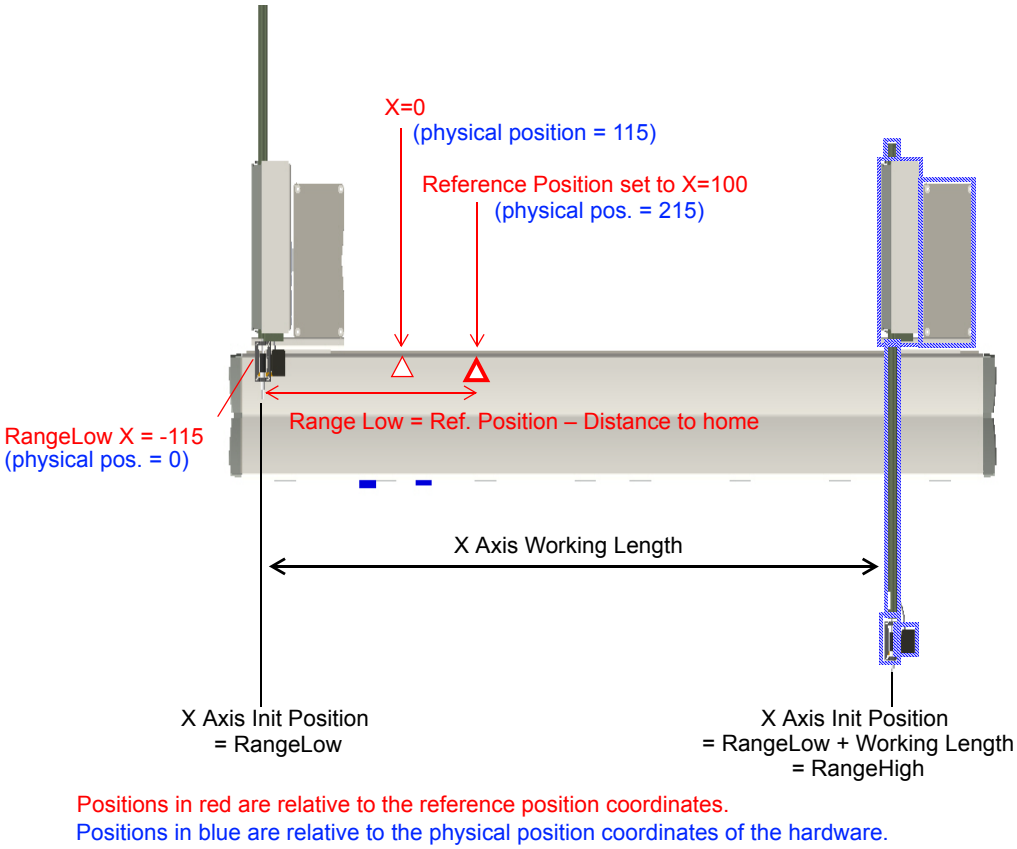


Figure 6-4 *X axis coordinate mappings relative to user-specified reference point*

Therefore, in the example in Figure 6-4, the positional parameters will be calculated as follows:

- *AxisOffset* = –115 (Reference position – distance to home; 100-215).
- *RangeLow* = –115.
- Assuming an X axis working length of 800, *RangeHigh* = 685.

This means that the X axis can move to positions between X = –115 and X = 685 as its working range.

6.2.1 Coordinates in a Dual Arm Configuration

For a dual arm configuration, the Y axis and Z axis positions and coordinate parameters are the same as for a single arm configuration.

Figure 6-5 shows the X axis coordinates for a dual arm configuration—in this case a configuration where the Z axis is mounted on the left side of each arm.

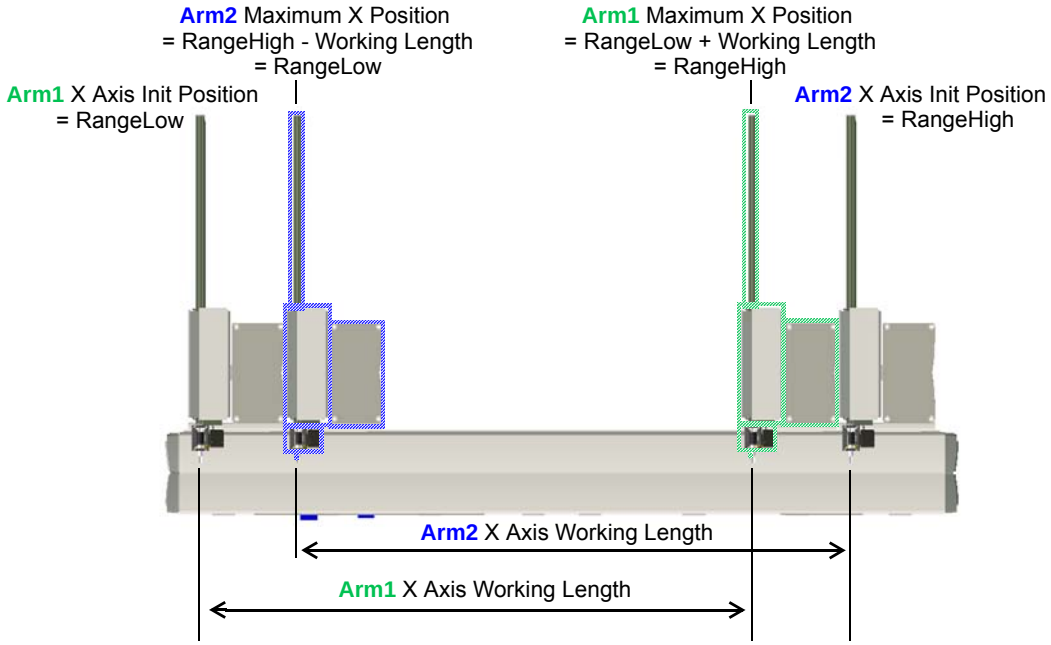


Figure 6-5 *Dual arm Omni Robot X axis positions and parameters*

In this configuration, one arm homes to the left, and the other homes to the right. Each X axis has its own settings for *AxisOffset*, *RangeLow* and *RangeHigh*. Note that for the X axis on Arm1 (the left-homed arm) *AxisOffset* = *RangeLow*, while for the X axis on Arm2 (the right-homed arm) *AxisOffset* = RangeHigh.

In a dual arm system, both X axes can have their coordinate systems re-mapped to a user-specified reference point. For example, you can map the X axis on Arm2 to the same reference point that is shown in Figure 6-4 for the X axis on Arm1, to enable both arms to address the same reference position using the same coordinate address.

When you are calibrating and working with a Dual Z, the rear-most Z must be used to establish a baseline to verify the true offset of the front-most Z (nominally 9 mm or 18 mm in the Y direction).

In this example:

- Assume the total length of the X axis is 1000 mm, so that the position of Arm2 after initialization is X=1000.
- Assume the Working Length for Arm2 X axis has been set to 800 mm.

The user can manually move Arm2 X axis to position 215, and set the reference X-coordinate to 100 with the Set Pos command.

When that is done, the system calculates the *AxisOffset*, *RangeLow*, and *RangeHigh* as follows:

- Arm2 X axis *AxisOffset* = distance from the reference position to Arm2 X axis home + value of the reference position new coordinate.

  The physical distance from the reference position to Arm2 X axis home is 1000-215, or 785 mm, so the *AxisOffset* = 785 + 100 = 885.
- *RangeHigh* = Arm2 X axis *AxisOffset* = 885.
- *RangeLow* = Arm2 X axis *AxisOffset* - Arm2 Working Length = 885-800 = 85.

Now, both Arm1 and Arm2 use the same reference position (physical position 215 mm) and refer to it as X=100.

WARNING! *The possibility of collision exists in a multiple arm system if one arm malfunctions and stops in the middle of a run. It is recommended that the OEM application software be designed such that if an asynchronous Device Error event occurs, the application will immediately terminate motion on all axes. See Section 6.6, "Status Codes, Error Codes, and Events" on page 6-24, for more information about asynchronous events.*

6.2.2 Changing Configuration Settings Programmatically

It is recommended that any specific configuration settings required by an application be made programmatically by the application itself, rather than relying on specific power-on default settings. The various Set commands described in Section 6.8, "Axis Parameter Commands" can be used for this purpose.

Furthermore, if the application requires that the robot be calibrated to a specific coordinate system, it is strongly recommended that the system calibration be handled under program control, to ensure that the settings are done correctly, rather than relying on the operator to make the settings manually using the Configurator software.

The following example illustrates the commands that could be used within a program to calibrate the robot before beginning the execution of the application's main functions. See Section 6.9, "Axis Action Commands" for more information about the commands in this example.

```
** Initialize the robot **
1/1Z/2Z/3Z

** Disable the axes so they can be moved manually **
1/1N0/2N0/3N0

** Communicate with the operator to move the arms to the desired
physical reference position, and indicate when they are in place. **

** Then, set the Omni position coordinates approriately for the
application (shown as <nn> below). **
1/1u0,<nn>/2u0,<nn>/3u0,<nn>

** Re-enable the axes **
1/1N1/2N1/3N1
```

If this is a dual arm system, repeat this procedure for the second arm.

6.3 Overview of the Embedded Command Set

The Omni Embedded Command Set (OECS) allows a custom application to communicate with the Omni Robot. The devices attached to the CIB, including axes and pumps, can be queried, controlled, or operated with OE command strings.

In addition, the OECS allows applications to communicate with Cavro® CAN device options which attach to the Omni robot using Cavro® OEM data protocol.

The flow of communication mostly falls into one of the scenarios shown in Figure 6-6, Figure 6-7, or Figure 6-8 depending on the command string contents.

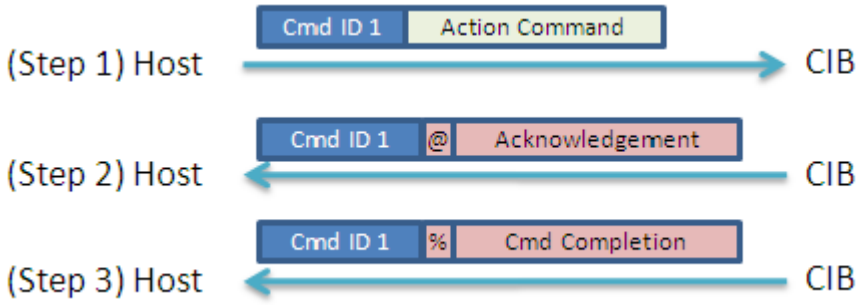


Figure 6-6 *Handshake when sending action command strings*

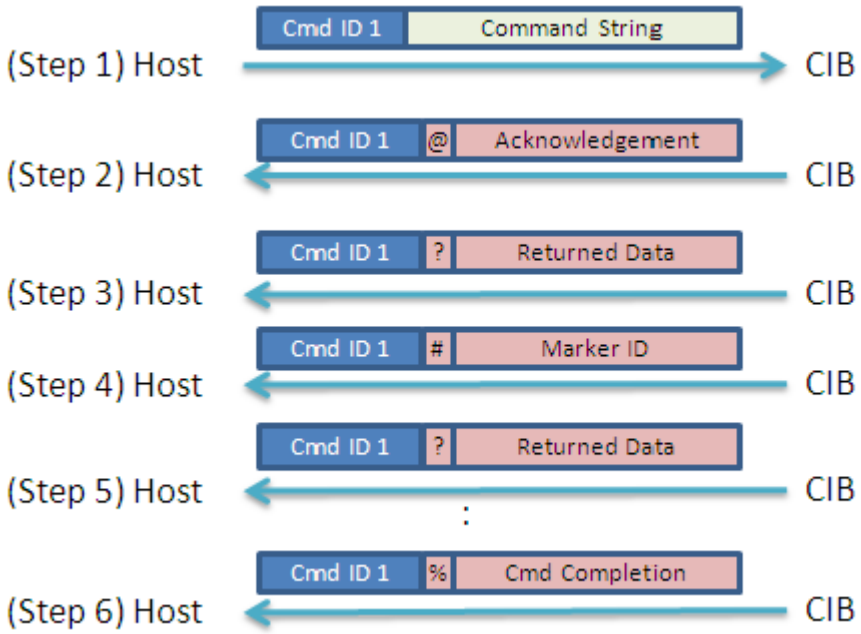


Figure 6-7 *Handshake when sending mixed response/action command strings*

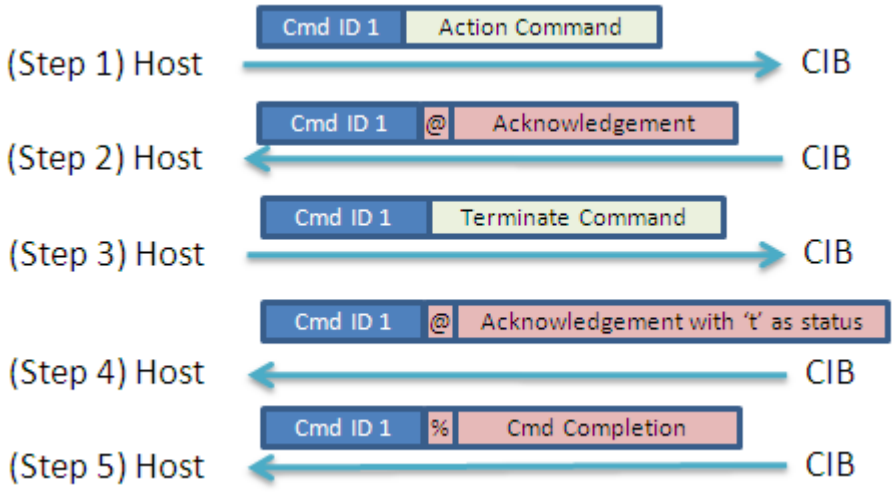


Figure 6-8 *Handshake when sending the Termination command while executing*

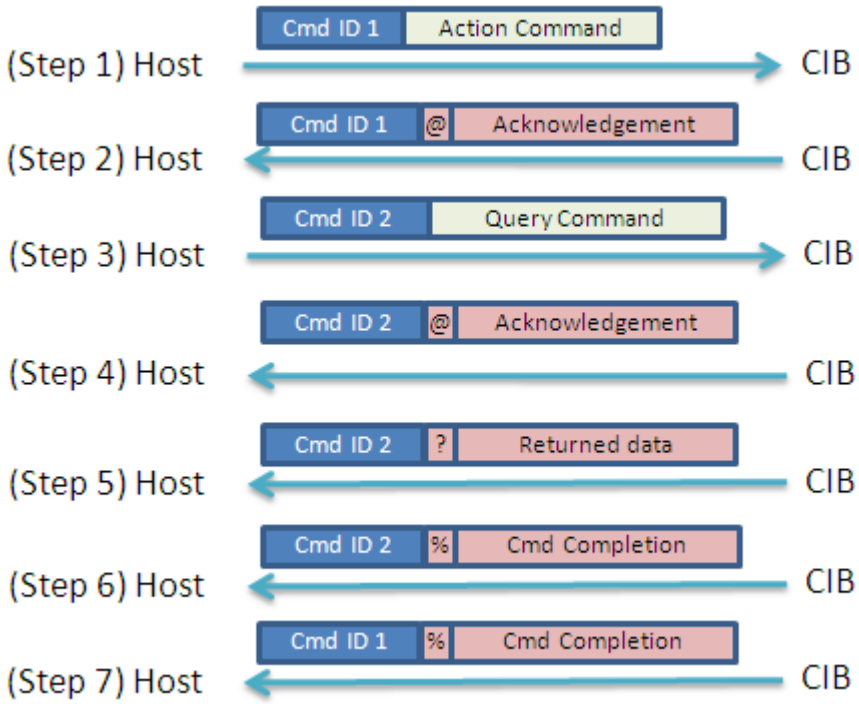


Figure 6-9 *Handshake when sending multiple command strings*

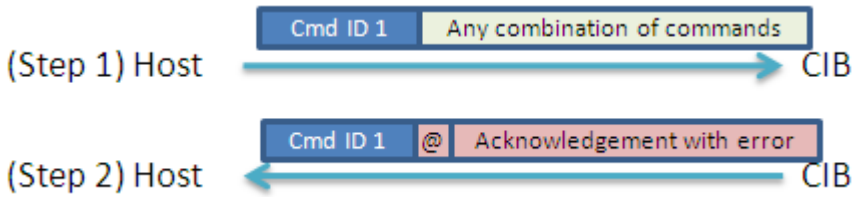


Figure 6-10 *Handshake involving immediate errors*

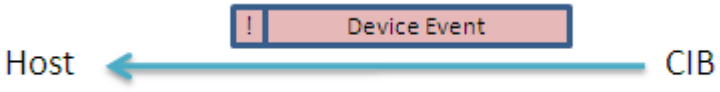
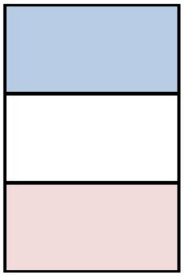


Figure 6-11 *Handshake when a device event is generated*

6.4 Command String Format

Command strings are always sent from the host to a device, or to a group of networked devices.

The following color scheme is used in the illustrations in this section.



Header and ending blocks

Required fields

Optional Fields

Each command string is capable of holding 256 bytes of data and consists of:

1. STX Character at the beginning of each command string
2. Omni ID
3. Sequence Number
4. Command String ID
5. Command Block to be sent to the device
6. ETX Character
7. Checksum at the end of each command string

Figure 6-12 illustrates the structure of a command string.

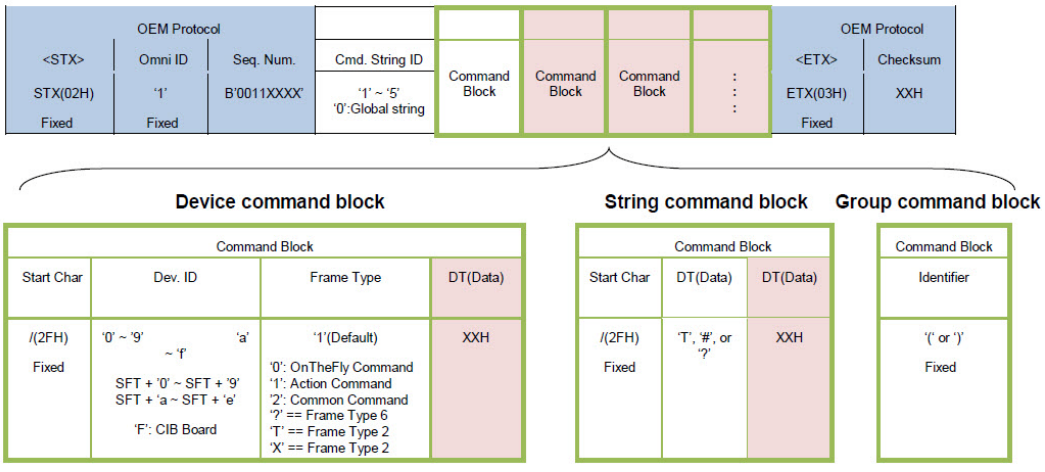


Figure 6-12 *Command string structure*

STX Character

The STX character indicates the beginning of the command string. All characters before the STX character are ignored. The STX character is always 0x02.

Omni ID

The value of the device address is always 0x31.

Sequence Number

The sequence number is a single byte that conveys both a sequence number (legal values are 0x31 through 0x37) and a bit-flag indicating that the command block is being repeated due to a communications breakdown. The sequence number is used as an identity stamp for the command block. Because it is only necessary that every command block carry a different sequence number from the previous command block, the sequence number may be toggled between two different values. During normal communication exchanges, the sequence number is ignored. If, however, the repeat flag is set, the system compares the sequence number with that of the previously received command block to determine if the command should be executed or merely acknowledged without executing.

The sequence number is constructed as follows:

Table 6-1 *Sequence number bits*

Bit #	7	6	5	4	3	2	1	0
Value	0	0	1	1	**REP** 1: repeat 0: no repeat	**SQ2** 1: 0 2: 0 3: 0 4: 1 5: 1 6: 1 7: 1	**SQ1** 1: 0 2: 1 3: 1 4: 0 5: 0 6: 1 7: 1	**SQ0** 1: 1 2: 0 3: 1 4: 0 5: 1 6: 0 7: 1

If the host chooses not to use sequence numbers, the sequence number should be set to a fixed value of 0x31.

Command String ID

A total of 6 command string IDs (0-5) are available; 1-5 for normal commands and 0 for global commands. If the global command string ID is used, then the string command block is used to address to all devices attached to the system. Currently the only cases where global string IDs can be used are the Global Termination and Global System Query commands.

Command strings with different command string IDs can be executed simultaneously. Only when terminating an in-process command string can the IDs

be the same. See Section 6.14.2, ”Terminate Command String” for more information.

Command Block

The command block is a string of displayable ASCII characters (0x20 <sp> through 0x7E <~>) that constitute legal OECS commands. Refer to Command Blocks for details.

ETX Character

The ETX character indicates the end of the command string. The ETX character indicates that the next character will be the checksum. The ETX character is always 0x03.

Checksum

The checksum is the last byte of the command block. All characters after the checksum are ignored until the STX character of the following command string is received. Calculate the checksum by XORing all bytes in the block, starting with the STX character and excluding the checksum byte, to form an 8-bit checksum.

Example:

First initialize both Z axes and the XCalibur pump simultaneously, then simultaneously initialize the X and Y axes. Immediately after the Z axes are initialized, the X and Y axes begin initializing, whether or not the XCalibur pump has finished initializing.

Command string ID and command blocks:

```
1(((/3Z/7Z)/1Z/2Z)(/0ZR))
```

Command string ID and command blocks(hex):

```
312828282F335A2F375A292F315A2F325A29282F305A522929
```

Actual command string format (hex) to send on RS-232 or Ethernet including headers and checksum:

```
023131312828282F335A2F375A292F315A2F325A29282F305A5229290320
```

Note: *This example shows the hex values of ASCII characters which are to be sent out (30 characters in total).*

6.4.1 Command Blocks

For Cavro® devices, the CIB firmware packages the command string in the command block into Cavro® CAN packets either using default, pre-defined characters (**?, T**, etc.), or specified frame types. CIB will automatically break down or assemble multiframe messages accordingly depending on the sent and received message sizes. For ACBs, because the TML library is used, commands

are broken down into 29-bit TML CAN packets and vice versa for responses automatically. Refer to the corresponding FW command set for Cavro® devices

This section describes how to construct a basic command block.

Formatting Conventions

Table 6-2 describes the formatting conventions used to describe command block syntax.

Note: *Commands are case-sensitive.*

Table 6-2 Formatting conventions

Convention	Description
bold font	Commands and keywords.
<*italic*> font with angle brackets	Arguments that accept values.
italic font (without angle brackets)	Parameters that affect a command but are not arguments to the command.
[ ]	Keywords or arguments that appear within square brackets are optional.
{x | y | z}	A choice of required keywords appears in braces separated by vertical bars. Select one.
`courier` font	Examples of information displayed on the screen.
`bold courier` font	Examples of information entered by the programmer or user

6.4.2 Command Block Syntax

Start Character

Each command in the command block begins with the ***/*** character.

Device ID Field

This field specifies the device that will receive the command block. The device IDs are:

- **0 - 9 and a - f:** These IDs reflect the same physical hardware IDs set on the devices (ID dials, ID DIP switches, ID jumpers, etc.) and are applicable to both axes and Cavro® CAN devices.
- **SHIFT + 0 - SHIFT + 9:** These IDs correspond to the characters of **), !, @, #, $, %, ^, &, *, and (**. These IDs are only applicable to Cavro® CAN devices of

group type 2 (diluters) with physical IDs set to **0 - 9** accordingly if IDs overlap with Omni axis IDs.

- **SHIFT + a - SHIFT + e:** These IDs are only applicable to Cavro® CAN devices of group type 2 (diluters) with physical IDs set to **a - e** accordingly if IDs overlap with Omni axis IDs.

Note: *Cavro® CAN devices can have IDs that overlap with Omni axes; however, Cavro® CAN devices cannot have IDs that overlap with any other Cavro® CAN device, and Omni axes cannot have IDs that overlap with any other Omni axis.*

- **F:** This ID addresses commands to the CIB board.

Note: *As shown in the system architecture, if there are less than or equal to 16 connected devices, each device can have an unique Device ID and is detected automatically upon startup. These devices can be controlled by using the Device ID ranges of 0 - 9 and a-f. When the number of devices exceeds this limit, IDs will overlap and the axes will have higher priority over diluters. For overlapped IDs, the axes have higher priority and therefore can use the 0 - 9 and a-f addresses.*

If both the diluter ID dial and ACB ID jumpers are set to 1, then the CIB firmware will automatically assign **1** as the ID for the ACB and **SHIFT + 1** as the ID for the diluter.

Frame Types - CAN Devices Only

This data field is applicable to Cavro® CAN devices only.

The frame types data field tells each Cavro® CAN device what type of command is coming in. For more information on the CAN frame type, refer to the CAN communication section of the manual for your diluter.

When this data field is omitted, the CIB firmware packages the command string in the command block into Cavro® CAN packets using default frame type: frame type 1.

When these following character(s) are used in place of the frame type, it is equivalent to a specific CAN frame type. Refer to the command set section of the manual for your diluter for more information:

- ? Frame type 6
- X Frame type 2, option 3
- T Frame type 2, option 4

The CIB automatically breaks down or assembles multiframe messages accordingly depending on the sent and received message sizes.

For ACBs, commands are automatically broken down into 29-bit TML CAN packets, and 29-bit TML CAN packets received from ACBs are translated into corresponding command responses.

6.4.3 Command Block Example

Note: *This Cavro® CAN Diluter command example illustrates the usage of the command block. Some of these commands may not be usable for some specific Cavro® CAN devices.*

Note: *All commands in this example are sent to Cavro® CAN diluter ID 1.*

1 Initialize command:

```
/1ZR              frame type 1 omitted
/11ZR             with frame type 1
```

2 Move Absolute command:

```
/1A300R           frame type 1 omitted
/11A300R          with frame type 1
```

3 Initialize, turn valve to Input, move to absolute 300, turn valve to Output, move to absolute 0:

```
/1ZIA300OA0R      frame type 1 omitted
/11ZIA300OA0R     with frame type 1
```

4 Report FW Revision:

```
/1?23             Frame type 6 substituted with ?
/1623             with frame type 6
```

5 Terminate Command:

```
/1T               Frame type omitted
/124              Frame type 2, option 4
```

6 Repeat Last Command:

```
/1X               Frame type omitted
/123              Frame type 2, option 3
```

7 Change Speed on-fly command:

```
/10V900          Frame type 0, changing speed on-fly
```

6.5 Response String Formats

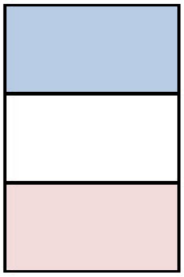
Response strings are always sent from an individual device to the host, in response to a command string. A device may only send a response string in response to a command string addressed to that specific device. Devices can only report events to the host, and can otherwise never initiate communications with the host. Each received data response string consists of:

1 STX Character at the beginning of each answer block

2 Host ID

3 Command String Status Byte
4 Response Data String
5 ETX Character
6 Checksum at the end of each answer block

The following color scheme is used in the illustrations in this section.



Header and ending blocks

Required fields

Optional Fields

STX Character

The STX character indicates the beginning of the response string. All characters before the STX character are ignored. The STX character is always 0x02.

Host ID

The host address is always set to 0x30.

Command String Status Byte

This byte is for status and error information that relates to the command string execution status. For more information about status and error codes, see Section 6.6.5, ”Command Response String Status Codes”.

Response Data String

The response data string is a string of displayable ASCII characters (0x20 <sp> through 0x7E <~>). The content and allowable length of the block depends on the specific type of device.

ETX Character

The ETX character indicates the end of the command string. The ETX character indicates that the next character will be the checksum. The ETX character is always 0x03.

Checksum

The checksum is the last byte of the response. The checksum is calculated by XORing all bytes in the block, starting with the STX character and excluding the checksum byte, to form an 8-bit checksum.

6.5.1 Response Types

There are several types of responses that the CIB makes:

- System Acknowledgement Responses
- Command String Completion Responses
- Query Responses
- Marker Responses
- Event Responses

All response strings start with STX and end with ETX and a checksum as described in Response String Formats. This section describes only the String Status, and Response Data String portions of the response format.

6.5.2 System Acknowledgement Responses

An acknowledgement string is returned whenever a properly formatted command string is received. The system acknowledgement string format is shown in Figure 6-13.

OEM Protocol			Response Data String		OEM Protocol	
<STX>	Host ID	String Status	Cmd. String ID	Identifier	<ETX>	Checksum
STX(02H) Fixed	'0' Fixed	B'01X0XXXX' or B'01110100' 't' if it's acknowledgement for a 'T' command	'1' ~ '5' '0':Global string	'@' Fixed	ETX(03H) Fixed	XXH

- String Status: This specifies the status of the command string. See Table 6-8 for more information about status codes.
- Command String ID: The number that corresponds to the Command ID that was provided when the command was issued. See Command String ID for more information.
- @: All acknowledgement responses use the @ in the Identifier field. If you see an @ in a response, it is always an acknowledgement response.

Figure 6-13 *System acknowledgement string format*

Note: *If the "repeat bit" is set in the sequence number field of the incoming command and the sequence number equates to the previously executed command, an acknowledgement packet will be returned with "String Status" set to "0 - No Error, command string finished." The contents of this OEM protocol packet were completely ignored and there will be no final response, only the acknowledgment. This is also the only case where an acknowledgement string is returned with the "0 - No Error, command string finished" error code.*

6.5.3 Command String Completion Responses

OEM Protocol			Response Data String							OEM Protocol	
<STX>	Host ID	String Status	Cmd. String ID	Identifier	Dev. ID	Device Status	Device Status	Device Status	Device Status	<ETX>	Checksum
STX(02H) Fixed	'0' Fixed	B'01X0XXXX' or B'01101011' (string completed with Device Error)	'1' ~ '5' '0':Global string	'%' Fixed	'0' ~ '9' 'a' ~ 'f' SFT + '0' ~ SFT + '9' SFT + 'a ~ SFT + 'e' 'F': CIB Board	HIGH NIBBLE of 1st Status byte XXH	LOW NIBBLE of 1st Status byte XXH	HIGH NIBBLE of 2nd Status byte XXH	LOW NIBBLE of 2nd Status byte XXH	ETX(03H) Fixed	XXH

- String Status: This specifies the status of the command string. See Table 6-8 for more information about status codes.
- Command String ID: The number that corresponds to the Device ID that was provided when the command was issued. See Command String ID for more information.
- %: Indicates that the command string completed.
- Device ID: The device that generated the status. See Device ID Field for more information.
- Device Status: A 4-byte Hex value ASCII string that reports the CAN device status, and any ACB or CIB error codes. See Status Codes, Error Codes, and Events for more information about these error codes.

Figure 6-14 *Command string completion response format*

Note: *The command string ends as soon as any device encounters an error.*

6.5.4 Query Responses

There are two types of response formats used for query responses. Global system block query responses follow the format shown in Figure 6-15, and device command block query responses follow the format shown in Figure 6-16. Note that in Figure 6-16 the OEM Protocol fields are not shown, but they are the same as shown in Figure 6-15.

OEM Protocol			Response Data String								OEM Protocol	
<STX>	Host ID	String Status	Cmd. String ID	Identifier	Identifier	Error Code	Error Code	Error Code	Error Code	DT(Data)	<ETX>	Checksum
STX(02H) Fixed	'0' Fixed	B'01X0XXXX'	'0'	'?' Fixed	'S' Fixed	'0' Fixed	'0' Fixed	'0' Fixed	'0' Fixed	XXH	ETX(03H) Fixed	XXH

- String Status: This specifies the status of the command string. See Table 6-8 for more information about status codes.
- Command String ID: The number that corresponds to the Command String ID that was provided when the command was issued. See Command String ID for more information.
- ?S: The identifier fields are fixed to ? and S. If present, this indicates that the response is the result of a system query command.
- 0x0000: String-level error codes are fixed to 0 as place holders.
- DT (Data): This field is optional depending on the string status and type of query command being issued. It is a string of ASCII characters and the allowable length depends on the type of device.

Figure 6-15 *Global system query response format*

OEM Protocol			Response Data String								OEM Protocol	
<STX>	Host ID	String Status	Cmd. String ID	Identifier	Dev. ID	Device Status	Device Status	Device Status	Device Status	DT (Data)	<ETX>	Checksum
STX(02H) Fixed	'0' Fixed	B'01X0XXXX'	'1' ~ '5'	'?' Fixed	'0' ~ '9' 'a' ~ 'f' SFT + '0' ~ SFT + '9' SFT + 'a ~ SFT + 'e' 'F': CIB Board	HIGH NIBBLE of 1st Status byte XXH	LOW NIBBLE of 1st Status byte XXH	HIGH NIBBLE of 2nd Status byte XXH	LOW NIBBLE of 2nd[1] Status byte XXH	XXH	ETX(03H) Fixed	XXH

- String Status: This specifies the status of the command string. See Table 6-8 for more information about status codes.
- Command String ID: The number that corresponds to the Command String ID that was provided when the command was issued. See Command String ID for more information.
- ?: The identifier fields are fixed to ?. If present, this indicates that the response is the result of a system query command.
- Device ID: The device that generated the response. See Device ID Field for more information.
- Device Status: A 4-byte Hex value ASCII string that represents the ASCII presentation of 2 data bytes in Hex. For example:
  - Axis invalid operand 30 30 30 33:
    - Hex value 0003 is invalid operand
  - Cavro® CAN device 32 31 36 30:
    - 0x21 0x60: In Cavro® CAN devices, the second byte is not used and varies by device. In this scenario, the error is 1, the initialization error.
- DT (Data): This field is optional depending on the string status and type of query command being issued. It is a string of ASCII characters and the allowable length depends on the type of device.

See Status Codes, Error Codes, and Events for more information about these error codes.

Figure 6-16 *Device command block query response format*

6.5.5 Marker Responses

Comment: Marker ID range is limited to 0~5 because markers can be reused in a command string. The sequence of returned markers can also be used as an identifier for command execution progress.

OEM Protocol			Response Data String			OEM Protocol	
<STX>	Host ID	String Status	Cmd. String ID	Identifier	Marker Label	<ETX>	Checksum
STX(02H) Fixed	'0' Fixed	B'01X0XXXX'	'1' ~ '5'	'#' Fixed	'0' ~ '5'	ETX(03H) Fixed	XXH

- String Status: This specifies the status of the command string. See Table 6-8 for more information about status codes.
- Command String ID: The number that corresponds to the Device ID that was provided when the command was issued. See Command String ID for more information.
- #: Indicates that this is a command marker response.
- Marker Label: The marker label specified in the command string.

Figure 6-17 *Command progress marker response format*

Note: *Marker command string ID numbers can be used more than once in a command string. The sequence of returned markers can be used to track the progress of a command string as it executes. For example:*

Command:

```
1/1A100/#0/1A200/#1R
```

Response:

```
1@
** Device Moves to position 100 **
```

Response:

```
1#0
** Device Moves to position 200 **
```

Response:

```
1#1
```

Response:

```
1%
```

6.5.6 Event Responses

OEM Protocol			Response Data String			OEM Protocol	
<STX>	Host ID	String Status	Identifier	Dev. ID	DT(Data)	<ETX>	Checksum
STX(02H) Fixed	'0' Fixed	B'01100101' 'e' Fixed	'!' Fixed	'0' ~ '9' 'a' ~ 'f' SFT + '0' ~ SFT + '9' SFT + 'a ~ SFT + 'e' 'F': CIB Board	XXH	ETX(03H) Fixed	XXH

- e!: The string status for an event is always e and the identifier for an event is always !
- Device ID: the ID of the device that generated the event.
- DT (Data): This field varies depending on the type of event generated. It is a string of ASCII characters and the allowable length depends on the type of device. See Table 6-5 and Table 6-7 for more information.

Figure 6-18 *Event format*

6.6 Status Codes, Error Codes, and Events

All status codes, error codes, and event codes are sent from the CIB to the host.

6.6.1 ACB and Gripper Error Codes

Table 6-3 lists the ACB supported error codes.

Table 6-3 *ACB and Gripper Error Codes*

Value	Description
0x0000	No error with command execution
0x0002	Command not recognized for the specified axis type.
0x0003	Command operand not recognized for the specified command.
0x0004	Device response timed out. Device requires clearing the error and enabling the axis to resume operation
0x0005	Device not implemented (device found but TML script version is not supported)
0x0007	Device initialization command has not been issued.
0x0008	Device is busy and unable to accept commands.

Table 6-3 *ACB and Gripper Error Codes (continued)*

Value	Description
0x0009	Device error occurred or a persistent error has not been cleared. Detailed information can be obtained by querying LastDeviceErrorInfo. Possible returned error types are listed in Section Table 6-4, ”ACB device error types”.
0x000B	Delta in hard stop position in double initialization for Z axis types exceeded LLDAcceptanceWindow variable value.
0x000D	Device is disabled and unable to accept action commands.
0x000E	Invalid travel position.
0x000F	Liquid surface is not found during detect liquid operation.
0x0010	Position of liquid surface detected is not enough to accommodate the expected submerge distance.
0x0011	Liquid surface is not found during detect liquid operation.
0x0012	Liquid exit signal found after ExitLimit has reached.
0x0013	Liquid exit signal not found during check for exit liquid signal operation.
0x0014	cLLD is not ready to detect liquid. Most likely tip is submerged in liquid.
0x0015	Setting parameter to ACB failed. Value set does not equate to value queried.
0x0016	DiTi tip not dropped during a drop tip operation.
0x0017	Hard stop occurred before the DropLimit function parameter has been reached.
0x0018	Hard stop not found after reaching DropLimit function parameter.
0x0019	Tip picked within PickLimit function parameter but not mounted after moving back to PickStartPos starting position.
0x001A	Hard stop found prior to reaching PickAcceptanceLimit function parameter.
0x001B	DiTi tip is not mounted.
0x001C	DiTi tip is already mounted.
0x001D	DiTi tip is not found during pick tip operation.
0x001E	DiTi tip is picked but specified PickForceFactor is not reached.
0x0029	Gripper is not attached to system.
0x002A	Gripper is not initialized.
0x002B	Problem occurred when moving gripper.
0x002C	Only reported if gripper is neither fully open nor fully closed and GripperInitMode mode is either in Normal Open or Normal close.
0x002D	Reported if an object is detected during gripper initialize operation.

Table 6-3 ACB and Gripper Error Codes (continued)

Value	Description
0x002E	Reported if no object is detected during gripper close operation.
0x0030	Errors with either one of the 3 gripper photo interrupters.
0x003C	Error occurred with brake self-test operation.
0x0065	Axis is blocked during pre-initialization relative move operation.
0x0066	Pre-initialization relative move operation is only allowed before axis initialization move has been performed.
0x0067	Error either with encoder or limit switch.
0x0068	Response related to command is not expected.

Note: *To unify device busy status among all device types, all Cavro® CAN devices report the same device busy error code (0x0008) as ACBs.*

Table 6-4 ACB device error types

Bit	Description	Value
0	CAN bus error	0x0001
1	Short circuit	0x0002
2	Invalid setup data	0x0004
3	Control error	0x0008
4	Serial communication error	0x0010
5	Position wraparound	0x0020
6	LSP (limit +) active	0x0040
7	LSP (limit -) active	0x0080
8	Over current	0x0100
9	I^2t	0x0200
10	Over temperature - Motor	0x0400
11	Over temperature - Drive	0x0800
12	Over voltage	0x1000
13	Under voltage	0x2000
14	Command error	0x4000
15	Enable input is active	0x8000

Note: *This table specifies detailed information for ACB and gripper error code 9.*

Example 1

Invalid parameter used in A command to move ACB.

Command: **`1/1A10000`** (10000 is out of range)

Response: **`1@`** (Command string 1 acknowledgement)

Response: **`1%10003`** (Command string 1 ended with device 1 returning error code 0x0003)

Example 2

Device errors generated while executing simultaneous motion

Command: **`1(/1A100/2A100)`** (Devices are blocked prior to motion)

Response: **`1@`** (Command string 1 acknowledgement)

Response: **`1%10009 20009`** (Command string 1 ended with device 1 and 2 returning error code 0x0009)

6.6.2 ACB and Gripper Event Codes

Table 6-5 lists the ACB supported event codes.

Refer to the corresponding firmware command set for Cavro® devices for ACB and Gripper event codes.

Table 6-5 *ACB and Gripper Event Codes*

Value	Description
0x0065	LastDeviceErrorInfo is automatically updated when this event is generated. Refer to ‘Device Error Info’ section for decoding the stored value.
0x006A	Diti tip is dropped unexpectedly.
0x0071	Generated when gripper changes state from “Object Detected” to “Gripper Closed” unexpectedly.
0x0072	Gripper finger state changes unexpectedly other than scenarios where the object dropped event is generated.

Example 1

ACB tip loss event received on axis 3

Event: **`!3006A`**

6.6.3 CIB (Device F) Error Codes

Table 6-6 specifies the returned status and error codes in the Device Status fields for device 'F' specified in “Device Command Block Query Response” and “Command String Completion Response” formats. ALL Cavro® CAN diluters use only the 1st status byte leaving the 2nd unused. To be consistent with the 2 status bytes, the error codes for CIB device 'F' returns 2 status bytes as well with the exception of utilizing both bytes.

Table 6-6 CIB error codes

Value	Description
0x0000	No Error
0x0001	Reserved
0x0002	Invalid Command
0x0003	Invalid Operand

6.6.4 CIB (Device F) Event Codes

Table 6-7 shows the 4-byte ASCII representation of 2-byte hex value event codes returned in the DT (Data) field of the “Event” response format for device 'F'.

Table 6-7 *CIB event codes*

Value	Description
0xFFFC	CAN0 Receive Packet Dropped
0xFFF9	#INT5 Triggered
0xFFF8	CIB Voltage Unstable
0xFFF7	CAN0 Bus Error
0xFFEE	NVMEM Checksum Error

Example 1

Invalid parameter used in U command to set configuration of CIB device.

Command: `1/FU55` (55 is out of range)

Response: `1@` (Command string 1 acknowledgement)

Response: `1%F0003` (Command string 1 ended with device F returning error code 0x0003)

Example 2

CIB voltage unstable event received

Event: `!FFFF8`

WARNING! *The possibility of collision exists in a multiple axis system if one axis malfunctions and stops in the middle of a run. It is recommended that the OEM application software be designed such that if an asynchronous Device Error event occurs, the application will immediately terminate motion on all axes.*

Note: *The ADP U3R command must be issued to enable tip loss events. Refer to the ADP Manual for detailed information.*

6.6.5 Command Response String Status Codes

Table 6-8 returned in the String Status Field of all string command response formats. Code numbers used in acknowledgement responses are indicated in the ACK? column.

Table 6-8 *Response string status codes*

bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0	Error Code Number
0	1	1	0	0	0	0	0	No error, command string finished
0	1	0	0	0	0	0	0	No error, command string busy
0	1	0	0	0	0	0	1	Command string currently in use
0	1	1	0	0	0	1	1	Invalid command string operand
0	1	1	0	0	1	0	0	Invalid placement or pairing of group command blocks
0	1	1	0	0	1	0	1	e - Event
0	1	1	0	0	1	1	0	System device scanning in progress
0	1	1	0	0	1	1	1	Invalid system operation (such as T used in combination with other commands)
0	1	1	0	1	0	1	1	Device error. The device which generated the error and its corresponding error code are specified in the Command String Completion Responses.
0	1	1	0	1	1	0	0	Invalid Device ID
0	1	1	1	0	1	0	0	t - indicates 'T' command ACK

6.7 Levels of the OE Command Set

There are three levels of OECS commands:

- OE System-Level Control Commands. For more information, see:
  - Global Termination
  - Global System Query
  - Terminate Command String
  - Halt
  - Command Progress Markers
  - Controlling Command Execution by Grouping Commands
- CIB commands (device F). For more information, see CIB Commands
- Single axis control commands (devices 1-15 or 1-f Hex). For information about single axis control commands, see:
  - Axis Parameter Commands
  - Axis Action Commands
  - Gripper Commands

For a quick summary of all commands, see Appendix F, “Embedded Command and Error Quick Reference Guide”.

6.8 Axis Parameter Commands

This section describes the commands that get and set values of the parameters stored used by axis commands.

The **?<*n*>** command retrieves the value of parameter <*n*>. The reply to the **?** command includes the command string ID, the command target’s device ID, the error code, and the requested parameter value. For example, if the reply to the command 1/1?10 is ?10000500 and 1%, that means that axis 1 returned error code 0x0000 and has a default axis movement speed of 500 mm/sec and successful execution of command string 1. For more information about the command response format, see Response String Formats.

The **u<*n*>** command edits the value of a parameter. The completion of the command string indicates execution success. For example, the reply to the command /1u10,800 is 1%, indicating successful execution of command string 1. For more information about the command response format, see Response String Formats.

6.8.1 Possible Accuracy Loss

Axis commands accept floating point values for distance, position, speed, and acceleration related properties. These are converted to integer internal device units and downloaded to the device when needed for execution of a motion

command. For that reason, the user may expect to lose some accuracy due to conversion truncation.

The units used throughout the system are:

- mm - distance and position
- mm/s - speed
- mm/s^2 - acceleration
- Internal Unit - force

6.8.2 Get Commands

Synopsis: The ?<n> command gets the value of the ***Var_ID***.

Syntax: ***/<AxisID>?<Var_ID>***

AxisID: Valid values are 1-f.

Var_ID: The following table provides a list of Var_IDs and the parameters they correspond to:

Table 6-9 *Axis Get command arguments*

Var ID	Name	Page Num
0	Pos	6-55
1	AxisOffset	6-34
2	AxisOrientation	6-34
3	InitMode	6-48
4	InitAcceptanceWindow	6-46
5	InitForceFactor	6-46
6	InitForceRatio	6-46
7	DefInitAcceleration	6-37
8	DefInitSpeed	6-37
9	DefAcceleration	6-36
10	DefSpeed	6-37
11	Status (DeviceState)	6-57
12	DropSpeed	6-40
13	DropForceFactor	6-40

Var ID	Name	Page Num
28	LLDAcceptanceWindow	6-48
29	LLDSearchSpeed	6-50
30	LLDSelfTestSensitivity	6-51
31	LLDSelfTestLevel	6-50
32	LLDSelfTestLastResult	6-50
33	MoveAbsHiForceSpeed	6-52
34	PWMFrequency	6-56
35	PWMDutyCycle	6-55
36	TipPresence	6-57
37	PickSpeed	6-54
38	PickForceFactor	6-53
39	RangeHigh	6-56
40	RangeLow	6-57
41	WorkingLength	6-58

Table 6-9 *Axis Get command arguments (continued)*

Var ID	Name	Page Num
14	EncoderDirection	6-41
15	ExitRetractSpeed	6-42
16	ExitRetractDist	6-42
17	ExitLimit	6-42
18	FirmwareID	6-43
19	GripperInitMode	6-45
20	GripperState	6-45
21	LLDSubmergeDist	6-52
22	LLDSensitivity	6-51
23	AxisDesc	6-34
24	LastDeviceErrorInfo	6-48
25	LLDDetectionMethod	6-49
26	LLDRetractOnError	6-49
27	LLDRetractDist	6-49

Var ID	Name	Page Num
71	Number of ACB move counts since last reset	6-33
72	Total number of recorded ACB move counts	6-33
73	Highest recorded ACB temperature in Celsius since last reset	6-33
74	Highest ever recorded ACB temperature in Celsius	6-32
76	General ACB Configuration	6-43
77	Initialization Configuration	6-47
78	Get Move Configuration	6-52
79	Detect Configuration	6-38
80	Exit Configuration	6-35
81	cLLD Self Test Configuration	6-35
82	Get Pick Tip Configuration	6-54
83	Drop Configuration	6-39
84	Gripper Configuration	6-44

The following errors can be generated by the **?<*n*>** command:

Generated Errors:
3 - Invalid Operand
5 - Device not implemented
8 - Device busy

Get ACB Highest Temperature Ever Recorded

Synopsis: The highest ACB temperature ever recorded in Celsius

Values: Any valid ACB temperature in Celsius

Syntax: **`/<Axis_ID>?74`**

Description: Gets the highest ACB temperature ever recorded in Celsius. This is a read-only parameter and cannot be set.

Example: Get the ACB highest temperature ever recorded:

```
/1?74
```

Get ACB Highest Temperature Recorded Since Last Reset

Synopsis: The highest ACB temperature recorded since the last temperature reset

Values: Any valid ACB temperature

Syntax: **/<*Axis_ID*>?73**

Description: Gets the highest ACB temperature recorded since the last reset.

To reset the highest recorded ACB temperature, see Set Highest Recorded ACB Temperature.

Example: Get the ACB highest temperature:

```
/1?73
```

Get ACB Move Count Since Last Reset

Synopsis: The number of ACB moves since the last move count reset

Values: 0 to 4294967295 (0xFFFFFFFF)

Syntax: **/<*Axis_ID*>?71**

Description: Gets the number of ACB moves recorded since the last reset.

To reset the number of ACB moves, see Set ACB Move Count.

Example: Get the ACB move count:

```
/1?71
```

Get ACB Move Count Total

Synopsis: The total recorded number of ACB moves

Values: 0 to 4294967295 (0xFFFFFFFF)

Syntax: **/<*Axis_ID*>?72**

Description: Gets the number of ACB moves recorded for all time. This is a read-only parameter and cannot be set.

Example: Get the total ACB move count:

```
/1?72
```

Get AxisDesc

Synopsis: The axis type, including the TML script version number

Values: Valid axis type

Syntax: **/<*AxisID*>?23**

Description: AxisDesc is a read-only parameter and its value cannot be changed.

Example: Get the type of axis 1:

```
/1?23
```

Get AxisOffset

Synopsis: The coordinates at the axis' home position, in mm

Values: Valid coordinates

Syntax: **/<*AxisID*>?1**

Description: AxisOffset is a read-only parameter and its value cannot be changed.

Example: Get the coordinates of the home position of axis 1:

```
/1?1
```

Get AxisOrientation

Synopsis: The orientation of the axis

Values: 0 or 1

Syntax: **/<*AxisID*>?2**

Description: The orientation of the axis. Either left or right oriented:

- 0: Left oriented
- 1: Right oriented

To set, see the Set AxisOrientation command.

Example: Get the orientation of axis 1:

```
/1?2
```

Get Check For Exit Configuration

Synopsis: ASCII text report of configuration for parameters that affect the Check For Exit command

Values: ASCII text string

Syntax: **`/<AxisID>?80`**

Description: Exit Configuration is a read-only parameter and its value cannot be changed. The Check For Exit command can be configured with Set ExitRetractSpeed, Set DefAcceleration, Set LLDSensitivity, Set ExitLimit and Set ExitRetractDist.

The format of the response to this command is as follows, with each field separated by a '|' character:

`'field1|field2|field3|field4|field5'`

Response format fields and values are shown below.

Field #	ASCII text	Description
1	Speed	Value of ExitRetractSpeed
2	Accel	Value of DefAcceleration
3	Sensitivity	Value of LLDSensitivity
4	Limit	Value of ExitLimit
5	ExitDist	Value of ExitRetractDist

Example: Get the exit configuration:

```
/1?80
```

Get cLLD Self Test Configuration

Synopsis: ASCII text report of configuration for parameters that affect the cLLD Self Test command

Values: ASCII text string

Syntax: **`/<AxisID>?81`**

Description: cLLD Self Test Configuration is a read-only parameter and its value cannot be changed. The cLLD Self Test command can be configured with Set LLDSelfTestSensitivity and Set LLDSelfTestLevel.

The format of the response to this command is as follows, with each field separated by a '|' character:

`'field1|field2'`

Response format fields and values are shown below.

Field #	ASCII text	Description
1	Sensitivity	Value of LLDSelfTestSensitivity
2	TestLevel	Value of LLDSelfTestLevel

Example: Get the cLLD Self Test configuration:

```
/1?81
```

Get DefAcceleration

Synopsis: The acceleration of the axis during any move command, except the Initialize Axis command (see the Get DefInitAcceleration parameter for the Initialize Axis command)

Values: 1 to 100,000 mm/s^2

Syntax: **`/<AxisID>?9`**

Description: Acceleration values are based on calculations. The actual acceleration of the X axis and Y axis could be lower due to the positioning algorithm depending on travel distance and speed settings. The default values are:

- X axis: 3500 mm/s^2
- Y axis: 3900 mm/s^2
- Standard Z axis: 6000 mm/s^2
- Universal Z axis: 1000 mm/s^2
- Dual Z axis: 6000 mm/s^2

To set, see the Set DefAcceleration command.

Example: Get the acceleration of axis 1:

```
/1?9
```

Get DefInitAcceleration

Synopsis: The acceleration of the axis during the Initialize Axis command

Values: 1 to 100,000 mm/s^2

Syntax: **`/<AxisID>?7`**

Description: Acceleration values are based on calculations. The actual acceleration of the X axis and Y axis could be lower due to the positioning algorithm depending on travel distance and speed settings. The default values are:

- X axis: 3500 mm/s^2
- Y axis: 3900 mm/s^2
- Standard Z axis: 6000 mm/s^2
- Universal Z axis: 1000 mm/s^2
- Dual Z axis: 6000 mm/s^2

To set, see the Set DefInitAcceleration command.

Example: Get the acceleration of axis 1:

```
/1?7
```

Get DefInitSpeed

Synopsis: The initialization speed of the axis, which is used when executing the Initialize Axis command

Values: 0.1 to 200 mm/s

Syntax: **`/<AxisID>?8`**

Description: The default values are:

- X axis: 80 mm/s
- Y axis: 80 mm/s
- Z axis: 40 mm/s

To set, see the Set DefInitSpeed command.

Example: Get the initialization speed of axis 1:

```
/1?8
```

Get DefSpeed

Synopsis: The default axis movement speed

Values: 0.1 to 10000 mm/s

Syntax: **`/<AxisID>?10`**

Description: The default values are:

- X axis: 800 mm/s
- Y axis: 600 mm/s
- Standard Z axis: 600 mm/s
- Universal Z axis: 250 mm/s
- Dual Z axis: 600 mm/s

To set, see the Set DefSpeed command.

Example: Get the default movement speed of axis 1:

```
/1?10
```

Get Detect Liquid Configuration

Synopsis: ASCII text report of configuration for parameters that affect the Detect Liquid command

Values: ASCII text string

Syntax: **`/<AxisID>?79`**

Description: Detect Configuration is a read-only parameter and its value cannot be changed. The Detect Liquid command can be configured with Set LLDSearchSpeed, Set DefAcceleration, Set LLDSensitivity, Set LLDSubmergeDist, Set LLDRetractDist, Set LLDAcceptanceWindow, Set PWMFrequency, Set LLDDetectionMethod, and Set LLDRetractOnError.

The format of the response to this command is as follows, with each field separated by a '|' character:

`'field1|field2|field3|field4|field5|field6|field7|field8|field9'`

Response format fields and values are shown below.

Field #	ASCII text	Description
1	Speed	Value of LLDSearchSpeed
2	Accel	Value of DefAcceleration
3	Sensitivity	Value of LLDSensitivity
4	SubDist	Value of LLDSubmergeDist
5	RetractDist	Value of LLDRetractDist
6	Window	Value of LLDAcceptanceWindow
7	PWMFrequency	Value of PWMFrequency
8	Method	Value of LLDDetectionMethod
9	RetractOnError	Value of LLDRetractOnError

Example: Get the detect liquid configuration:

```
/1?79
```

Get Drop Tip Configuration

Synopsis: ASCII text report of configuration for parameters that affect the Drop Tip command

Values: ASCII text string

Syntax: **`/<AxisID>?83`**

Description: Drop Configuration is a read-only parameter and its value cannot be changed. The Drop Tip command can be configured with Set DropSpeed, Set DefAcceleration, and Set DropForceFactor. The AxisOffset parameter cannot be changed.

The format of the response to this command is as follows, with each field separated by a '|' character:

`'field1|field2|field3|field4|field5|field6'`

Response format fields and values are shown below.

Field #	ASCII text	Description
1	Speed	Value of DropSpeed
2	Accel	Value of DefAcceleration
3	ForceFactor	Value of DropForceFactor
4	StartPos	Value of AxisOffset minus 20.0 mm
5	Limit	Value of AxisOffset plus 20.0 mm
6	AcceptanceLimit	Value of AxisOffset

Example: Get the drop configuration:

```
/1?83
```

Get DropForceFactor

Synopsis: The force used to drop the disposable tip

Values: 8000 to 22000

Syntax: **`/<AxisID>?13`**

Description: The default value is 17000. The higher the force factor, the stronger the applied force is.

Note: Because the Drop Tip hardware mechanism is not yet supported for the Universal Z axis, issuing Drop Tip with DropForceFactor while using the Universal Z axis will have no physical drop-tip effect.

To set, see the Set DropForceFactor command.

Example: Get the drop force of axis 1:

```
/1?13
```

Get DropSpeed

Synopsis: The travel speed used by the Standard Z or Dual Z axis to drop a disposable tip

Values: 1 to 150 mm/s

Syntax: **`/<AxisID>?12`**

Description: The default value is 60 mm/s.

Note: Because the Drop Tip hardware mechanism is not yet supported for the Universal Z axis, issuing the Drop Tip command with DropSpeed while using the Universal Z axis will have no physical drop-tip effect.

To set, see the Set DropSpeed command.

Example: Get the drop speed of axis 1:

```
/1?12
```

Get EncoderDirection

Synopsis: Direction of the encoder

Values: 0, 1

Syntax: `/<AxisID>?14`

Description: The direction of the encoder that performs increment and decrement. The legal values are:

- 0: Hardware uses normal encoder direction
- 1: Hardware uses reversed encoder direction

Default is 0. To set, see the Set EncoderDirection command.

Example: Get the encoder direction of axis 1:

```
/1?14
```

Get ExitLimit

Synopsis: The acceptance limit for a successful exit signal check

Values: 0 to 50 mm

Syntax: **`/<AxisID>?17`**

Description: The default value is 5 mm.

To set, see the Set ExitLimit command.

Example: Get the exit limit of axis 1:

```
/1?17
```

Get ExitRetractDist

Synopsis: The travel distance in the Z direction that the Z axis will move when searching for an exit signal with the Check For Exit command

Values: 1 to 100 mm

Syntax: **`/<AxisID>?16`**

Description: The default value is 10.

To set, see the Set ExitRetractDist command.

Example: Get the exit travel distance of axis 1:

```
/1?16
```

Get ExitRetractSpeed

Synopsis: The speed that the Z axis will use when retracting and searching for an exit signal

Values: 1 to 150 mm/s

Syntax: **`/<AxisID>?15`**

Description: The default value is 60.

To set, see the Set ExitRetractSpeed command.

Example: Get the exit travel speed of axis 1:

```
/1?15
```

Get FirmwareID

Synopsis: The revision number of the firmware in the Axis PCB

Values: Valid axis revision number

Syntax: **`/<AxisID>?18`**

Description: FirmwareID is a read-only parameter and its value cannot be changed.

Example: Get the firmware ID numbers for axis 1:

```
/1?18
```

Get General ACB Configuration Information

Synopsis: ASCII text report of ACB configuration

Values: ASCII text string

Syntax: **`/<AxisID>?76`**

Description: ACB Configuration is a read-only parameter and its value cannot be changed. The ACB can be configured with the commands described in the sections Set WorkingLen, Set EncoderDirection, and Set AxisOrientation. The other fields returned by this command cannot be set.

The format of the response to this command is as follows, with each field separated by a '|' character:

```
'field1|field2|field3|field4|field5'
```

Applicability of fields depends on axis type.

Response format fields and values are shown below.

Field #	ASCII text	Description
1	WorkLen	Value of WorkingLength
2	RangeLow	Value of RangeLow
3	RangeHi	Value of RangeHigh
4	Offset	Value of AxisOffset
5	EncDir	Value of EncoderDirection, displayed as either NORMAL or REVERSE
	Orientation	Value of AxisOrientation, displayed as either LEFT or RIGHT

Example: Get general ACB configuration information:

```
/1?76
```

Get Gripper Configuration

Synopsis: ASCII text report of configuration for parameters that affect the Gripper commands

Values: ASCII text string

Syntax: **`/<AxisID>?84`**

Description: Gripper Configuration is a read-only parameter and its value cannot be changed. The Gripper commands can be configured with Set GripperInitMode.

The format of the response to this command is as follows:

```
'field1'
```

Response format fields and values are shown below.

Field #	ASCII text	Description
1	InitMode	Value of GripperInitMode

Example: Get the gripper configuration:

```
/1?84
```

Get GripperInitMode

Synopsis: The initialization mode of the gripper

Values: 0, 1, 2, or 3

Syntax: `/<AxisID>?19`

Description: Gets the initialization mode of the gripper. Possible values are:

- 0: Normal init, closed
- 1: Alternate init, closed
- 2: Normal init, open
- 3: Alternate init, open

The default value is 0. Closed or open specifies the gripper fingers' position after the initialization is performed.

In a Normal init, the Initialize Gripper command checks for the presence of an object in the gripper before initialization. If an object is detected, initialization is not performed. Initialize Gripper will also not execute if the CAM is in an intermediate position (neither fully closed nor fully open).

In an Alternate init, Initialize Gripper will not check for gripper status before initializing.

Example: Get the initialization mode of the gripper for axis 1:

```
/1?19
```

To set, see the Set GripperInitMode command.

Get GripperState

Synopsis: The state of the gripper

Values:
0 - Gripper Open
1 - Gripper Not Attached
2 - Gripper Move Error
3 - Gripper Not Initialized
4 - Gripper Object Detected
5 - Gripper Closed

Syntax: `/<AxisID>?20`

Description: GripperState is a read-only parameter and its value cannot be changed.

Example: Get the state of the gripper for axis 1:

```
/1?20
```

Get InitAcceptanceWindow

Synopsis: The allowable position difference between the two initializations in a double initialization performed with the Initialize Axis command

Values: 0 to 10 mm

Syntax: **`/<AxisID>?4`**

Description: Applies to Z axis only. The default value is 0.5 mm.

To set, see the Set InitAcceptanceWindow command.

Example: Get the allowable initialization position difference of axis 1:

```
/1?4
```

Get InitForceFactor

Synopsis: The force that is used to search for a hard stop during the execution of the Initialize Axis command

Values: 6000 to 22000

Syntax: **`/<AxisID>?5`**

Description: Applies to Z axis only. The higher the force factor, the stronger the applied force is. Default values are:

- Standard Z axis: 8000
- Universal Z axis: 22000
- Dual Z axis: 8000

To set, see the Set InitForceFactor command.

Example: Get the initialization force factor of axis 1:

```
/1?5
```

Get InitForceRatio

Synopsis: The percentage of the *InitForceFactor* used for the first initialization in a double hardstop initialization process

Values: 1-100

Syntax: **`/<AxisID>?6`**

Description: Applies to Universal Z axis during double initialization only. The higher the force ratio, the stronger the applied force is for the first of the double hardstop initializations. The default value is **55**.

The force that is used to search for a hard stop during the execution of the Initialize Axis command.

This variable is not applicable to the Standard Z or Dual Z axis, because it always applies half of the full force factor for the first of the double hardstop initializations.

To set, see the Set InitForceRatio command.

Example: Get the initialization force factor of axis 1:

```
/1?6
```

Get Initialize Axis Configuration

Synopsis: ASCII text report of configuration for parameters that affect the Initialize Axis command

Values: ASCII text string

Syntax: **/<*AxisID*>?77**

Description: Initialization Configuration is a read-only parameter and its value cannot be changed. The Initialize Axis command can be configured with Set DefInitSpeed, Set DefInitAcceleration, Set InitForceFactor, Set InitAcceptanceWindow, Set InitMode, and Set InitForceRatio.

The format of the response to this command is as follows, with each field separated by a '|' character:

'field1|field2|field3|field4|field5|field6'

Applicability of fields depends on Axis type. Response format fields and values are shown below.

Field #	ASCII text	Description
1	Speed	Value of DefInitSpeed
2	Accel	Value of DefInitAcceleration
3	ForceFactor	Value of InitForceFactor
4	Window	Value of InitAcceptanceWindow
5	Mode	Value of InitMode
6	ForceRatio	Value of InitForceRatio

Example: Get the initialization configuration:

```
/1?77
```

Get InitMode

Synopsis: The value that indicates the initialization mode (single or double) of the module

Values: 0 (double init) or 1 (single init)

Syntax: **`/<AxisID>?3`**

Description: Applies to Z axis only. The default value is 0 (double init).

To set, see the Set InitMode command.

Caution! *Because the axis uses a hard-stop for initialization, using single init increases the risk of the axis initializing at an incorrect position due to mechanical interference.*

Example: Get the initialization mode of axis 1:

```
/1?3
```

Get LLDAcceptanceWindow

Synopsis: The maximum acceptable difference between the two detection positions used in double detection

Values: 0.5 to 50 mm

Syntax: **`/<AxisID>?28`**

Description: This setting applies to both positive and negative differences. The default value is 5.

To set, see the Set LLDAcceptanceWindow command.

Example: Get the maximum acceptable difference between detection positions for axis 1:

```
/1?28
```

Get LastDeviceErrorInfo

Synopsis: Information about the most recent device error

Values: See Table 6-4.

Syntax: **`/<AxisID>?24`**

Description: LastDeviceErrorInfo is a read-only parameter and its value cannot be set. See Table 6-4, ACB device error types for a list of error types.

Example: Get information about the most recent error for axis 1:

```
/1?24
```

Get LLDDetectionMethod

Synopsis: The value that indicates the detection method of the LLD system

Values: 0 or 1

Syntax: **`/<AxisID>?25`**

Description: Valid values are 0 (for double detection) and 1 (for single detection); the default value is 0.

To set, see the Set LLDDetectionMethod command.

Example: Get the detection method for axis 1:

```
/1?25
```

Get LLDRetractDist

Synopsis: The retract distance for the Z axis after the first detection of a double detection

Values: *LLDAcceptanceWindow* to 100 mm

Syntax: **`/<AxisID>?27`**

Description: Valid values are between the current value of *LLDAcceptanceWindow* and 100 mm. The default value is 10 mm.

To set, see the Set LLDRetractDist command.

Example: Get the retract distance of axis 1:

```
/1?27
```

Get LLDRetractOnError

Synopsis: The value that represents whether or not the Z axis will retract to the start position after a “No Liquid Detected” error

Values: 0 or 1

Syntax: **`/<AxisID>?26`**

Description: Valid values are 0 (retract) and 1 (do not retract); the default value is 0.

To set, see the Set LLDRetractOnError command.

Example: Get the retract on error status of axis 1:

```
/1?26
```

Get LLDSearchSpeed

Synopsis: The speed to be used when searching for liquid, in mm/s

Values: 1 to 150 mm/s

Syntax: **`/<AxisID>?29`**

Description: The default search speed value is 60 mm/s.

To set, see the Set LLDSearchSpeed command.

Example: Get the liquid search speed for axis 1:

```
/1?29
```

Get LLDSelfTestLastResult

Synopsis: Returns the result of the most recent call to cLLD Self Test command.

Values: A four-digit binary number. See cLLD Self Test for more information.

Syntax: **`/<AxisID>?32`**

Description: This is a read-only parameter and its value cannot be set.

Example: Get the most recent results of the cLLD Self Test command for axis 1:

```
/1?32
```

Get LLDSelfTestLevel

Synopsis: The sensitivity test level

Values: 0 or 1

Syntax: **`/<AxisID>?31`**

Description: The sensitivity test level. Valid values are 0: low sensitivity and 1: high sensitivity; the default value is 0.

To set, see Set LLDSelfTestLevel.

Example: Get the self test level of axis 1:

```
/1?31
```

Get LLDSelfTestSensitivity

Synopsis: The sensitivity, in increments, to be used when performing a diagnostic test

Values: 3 (most sensitive) to 120 (least sensitive)

Syntax: **`/<AxisID>?30`**

Description: Valid values are from 3 (highest sensitivity) to 120 (lowest sensitivity); the default value is 63.

The LLDSelfTestSensitivity setting results in a PWM duty cycle setting on the ACB J7 pin 1 only during command operations that use this parameter.

Caution! *An inappropriate sensitivity setting may cause false liquid level detection.*

To set, see the Set LLDSelfTestSensitivity command.

Example: Get the self test sensitivity of axis 1:

```
/1?30
```

Get LLDSensitivity

Synopsis: The sensitivity, in increments, to be used when searching for liquid

Values: 3 (most sensitive) to 120 (least sensitive)

Syntax: **`/<AxisID>?22`**

Description: Valid values are from 3 (highest sensitivity) to 120 (lowest sensitivity); the default value is 63.

The *LLDSensitivity* setting results in a PWM duty cycle setting on the ACB J7 pin 1 only during command operations that use this parameter.

Note that in addition to increasing the liquid sensitivity, setting the *LLDSensitivity* parameter to a more sensitive setting makes the cLLD system more susceptible to outside interference. Conversely, setting the cLLD system to a less sensitive setting makes it less susceptible to interference.

See Section C.10, ”Preventing Interference Between Two Robots” on page C-38 and Section C.11, ”Preventing Z Axis cLLD Interference in Robots with Multiple Z Axes” on page C-39 for restrictions on the sensitivity settings.

Caution! *An inappropriate sensitivity setting may cause false liquid level detection.*

To set, see the Set LLDSensitivity command.

Example: Get the search sensitivity of axis 1:

```
/1?22
```

Get LLDSubmergeDist

Synopsis: The distance that the Z axis will extend after a successful detection is performed

Values: 0 to 100 mm

Syntax: **/<*AxisID*>?21**

Description: The default value is 0.

To set, see the Set LLDSubmergeDist command.

Example: Get the submerge distance of axis 1:

```
/1?21
```

Get MoveAbsHiForceSpeed

Synopsis: Returns the current value of the speed setting for the Move Axis to Absolute Position with High Force command.

Values: 0.1 to 10000 mm/sec

Syntax: **/<*AxisID*>?33**

Description: The movement speed of the axis during execution of the Move Axis to Absolute Position with High Force command. Default is 40 mm/sec.

This parameter is set through the [Speed] argument to the Move Axis to Absolute Position with High Force command, or with the Set MoveAbsHiForceSpeed command.

Example: Get the current speed setting for axis 1:

```
/1?33
```

Get Move Axis to Absolute Position Configuration

Synopsis: ASCII text report of configuration for parameters that affect the Move Axis to Absolute Position, and Move Axis to Absolute Position with High Force commands

Values: ASCII text string

Syntax: **/<*AxisID*>?78**

Description: Move Configuration is a read-only parameter and its value cannot be changed. The Move commands can be configured with Set DefSpeed, Set DefAcceleration, and Set MoveAbsHiForceSpeed.

The format of the response to this command is as follows, with each field separated by a '|' character:

`'field1|field2|field3'`

Applicability of fields depends on axis type.

Response format fields and values are shown below.

Field #	ASCII text	Description
1	Speed	Value of DefSpeed
2	Accel	Value of DefAcceleration
3	HiForceSpeed	Value of MoveAbsHiForceSpeed

Example: Get the move configuration:

`/1?78`

Get PickForceFactor

Synopsis: The force used to engage the disposable tip during pickup

Values: 1000 to 20000

Syntax: **`/<AxisID>?38`**

Description: This force can be set to allow for variances in disposable tips. The higher the force factor, the stronger the applied force is. When using Tecan disposable tips, do not change this value from the default setting. Default values are:

- Standard Z axis: 11800
- Universal Z axis: 7000
- Dual Z axis: 11800

To set, see the Set PickForceFactor command.

Example: Get the pick-up force for axis 1:

`/1?38`

Get PickSpeed

Synopsis: The travel speed used by the Standard Z or Dual Z axis to pick up a disposable tip

Values: 1 to 150 mm/s

Syntax: **`/<AxisID>?37`**

Description: This speed can be set to allow for variances in disposable tips. When using Tecan disposable tips, do not change this value from the default setting. The default value is 60 mm/s.

To set, see the Set PickSpeed command.

Example: Get the pick-up speed of axis 1:

```
/1?37
```

Get Pick Tip Configuration

Synopsis: ASCII text report of configuration for parameters that affect the Pick Tip command

Values: ASCII text string

Syntax: **`/<AxisID>?82`**

Description: Pick Tip Configuration is a read-only parameter and its value cannot be changed. The Pick Tip command can be configured with Set PickSpeed, Set DefAcceleration, and Set PickForceFactor.

The format of the response to this command is as follows, with each field separated by a '|' character:

`'field1|field2|field3'`

Response format fields and values are shown below.

Field #	ASCII text	Description
1	Speed	Value of PickSpeed
2	Accel	Value of DefAcceleration
3	ForceFactor	Value of PickForceFactor

Example: Get the pick tip configuration:

```
/1?82
```

Get Pos

Synopsis: The position of the axis per the device encoder

Values: Axis coordinates

Syntax: **`/<AxisID>?0`**

Description: Returns the actual (physical) position per encoder feedback.

To set, see the Set Pos command.

WARNING! *Using the Set command will change the Omni Robot coordinate system and the behavior of the system!*

Example: Get the position of axis 1:

```
/1?0
```

Get PWMDutyCycle

Synopsis: The high (on) state interval of the PWM output signal period for all Z axis types, in 25ns increments. Not available for X and Y axes. Output signal frequency on J7 pin 1 of the ACB board.

Values: Valid range: 0 (output signal stead at low state) to (PWMFrequency-1); the default value is 63.

Syntax: **`/<AxisID>?35`**

Description: Gets the On (= high) state time. The Off (= low) state time is calculated to maintain the period of the output signal as set by the Set PWMFrequency command.

A special case represents the value of 0 (zero), which disables the toggling of the output, leaving the output permanently at its low state.

To set, see the Set PWMDutyCycle command.

Example: Get the PWM output signal interval of axis 1:

```
/1?35
```

Get PWMFrequency

Synopsis: The PWM output signal period for all Z-axis types, in 25ns increments. Not available for X and Y axes. Output signal frequency on J7 pin 1 of the ACB board.

Values: (PWMDutyCycle+1) to 65535; the value is initialized at system startup for non-X or ^ axes starting from 127 with an increment of 2 to avoid conflict.

Syntax: **`/<AxisID>?34`**

Description: The start up frequencies are set based on the address sequence of all applicable axis types (Z-axes types only) starting with 127 and incrementing by 2 as shown here:

X axis Arm1:	Address 1	--> N/A
Y axis Arm1:	Address 2	--> N/A
Z axis Arm1:	Address 3	--> PWMFrequency = 127
X axis Arm2:	Address 4	--> N/A
Y axis Arm2:	Address 5	--> N/A
Z axis Arm2:	Address 6	--> PWMFrequency = 129
2nd Z axis Arm2:	Address 7	--> PWMFrequency = 131

Caution! *An inappropriate PWMFrequency setting may cause false liquid level detection. See Section C.10, "Preventing Interference Between Two Robots" on page C-38 and Section C.11, "Preventing Z Axis cLLD Interference in Robots with Multiple Z Axes" on page C-39 for more details.*

To set, see the Set PWMFrequency command.

Example: Get the PWM output signal period of axis 1:

```
/1?34
```

Get RangeHigh

Synopsis: The highest reachable axis position, in mm

Values: X axis left: **AxisOffset** + **WorkingLen**
X axis right: **AxisOffset**
Y axis: **AxisOffset + WorkingLen**
Z axis: **AxisOffset**

Syntax: **`/<AxisID>?39`**

Description: RangeHigh is calculated; it cannot be set.

Example: Get the highest reachable position of axis 1:

```
/1?39
```

Get RangeLow

Synopsis: The lowest reachable axis position, in mm

Values: X axis left: **AxisOffset**
X axis right: **AxisOffset** - **WorkingLen**
Y axis: **AxisOffset**
Z axis: **AxisOffset - WorkingLen**

Syntax: **/<*AxisID*>?40**

Description: RangeLow is calculated; it cannot be set.

Example: Get the lowest reachable position of axis 1:

```
/1?40
```

Get Status

Synopsis: The current status of the axis

Values: Initialized or Not Initialized. Also may include one or more additional messages:

- 0x00: Initialized
- 0x10: Not Ready
- 0x20: Not Initialized
- 0x01: Disabled
- 0x02: Device Error
- 0x04: Device Busy

Syntax: **/<*AxisID*>?11**

Description: Status is a read-only parameter and its value cannot be changed.

Example: Get the current status of axis 1:

```
/1?11
```

Get TipPresence

Synopsis: The value that indicates if a tip is attached to the Z axis

Values: 0 (not present) or 1 (present)

Syntax: **/<*AxisID*>?36**

Description: TipPresence is a read-only parameter and its value cannot be changed.

Example: Get tip presence status of axis 1:

```
/1?36
```

Get WorkingLen

Synopsis: The distance between the axis home position and the maximum reachable axis position

Values: 0 to 10,000 mm

Syntax: **`/<AxisID>?41`**

Description: This value is used to calculate *RangeLow* and *RangeHigh*. Default values:

- Axis type 1: 750
- Axis type 2: 300
- Axis types 3, 8, 9: 210

To set, see the Set WorkingLen command.

Example: Get the distance between the home position and maximum reachable axis position for axis 1:

```
/1?41
```

6.8.3 Set Commands that Write ACB Configuration Variables to RAM

Synopsis: The u<n> command sets the value of parameters that can only be set in RAM.

Syntax: **`/<AxisID>u<Var_ID>,<Values>`**

AxisID: Valid values are 1-f.

Var_ID: The following table provides a list of Var_IDs and the parameters they correspond to:

Table 6-10 *Axis Set command (RAM) arguments*

Var ID	Name	Page Num
0	Pos	6-71
2	AxisOrientation	6-60
3	InitMode	6-66
4	InitAcceptanceWindow	6-64
5	InitForceFactor	6-65
6	InitForceRatio	6-65
7	DefInitAcceleration	6-60

Var ID	Name	Page Num
19	GripperInitMode	6-64
21	LLDSubmergeDist	6-69
22	LLDSensitivity	6-68
25	LLDDetectionMethod	6-66
26	LLDRetractOnError	6-67
27	LLDRetractDist	6-67
28	LLDAcceptanceWindow	6-66

Table 6-10 *Axis Set command (RAM) arguments*

Var ID	Name	Page Num
8	DefInitSpeed	6-61
9	DefAcceleration	6-60
10	DefSpeed	6-61
12	DropSpeed	6-62
13	DropForceFactor	6-62
14	EncoderDirection	6-62
15	ExitRetractSpeed	6-63
16	ExitRetractDist	6-63
17	ExitLimit	6-63

Var ID	Name	Page Num
29	LLDSearchSpeed	6-67
30	LLDSelfTestSensitivity	6-68
31	LLDSelfTestLevel	6-68
33	MoveAbsHiForceSpeed	6-69
34	PWMFrequency	6-72
35	PWMDutyCycle	6-71
37	PickSpeed	6-70
38	PickForceFactor	6-70
41	WorkingLength	6-73

Values: Valid values are specified for each command.

The following errors can be generated by the **u<*n*>** command:

Generated Errors: 3 - Invalid Operand
5 -- Device Not Implemented
7 - Device not initialized (NOTE*1)
8 - Device Busy

NOTE*1: Applicable only to AxisOrientation, EncoderDirection: After setting these variables, the axis will change state to “not initialized” automatically.

WARNING! *Take care to ensure that the values of any parameters you set are within the valid range for your robot hardware (such as axis dimensions and travel length).*

Caution! *Setting parameters to values lower than the recommended minimum or higher than the recommended maximum may affect Omni Robot performance. Operation may become unstable even though the parameter value was accepted.*

Set AxisOrientation

Synopsis: The orientation of the X or Y axis

Values: 0, or 1

Syntax: **`/<AxisID>u2,<Orientation>`**

Description: The orientation of the X or Y axis can be set to one of:

- 0: Left oriented, initialize to the left
- 1: Right oriented, initialize to the right

To get the current value, see the Get AxisOrientation command.

Example: Set the orientation of axis 1 to initialize to the left:

```
/1?2,0
```

Set DefAcceleration

Synopsis: The acceleration of the axis during any move command, except the Initialize Axis command (see the Get DefInitAcceleration parameter for the Initialize Axis command)

Values: 1 to 100,000 mm/s^2

Syntax: **`/<AxisID>u9,<Accel>`**

Description: Acceleration values are based on calculations. The actual acceleration of the X axis and Y axis could be lower due to the positioning algorithm depending on travel distance and speed settings. The default values are:

- X axis: 3500 mm/s^2
- Y axis: 3900 mm/s^2
- Standard Z axis: 6000 mm/s^2
- Universal Z axis: 1000 mm/s^2
- Dual Z axis: 6000 mm/s^2

To get the current value, see the Get DefAcceleration command.

Example: Set the acceleration of axis 1 to 1200 during the command (except Initialize Axis):

```
/1u9,1200
```

Set DefInitAcceleration

Synopsis: The acceleration of the axis during the Initialize Axis command

Values: 1 to 100,000 mm/s^2

Syntax: **`/<AxisID>u7,<Accel>`**

Description: Acceleration values are based on calculations. The actual acceleration of the X axis and Y axis could be lower due to the positioning algorithm depending on travel distance and speed settings. The default values are:

- X axis: 3500 mm/s^2
- Y axis: 3900 mm/s^2
- Standard Z axis: 6000 mm/s^2
- Universal Z axis: 1000 mm/s^2
- Dual Z axis: 6000 mm/s^2

To get the current value, see the Get DefInitAcceleration command.

Example: Set the acceleration of axis 1 to 2000 during the Initialize Axis command:

```
/1u7,2000
```

Set DefInitSpeed

Synopsis: The initialization speed of the axis, which is used when executing the Initialize Axis command

Values: 0.1 to 200 mm/s

Syntax: **`/<AxisID>u8,<Speed>`**

Description: The default values are:

- X axis: 80 mm/s
- Y axis: 80 mm/s
- Z axis: 40 mm/s

To get the current value, see the Get DefInitSpeed command.

Example: Set the initialization speed of axis 1 to 80:

```
/1u8,80
```

Set DefSpeed

Synopsis: The default axis movement speed

Values: 0.1 to 10000 mm/s

Syntax: **`/<AxisID>u10,<Speed>`**

Description: The default values are:

- X axis: 800 mm/s
- Y axis: 600 mm/s
- Standard Z axis: 600 mm/s
- Universal Z axis: 250 mm/s
- Dual Z axis: 600 mm/s

To get the current value, see the Get DefSpeed command.

Example: Set the default speed of axis 1 to 800:

```
/1u10,800
```

Set DropForceFactor

Synopsis: The force used to drop the disposable tip

Values: 8000 to 22000

Syntax: **/<*AxisID*>u13,<*Force*>**

Description: The default value is 17000. The higher the force factor, the stronger the applied force is. To get the current value, see the Get DropForceFactor command.

Note: Because the Drop Tip hardware mechanism is not yet supported for the Universal Z axis, issuing Drop Tip with DropForceFactor while using the Universal Z axis will have no physical drop-tip effect.

Example: Set the force used to drop the disposable tip to 2000:

```
/3u13,2000
```

Set DropSpeed

Synopsis: The travel speed used by the Standard Z and Dual Z axis to drop a disposable tip

Values: 1 to 150 mm/s

Syntax: **/<*AxisID*>u12,<*Speed*>**

Description: The default value is 60 mm/s. To get the current value, see the Get DropSpeed command.

Note: Because the Drop Tip command is not yet supported for the Universal Z axis, using the DropSpeeed command with the Universal Z axis will have no effect.

Example: Set the force used to drop the disposable tip to 80:

```
/3u12,80
```

Set EncoderDirection

Synopsis: Changes the direction of encoder performing increment and decrement.

Values: 0, 1

Syntax: **/<*AxisID*>u14,<*Direction*>**

Description: Changes the direction of encoder performing increment and decrement. The legal values are:

- 0: Hardware uses normal encoder direction
- 1: Hardware uses reversed encoder direction

Default is 0. To get, see the Get EncoderDirection command.

Example: Set the encoder direction of axis 3 to reverse direction:

```
/3u14,1
```

Set ExitLimit

Synopsis: The acceptance limit for a successful exit signal check

Values: 0 to 50 mm

Syntax: **/<*AxisID*>u17,<*Limit*>**

Description: The default value is 5 mm. To get the current value, see the Get ExitLimit command.

Example: Set the acceptance limit for a successful exit signal check to 50:

```
/3u17,50
```

Set ExitRetractDist

Synopsis: The travel distance in the Z direction that the Z axis will move when searching for an exit signal with the Check For Exit command

Values: 1 to 100 mm

Syntax: **/<*AxisID*>u16,<*RetractDist*>**

Description: The default value is 10. To get the current value, see the Get ExitRetractDist command.

Example: Set the retract travel distance to 40:

```
/3u16,40
```

Set ExitRetractSpeed

Synopsis: The speed that the Z axis will use when retracting and searching for an exit signal

Values: 1 to 150 mm/s

Syntax: **/<*AxisID*>u15,<*Speed*>**

Description: The default value is 60. To get the current value, see the Get ExitRetractSpeed command.

Example: Set the retract travel speed to 35:

```
/3u15,35
```

Set GripperInitMode

Synopsis: The initialization mode of the gripper

Values: 0, 1, 2, or 3

Syntax: ***/<AxisID>u19,<InitMode>***

Description: Sets the initialization mode of the gripper. Possible modes are:

- 0: Normal init, closed
- 1: Alternate init, closed
- 2: Normal init, open
- 3: Alternate init, open

The default value is 0. Closed or open specifies the gripper fingers' position after the initialization is performed.

In a Normal init, the Initialize Gripper command checks for the presence of an object in the gripper before initialization. If an object is detected, initialization is not performed. Initialize Gripper will also not execute if the CAM is in an intermediate position (neither fully closed nor fully open).

In an Alternate init, Initialize Gripper will not check for gripper status before initializing.

Example: Set the initialization mode of the gripper for axis 3 to Normal init, closed:

```
/3u19,0
```

To get the current value, see the Get GripperInitMode command.

Set InitAcceptanceWindow

Synopsis: The allowable position difference between the two initializations in a double initialization performed with the Initialize Axis command

Values: 0 to 10 mm

Syntax: ***/<AxisID>u4,<Window>***

Description: Applies to Z axis only. The default value is 0.5 mm. To get the current value, see the Get InitAcceptanceWindow command.

Example: Set the allowable position difference between two initializations to 5.5 mm:

```
/3u4,5.5
```

Set InitForceFactor

Synopsis: The force that is used to search for a hard stop during the execution of the Initialize Axis command

Values: 6000 to 22000

Syntax: **`/<AxisID>u5,<Force>`**

Description: Applies to Z axis only. The higher the force factor, the stronger the applied force is. Default values are:

- Standard Z axis: 8000
- Universal Z axis: 22000
- Dual Z axis: 8000

To get the current value, see the Get InitForceFactor command.

Example: Set the force used to search for a hard stop to 2000:

```
/3u5,2000
```

Set InitForceRatio

Synopsis: The percentage of the *InitForceFactor* used for the first initialization in a double hardstop initialization process

Values: 1-100

Syntax: **`/<AxisID>u6,<Force>`**

Description: Applies to Universal Z axis only. The higher the force ratio, the stronger the applied force is for the first of the double hardstop initializations. The default value is **55**.

Increasing the force ratio from the default decreases the effectiveness of detecting potential blockages when using hardstop to find the home position. Decreasing the force ratio from the default might impact the ability of the axis to perform the first initialization movement. See the Initialize Axis command for more information.

To get the current value, see the Get InitForceRatio command.

Example: Set the first force used to search for a hard stop to 60% of InitForceFactor:

```
/3u6,60
```

Set InitMode

Synopsis: The value that indicates the initialization mode (single or double) of the module

Values: 0 (double init) or 1 (single init)

Syntax: **`/<AxisID>u3,<Mode>`**

Description: Applies to Z axis only. The default value is 0 (double init). To get the current value, see the Get InitMode command.

Caution! *Because the axis uses a hard-stop for initialization, using single init increases the risk of the axis initializing at an incorrect position due to mechanical interference.*

Example: Set the initialization mode of the module to single init:

```
/3u3,1
```

Set LLDAcceptanceWindow

Synopsis: The maximum acceptable difference between the two detection positions used in double detection

Values: 0.5 to 50 mm

Syntax: **`/<AxisID>u28,<Window>`**

Description: This setting applies to both positive and negative differences. The default value is 5. To get the current value, see the Get LLDAcceptanceWindow command.

Example: Set the maximum acceptable difference between two detection positions to 10 mm:

```
/3u28,10
```

Set LLDDetectionMethod

Synopsis: The value that indicates the detection method of the LLD system

Values: 0 or 1

Syntax: **`/<AxisID>u25,<Method>`**

Description: Valid values are 0 (for double detection) and 1 (for single detection); the default value is 0. To get the current value, see the Get LLDDetectionMethod command.

Example: Set the detection method to single detection:

```
/3u25,1
```

Set LLDRetractDist

Synopsis: The retract distance for the Z axis after the first detection of a double detection

Values: *LLDAcceptanceWindow* to 100 mm

Syntax: **`/<AxisID>u27,<RetractDist>`**

Description: Valid values are between the current value of *LLDAcceptanceWindow* and 100 mm. The default value is 10 mm. To get the current value, see the Get LLDRetractDist command.

Example: Set the retract distance to 5 mm:

```
/3u27,5
```

Set LLDRetractOnError

Synopsis: The value that represents whether or not the Z axis will retract to the start position after a “No Liquid Detected” error

Values: 0 or 1

Syntax: **`/<AxisID>u26,<Retract>`**

Description: Valid values are 0 (retract) and 1 (do not retract); the default value is 0. To get the current value, see the Get LLDRetractOnError command.

Example: Set the Z axis to retract to the start position after a “No Liquid Detected” error:

```
/3u26,0
```

Set LLDSearchSpeed

Synopsis: The speed to be used when searching for liquid, in mm/s

Values: 1 to 150 mm/s

Syntax: **`/<AxisID>u29,<SearchSpeed>`**

Description: The default search speed value is 60 mm/s. To get the current value, see the Get LLDSearchSpeed command.

Example: Set the search speed to 50 mm/s:

```
/3u29,50
```

Set LLDSelfTestLevel

Synopsis: The sensitivity test level

Values: 0 or 1

Syntax: **`/<AxisID>u31,<TestLevel>`**

Description: The sensitivity test level. Valid values are 0: Low sensitivity, and 1: High sensitivity; the default value is 0.

To set, see Get LLDSelfTestLevel

Example: Set the self test level of axis 3 to high sensitivity:

```
/3u31,1
```

Set LLDSelfTestSensitivity

Synopsis: The sensitivity, in increments, to be used when performing a diagnostic test

Values: 3 (most sensitive) to 120 (least sensitive)

Syntax: **`/<AxisID>u30,<LLDSensitivity>`**

Description: Valid values are from 3 (highest sensitivity) to 120 (lowest sensitivity); the default value is 63.

The LLDSelfTestSensitivity setting results in a PWM duty cycle setting on the ACB J7 pin 1 only during command operations that use this parameter.

Caution! *An inappropriate sensitivity setting may cause false liquid level detection.*

To get the current value, see the Get LLDSelfTestSensitivity command.

Example: Set the self-test sensitivity level to 90:

```
/3u30,90
```

Set LLDSensitivity

Synopsis: The sensitivity, in increments, to be used when searching for liquid

Values: 3 (most sensitive) to 120 (least sensitive)

Syntax: **`/<AxisID>u22,<LLDSensitivity>`**

Description: Valid values are from 3 (highest sensitivity) to 120 (lowest sensitivity); the default value is 63.

The *LLDSensitivity* setting results in a PWM duty cycle setting on the ACB J7 pin 1 only during command operations that use this parameter.

Note that in addition to increasing the liquid sensitivity, setting the *LLDSensitivity* parameter to a more sensitive setting makes the cLLD system more susceptible to outside interference. Conversely, setting the cLLD system to a less sensitive setting makes it less susceptible to interference.

See Section C.10, "Preventing Interference Between Two Robots" on page C-38 and Section C.11, "Preventing Z Axis cLLD Interference in Robots with Multiple Z Axes" on page C-39 for restrictions on the sensitivity settings.

Caution! *An inappropriate sensitivity setting may cause false liquid level detection.*

To get the current value, see the Get LLDSensitivity command.

Example: Set the sensitivity level to 90:

```
/3u22,90
```

Set LLDSubmergeDist

Synopsis: The distance that the Z axis will extend after a successful detection is performed

Values: 0 to 100 mm

Syntax: **`/<AxisID>u21,<SubmergeDist>`**

Description: The default value is 0. To get the current value, see the Get LLDSubmergeDist command.

Example: Set the submerge distance to 5 mm:

```
/3u21,5
```

Set MoveAbsHiForceSpeed

Synopsis: Sets the current value of the speed setting for the Move Axis to Absolute Position with High Force command.

Values: 0.1 to 10000 mm/sec

Syntax: **`/<AxisID>u33,<Speed>`**

Description: Applies only to the Standard Z or Dual Z axis. The movement speed of the axis during execution of the Move Axis to Absolute Position with High Force command. Default is 40 mm/sec.

To get the current value, see the Get MoveAbsHiForceSpeed command.

Example: Set the current speed setting for axis 1 to 50:

```
/3u33,50
```

Set PickForceFactor

Synopsis: The force used to engage the disposable tip during pickup

Values: 1000 to 20000

Syntax: **`/<AxisID>u38,<SubmergeDist>`**

Description: This force can be set to allow for variances in disposable tips. The higher the force factor, the stronger the applied force is. When using Tecan disposable tips, do not change this value from the default setting. Default values are:

- Standard Z axis: 11800
- Universal Z axis: 7000
- Dual Z axis: 11800

To get the current value, see the Get PickForceFactor command.

Example: Set the force factor of axis 3 to 3000:

```
/3u38,3000
```

Set PickSpeed

Synopsis: The travel speed used by the Standard Z or Dual Z axis to pick up a disposable tip

Values: 1 to 150 mm/s

Syntax: **`/<AxisID>u37,<Speed>`**

Description: This speed can be set to allow for variances in disposable tips. When using Tecan disposable tips, do not change this value from the default setting. The default value is 60 mm/s. To get the current value, see the Get PickSpeed command.

Example: Set the travel speed of axis 3 to 80 mm/s:

```
/3u37,80
```

Set Pos

Synopsis: The position of the axis per the device encoder

Values: Axis coordinates

Syntax: `/<AxisID>u0,<Pos>`

Description: This command can be used to calibrate axis coordinates to worktable coordinates. Once set, the system automatically re-calculates the axis offset, *RangeLow* and *RangeHigh* parameters.

WARNING! *Using this command will change the Omni Robot coordinate system and the behavior of the system!*

To get the current value, see the Get Pos command.

Example: Set the physical position of axis 1 to 100.5:

```
/1u0,100.5
```

Set PWMDutyCycle

Synopsis: The high (on) state interval of the PWM output signal period for all Z axis types, in 25ns increments. Not available for X and Y axes. Output signal frequency on J7 pin 1 of the ACB board.

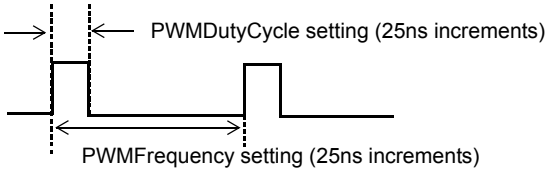
Values: Valid range: 0 (output signal stead at low state) to (PWMFrequency-1); the default value is 63.

Syntax: `/<AxisID>u35,<Value>`

Description: This command sets the On (= high) state time, while the Off (= low) state time is calculated to maintain the period of the output signal as set by the Set PWMFrequency command.

A special case represents the value of 0 (zero), which disables the toggling of the output, leaving the output permanently at its low state. Changing the setting from 0 to a different value will cause a delay in restoring the cLLD functionality. The required time for the cLLD to be fully operable depends on the new sensitivity setting, for example:

- ≥ 50 ms for sensitivity 63 (default), or
- ≥ 500 ms for sensitivity 3 (worst case)



The PWM duty cycle is also set by altering the **LLDSensitivity** and **LLDSelfTestSensitivity** parameters. The **LLDSensitivity** setting becomes effective only during execution of the **Detect Liquid** command or the **Check For Exit** command whereas **LLDSelfTestSensitivity** becomes effective only during the execution of the **cLLDSelfTest** command.

The **Set PWMDutyCycle** command applies the new settings on the ACB immediately. The setting remains valid until one of the liquid detection or self test commands is sent, when the duty cycle set by **LLDSensitivity** or **LLDSelfTestSensitivity** takes effect. The duty cycle is restored to the original **PWMDutyCycle** value after the command operation.

The **PWMDutyCycle** parameter makes the PWM capabilities of the ACB board available for general purpose unrestricted use.

To get the current value, see the Get PWMDutyCycle command.

Example: Set the interval of the PWM output signal period to 63:

```
/3u35,63
```

Set PWMFrequency

Synopsis: The PWM output signal period for all Z axis types, in 25ns increments. Not available for X and Y axes. Output signal frequency on J7 pin 1 of the ACB board.

Values: (PWMDutyCycle+1) to 65535; the value is initialized at system startup for Z axes starting from 127 with an increment of 2 to avoid conflict.

Syntax: ***/<AxisID>u34,<Value>***

Description: Setting PWMFrequency overwrites the frequency for operating the cLLD board that is set automatically at start up. It is used to separate the frequencies in configurations with multiple Z axes to prevent cLLD interference.

The start up frequencies are set based on the address sequence of all applicable axes starting with 127 and incrementing by 2 as shown here:

X axis Arm1:	Address 1	--> N/A
Y axis Arm1:	Address 2	--> N/A
Z axis Arm1:	Address 3	--> PWMFrequency = 127
X axis Arm2:	Address 4	--> N/A
Y axis Arm2:	Address 5	--> N/A

Z axis Arm2:	Address 6	--> PWMFrequency = 129
2nd Z axis Arm2:	Address 8	--> PWMFrequency = 131

When setting the PWMFrequency, it is the user's responsibility to ensure that each Z axis ACB has a unique PWMFrequency setting that maintains at least a 2 increment difference from any other Z axis on an Omni robot. The setting range for ACB boards connected to a cLLD board should be limited to 127 – 141.

This command sets the total period of the output signal; the high (On) state is controlled by the *PWMDutyCycle*; the low state is calculated as the *PWMFrequency –PWMDutyCycle*.

For example, if *PWMDutyCycle* is set to 30, setting *PWMFrequency* to 127 will generate an output signal with an On time of 30 x 25ns = 750ns,a period of (127+1) x 25ns = 3200ns, and an Off time of (127 – On time) x 25ns = 98 x 25ns = 2450ns.

```
Actual Frequency
= 1/{ (PWMFrequency + 1) * 25e-09 s} Hz
= 1/{ (127 + 1) * 25e-09 s} Hz
= 1/{3200e-09 s} Hz
= 312.5kHz
```

Caution! *An inappropriate PWMFrequency setting may cause false liquid level detection.* See Section C.10, "Preventing Interference Between Two Robots" on page C-38 and Section C.11, "Preventing Z Axis cLLD Interference in Robots with Multiple Z Axes" on page C-39 for more details.

To get the current value, see the Get PWMFrequency command.

Example: Set the *PWMFrequency* to 127 on the Z axis (Address 3).

```
/3u34,127
```

Set WorkingLen

Synopsis: The distance between the axis home position and the maximum reachable axis position

Values: 0 to 10,000 mm

Syntax: **`/<AxisID>u41,<Len>`**

Description: This value is used to calculate *RangeLow* and *RangeHigh*.

Default values for ACBs that have not been configured by the Configurator:

- Axis type 1: 750
- Axis type 2: 300
- Axis types 3, 8, 9: 210

To get the current value, see the Get WorkingLen command.

Example: Set the distance between axis 1 home position and the maximum reachable axis position to 1250:

```
/1u41,1250
```

6.8.4 Set Commands that Write ACB Configuration to Non-Volatile Memory

You can use the Configurator software to permanently change the value of a parameter (i.e., to change its power-on default value). Changes made through the Configurator are permanently saved by the CIB. See the Configurator documentation for more information.

Synopsis: The U<n> command sets the value of the ***Var_ID***.

Syntax: **/<*Axis_ID*>U,<*Var_ID*>**

Var_ID: The following table provides a list of Var_IDs and the variables they correspond to:

Table 6-11 *ACB set command (non-volatile memory) arguments*

Var ID	Name	See Page
71	Number of ACB move counts	6-74
73	Highest recorded ACB temperature	6-75

The following errors can be generated by the **U<*n*>** command:

Generated Errors: 3 - Invalid Operand

Set ACB Move Count

Synopsis: Sets the number of ACB move counts.

Values: 0 to 4294967295 (0xFFFFFFFF)

Syntax: **/<*Axis_ID*>U71,<*Value*>**

Description: ACB move counts automatically rolls over if the maximum value is exceeded.

To get, see the Get ACB Move Count Since Last Reset command.

Example: Set the number of ACB move counts to 100:

```
/U71,100
```

Set Highest Recorded ACB Temperature

Synopsis: Resets the highest recorded ACB temperature to 0.

Values: 0

Syntax: **/<*Axis_ID*>U73,0**

Description: Use this command to reset the ACB recorded temperature to 0 so that a new highest temperature can be recorded.

To get, see the Get ACB Highest Temperature Recorded Since Last Reset command.

Example: Reset the ACB highest recorded temperature:

```
/U73,0
```

6.9 Axis Action Commands

The following table summarizes the Cavro® Omni Robot action commands.

Table 6-12 *Action commands*

Description	Embedded Command	See Page
Brake Test: Performs a diagnostic brake test.	**/<*AxisID*>b**	6-77
Check for Exit: Retracts tip and checks for exit from liquid.	**/<*AxisID*>O**	6-81
Clear Error: Clears and resets status of the axis.	**/<*AxisID*>C**	6-83
cLLD Self Test: Activates the cLLD self test for the Z axis.	**/<*AxisID*>c**	6-77
Detect Liquid: Moves axis to detect the presence of liquid.	**/<*AxisID*>B**	6-84
Disable: Disables the current/drive for the axis.	**/<*AxisID*>N0**	6-94
Drop Tip: Drops a disposable tip.	**/<*AxisID*>E**	6-91
Enable: Enables the current/drive for the axis.	**/<*AxisID*>N1**	6-94

Table 6-12 *Action commands (continued)*

Description	Embedded Command	See Page
Initialize Axis: Initializes axis.	**/<*AxisID*>Z**	6-78
Move Axis to Absolute Position: Moves axis to specified coordinates.	**/<*AxisID*>A**	6-94
Move Axis to Absolute Position with High Force: Moves axis to an absolute position with higher ending current.	**/<*AxisID*>F**	6-95
Pick Tip: Picks up a disposable tip.	**/<*AxisID*>P**	6-97
Move Axis to Relative Position Before Initialization: Moves axis to a relative axis offset position using a fixed speed of 25 mm/s. This command will only be accepted when the axis is ***not*** initialized.	**/<*AxisID*>I**	6-96
Terminate Axis Motion: Immediately terminates axis motion.	**/<*AxisID*>T**	6-102

WARNING! *The Omni Robot contains pointed tips and other sharp-edged elements, which might cause injuries when you reach into the working area.*

- *Always be aware of mechanical hazards. Do not move your head or hands into the path of moving parts during deployment; a risk of injury, blunt force trauma, or piercing exists.*
- *Wear proper laboratory apparel (such as rubber gloves and safety goggles) as appropriate for your deployment.*
- *Personnel who are not operating the Omni Robot module should take extra care when standing near the module, as an unanticipated movement by one or more axes could cause injury to a bystander. This is especially important if the Omni Robot is being operated remotely through the Ethernet connection.*

6.9.1 Axis Action Command Descriptions

The commands included in this section operate a single axis.

To be consistent with Cavro Devices, commands sent to an individual ACB can be concatenated. However, '?' and 'T' commands must be sent as individual commands. Concatenated commands are executed sequentially.

Note*:* Executing commands for the ACB does not require the [R] command. Commands are executed immediately when they are received. To be consistent with Cavro® devices, the [R] command can be appended to the end of a command, but it has no effect. For more information on the [R] command, see the manual for your Cavro® device.

Example: Initialize axis 1 and move to absolute positions 100, 200, and 300 respectively:

```
/1ZA100A200A300R
```

Brake Test

Synopsis: Performs a diagnostic brake test.

Syntax: **/<*AxisID*>b**

AxisID: Valid values are 1-f.

Description: This command moves the Universal Z downward twice at 100 mm/s speed and 1000 mm/s^2 acceleration from the current position to 40 mm away. The first time it moves with the brake disengaged, monitors the current consumption during the motion, and moves back to the original position. The second time it moves with brake engaged, monitors the current consumption, then moves back to the original position. If the total current consumption of the second move is at least four times as much as the first, then the brake should be in a good state. Otherwise, maintenance or part replacement is required.

To prevent a move with an engaged brake, if the brake test is terminated by a terminate motion command, the state of the axis is set to uninitialized as soon as the brake test command is issued.

Generated Errors:
2 -- Invalid Command
3 -- Invalid Operand
5 -- Device Not Implemented
7 -- Device Not Initialized
8 -- Device Busy
9 -- Device Error (Hardware Error)
13 -- Device Disabled
14 -- Invalid Travel Pos
60 -- Brake test failed

Example: Perform a brake test on axis 3:

```
1/3b
```

cLLD Self Test

Synopsis: Activates the cLLD self test.

Syntax: /**<*AxisID*>c**

AxisID: Valid values are 1-f.

Description: Activates the cLLD self test.

This command activates the cLLD self test. The result of the self test is a four-digit number in the following binary format:

1	0	0	1
Dip-in signal response at capacitance increase	Dip-out signal response at capacitance increase	Dip-in signal response at capacitance decrease	Dip-out signal response at capacitance decrease

This four-digit number can be accessed with the Get LLDSelfTestLastResult command. For each digit, 0 indicates no detection signal was produced and 1 indicates that a detection signal was produced.

This command is specific to Z axes with cLLD.

Generated Errors:
2 -- Invalid Command
3 -- Invalid Operand
5 -- Device Not Implemented
8 -- Device Busy
20 -- cLLD Not Ready
27 -- DiTi - No Tip Mounted

The following parameters have an effect on the behavior of this command:

LLDSelfTestSensitivity The sensitivity, in increments, to be used when performing a diagnostic test. Valid values are from 3 (highest sensitivity) to 120 (lowest sensitivity); the default is 63 and can be changed with the Set LLDSelfTestSensitivity command.

LLDSelfTestLevel The sensitivity test level. Valid values are 0: Low sensitivity, and 1: High sensitivity; the default value is 0 and can be changed with the Set LLDSelfTestLevel command.

Example: Perform self test, high capacitance test level, lowest sensitivity:

Command: `/3u31,0u30,120c/3?32`
Response: `?300000000`

Initialize Axis

Synopsis: Initializes a single axis of motion.

Syntax: /<***AxisID***>**Z**

AxisID: Valid values are 1-f.

Description: This command initializes a single axis. An axis will not perform any action commands without being initialized.

For the X axis and Y axis, the Z command causes the axis to search for the home sensor using the current AxisOrientation. To get the current value of AxisOrientation, see Get AxisOrientation.

For Z axes, the Z command causes the axis to search for the mechanical hard stop.

If the axis has a brake attached, the brake will be mechanically disengaged after the command is issued and remain disengaged.

WARNING! *Before initializing the Z axis, be sure the probe tip is not inside a cap. Initializing the Z axis while the probe is still inside a cap can result in contamination or damage to labware.*

Generated Errors:

3 -- Invalid Operand
5 -- Device Not Implemented
8 -- Device Busy
9 -- Device Error (Hardware Error)
11 -- Double initialization acceptance window exceeded (for Z axes only)
13 -- Device disabled
103 -- Encoder Limit Switch Error

The following parameters have an effect on the behavior of this command:

AxisOrientation	The orientation of the axis. Valid values are 0: Left oriented, and 1: Right oriented. To get the current value of AxisOrientation, see Get AxisOrientation.
EncoderDirection	The direction of the encoder. Valid values are 0: Normal, and 1: Reversed. The default value is 0: Normal. To get the current value of AxisOrientation, see Get EncoderDirection.
InitMode (Z axes only)	The initialization mode (single or double) of the module. Valid values are 0 (double init) or 1 (single init). The default value is 0 (double init) and can be changed with the Set InitMode command.
InitAcceptanceWindow (Z axes only)	The allowable position difference between the two initializations in a double initialization. Valid values are 0 to 10 mm. The default value is 0.5 mm and can be changed with the Set InitAcceptanceWindow command.
InitForceFactor (Z axes only)	The force that is used to search for a hard stop. Valid values are 6000 to 22000. The default values are: ◆ Standard Z axis: 8000 ◆ Universal Z axis: 22000 ◆ Dual Z axis: 8000 They can be changed with the Set InitForceFactor command.
InitForceRatio (Universal Z axis only)	The percentage of the *InitForceFactor* used for the first initialization in a double hardstop initialization process. Valid values are 1-100. The default value is 55 and can be changed with the Set InitForceRatio command.
DefInitAcceleration	The acceleration rate of the axis during initialization. Valid values are 1 to 100,000 mm/s^2. The actual acceleration of the X axis and Y axis could be lower due to the positioning algorithm depending on travel distance and speed settings. The default values are: ◆ X axis: 3500 mm/s^2 ◆ Y axis: 3900 mm/s^2 ◆ Standard Z axis: 6000 mm/s^2 ◆ Universal Z axis: 1000 mm/s^2 ◆ Dual Z axis: 6000 mm/s^2 They can be changed with the Set DefInitAcceleration command.
DefInitSpeed	The initialization speed of the axis. Valid values are 0.1 to 200 mm/s. The default values are: ◆ X axis: 80 mm/s ◆ Y axis: 80 mm/s ◆ Z axis: 40 mm/s They can be changed with the Set DefInitSpeed command.

Example: Initialize axis 1:

```
1/1z
```

Check For Exit

Synopsis: Checks for cLLD exit signal.

Syntax: /<*AxisID*>O

AxisID: Valid values are 1-f.

Description: This command is illustrated step-by-step in Figure 6-19 on page 6-83.

The following parameters have an effect on the behavior of this command:

DefAcceleration The acceleration of the axis during any move command, except the Initialize Axis command. Valid values are 1 to 100,000 mm/s^2. The default values are:

- X axis: 3500 mm/s^2
- Y axis: 3900 mm/s^2
- Standard Z axis: 6000 mm/s^2
- Universal Z axis: 1000 mm/s^2
- Dual Z axis: 6000 mm/s^2

They can be changed with the Set DefAcceleration command.

ExitRetractSpeed The retract speed. Valid values are from 1 to 150 mm/s; the default is the 60 and can be changed with the Set ExitRetractSpeed command,.

ExitRetractDist The travel distance. Valid values are from 1 to 100 mm; the default is the 10 and can be changed with the Set ExitRetractDist command.

ExitLimit The acceptance limit for a successful exit signal check. Valid values are from 0 to 50 mm; the default is 5 and can be changed with the Set ExitLimit command.

LLDSensitivity The sensitivity in increments during exit search. Valid values are from 3 (highest sensitivity) to 120 (lowest sensitivity); the default is 63 and can be changed with the Set LLDSensitivity command.

This command moves the tip out of the liquid starting from the current position and checks for an exit signal. The exit detection is successful if the exit signal (low to high transition) appears between start of the move and the *ExitLimit.*

This command is specific to Z axes with cLLD.

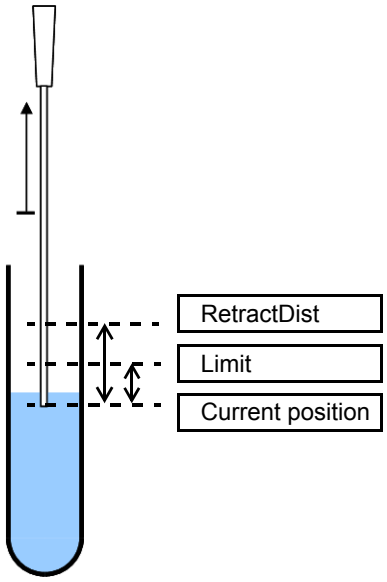
Generated Errors:

2 -- Invalid Command
3 -- Invalid Operand
5 -- Device Not Implemented
7 -- Device Not Initialized
8 -- Device Busy
9 -- Device Error (Hardware Error)
13 -- Device Disabled
18 -- cLLD Exit Limit Exceeded at Exit Detection (if the exit signal appears after the position set by *ExitLimit*)
19 -- cLLD No Exit Signal at Exit Detection
20 -- cLLD Not Ready
27 -- DiTi - No Tip Mounted

Example: Check for exit signal of Axis 3 with the default retract speed, a retract distance of 20 mm, the default limit, and a sensitivity of 63:

```
/3u27,20u22,63O
```

The following diagram (Figure 6-19) illustrates the specific actions that are taken to execute the Check For Exit command.



Actions	Move from current position by *RetractDist* using *RetractSpeed*. Checks for exit signal during entire motion. Does not stop motion upon exit signal.
Related Error Messages	**Error 20**: cLLD Not Ready (if output is already high at search begin). **Error 27**: DiTi - No tip mounted. **Error 18**: cLLD exit limit exceeded at exit detection (If the Exit signal appears after the position set by limit). **Error 19**: No exit signal at exit detection.
Notes	*RetractDist* and *Limit* are relative positions (based on current position).

Figure 6-19 *Actions taken to execute a Check For Exit command*

Clear Error

Synopsis: Clears device errors.

Syntax: /<***AxisID***>C

AxisID: Valid values are 1-f.

Description: This commands clears all errors that prevent action commands from executing on the axis, and resets the error status. These errors are listed in Table 6-4.

Example: Clear errors from axis 1

```
/1C
```

Generated Errors:	3 -- Invalid Operand 5 -- Device Not Implemented 8 -- Device Busy

Detect Liquid

Synopsis: Performs cLLD liquid detection.

Syntax: **`/<AxisID>B<LLDStartPos>,<SearchLimit>`**

AxisID: Valid values are 1-f.

LLDStartPos: Any number between *RangeLow* and *RangeHigh*

SearchLimit: Any number between *RangeLow* and *LLDStartPos*

Description: <*LLDStartPos*> and <*SearchLimit*> are required parameters and can only be set here.

This command is specific to Z axes with cLLD.

<*LLDStartPos*>	Start position for the liquid search. Valid values are from *RangeLow* to *RangeHigh*. If you are using disposable tips, the highest position should be no higher than *RangeHigh* –10 mm. For more information on the values of *RangeLow* and *RangeHigh*, see the Get Commands and Set Commands that Write ACB Configuration Variables to RAM commands.
<*SearchLimit*>	The position limit to search for liquid, starting from the position given in *LLDStartPos*. If liquid is not found during this entire travel, an error is returned. Valid values are from *LLDStartPos* to *RangeLow*. For more information on *RangeLow*, see the Get Commands command.
Generated Errors:	2 -- Invalid Command 3 -- Invalid Operand 5 -- Device Not Implemented 7 -- Device Not Initialized 8 -- Device Busy 9 -- Device Error (Hardware Error) 13 -- Device Disabled 15 -- cLLD - No Liquid Detected 16 -- cLLD - Not Enough Liquid Detected 17 -- cLLD - Liquid Detection Failure - Double Detection Acceptance Window Exceeded 20 -- cLLD Not Ready 27 -- DiTi - No Tip Mounted

The following parameters have an effect on the behavior of this command:

DefAcceleration	The acceleration of the axis during any move command, except the Initialize Axis command. Valid values are 1 to 100,000 mm/s^2. The default values are: ◆ X axis: 3500 mm/s^2 ◆ Y axis: 3900 mm/s^2 ◆ Standard Z axis: 6000 mm/s^2 ◆ Universal Z axis: 1000 mm/s^2 ◆ Dual Z axis: 6000 mm/s^2 They can be changed with the Set DefAcceleration command.
LLDAcceptanceWindow	The maximum acceptance difference between the two detection positions when using double detection. Valid values are from 0.5 to 50 mm. The default is 5 mm and can be set with the Set LLDAcceptanceWindow command.
LLDRetractDist	The Z axis retraction distance. Valid values are from *LLDAcceptanceWindow* to 100 mm. The default is 100 mm/s and can be set with the Set LLDRetractDist command.
LLDSearchSpeed	The speed to be used when searching for liquid, in mm/s. Valid values are from 1 to 150 mm/s; the default is 60 and can be changed with the Set LLDSearchSpeed command.
LLDSubmergeDist	The distance the Z axis extends after a successful detection. Valid values are from 0 to 100 mm; the default is 0 and can be changed with the Set LLDSubmergeDist command.
LLDSensitivity	The sensitivity to use when searching for liquid. Valid values are from 3 to 120; the default is 63 and can be changed with the Set LLDSensitivity command.
LLDDetectionMethod	0 for double detection; 1 for single detection. The default is 0 and can be changed with the Set LLDDetectionMethod command.
LLDRetractOnError	0 for retract on error; 1 for do not detract on error. The default is 0 and can be changed with the Set LLDRetractOnError command.

The steps described in the next paragraphs are illustrated in the diagrams (Figure 6-20 and Figure 6-21) on pages 6-87 and 6-88.

The tip presence signal of the cLLD board is checked for low state before any further action. When the Detect Liquid command is issued, the Z-drive moves to the start position <*LLDStartPos*> using the default move parameters (step A in Figure 6-20), and starts searching for liquid from that position using the speed and search sensitivity specified by *LLDSearchSpeed* and *LLDSensitivity* (step B). When the liquid surface is detected by the tip, the Z-move immediately stops (step C, Figure 6-21).

If performing a single detection (that is, *DetectionMethod* = 1), the detection is considered successful if the detection position minus the given submerge

distance, *LLDSubmergeDist,* is higher than or equal to the value of <*SearchLimit*>. In this case, steps D though F in Figure 6-21 are skipped.

If performing a double detection (that is, *DetectionMethod* = 0), the tip retracts to the distance specified by *LLDRetractDist* (step D in Figure 6-21) using the given search speed, *LLDSearchSpeed*, and starts the liquid search again (step E). It is considered successful if the second detection position minus the given submerge distance, *LLDSubmergeDist,* is higher than or equal to the value of <*SearchLimit*>, and the difference between the two detection positions is less than or equal to the acceptance window (step F) specified by *LLDAcceptanceWindow*.

If the liquid detection is successful, a subsequent down move is performed to submerge the tip, using the speed specified by *LLDSearchSpeed* (step G in Figure 6-21).

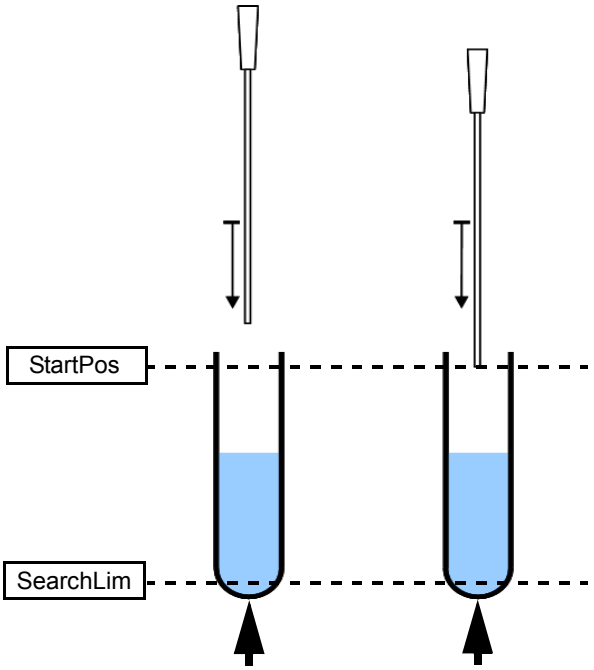
An error is generated if the limit specified in <*SearchLimit*> is reached without detecting the liquid. This command replies with an error code 0 (no error) if all Z-drives executed successfully; otherwise, the last error encountered is returned.

The *LLDRetractOnError* parameter determines if the tip remains at the current position, <*SearchLimit*>, or if it retracts to the start position using *LLDSearchSpeed* after an error is generated.

Example: Detect liquid for Axis 3 with a start position of 50 mm and a search limit of 20 mm:

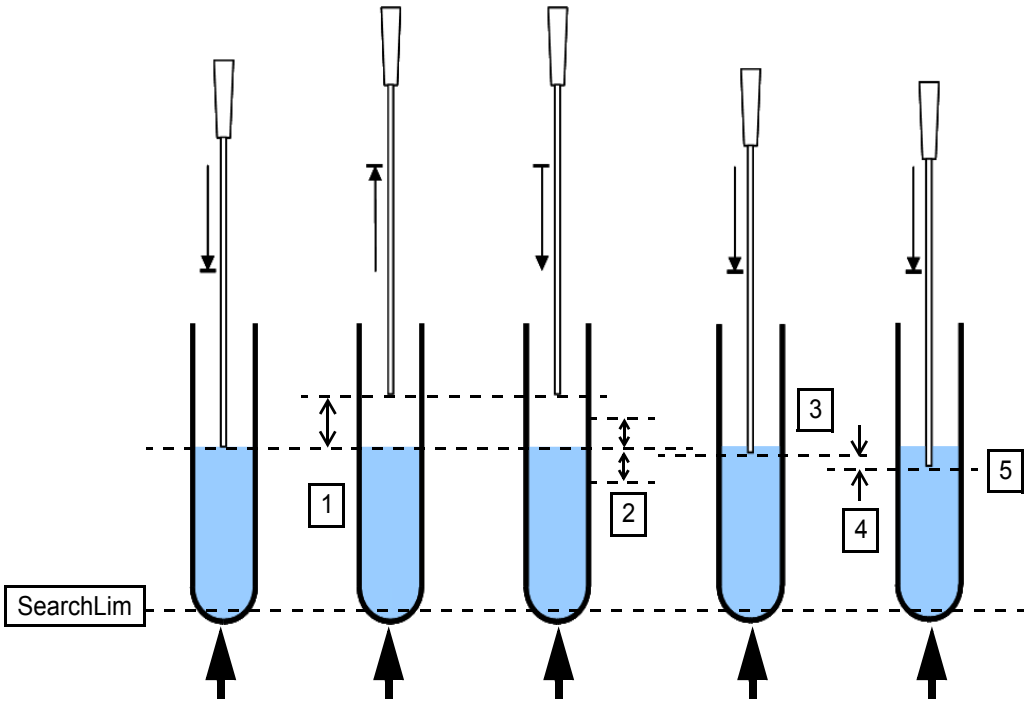
```
/3u26,1B50,20
```

The following diagrams (Figure 6-20 and Figure 6-21), show the individual actions taken to execute the Detect Liquid command. Each column in the table describes the movement illustrated in the diagram directly above it.



Actions	A) Move from current position to *LLDStartPos* with default speed	B) Start liquid search from *LLDStartPos* with *LLDSearchSpeed* towards *SearchLimit*
Related Error Messages	**Error 27**: DiTi - No tip mounted.	**Error 20**: cLLD Not Ready (if output is already high at search begin).

Figure 6-20 *Actions to execute the Detect Liquid command, part 1*



Actions	C) Move stops when the tip touches the liquid surface.	D) Upwards move relative to current position (*LLDRetract-Dist* **[1]**)	E) Re-start liquid search from current position with *LLDSearch-Speed* towards *SearchLimit*	F) Move stops when the tip touches the liquid surface. Checks if detection position **[3]** is within the Acceptance Window **[2]**	G) Additional downward move by the amount of *LLDSubmergeDist* **[4]**
Related Error Messages	**Error 15**: cLLD - No liquid detected (if *SearchLimit* was reached without detection).			**Error 15**: cLLD - No liquid detected (if *SearchLimit* reached without detection). **Error 17**: cLLD - Liquid Detection Failure - double detection acceptance window exceeded.	**Error 16**: cLLD - Not enough liquid detected (if calculated final position **[5]** would be lower than *SearchLim*.
Notes		This step is skipped if *LLDDetection-Method* = 1	This step is skipped if *LLDDetection-Method* = 1	This step is skipped if *LLDDetection-Method* = 1	

Figure 6-21 *Actions to execute the Detect Liquid command, continued (part 2)*

Liquid Level Detection with 8-Channel Pipetting Head

Only applications with equal fill levels qualify for 8-channel head liquid level detection. It is the user's / programmer's responsibility to set the submerge distance appropriately to overcome minor differences between the individual fill levels and / or tolerances of the 8-channel head.

Caution! *When using the 8-channel pipetting head, users should be aware that the first probe making contact with liquid will cause the Z axis to stop moving. If the containers being used have uneven liquid fill levels, liquid level detection may have unexpected results.*

In addition, other factors may affect the ability of the 8-channel pipetting head to accurately detect liquid levels. For best results, check the following factors:

Levelness of Robotics

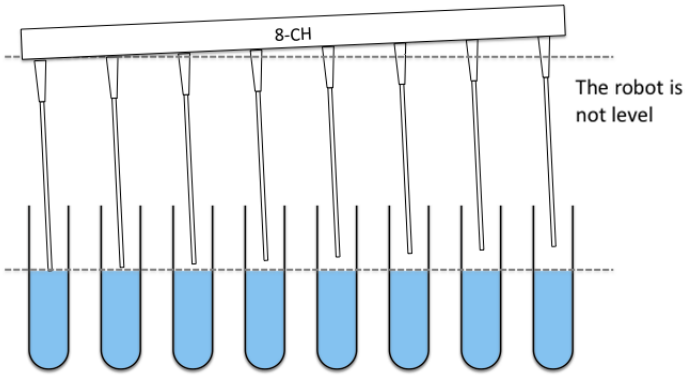


Figure 6-22 *Impact of robot not leveled properly*

Levelness of Labware

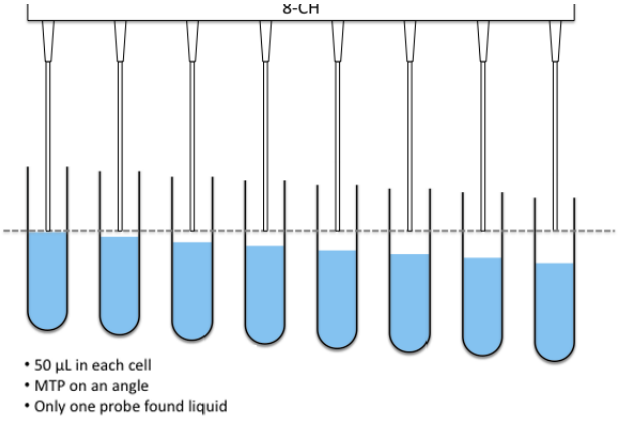


Figure 6-23 *Impact of uneven labware*

Uneven Liquid Levels

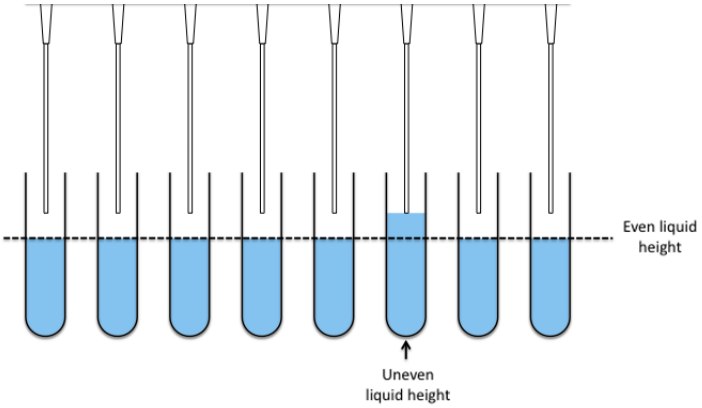


Figure 6-24 *Impact of uneven liquid levels*

Drop Tip

Synopsis: Drops a disposable tip.

Syntax: **`/<AxisID>E[DropStartPos],[DropLimit],[DropAcceptanceLimit]`**

AxisID: Valid values are 1-f.

DropStartPos: Any number between *RangeLow* and *RangeHigh minus 10*

DropLimit: Any number between *DropStartPos* and *RangeHigh plus 50*

DropAcceptanceLimit:
Any number between *DropStartPos* and *DropLimit*

Description: This command drops a disposable tip from the current position.

[DropStartPos], [DropLimit] and [*DropAcceptanceLimit]* can only be set here. If they are not supplied as arguments to the Drop Tip command, the default values are used. Either all three arguments must be supplied, or all three can be omitted to use the default settings.

This command is specific to Z axes with cLLD.

[DropStartPos]	The Z-drive position, in mm, for the beginning of the drop tip action. Valid values are from *RangeLow* to (*RangeHigh* – 10 mm); the default value is *AxisOffset* – 20 mm. For more information on the values of *RangeLow*, *RangeHigh*, and *AxisOffset*, see the Get Commands command.
[DropLimit]	The position limit for the drop search. Valid values are from *DropStartPos* to (*RangeHigh* + 50 mm); the default value is *AxisOffset* + 20 mm. For more information on the value of *RangeHigh*, see the Get Commands command.
[DropAcceptanceLimit]	The limit for a valid stop position. Valid values are from *DropStartPos* to the value of *DropLimit*; the default is the value of *AxisOffset*.
Generated Errors:	2 -- Invalid Command 3 -- Invalid Operand 5 -- Device Not Implemented 7 -- Device Not Initialized 8 -- Device Busy 9 -- Device Error (Hardware Error) 13 -- Device Disabled 22 -- DiTi - Tip Not Dropped (If the Tip Presence Signal State Remains High) 23 -- DiTi - Tip Drop Error (If the Stop Occurs Before the *AcceptanceLimit*) 24 -- DiTi - Hard Stop Not Found (If Limit is Reached) 27 -- DiTi - No Tip Mounted

The following parameters have an effect on the behavior of this command:

DefAcceleration — The acceleration of the axis during any move command, except the Initialize Axis command. Valid values are 1 to 100,000 mm/s^2. The default values are:

- X axis: 3500 mm/s^2
- Y axis: 3900 mm/s^2
- Standard Z axis: 6000 mm/s^2
- Universal Z axis: 1000 mm/s^2
- Dual Z axis: 6000 mm/s^2

They can be changed with the Set DefAcceleration command.

DropSpeed — The travel speed used by the Standard Z or Dual Z axis to drop a disposable tip. Valid values are from 1 to 150 mm/s; the default is 60 and can be changed with the Set DropSpeed command.

DropForceFactor — The force used to drop a disposable tip. Valid values are from 8000 to 22000; the default is 17000 and can be changed with the Set DropForceFactor command.

The steps described in the next paragraphs are illustrated in the diagram (Figure 6-25) on page 6-101.

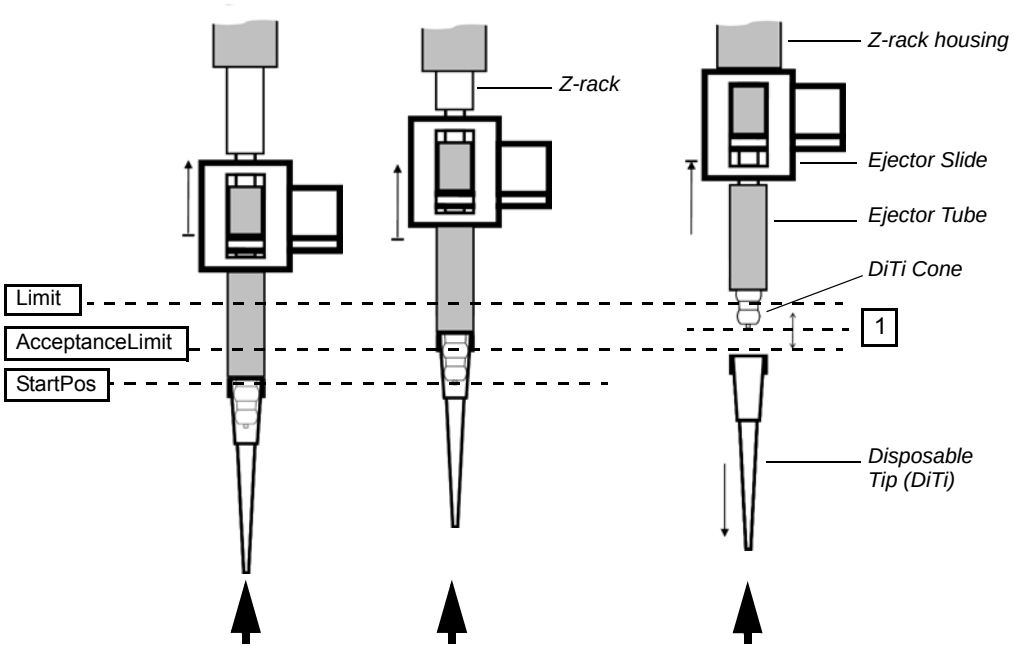
The tip presence signal of the cLLD board is checked for low state prior to any further action. The Z-drive moves to the start position, <*DropStartPos*> using regular move parameters (Step A in *Figure 6-25*). Then it starts the search function from the current XY position (Step B) by moving up at the given speed and force (*DropSpeed* and *DropForceFactor*) until the drive is stopped by hitting the hard stop (Step C). An error is generated if the Z-drive does not reach the stop position limit (specified by <*DropAcceptanceLimit*>) or if it reaches the drop position limit (specified by <*DropLimit*>). The Z-drive retracts to <*DropStartPos*> using regular move parameters and checks for high state of the tip presence signal.

Note: *This command is specific to Z axes with cLLD. The Drop Tip command is not yet supported for the Universal Z axis.*

Example: Drop a disposable tip from Axis 3 with a start position of 200 mm, a search limit of 220 mm, an acceptance limit of 210, the default speed, and a force factor of 1500:

```
/3u13,1500E200,220,210
```

Figure 6-25 shows the specific actions taken to execute the Drop Tip command. Each step (column) in the table describes the movement illustrated in the diagram directly above it.



Intended Actions	A) Move from current position to *DropStartPos* with default speed parameters.	B) Start hard stop search from *DropStartPos* with *DropSpeed* towards *DropLimit.*	C) The ejector slide hits the z-rack housing and continues the move. The ejector slide is pushed down with the force set by *DropForceFactor*. The force is transmitted through the ejector tube and results in tip ejection. The axis is forced to stop when the ejector slide is pushed all the way down. The stop position **[1]** should occur between *DropAcceptanceLimit* and *DropLimit.*
Related Error Messages	**Error 27**: DiTi - No Tip mounted (if the tip presence signal state is high prior to the search).		**Error 23**: DiTi - Tip Drop error (if the stop occurs below *DropAcceptanceLimit*. **Error 22**: DiTi - Tip not dropped (if the tip presence signal state remains high after the axis has stopped). **Error 24**: DiTi - Hard stop not found (if *DropLimit* is reached).

Figure 6-25 *Actions taken to execute the Drop Tip command*

Enable / Disable

Synopsis: Enables or disables motor current/drive.

Syntax: /<*AxisID*>N<*Value*>

AxisID: Valid values are 1-f.

Values: 0 disables the axis; 1 enables the axis

Description: This command disables the motor current/drive.

The Disable command cannot be sent while the axis is in motion.

WARNING! *For an axis with a brake, this command may trigger the axis to drop.*

Generated Errors:
3 -- Invalid Operand
5 -- Device Not Implemented
8 -- Device Busy

Example: Enable axis 1:

```
/1N1
```

Disable axis 1:

```
/1N0
```

Move Axis to Absolute Position

Synopsis: Moves a single axis to an absolute position.

Syntax: /<*AxisID*>A<*TargetPos*>

AxisID: Valid values are 1-f.

TargetPos: The coordinate of the axis, in mm. The valid range is from RangeLow to RangeHigh. For more information on the values of RangeLow and RangeHigh, see Get RangeLow and Get RangeHigh, respectively.

Description: This command moves the axis to an absolute position, using the speed (*DefSpeed*) assigned with the Set DefSpeed command.

Generated Errors:
3 -- Invalid Operand
5 -- Device Not Implemented
7 -- Device Not Initialized
8 -- Device Busy
9 -- Device Error (Hardware Error)
13 -- Device Disabled

The following parameters have an effect on the behavior of this command:

DefSpeed — The default axis movement speed. Valid values are 0.1 to 10000 mm/s. The default values are:

- X axis: 800 mm/s
- Y axis: 600 mm/s
- Standard Z axis: 600 mm/s
- Universal Z axis: 250 mm/s
- Dual Z axis: 600 mm/s

They can be changed with the Set DefSpeed command.

DefAcceleration — The acceleration of the axis during any move command, except the Initialize Axis command. Valid values are 1 to 100,000 mm/s^2. The default values are:

- X axis: 3500 mm/s^2
- Y axis: 3900 mm/s^2
- Standard Z axis: 6000 mm/s^2
- Universal Z axis: 1000 mm/s^2
- Dual Z axis: 6000 mm/s^2

They can be changed with the Set DefAcceleration command.

Example: Move axis 1 to absolute position 100.54

```
/1A100.54
```

Move Axis to Absolute Position with High Force

Synopsis: Moves the specified axis to an absolute axis coordinate position using specified speed parameter and DefAcceleration.

Syntax: /<*AxisID*>F<*TargetPos*>

AxisID: Valid values are 1-f.

TargetPos: The coordinate of the axis, in mm. The valid range is from RangeLow to RangeHigh. For more information on the values of RangeLow and RangeHigh, see Get RangeLow and Get RangeHigh, respectively.

Description: This command moves the axis to the given coordinate positions, and is applicable only to standard Z axis types.

Generated Errors:
2 -- Invalid Command
3 -- Invalid Operand
5 -- Device Not Implemented
7 -- Device Not Initialized
8 -- Device Busy
9 -- Device Error (Hardware Error)
13 -- Device Disabled

The following parameter has an effect on the behavior of this command:

DefAcceleration — The acceleration of the axis during any move command, except the Initialize Axis command. Valid values are 1 to 100,000 mm/s^2. The default values are:

- X axis: 3500 mm/s^2
- Y axis: 3900 mm/s^2
- Standard Z axis: 6000 mm/s^2
- Universal Z axis: 1000 mm/s^2
- Dual Z axis: 6000 mm/s^2

They can be changed with the Set DefAcceleration command.

MoveAbsHiForceSpeed — The value of the speed setting for the Move Axis to Absolute Position with High Force command. Valid values are 0.1 to 10000 mm/sec. The default is 40 mm/sec. This can be changed with the Set MoveAbsHiForceSpeed command.

Example: Move the axis to position 50.5:

```
/3F50.5
```

Move the axis to position 50.5 with a speed of 100: *

```
/3u33,100F50.5
```

Move Axis to Relative Position Before Initialization

Synopsis: Moves a single axis to a relative position before initialization.

Syntax: /<***AxisID***>I<***offset***>

AxisID: Valid values are 1-f.

Description: This command moves the specified axis to a relative axis offset position at a fixed speed of 25 mm/s. Motion is stopped if a hard-stop is hit. This command will only be accepted when the axis is ***not*** initialized. This command only works with X and Y axes.

Caution! Because the offset value can not be range limited by the software, the user is responsible for selecting a value which will not result in a collision with obstacles on the work table or the axis hard-stops.

<Offset>	The relative offset to current position of each axis, definable by user. Valid values are numeric units in mm.
Generated Errors:	2 -- Invalid Command 3 -- Invalid Operand 5 -- Device Not Implemented 8 -- Device Busy 9 -- Device Error (Hardware Error) 13 -- Device disabled 101- PreInitMoveRel move blocked 102 - PreInitMoveRel not allowed at current state 103 - PreInitMoveRel encoder limit switch error

Example: Move axis 1 to relative position -2000

```
/1I-2000
```

Pick Tip

Synopsis: Picks up a disposable tip.

Syntax: **`/<AxisID>P<StartPos>,<Limit>,<AcceptanceLimit>`**

Description: This command picks up a disposable tip.

The steps described in the next paragraph are illustrated Figure 6-26 on page 6-100. Error conditions that may occur are shown in Figure 6-27.

<StartPos>, *<Limit>*, and *<AcceptanceLimit>* are required arguments and can only be set here.

This command is specific to Standard Z, Dual Z, and Universal Z axis with ADP only.

<StartPos>	The Z-drive position, in mm, for the beginning of the pick tip action. Valid values are from *RangeLow* to (*RangeHigh* – 10 mm). For more information on the values of *RangeLow* and *RangeHigh*, see the Get RangeLow and Get RangeHigh commands.
<Limit>	The Z distance to travel, in mm, when searching for a disposable tip before reporting an error. Valid values are from *StartPos* to *RangeLow*.
<AcceptanceLimit>	The limit for a valid stop position. Valid values are from *StartPos* to *PickLimit*.
Generated Errors:	2 -- Invalid Command 3 -- Invalid Operand 5 -- Device Not Implemented 7 -- Device Not Initialized 8 -- Device Busy 9 -- Device Error (Hardware Error) 13 -- Device Disabled 25 -- DiTi - Tip Not Fetched 26 -- DiTi - Tip Crash (If the Stop occurs before AcceptanceLimit) 28 -- DiTi - Tip Already Mounted (If the tip presence signal state is low prior to the search) 29 -- DiTi - Tip Not Found 30 -- DiTi - Tip Not Mounted Properly

The following parameters have an effect on the behavior of this command:

DefAcceleration	The acceleration of the axis during any move command, except the Initialize Axis command. Valid values are 1 to 100,000 mm/s^2. The default values are: ◆ X axis: 3500 mm/s^2 ◆ Y axis: 3900 mm/s^2 ◆ Standard Z axis: 6000 mm/s^2 ◆ Universal Z axis: 1000 mm/s^2 ◆ Dual Z axis: 6000 mm/s^2 They can be changed with the Set DefAcceleration command.
PickSpeed	The travel speed used by the Z axis to pick up a disposable tip. Valid values are from 1 to 150 mm/s; the default is 60 and can be changed with the Set PickSpeed command.
PickForceFactor	The force used to engage a disposable tip during pickup. Valid values are from 1000 to 20000; the default is 11800 for the Standard or Dual Z and 7000 for the Universal Z. It can be changed with the Set PickForceFactor command.

This command picks up a disposable tip. The tip presence signal of the cLLD board is checked for high state prior to any further action. The Z-drive moves to

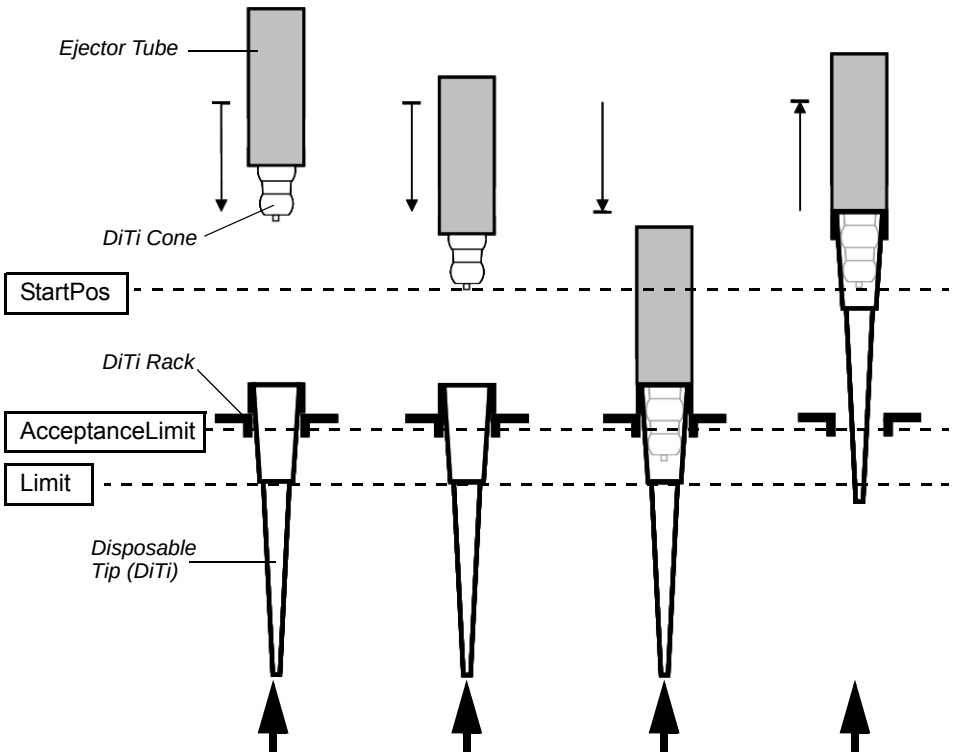
the start position, <*StartPos*>, using regular move parameters (Step A in Figure 6-26). Then it starts the pick function from the current XY position (Step B) by moving down with the given speed and force (specified with the Set PickSpeed and Set PickForceFactor commands for that axis) until the drive is stopped by hitting the tip (Step C). An error is generated if the Z-drive does not reach the stop position limit (specified by <*AcceptanceLimit*>) or if it reaches the drop position limit (specified by <*Limit*>). The Z-drive retracts to <*StartPos*> using regular move parameters and checks for the low state of the tip presence signal (Step D).

Caution! *For a Dual Z axis, there should be a minimum 0.5 second time gap between each axis' **Pick Tip** to ensure that they will not both engage the tips at the same time.*

Example: Pick up a disposable tip from Axis 3 with a start position of 50 mm, a search limit of 20 mm, an acceptance limit of 20.5, the default pick speed, and a force factor of 1200:

```
/3u38,1200P50,20,20.5
```

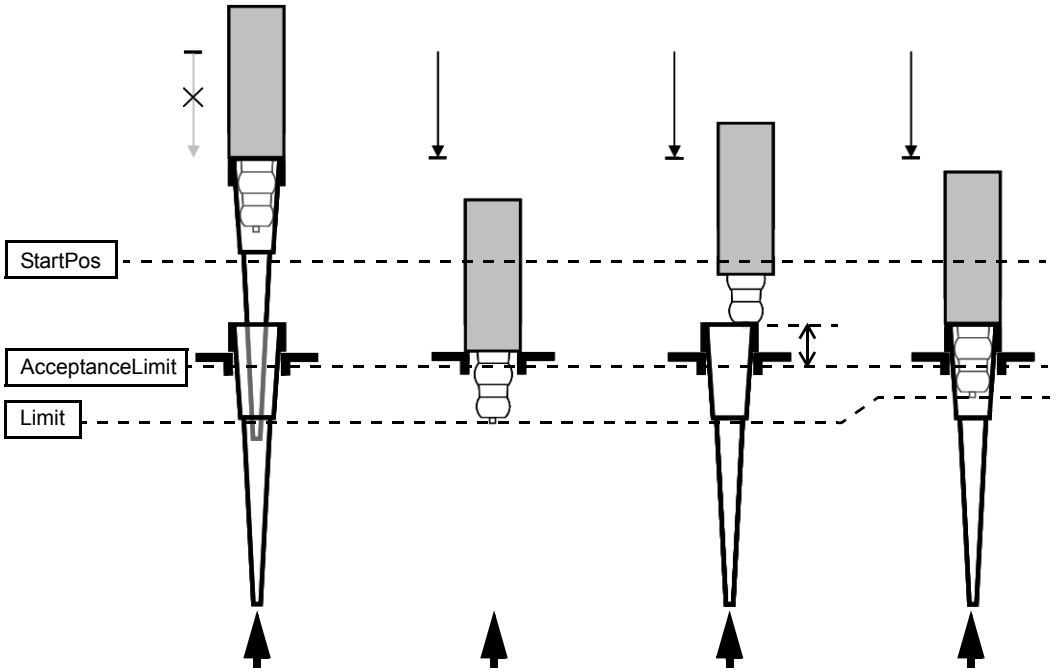
Figure 6-20 shows the specific actions taken to execute the Pick Tip command. Each step (column) in the table describes the movement illustrated in the diagram directly above it.



Actions	A) Move from current position to *StartPos* with default speed parameters.	B) Start tip search from *StartPos* with *PickSpeed* towards *Limit.*	C) DiTi cone engages in the tip with a force set by *PickForceFactor.* The Z axis move is stopped at a position below *AcceptanceLimit* but above *Limit.*	D) Upwards move back to *StartPos*.
Related Error Messages	**Error 28**: DiTi - Tip already mounted (if the tip presence signal state is low prior to the search).		**Error 26**: DiTi - Tip crash (if the stop occurs above *AcceptanceLimit*. **Error 29**: DiTi - Tip not found (if the Z axis has reached *Limit*.	**Error 30**: DiTi - Tip not properly mounted (if the Z axis reached *Limit* in step C) and the tip got mounted).

Figure 6-26 *Actions taken to execute the Pick Tip command, normal operation*

Figure 6-27 describes the error conditions that may occur with the Pick Tip command. The steps refer to the actions described in Figure 6-26.



Intended Actions	Move from current position to *StartPos* with default speed parameters (step A).	DiTi cone engages in the tip with a force set by *PickForceFactor*. The Z axis move is stopped below *AcceptanceLimit* but above *Limit* (step C).	DiTi cone engages in tip with a force set by *PickForceFactor*. The Z axis move is stopped below *AcceptanceLimit* but above *Limit* (step C).	DiTi cone engages in the tip with a force set by *PickForceFactor*. The Z axis move is stopped below *AcceptanceLimit* but above *Limit* (steps C/D).
Error Condition	A tip was already mounted and not discarded prior to this operation.	The axis reaches *Limit* because there is no tip at the current position.	Inappropriate X/Y position results in crash above *AcceptanceLimit*.	*Limit* is reached during tip search due to inappropriate *Limit* (too high). The tip can be mounted or not, depending on the pick force.
Resulting Action	The axis does not move to start position, and responds with **Error 28**: DiTi - Tip already mounted.	Axis retracts to *StartPos* and responds with **Error 29:** DiTi - Tip not found.	Axis stops, retracts to StartPos, and responds with **Error 26**: DiTi - Tip crash.	Axis stops, retracts to *StartPos*, and responds with either **Error 29**: DiTi - Tip not found, or **Error 30**: Tip not properly mounted (tip is mounted.

Figure 6-27 *The Pick Tip command, error conditions*

Terminate Axis and Gripper Motion

Synopsis: Terminates the axis and gripper motion immediately.

Syntax: **`/<AxisID>T`**

Description: This command terminates the motion of the axis. The motor remains enabled after ramping down according to the most recent acceleration parameters. No action takes place if the command is sent to an axis in an idle state or if the axis is not initialized. If this command is used to terminate gripper motion, the gripper state changes to 3 - Gripper Not Initialized.

Generated Errors:
3 -- Invalid operand
5 -- Device Not Implemented
9 -- Device Error (Hardware Error)

Example: Terminate motion of device 1 on string id1:

```
1/1T
```

6.10 Gripper Commands

This section provides reference information for Omni Robot software commands for the gripper.

Caution! *Don't send commands to the gripper which are not intended for execution with a gripper, because this could cause unintended motion of the gripper.*

WARNING! *There is a potential risk of collision during the initialization sequence. Take precautions to avoid collisions.*

WARNING! *Input a collision avoidance number larger than the size of the largest gripper with its maximum load, based on your system configuration and/or gripper finger type/orientation.*

WARNING! *If there are two grippers, there is a risk of collision. The collision avoidance number needs to be larger than the size of the larger gripper with its maximum load.*

WARNING! *Confirm that the gripper is mounted in the correct orientation and with the correct position offset. If you remove the gripper and replace it in a different mounting position, you must tell the software about the new position. The software cannot detect changes in gripper mounting position.*

Table 6-13 *Gripper commands*

Description	Embedded Command	See Page
Close Gripper: Closes the gripper.	**/<*AxisID*>W1**	6-103
Initialize Gripper: Initializes the gripper.	**/<*AxisID*>W0**	6-103
Open Gripper: Opens the gripper.	**/<*AxisID*>W2**	6-104
Terminate Gripper Motion: Stops gripper motion.	**/<*AxisID*>T**	6-105

See Section 6.6.2, "ACB and Gripper Event Codes" for a list of gripper events.

Close Gripper

Synopsis: Closes the gripper.

Syntax: **/<AxisID>W1**

AxisID: Valid values are 1-f.

Description: Performs the gripping operation. If an object is NOT detected after close operation is performed, an error code 46 - Gripper no object detected will be returned.

Generated Errors:
2 -- Invalid Command
3 -- Invalid Operand
5 -- Device Not Implemented
8 -- Device Busy
9 -- Device Error (Hardware Error)
40 -- Gripper Not Attached
42 -- Gripper Not Initialized
43 -- Gripper Move Error
46 -- No Object Detected

Initialize Gripper

Synopsis: Initializes the gripper.

Syntax: **/<AxisID>W0**

AxisID: Valid values are 1-f.

Description: Checks for object presence prior to init and if object is detected, the function will not perform an initialization. The function will also not initialize if the CAM is in an intermediate position (neither fully closed nor opened).

Initializing the gripper must be done by calling this command. The Initialize Axis command will not initialize the gripper.

Generated Errors:
2 -- Invalid Command
3 -- Invalid Operand
5 -- Device Not Implemented
8 -- Device Busy
9 -- Device Error (Hardware Error)
40 -- Gripper Not Attached
43 -- Gripper Move Error
44 -- Gripper CAM at Intermediate Starting Position
45 -- Gripper Object Detected
48 -- Gripper Sensor Error

The following parameter has an effect on the behavior of this command:

GripperInitMode — The initialization mode of the gripper. Valid values are:

- 0: Normal init, closed
- 1: Alternate init, closed
- 2: Normal init, open
- 3: Alternate init, open

The default value is 0. The value can be changed with the Set GripperInitMode command.

Open Gripper

Synopsis: Opens the gripper.

Syntax: **`/<AxisID>W2`**

AxisID: Valid values are 1-f.

Description: Opens the gripper fingers.

Generated Errors:
2 -- Invalid Command
3 -- Invalid Operand
5 -- Device Not Implemented
8 -- Device Busy
9 -- Device Error (Hardware Error)
40 -- Gripper Not Attached
42 -- Gripper Not Initialized
43 -- Gripper Move Error

Terminate Gripper Motion

Synopsis: Terminating gripper motion can be performed by issuing the Terminate Axis and Gripper Motion command.

6.10.1 Returning to Home Position With the Gripper Attached

Under normal circumstances, the program controlling the Omni should return the gripper to home position (or another chosen safe position) when it is not in use. The choice of using home position or another designated “Safe” position is at the programmers discretion.

If there has been an event such as a power failure, the program controlling the gripper may not be able to return the gripper to home position because the Omni may no longer have information about the size and location of attached fingers on the gripper, or the gripper payload. It is also possible when power is lost that the gripper might move to a position from which it is blocked from moving to a safe position during the normal initialization process. In either case, the gripper can be returned to home using a different sequence of commands that allows the program to more slowly feel its way back to a safe position moving in smaller increments. When the gripper has returned to a safe position, proper initialization (which is required for operation) can occur.

The following example illustrates how to help the Omni return to home position when the gripper is attached, with eccentric (L-shaped) fingers mounted pointing toward the X axis. In this example, the gripper fingers are located underneath the X axis, preventing normal initialization by mechanical interference, as shown in Figure 6-28.

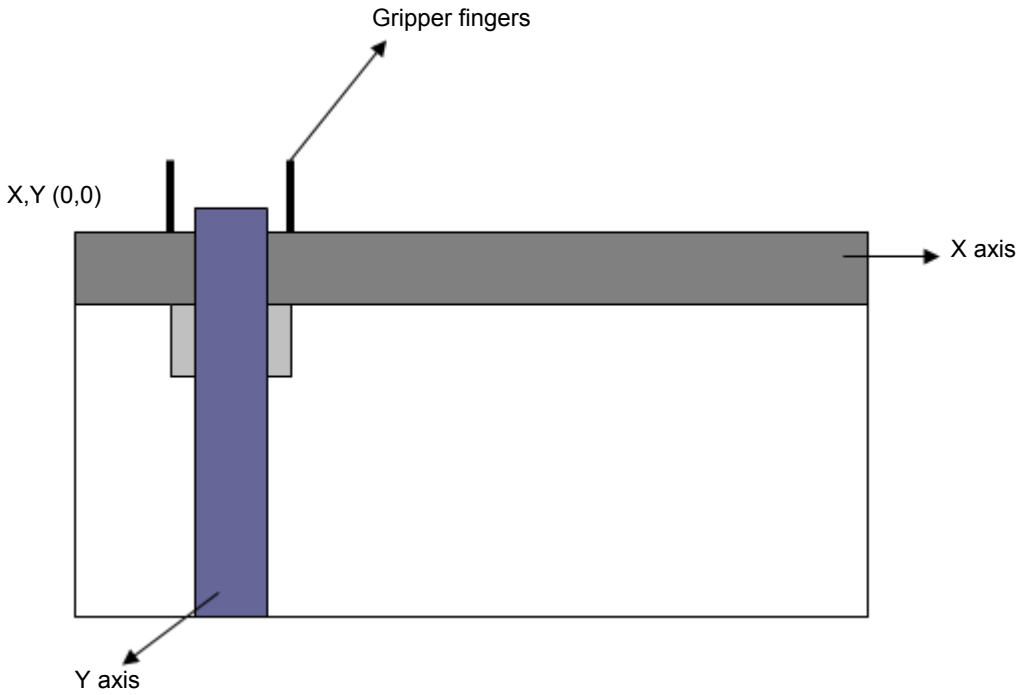


Figure 6-28 *Gripper with fingers underneath X axis*

6.10.2 Script Excerpt Example: Initialize arm procedure

Note: *This example script does not handle possible errors that could occur for each of the commands. See Appendix F, "Embedded Command and Error Quick Reference Guide" for a list of possible errors for each command. If an error occurs, it is returned when the command is executed.*

Note: *This is not a complete script. It uses script excerpts to illustrate the steps in this procedure.*

Note: *The move parameter values shown in these examples are for illustration only. They should be replaced with values appropriate for each application.*

Note: *The Move Axis to Relative Position Before Initialization command operates at a slow fixed speed and is used to move one X or Y axis at a time. Choose the appropriate axis to start with for your scenario.*

Note: *For the re-homing in this sample script, one axis at a time is initialized. This maximizes the ability to perform each initialization safely.*

1 Use the Move Axis to Relative Position Before Initialization command to guide axes out to a safe position where the fingers are clear of the X axis and all axes can return home freely.

2 Move the gripper on the Y axis to a safe position where the Universal Z axis can initialize freely. (See Figure 6-29)

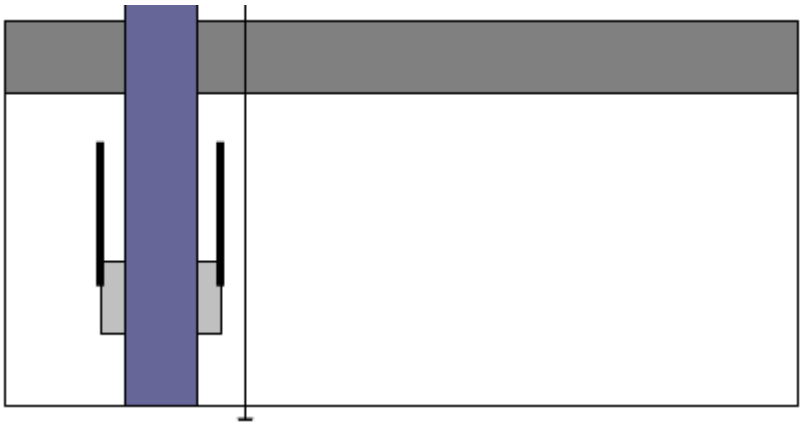


Figure 6-29 *Gripper moved to safe position*

```
//Relative move, 150 mm away from current position in positive
//direction
//Assume that 150 mm clears the finger without colliding with
//the X-extrusion
 //Assume that X axis is at ID1, Y axis is at 2,
 //and Z axis is at 3
 1/2I150
```

3 Initialize the Universal Z axis, retracting upward.

```
1/3Z
```

4 Initialize the X axis. (See Figure 6-30)

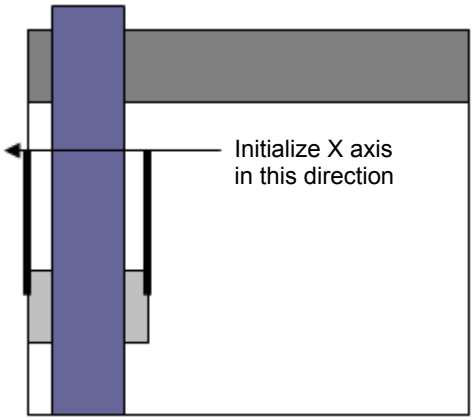


Figure 6-30 *Initialize X axis*

```
1/1Z
```

5 Move the Universal Z axis down to a safe position where it will not collide with the X axis body when the Y axis initializes.

```
//Assuming 150 mm will bring the gripper to safe position
//for initialization on Y axis.
1/3A150
```

6 Initialize theY axis. (See Figure 6-31)

```
1/2Z
```

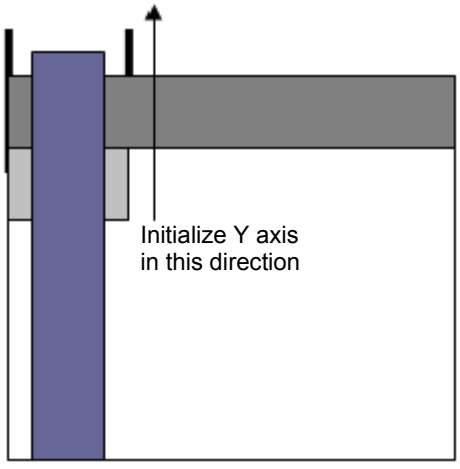


Figure 6-31 *Initialize Y axis*

Note: *The X axis and Y axis are now at home position, but the Universal Z axis is not. It has to move down away from home position so that the Y axis can return to home position. The important thing is that all axes now have been initialized and are ready for use.*

7 The application program can now initialize the gripper and other system peripherals.

6.10.3 Script Excerpt Example: Pick up and drop a plate

The following script excerpt example illustrates the commands that control picking up and dropping a plate:

```
//Goto position where a plate is located and where the
//fingers can grab the object
//Assuming X/Y/Z coords 300,150,150 is the position

//Go to X/Y coord
1(/1A300/2A150)

//Open the gripper
1/3W2

//Go down to Z-coord where the gripper can grab the plate
1/3A150

//Grabbing plate
1/3W1

//Retracting Universal Z
1/3A200

//Move to different X/Y coord
1(/1A200/2A100)

//Universal Z goes down to a position where a plate can
//be safely placed.
1/3A150

//Placing a plate by releasing the gripper
1/3W2

//Retracting Universal Z
1/3A200
```

6.11 CIB Commands

Unlike Cavro® devices, executing internal commands for the CIB does not require the R command. Commands are executed immediately when they are received. To be consistent with Cavro® devices, the R command can be appended to the end of a command, but it has no effect. For more information on the R command, see the manual for your Cavro® device.

6.11.1 Get Commands

Synopsis: The F?<n> command gets the value of the ***Var_ID***.

Syntax: /F?<***Var_ID***>

Var_ID: The following table provides a list of Var_IDs and the variables they correspond to:

Table 6-14 *CIB Get command arguments*

Var ID	Name	See Page
0	IO1	6-115
1	IO2	6-116
2	Input1	6-113
3	Input2	6-113
4	Input3	6-114
5	Input4	6-114
6	Input5	6-114
7	Input7	6-115
8	Output3	6-118
9	Output4	6-118
10	Output1	6-117
11	Output2	6-117
12	Input6	6-114
13	IO1Mode	6-115
14	IO2Mode	6-116

Var ID	Name	See Page
25	BootVersion	6-111
30	VoltageFU1	6-120
31	VoltageFU2	6-121
32	VoltageFU3	6-121
33	VoltageFU4	6-121
34	VoltageV3.3D	6-121
35	VoltageV1.8D	6-122
36	VoltageTPPO	6-122
40	MACAddress	6-117
41	IP Address	6-116
42	PortNumber: Command Processor	6-118
44	PortNumber: Embedded	6-119
50	Power On Minutes	6-119
51	Absolute Power On Minutes	6-119
52	Power Ups	6-120

Table 6-14 CIB Get command arguments (continued)

Var ID	Name	See Page
23	FWRevision	6-112
24	FWVersion	6-113

Var ID	Name	See Page
53	Absolute Power Ups	6-120
76	CIB configuration	6-111

The following error can be generated by the **?<*n*>** command:

Generated Error: 3 - Invalid Operand.

Get BootVersion

Synopsis: The boot code version

Values: Valid version number

Syntax: **`/F?25`**

Description: BootVersion is a read-only parameter and its value cannot be changed.

Example: Get the version number of the boot code for the CIB:

```
/F?25
```

Get CIB Configuration

Synopsis: ASCII text report of CIB configuration

Values: ASCII text string

Syntax: **`/F?76`**

Description: CIB Configuration is a read-only parameter and its value cannot be changed. The CIB can be configured with the commands described in the sections Set Commands That Write CIB Configuration Variables to RAM and Set Commands That Write CIB Configuration Variables to Non-Volatile Memory.

The format of the response to this command is as follows, with each field separated by a '|' character:

```
'field1|field2|field3|field4|field5|field6|field7|field8'
```

Fields and their values are shown in Table 6-15, Response format fields and values.

Table 6-15 *Response format fields and values*

Field #	Category	ASCII text	Description
1	RS-232 baud rate	'RS232-9600	'9600 bps
		'RS232-38400	'38400 bps
2	RS-485 baud rate	'RS485-9600	'9600 bps
		'RS485-38400	'38400 bps
3	CAN0 baud rate	'CAN0-100K	'100 Kbps
		'CAN0-125K	'125 Kbps
		'CAN0-250K	'250 Kbps
		'CAN0-500K	'500 Kbps
4	CAN1 baud rate	'CAN1-100K	'100 Kbps
		'CAN1-125	'125 Kbps
		'CAN1-250K	'250 Kbps
		'CAN1-500K	'500 Kbps
5	Device protocol	'OEM	'OEM serial communication protocol only
6	Device error termination type	'ERR_GlobalT	'Reflects u15,1 setting
		'ERR_CmdT	'Reflects u15,0 setting
7	Device event termination type	'EVNT_GlobalT	'Reflects u16,1 setting
		'EVNT_CmdT	'Reflects u16,0 setting
8	INT5 trigger type	'INT5_EVNT	'Reflects u17,0 setting
		'INT5_AxesT	'Reflects u17,1 setting

Example: Get the CIB configuration:

```
1/F?76
```

Get FWRevision

Synopsis: The CIB firmware part number and revision number

Values: Valid firmware part and revision number

Syntax: **`/F?23`**

Description: FWRevision is a read-only parameter and its value cannot be changed.

Example: Get the firmware part number and revision:

```
/F?23
```

Get FWVersion

Synopsis: The CIB firmware version number

Values: Valid CIB firmware version number

Syntax: `/F?24`

Description: FWVersion is a read-only parameter and its value cannot be changed.

Example: Get the CIB firmware version number:

```
/F?24
```

Get Input1

Synopsis: Status of Input 1 (J9, pin 13, and J8, pin 10)

Values: 0 or 1

Syntax: `/F?2`

Description: Input1 is a read-only parameter and its value cannot be changed.

Example: Get the status of Input1:

```
/F?2
```

Get Input2

Synopsis: Status of Input 2 (J9, pin 14, and J8, pin 12)

Values: 0 or 1

Syntax: `/F?3`

Description: Input2 is a read-only parameter and its value cannot be changed.

Example: Get the status of Input2:

```
/F?3
```

Get Input3

Synopsis: Status of Input 3 (J9, pin 15, and J8, pin 14)

Values: 0 or 1

Syntax: `/F?4`

Description: Input3 is a read-only parameter and its value cannot be changed.

Example: Get the status of Input3:

```
/F?4
```

Get Input4

Synopsis: Status of Input 4 (J4, pin 6)

Values: 0 or 1

Syntax: `/F?5`

Description: Input4 is a read-only parameter and its value cannot be changed.

Example: Get the status of Input4:

```
/F?5
```

Get Input5

Synopsis: Status of Input 5 (J5, pin 6)

Values: 0 or 1

Syntax: `/F?6`

Description: Input5 is a read-only parameter and its value cannot be changed.

Example: Get the status of Input5:

```
/F?6
```

Get Input6

Synopsis: Status of Input 6 (J18, pin 5)

Values: 0 or 1

Syntax: `/F?12`

Description: Input6 is a read-only parameter and its value cannot be changed.

Example: Get the status of Input6:

```
/F?12
```

Get Input7

Synopsis: Status of Input 7/INT5 (J17, pin 4)

Values: 0 or 1

Syntax: `/F?7`

Description: Input7 is a read-only parameter and its value cannot be changed.

Example: Get the status of Input7:

```
/F?7
```

Get IO1

Synopsis: The status of I/O1 (J4, pin 5)

Values: 0 or 1

Syntax: `/F?0`

Description: Reflects pin state if /O1 is acting as input (*I/O1Mode* = 2); otherwise reflects data bit set in Data Register.

To set, see the Set IO1 command

Example: Get the I/O 1 status:

```
/F?0
```

Get IO1Mode

Synopsis: I/O direction for I/O1 (J4, pin 5)

Values: Numbers that represent Input or Output mode

Syntax: Get: `/F?13`

Description: I/O1 is bidirectional, and can be Input or Output. Default is Input. Valid values are:

- 0: Input
- 1: Output

To set, see the Set IO1Mode command. Note that the values for Input and Output are different for the Set IO1Mode command.

Example: Get the I/O 1 direction:

```
/F?13
```

Get IO2

Synopsis: The status of I/O2 (J5, pin 5)

Values: 0 or 1

Syntax: **`/F?1`**

Description: Reflects pin state if /O1 is acting as input (*I/O1Mode* = 2); otherwise reflects data bit set in Data Register.

To set, see the Set IO2 command

Example: Get the I/O 2 status:

```
/F?1
```

Get IO2Mode

Synopsis: I/O direction for I/O2 (J5, pin 5)

Values: Numbers that represent Input or Output mode

Syntax: **`/F?14`**

Description: I/O2 is bidirectional, and can be Input or Output. Default is Input. Valid values are:

- 0: Input
- 1: Output

To set, see the Set IO2Mode command. Note that the values for Input and Output are different for the Set IO2Mode command.

Example: Get the I/O 2 direction:

```
/F?14
```

Get IP Address

Synopsis: The IP address of the CIB

Values: A valid IP address

Syntax: **`/F?41`**

Description: The address must be a valid IP address in the form *nnn.nnn.nnn.nnn*, where nnn is a value ranging from 0 to 255. The CIB subnet mask is 255.255.255.0; the host PC must be configured to the same subnet. The default value is 192.168.0.198.

To set, see the Set IP Address command

Example: Get the IP address:

```
/F?41
```

Get MAC Address

Synopsis: The MAC address of the CIB

Values: A valid MAC address

Syntax: `/F?40`

Description: MACAddress is a read-only parameter and its value cannot be changed.

Example: Get the MAC address:

```
/F?40
```

Get Output1

Synopsis: The status of Output 1 (J9, pin 7 and J8, pin 13)

Values: 0 or 1

Syntax: `/F?10`

Description: Valid values are 0 (Low) and 1 (High). Default is Low (0).

To set, see the Set Output1 command

Example: Get the Output1 status:

```
/F?10
```

Get Output2

Synopsis: The status of Output 2 (J9, pin 8 and J8, pin 15)

Values: 0 or 1

Syntax: `/F?11`

Description: Valid values are 0 (Low) and 1 (High). Default is Low (0).

To set, see the Set Output2 command

Example: Get the Output2 status:

```
/F?11
```

Get Output3

Synopsis: The status of Output 3 (J17, pin 3)

Values: 0 or 1

Syntax: **`/F?8`**

Description: Valid values are 0 (Low) and 1 (High). Default is Low (0).

To set, see the Set Output3 command

Example: Get the Output3 status:

```
/F?8
```

Get Output4

Synopsis: The status of Output 4 (J18, pin 3)

Values: 0 or 1

Syntax: **`/F?9`**

Description: Open drain power output: valid values are 0 (Low impedance to ground) and 1 (High impedance). Initial value is 0 (Low impedance to ground).

To set, see the Set Output4 command

Example: Get the Output4 status:

```
/F?9
```

Get Port Number for Command Processor Connection

Synopsis: The port number on the host PC that the CIB uses for communications

Values: 1 to 65535

Syntax: **`/F?42`**

Description: The default value is 9898.

To set, see the Set Port Number for Command Processor Connection command

Example: Get the port number:

```
/F?42
```

Get PortNumber for Embedded Connection

Synopsis: The port number for the Omni Embedded communication connection

Values: 1 to 65535

Syntax: `/F?44`

Description: The default value is 9897.

To set, see the Set Port Number for Embedded Connection command

Example: Get the port number:

```
/F?44
```

Get Power On Minutes

Synopsis: The value of the counter for the cumulative number of power on minutes

Values: Any valid number of minutes

Syntax: `/F?50`

Description: The Get command returns the number of power on minutes since the most recent Set command.

To set, see the Set Power On Minutes command

Example: Get the power on minutes:

```
/F?50
```

Get Power On Minutes Abs

Synopsis: The highest number of power on minutes ever recorded

Values: Any valid number of minutes

Syntax: `/F?51`

Description: Power On Minutes Abs is a read-only parameter and its value cannot be changed.

Example: Get the highest recorded power on minutes:

```
/F?51
```

Get Power Ups

Synopsis: The value of the counter for the cumulative number of power ups

Values: Any valid number of power ups

Syntax: Get: **`/F?52`**

Description: The Get command returns the number of power ups since the most recent Set command.

To set, see the Set Power Ups command

Example: Get the number of power ups:

```
/F?52
```

Get Power Ups Abs

Synopsis: The highest number of power ups ever recorded

Values: Any valid number of power ups

Syntax: **`/F?53`**

Description: Power Ups Abs is a read-only parameter and its value cannot be changed.

Example: Get the highest recorded power ups:

```
/F?53
```

Get VoltageFU1

Synopsis: The power supply voltage after FU1 from the CIB

Values: Valid voltage in mV

Syntax: **`/F?30`**

Description: VoltageFU1 is a read-only parameter and its value cannot be changed.

Example: Get the power supply voltage after FU1:

```
/F?30
```

Get VoltageFU2

Synopsis: The power supply voltage after FU2 from the CIB

Values: Valid voltage in mV

Syntax: `/F?31`

Description: VoltageFU2 is a read-only parameter and its value cannot be changed.

Example: Get the power supply voltage after FU2:

```
/F?31
```

Get VoltageFU3

Synopsis: The power supply voltage after FU3 from the CIB

Values: Valid voltage in mV

Syntax: `/F?32`

Description: VoltageFU3 is a read-only parameter and its value cannot be changed.

Example: Get the power supply voltage after FU3:

```
/F?32
```

Get VoltageFU4

Synopsis: The power supply voltage after FU4 from the CIB

Values: Valid voltage in mV

Syntax: `/F?33`

Description: VoltageFU4 is a read-only parameter and its value cannot be changed.

Example: Get the power supply voltage after FU4:

```
/F?33
```

Get VoltageV3.3D

Synopsis: The 3.3V power supply rail

Values: Valid voltage in mV

Syntax: `/F?34`

Description: VoltageV3.3D is a read-only parameter and its value cannot be changed.

Example: Get the voltage of the 3.3V power supply rail:

```
/F?34
```

Get VoltageV1.8D

Synopsis: The 1.8V power supply rail

Values: Valid voltage in mV

Syntax: `/F?35`

Description: VoltageV1.8D is a read-only parameter and its value cannot be changed.

Example: Get the voltage of the 1.8V power supply rail:

```
/F?35
```

Get VoltageTPPO

Synopsis: The On/Off state of J18 pin 3

Values: Valid voltage in mV

Syntax: `/F?36`

Description: Checks the On/Off state of J18 pin 3. VoltageTPPO is a read-only parameter and its value cannot be changed.

Example: Get the On/Off state:

```
/F?36
```

6.11.2 Set Commands That Write CIB Configuration Variables to RAM

Synopsis: The u<n> command sets the value of the ***Var_ID***.

Syntax: ***/Fu,<Var_ID>***

Var_ID: The following table provides a list of Var_IDs and the variables they correspond to:

Table 6-16 *CIB Set command (RAM) arguments*

Var ID1	Name	See Page
0	Output1	6-126
1	Output2	6-126
2	IO1	6-124
3	IO2	6-125
4	Output3	6-126
5	Output4	6-127
15	Terminate motion all upon error	6-127
16	Terminate motion all upon event	6-127
17	Set INT5	6-123

The following error can be generated by the **Fu<*n*>** command:

Generated Error: 3 - Invalid Operand.

WARNING! *Take care to ensure that the values of any parameters you set are within the valid range for your robot hardware (such as axis dimensions and travel length).*

Caution! *Setting parameters to values lower than the recommended minimum or higher than the recommended maximum may affect Omni Robot performance. Operation may become unstable even though the parameter value was accepted.*

Set INT5

Synopsis: Sets INT5 behavior.

Values: 0 or 1

Syntax: **`/Fu17,<Value>`**

Description: Valid values are 0 (set INT5 to report an event only if triggered) and 1 (set INT5 to terminate motion with all devices if triggered). Default is 0.

Note: It is strongly recommended to exit the application and cycle the CIB power after a stop.

Stoppage of axes is not instant but should be less than one second. Stoppage depends on four factors:

- The time it takes for the CIB to receive the INT5 signal
- The time it takes for the TML script to send the stop command
- The speed of the axis when the stop command is received
- The acceleration of the axis when the stop command is received

Stoppage of diluters varies depending on current executing diluter command. Please refer to diluter manuals for details.

The stop function is dependent upon communication between the devices and the CIB. If a communication breakdown should happen during the stop, it is possible that not all devices will be stopped.

Note: *Terminating axis motion using INT5 is the only termination type that results in a disabled axis.*

Example: Set to terminate motion for all devices if triggered:

```
/Fu17,1
```

Set IO1

Synopsis: The status of I/O1 (J4, pin 5), when I/O1 mode is set to output

Values: 0 or 1

Syntax: **`/Fu2,<Value>`**

Description: If I/O1 is acting as output (*I/O1Mode* = 1), valid values are 0 (Low) or 1 (High).

If I/O1 is acting as input (*I/O1Mode* = 0), setting I/O1 has no effect until the mode is changed to output.

To get, see the Get IO1 command

Example: Set the I/O 1 status to Low:

```
/Fu2,0
```

Set the I/O 1 status to High:

```
/Fu2,1
```

Set IO1Mode

Synopsis: I/O direction for I/O1 (J4, pin 5)

Values: Numbers that represent Input or Output mode

Syntax: **`/Fu2,2 or /Fu2,3`**

Description: I/O1 is bidirectional, and can be set as Input or Output. Default is Input. Valid values are:

- 2: Input
- 3: Output

To get, see the Get IO1Mode command

Example: Set the I/O 1 direction to Input:

```
/Fu2,2
```

Set the I/O 1 direction to Output:

```
/Fu2,3
```

Set IO2

Synopsis: The status of I/O2 (J5, pin 5), when I/O2 mode is set to output

Values: 0 or 1

Syntax: **`/Fu3,<Value>`**

Description: If I/O2 is acting as output (*I/O1Mode* = 1), valid values are 0 (Low) or 1 (High).

If I/O2 is acting as input (*I/O1Mode* = 0), setting I/O1 has no effect until the mode is changed to output.

To get, see the Get IO2 command

Example: Set the I/O 2 status to Low:

```
/Fu3,0
```

Set the I/O 2 status to High:

```
/Fu3,1
```

Set IO2Mode

Synopsis: I/O direction for I/O2 (J5, pin 5)

Values: Numbers that represent Input or Output mode

Syntax: **`/Fu3,2 or /Fu3,3`**

Description: I/O2 is bidirectional, and can be set as Input or Output. Default is Input. Valid values are:

- 2: Input
- 3: Output

To get, see the Get IO2Mode command

Example: Set the I/O 2 direction to Input:

```
/Fu3,2
```

Set the I/O 2 direction to Output:

```
/Fu3,3
```

Set Output1

Synopsis: The status of Output 1 (J9, pin 7 and J8, pin 13)

Values: 0 or 1

Syntax: **`/Fu0,<Value>`**

Description: Valid values are 0 (Low) and 1 (High). Default is Low (0).

To get, see the Get Output1 command

Example: Set the Output1 status to high:

```
/Fu0,1
```

Set Output2

Synopsis: The status of Output 2 (J9, pin 8 and J8, pin 15)

Values: 0 or 1

Syntax: **`/Fu1,<Value>`**

Description: Valid values are 0 (Low) and 1 (High). Default is Low (0).

To get, see the Get Output2 command

Example: Set the Output2 status to high:

```
/Fu1,1
```

Set Output3

Synopsis: The status of Output 3 (J17, pin 3)

Values: 0 or 1

Syntax: **`/Fu4,<Value>`**

Description: Valid values are 0 (Low) and 1 (High). Default is Low (0).

To get, see the Get Output3 command

Example: Set the Output3 status to high:

```
/Fu4,1
```

Set Output4

Synopsis: The status of Output 4 (J18, pin 3)

Values: 0 or 1

Syntax: **`/Fu5,<Value>`**

Description: Open drain power output: valid values are 0 (Low impedance to ground) and 1 (High impedance). Initial value is 0 (Low impedance to ground).

To get, see the Get Output4 command

Example: Set the Output4 status to high:

```
/Fu5,1
```

Terminate Motion All Upon Error

Synopsis: Terminates motion for the *Devices* due to an error (not an event).

Values: 0 or 1

Syntax: **`/Fu15,<Value>`**

Description: Valid values are 0 (terminate motion for all devices in the same command string ID if an error occurs with a device) and 1 (terminate motion for all devices in the system if an error occurs with a device). Axes are not disabled after termination. Default is 0.

Example: Set to terminate motion for all devices in the system if an error occurs:

```
/Fu15,1
```

Terminate Motion All Upon Event

Synopsis: Terminates motion for the *Devices* due to an event (not an error).

Values: 0 or 1

Syntax: **`/Fu16,<Value>`**

Description: Valid values are:

- 0: Report event for device
- 1: Terminate motion with all devices in the system if an event occurs with a device (excluding ADP streaming events and events used with H commands).

Default is 0.

Example: Set to terminate motion for all devices in the system when an event occurs:

```
/Fu16,1
```

6.11.3 Set Commands That Write CIB Configuration Variables to Non-Volatile Memory

Synopsis: The U<n> command sets the value of the ***Var_ID***.

Syntax: **`/FU,<Var_ID>`**

Var_ID: The following table provides a list of Var_IDs and the variables they correspond to:

Table 6-17 *CIB set command (non-volatile memory) arguments*

Var ID	Name	See Page
0	CAN0 Baud Rate	6-129
1	CAN1 Baud Rate	6-129
10	RS-232 Baud Rate	6-131
15	RS-485 Baud Rate	6-132
20	IP Address:1	6-130
21	IP Address:2	6-130
22	IP Address:3	6-130
23	IP Address:4	6-130
24	Port Number: Command Processor	6-130
34	Port Number: Embedded	6-130
50	Power On Minutes	6-131
52	Power Ups	6-131

The following error can be generated by the **FU<*n*>** command:

Generated Error: 3 - Invalid Operand.

Set CAN0 Baud Rate

Synopsis: Sets the baud rate of CAN0.

Values: 0

Syntax: **`/FU0,0`**

Description: The CAN0 baud rate can only be set to 500 Kbps.

To get, see the Get CIB Configuration command.

Example: Set the CAN0 baud rate:

```
/FU0,0
```

Set CAN1 Baud Rate

Synopsis: Sets the baud rate of CAN1.

Values: 0 to 3

Syntax: **`/FU1,<Value>`**

Description: Valid values are:

- 0: 500 Kbps
- 1: 250 Kbps
- 2: 125 Kbps
- 3: 100 Kbps

The default value is 0.

To get, see the Get CIB Configuration command.

Example: Set the CAN1 baud rate to 250 Kbps:

```
/FU1,1
```

Set IP Address

Synopsis: The IP address of the CIB

Values: A valid IP address

Syntax: **`/FU20,<AddressPart0>/FU21,<AddressPart1>/FU22,<AddressPart2>/FU23,<AddressPart3>`**

Description: The address must be a valid IP address in the form *nnn.nnn.nnn.nnn*, where nnn is a value ranging from 0 to 255. The CIB subnet mask is 255.255.255.0; the host PC must be configured to the same subnet. The default value is 192.168.0.198.

To get, see the Get IP Address command

Example: Set the IP address to 192.168.0.198:

```
/FU20,192/FU21,168/FU22,0/FU23,198
```

Set Port Number for Command Processor Connection

Synopsis: The port number on the host PC that the CIB uses for communications

Values: 1 to 65535

Syntax: **`/FU24,<Value>`**

Description: The default value is 9898.

To get, see the Get Port Number for Command Processor Connection command

Example: Set the port number to 9898:

```
/FU24,9898
```

Set Port Number for Embedded Connection

Synopsis: The port number on the host PC that the CIB uses for communications

Values: 1 to 65535

Syntax: **`/FU34,<Value>`**

Description: The default value is 9897.

To get, see the Get PortNumber for Embedded Connection command.

Example: Set the port number to 9897:

```
/FU34,9897
```

Set Power On Minutes

Synopsis: The value of the counter for the cumulative number of power on minutes

Values: 0 to 4294967295 (0xFFFFFFFF)

Syntax: **`/FU50,<Value>`**

Description: Sets the counter to the specified number. Recording the number of Power On Minutes resumes beginning from that counter.

To get, see the Get Power On Minutes command.

Example: Set power on minutes to 10 minutes:

```
/FU50,10
```

Set Power Ups

Synopsis: The value of the counter for the cumulative number of power ups

Values: 0 to 65535

Syntax: **`/FU52,<Value>`**

Description: Sets the counter to the specified number. Recording the number of Power Ups resumes beginning from that counter.

To get, see the Get Power Ups command.

Example: Set the number of power ups to 20 times:

```
/FU52,20
```

Set RS-232 Baud Rate

Synopsis: Selects port settings for RS-232 communications

Values: 0 or 1

Syntax: **`/FU10,<Value>`**

Description: The values represent:

- 0: 9600,n,8,1
- 1: 38400,n,8,1

The default value is 1.

To get, see the Get CIB Configuration command.

Example: Set the port settings to 9600,n,8,1:

```
/FU10,0
```

Set RS-485 Baud Rate

Synopsis: Selects port settings for RS-485 communications.

Values: 0 or 1

Syntax: **`/FU15,<Value>`**

Description: The values represent:

- 0: 9600,n,8,1
- 1: 38400,n,8,1

The default value is 0.

To get, see the Get CIB Configuration command.

Example: Set the port settings to 38400,n,8,1:

```
/FU15,1
```

6.12 OE System-Level Control Commands

6.12.1 Controlling Command Execution by Grouping Commands

In groups of commands, parentheses can be used to determine execution order, and to associate commands with devices. No predefined relation among devices is required. Also, this eliminates multiple intermediate responses reported by the devices during the command string execution.

In Figure 6-32, the six axes of the robotic system are labeled ID1 to ID6, and the two attached devices are labeled ID7 and ID8. To initialize the system and avoid colliding into labware while doing so, axes ID3 and ID6 are initialized simultaneously first and moved to their topmost positions. Axes ID1 and ID2 are initialized simultaneously as soon as ID3 finishes initializing, and axes ID4 and ID5 initialized simultaneously after ID6 finishes initializing. Because ID3 and ID6 might have different initializing parameters, the elapsed time for ID3 and ID6 might vary. For that reason, the initializing motions of axis pairs ID1-ID2 and ID4-ID5 are handled separately.

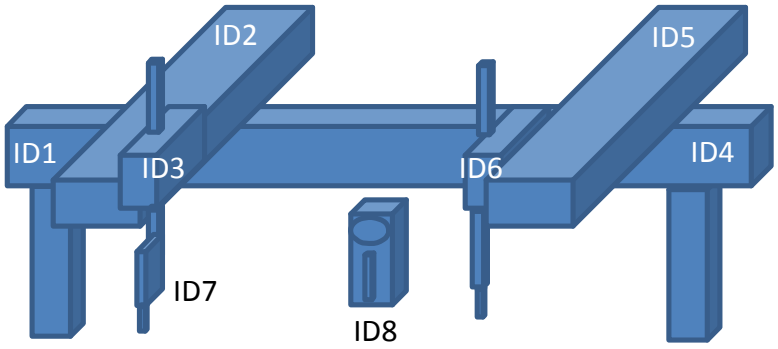


Figure 6-32 *Six-axis robotic system with two attached devices*

Devices ID7 and ID8 are seen as different entities and thus initialized simultaneously during the initialization of axes ID3 and ID6. The overall system reports an initialization completed status after all devices are finished.

Example 1

The following examples show two variations that initialize axes ID1 and ID2 simultaneously. Device command blocks in the parenthesis are seen as a group and commands within a group are executed simultaneously:

- `(/1Z/2Z)`
- `(/2Z/1Z)`

Example 2

Devices within different groups can have different execution priorities. Command execution for two concatenated groups is from left to right. To initialize axis ID3 prior to initializing the group of axes ID1 and ID2:

```
(/3Z)(/1Z/2Z)
```

Because ID3 is the only axis in its group, this command format can be abbreviated to:

```
/3Z(/1Z/2Z)
```

Example 3

Parentheses follow the mathematical order of operation only when nested. Because the inner nested group has higher execution priority, the command from Example 2 can also be written in any of the following ways:

- `((/3Z)/1Z/2Z)`
- `(/1Z/2Z(/3Z))`
- `(/1Z(/3Z)/2Z)`

Similarly, ID6 can be initialized prior to ID4 and ID5:

- `((/6Z)/4Z/5Z)`
- `(/4Z/5Z(/6Z))`

- `(/4Z(/6Z)/5Z)`

Example 4

Concatenating the two initializes axes ID1, ID2, and ID3 prior to axes ID4, ID5, and ID6:

```
((/3Z)/1Z/2Z)((/6Z)/4Z/5Z)
```

Example 5

Putting parenthesis around the command in Example 4 imposes the same level of precedence so that both groups are executed simultaneously. Axes ID3 and ID6 are initialized simultaneously first. Because ID1 - ID2 and ID4 - ID5 are in different groups, the initialization of ID1 - ID2 does not need to follow the initialization of axis ID6 and vice versa:

```
(((/3Z)/1Z/2Z)((/6Z)/4Z/5Z))
```

Example 6

Combined with the initialization of device ID7 and ID8, the overall system initialization can be described as the following command:

```
(((/3Z)/1Z/2Z)((/6Z)/4Z/5Z)(/7Z/8Z))
```

When spaces are inserted, you can clearly see there are 3 groups in the system that will be executed simultaneously. ID3 and ID6 have higher precedence in the first and second groups, respectively:

```
(   ((/3Z)/1Z/2Z)  ((/6Z)/4Z/5Z)  (/7Z/8Z)   )
```

Note: *Grouping commands to the same device at the same grouping level is prohibited. For example, the follow commands are all illegal:*

- `(/1Z/1Z)`
- `(/1Z/1?0)`
- `((/1Z)(/1Z/2Z))`

6.12.2 Command Progress Markers

The # character can be inserted in a command string to return a progress marker when the CIB reaches that point in executing the command string. If an error occurs, this assists with locating the failing point. A marker consists of the # paired with a number that identifies which marker it is. Markers can be used multiple times in a command string, but each numbered marker can only be used once within parentheses in a command group. For more information about command groups, see Controlling Command Execution by Grouping Commands.

Values: 0 to 5

Syntax: `/#<Value>`

Example1: Three markers in a command string:

```
1/3Z/#1/3A100/#2/3A1000000/#3/3A100R
```

Response if an error occurred while moving device ID 3 to A0:

`1@`	Acknowledgement
`1#1`	Marker 1 hit
`1#2`	Marker 2 hit
`1%30003`	String ID1 finished with error 3 (Invalid parameter) on device 3 after marker 2 was hit.

For more information about error codes, see Status Codes, Error Codes, and Events.

Example2: If command groups are used, the same numbered marker can only be used once within the same nested level of parentheses. The following command strings show incorrect use of markers in command groups.

No: `1(((/3Z)/#1/1Z/2Z)((/6Z)/#1/4Z/5Z))`

No: `1(/#1/#1/1ZR)`

For more information about command groups, see Controlling Command Execution by Grouping Commands.

Example3: For a command grouped with parentheses that includes one or more markers, the markers are returned before the execution of the commands within the parentheses. When there are multiple markers within a set a parentheses, the markers are returned in the order of their numbers, not necessarily in the order they appear in the command string.

Input:

```
1((/3Z)/#2/1Z/2Z/#1)
```

Response:

```
1@
1#1
1#2
1%
```

For more information about command groups, see Controlling Command Execution by Grouping Commands.

6.12.3 Halting Command Strings

Halt

Synopsis: Halts command string operation until the specified event occurs.

Syntax: /**H<*Value*>**

Values: A valid device event, beginning with !

Description: This command does not specify how long to wait for the event. The only argument it accepts is the event that will cause it to resume after halting.

This command does not generate any errors.

Example: Initialize axis 1, wait until axis 4 generates event code 65, then move to position 100:

```
1/1Z/H!40065/1A100
```

Caution! *When operating the Omni based on events, delays in command string execution must be added to avoid interference between commands. For example, an Omni system with an ADP must cycle between two command strings based on ADP Tip Loss events. The two command strings are sent together, and the second command string is Halted until the first one sends a Tip Loss event.*

The following event-driven code will eventually cause an error:

```
1/1A300/3P160,0,160(Other tasks are performed)/1A300/0ER
2/H!3006A/1A309/3P160,0,160(Other tasks are performed)
```

In this example, the ADP DropTip command fully extends the plunger to drop the tip, then the ADP performs an internal re-initialization. Because the Tip Loss event immediately starts the execution of the second command, it is possible for the axis to begin picking up a tip before the ADP has finished execution, and this can cause errors to occur.

To prevent this error, a proper implementation must account for this with delay in one of the following ways:

- with a pump delay command

- by slowing down axis motion,
- by placing tips farther apart
- or using any other method to ensure that all DropTip execution processes have finished before PickTip processes begin.

6.12.4 Global Commands

The commands that operate globally on the Omni system are:

- Terminate All
- Global System Query

Terminate All

Synopsis: Terminates motion for all devices in the system.

Syntax: 0**/T**

Description: When the T command is issued with the 0 string ID, it is a global terminate command and the motion of all devices is stopped.

For more information about ways to terminate motion, see Terminating Motion.

Example: Terminate all motion:

```
0/T
```

Global System Query

Synopsis: Returns system-related information.

Values: 0, 1, or 2

Syntax: **0/?*<Value>***

Description: Valid values are:

- 0 - Reports the number and IDs of all attached devices on the system.
- 1 - Reports the number and IDs of attached ACB devices on the system.
- 2 - Reports the number and IDs of attached Cavro® CAN devices on the system.

Generated Errors: String status error codes. See Section 6.6.5, "Command Response String Status Codes" for more information.

Example: Report on all attached devices on the system:

```
0/?0
```

6.13 Controlling Omni Hardware

6.13.1 Omni Hardware Configuration

There are two methods for configuring Omni in Embedded mode:

- Using the Omni Configurator supplied on the CD.
- Configure the Omni programmatically using the Omni Embedded Command Set.

For information about using Configurator software, please refer to Chapter 5, “Integration with Omni Configurator and Cavro Fusion”.

6.13.2 Basic Omni Hardware Control

The following links contain two examples of embedded command string sequences:

- Section G.3.1, ”Aspirate and Dispense with ADP”
- Section G.3.2, ”Aspirate and Dispense with XLP Pumps”

In both examples, the R command is required at the end of most pump commands. The R command is optional for axis commands.

6.14 Terminating Motion

There are several commands that can stop the motion of the Omni. Which one to use depends on the type of stop desired and the circumstances under which the Omni should stop:

- Global Termination: Terminates all commands currently in execution.
- Terminate Command String: Terminates a specified command string.
- Terminate Current Command for Device: Terminates a currently executing command of a device within a command string. If there are more action commands for the selected device that follow the terminate command in the command string, these commands are executed normally.
- Terminate on Event: Terminates command string execution when a device in that command string encounters the specified event.
- Terminate on Error: Terminates command string execution when a device in that command string encounters any error.
- Terminate on Triggering of INT5 Pin: See Set INT5 for more information.

Note: *In the Terminate String command you can use the same string ID to send the T command to multiple devices. In the Terminate Current Executing Action Command for Selected Device command you cannot use the same string ID to send the T command to multiple devices.*

6.14.1 Global Termination

When sending the Terminate Axis and Gripper Motion command globally, all commands currently in execution are terminated. The following example illustrates using the Terminate Axis Motion command to terminate all four current command strings.

Commands strings:

```
** Initialize Axis 1 then move to absolute position 200 **
      1/1Z/1A200
** Initialize Axis 2 then move to absolute position 100 **
      2/2Z/2A100
** Initialize Axis 3 then move to absolute position 150 **
      3/3Z/2A150
** Initialize Diluter 5 then move to absolute 200, then 100 **
      4/5ZA200A100R
** Terminate everything **
      0/T
```

Response:

```
** Ack for T command send (no completion response) **
      0t0@
** Command string ID 1 is completed (terminated) **
      0`1%
** Command string ID 2 is completed (terminated) **
      0`2%
** Command string ID 3 is completed (terminated) **
      0`3%
** Command string ID 4 is completed (terminated) **
      0`4%
```

6.14.2 Terminate Command String

To terminate only a specified command string, use the Terminate Axis and Gripper Motion command with a command string ID. This is the only circumstance when you can use the a command string ID while the same command string is busy executing. The following example shows how to terminate a command string.

Commands:

```
** Initialize Axis 1 and 2 then move to absolute positions **
** 200 and 100 respectively **
      1/1Z/2Z/1A200/2Z/2A100
** Terminate command string ID 1 (note: using the same **
** string ID) **
```

```
      1/T
** Ack for T command send (no completion response for this **
** command) **
      0t1@
```

Response:

```
** Command string ID 1 is terminated **
      0`1%
```

6.14.3 Terminate Current Command for Device

To terminate a specific command in a command string for a device, use the Terminate Axis and Gripper Motion command and specify the device. The terminate command will only terminate the current action command, and then the command string will resume execution. The following example shows how to terminate a command in a command string.

Commands:

```
** Initialize Axis 1 and 2 then move to absolute positions **
** 200 and 100 respectively **
      /1Z/2Z/1A200/2Z/2A100
** Send T command to terminate axis ID 2 during the first **
** 2Z command. **
** The first Z command for axis 2 will be terminated, **
** however the all the sequential commands (2Z and 2A100) **
** for axis 2 are still executed. **
      2/2T
```

Response:

```
** Ack for T command (no completion response) **
      0t2@
** Command string completed. **
      0'2%
** Command string completed. **
      0`1%
```

6.14.4 Terminate on Event

To preset an event that will cause a command string to terminate when the event occurs, use the Terminate Motion All Upon Event command. The following example sets the termination event and shows how the system behaves when that event occurs:

Commands:

```
** Set the termination event **
      1/Fu16,1
** Initialize axes 1, 2, and 3 **
      1/1Z/2Z/3Z
** Move to absolute position X: 300, Y: 100, Z: 150 **
```

```
        1/1A300/2A100/3A150
** Initialize gripper **
        1/3W0
** Open gripper **
        1/3W2
** Move Z to absolute 100 **
        1/3A100
** Close gripper to grab plate **
        1/3W1
** Move to absolute position  X: 100, Y: 50, Z: 180 **
        1/3A180(/1A100/2A50)/3A100
** Drop plate event received from device ID 3 **
        0e!30071
```

Response:

```
** Command aborted due to drop plate event **
        0`1%30071
```

6.14.5 Terminate on Error

To terminate a command string when an error occurs, use the Terminate Motion All Upon Error command. The following example sets the option to terminate the command string if an error occurs and shows how the system behaves when that error occurs:

Commands:

```
** Set Terminate Motion All Upon Error option **
        1/Fu15,1
** Initialize axes 1, 2, and 3 **
        1/1Z/2Z/3Z
** Move to absolute position X: 300, Y: 10000 (invalid **
** parameter that triggers an error) , Z: 150 **
        1/1A300/2A10000/3A150
```

Response:

```
** Command aborted, device 2 on command string ID 1 **
** encounters error 3, the command 3A150 never executes **
        0k1%20003
```

6.14.6 Terminate on Triggering of INT5 Pin

See Set INT5 for a description of INT5 termination.

This page has been left intentionally blank.

7 Operating in Command Processor Mode

This chapter provides reference information for the Omni Robot software command set. For an example of how to use a command in conjunction with a C# program, see Section 7.11.2, ”Omni API Usage Examples” on page 7-105.

This chapter is organized into the following sections:

7.1 Software Setup and Integration

The OEM user is responsible for developing a Windows-based application to transmit commands to the Cavro® Omni Robot CIB. This application must use the appropriate Command Processor .dll file provided to interface with the Omni Robot Communication Interface Board (CIB).

To make this possible, the Omni Robot comes with the software listed in Table 1-1.

This chapter describes the Command Processor and provides instructions for using the Command Processor Application Programming Interface (API) and the Omni Robot command set to integrate the operation of the Omni Robot with the software used to operate the OEM system.

7.1.1 Communication Flow

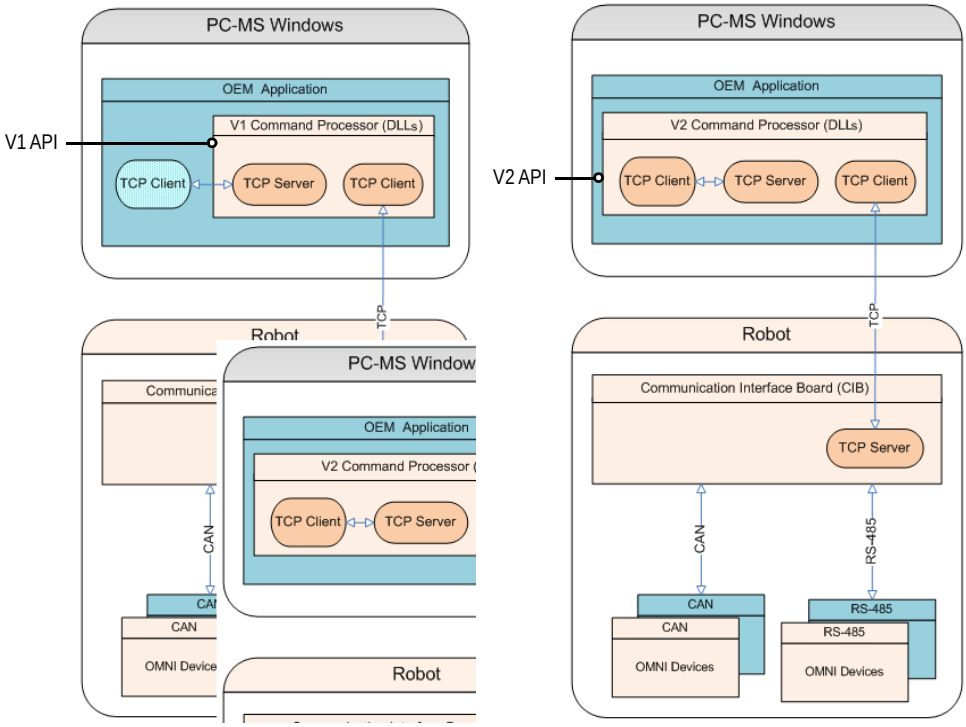


Figure 7-1 *Communication flow diagrams, V1 vs. V2 Command Processors*

The Omni Robot CIB and the Command Processor V1 API interact as follows:

1 The OEM application calls a function provided by the Command Processor. This initiates the connection between the Command Processor and the CIB, and activates the TCP/IP server in the Command Processor.
2 The OEM application establishes a communications channel to the TCP/IP server in the Command processor.
3 The OEM application uses the communications channel through standard TCP/IP I/O calls to send commands or receive replies from the CIB.

The Command processor V2 API has simplified these interactions: the OEM application now calls Omni V2 API functions to control the Omni Robot.

7.1.2 Installing the Software

The installation CD includes the Configurator and the v4.0 Command Processor and axis control software.

It is not necessary to uninstall older versions of Omni software first. However, users must be aware that multiple versions of the Command Processor file `nrccmdproc.dll` and axis driver file `axisdrivers.dll` may exist in different directories. To install the software, run the `setup.exe` executable on the installation CD. This will install the Configurator software and the Command Processor file (**`NRCCmdProc.dll`**) and axis driver file (`axisdrivers.dll`) in the ***`<drive>`*`\Program Files(x86)\Tecan Systems\Omni V4.0`** directory, where ***`<drive>`*** is the PC hard drive.

Note: *This is a 32 bit application.*

The CD also includes a C# demo program that can be used to quickly start using the Cavro® Omni Robot. There is also a LabVIEW demo program that illustrates the OMNI Command Process V1 API usage, available upon request.

7.1.3 Setting the CIB IP Address

The CIB must be configured with a unique IP address to communicate over a network with a client PC. The default CIB IP address is 192.168.0.198.

To set a different CIB IP address:

1 If you have not already done so, disconnect the PC from any network to which it is connected; Tecan recommends using a dedicated PC to operate the Omni Robot.

Caution! *Tecan recommends that the PC only run the software required to operate the Omni Robot and the OEM system. Running additional software, such as antivirus software, could cause a decrease in performance.*

2 Connect the PC directly to the CIB with a Category 5 patch cable.

3 Manually change the IP address of the client computer to a static IP address of 192.168.0.*x*, where *x* is a number other than 0, 1, 198, 255, or any other IP address already in use within the subnet.

Note: *The subnet of both the client PC and the Omni Robot must be the same. The default subnet mask is 255.255.255.0.*

4 Run the Configurator software; it will automatically attempt to connect to the Cavro® Omni Robot using the default CIB IP address (192.168.0.198) and port number (9898).

This establishes the connection between the PC and the Omni Robot.

5 Log into the Configurator. You can use any Operator Name you want; the password is ***TechOpr*** (case sensitive).

6 Use the Command Line tool to submit a **C0,Set,IPAddress,*[address]*|** command to the CIB. ***[address]*** is the new IP address to use.

See Set,IPAddress in Chapter 7, “Operating in Command Processor Mode” for more information.

Caution! *If you change the default IP address or listener port and subsequently forget the new values, you will not be able to pass communications between the CIB and Command Processor. In this case, you will need to reset the CIB to the factory default IP address and port number.*

7 Power cycle the Omni Robot to make the new setting take effect.

7.2 Omni Robot Coordinate System

The following images show the Omni Robot axis positions and parameters. Figure 7-2 shows the X and Z axis parameters for a single arm model.

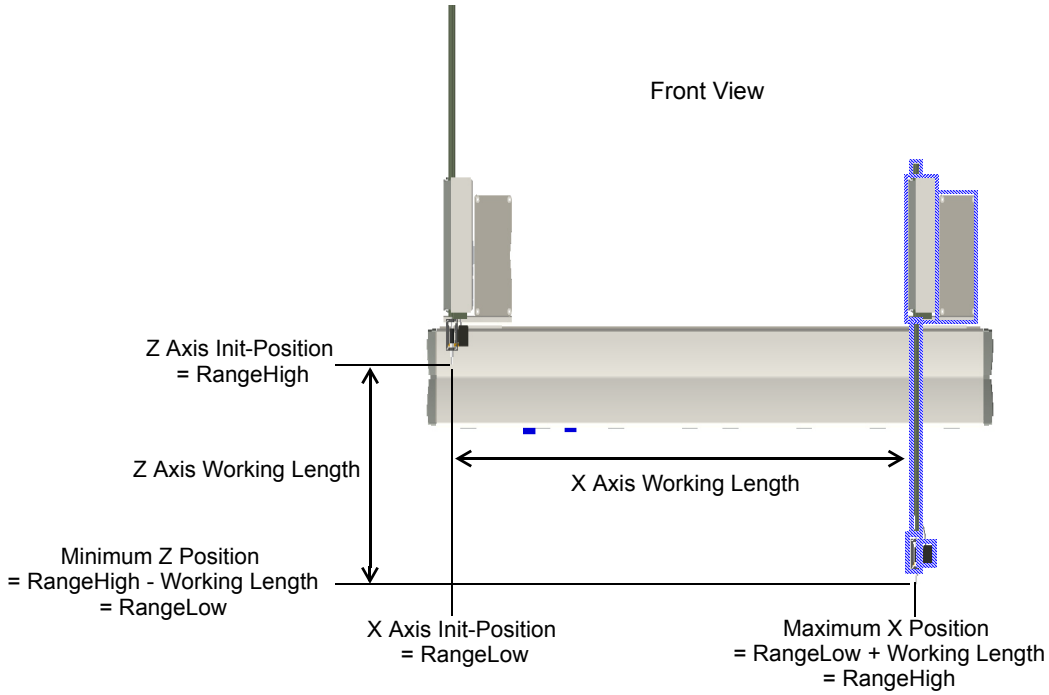


Figure 7-2 *Single arm Cavro Omni Robot axis positions (X and Z axes)*

Figure 7-3 shows the Y axis positions and parameters.

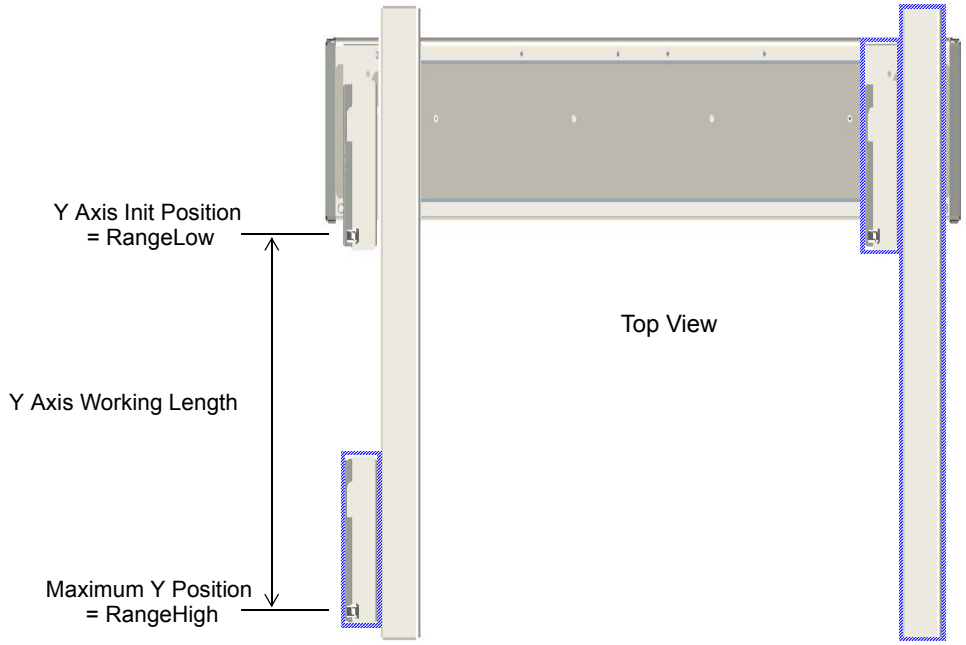


Figure 7-3 *Cavro Omni Robot Y axis positions*

Note: *For the Z axis (unlike X and Y), the Axis Initial Position (home) defines RangeHigh, and RangeLow is at the bottom, farthest from the home position.*

The Cavro® Omni Robot coordinate system provides significant flexibility in that it allows the user (application developer) to define the coordinate system to be used by the Omni Robot, relative to a position of interest to the user (for example, the X, Y, and Z coordinates of a specific location on the worktable or a test fixture). The Omni Robot will then use this relative coordinate system for all its functions, transparently mapping those coordinates to the physical coordinates of the robot.

The Cavro® Omni Robot coordinate system can use coordinates in the range of -10,000 to 10,000 mm.

Coordinate mapping is accomplished by moving the robot axes to a specific physical location, and then setting the coordinate values that are to be used to represent that location. The reference position can be set using any arbitrary values for the coordinates, completely independently of the physical coordinates, based on whatever makes sense in terms of the application. This is done by performing a calibration using the Omni Configurator software, or programmatically using the Omni command set.

For example, suppose you have a fixture for an array of test tubes, and want to use the top left tube as your reference point, call that location by coordinates 100, 100, 100. However, in physical coordinates, that tube is actually at X = 215 mm, Y = 149 mm, and the Z axis height is 128 mm. You can use the calibration

process to specify that the upper left test tube coordinates shall be called 100, 100, 100. Thereafter, all positional references in the application will be in terms of a coordinate system where 100, 100, 100 is at the designated test tube.

This is explained in more detail below. For simplicity, the following discussion will focus on the X axis position, but the principles apply to the Y and Z axes as well.

The X axis parameters shown in Figure 7-2 are:

- X axis Initial Position: the home physical position of the X axis.
- X axis Maximum position: the farthest physical location away from the axis home position.
- Working Length: the distance between the axis home and the maximum X position.
- *RangeLow*: the lowest coordinate address valid for the axis, relative to the reference position.
- *RangeHigh*: the highest coordinate address valid for the axis, relative to the reference position.

There is also an additional parameter, *AxisOffset*, which is the Axis home position coordinate (relative to the reference position), and is equal to *RangeLow* for the X or Y axis.

Figure 7-4, “X axis coordinate mappings relative to user-specified reference point” on page 7-8 shows the use of a reference position, and how it affects all the other parameters.

In terms of *physical* units, the initial (home) position of the X axis is at coordinate zero. For this example, assume the reference point is 215 physical units (mm) from the home position. As part of the calibration process, the user can manually move the robot arm to position 215, and then set this position to be 100 (via the configurator calibration function, or by using the Set,Pos,100 command.

When the reference point is set, other parameters such as the *AxisOffset*, *RangeLow*, and *RangeHigh*, are calculated relative to this reference point. For subsequent commands, such as move commands, the coordinates are assumed to be relative to the new coordinate range.

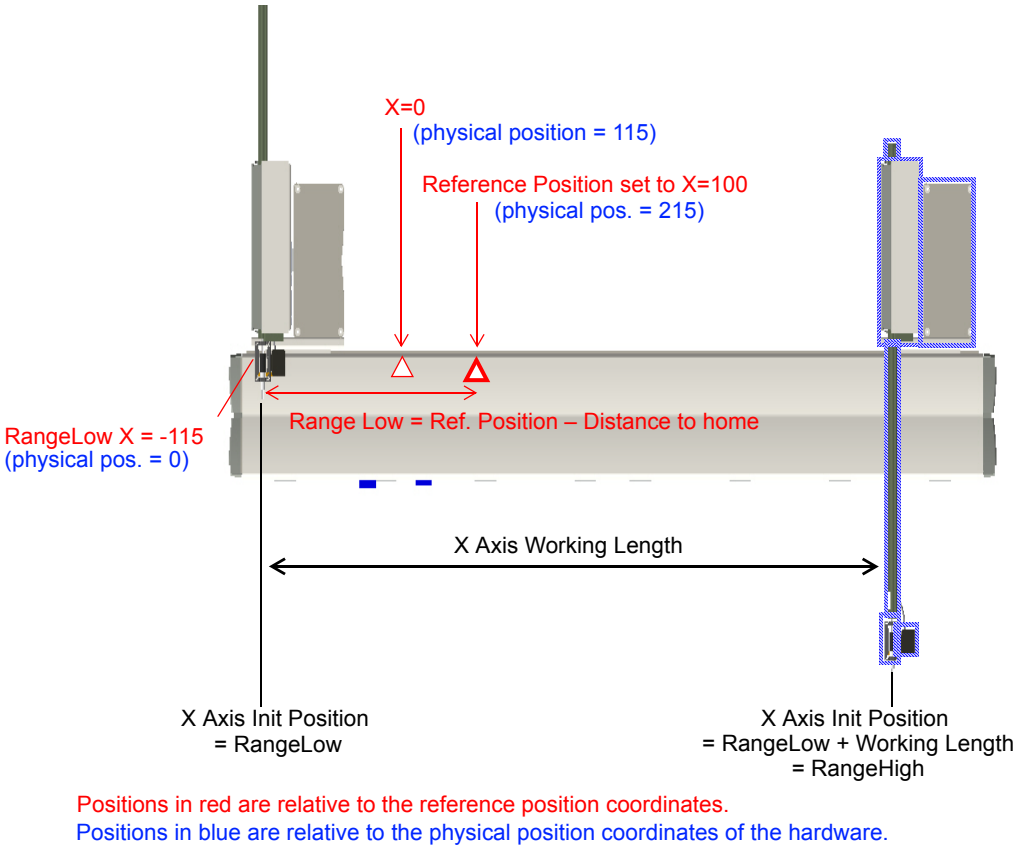


Figure 7-4 *X axis coordinate mappings relative to user-specified reference point*

Therefore, in the example in Figure 7-4, the positional parameters will be calculated as follows:

- *AxisOffset* = –115 (Reference position – distance to home; 100-215).
- *RangeLow* = –115.
- Assuming an X axis working length of 800, *RangeHigh* = 685.

This means that the X axis can move to positions between X = –115 and X = 685 as its working range.

Note: *If you re-use the same axis ID in two different arm groupings and then perform calibration on both, the second calibration will overwrite the offsets from the first.*

7.2.1 Coordinates in a Dual Arm Configuration

For a dual arm configuration, the Y axis and Z axis positions and coordinate parameters are the same as for a single arm configuration.

Figure 7-5 shows the X axis coordinates for a dual arm configuration—in this case a configuration where the Z axis is mounted on the left side of each arm.

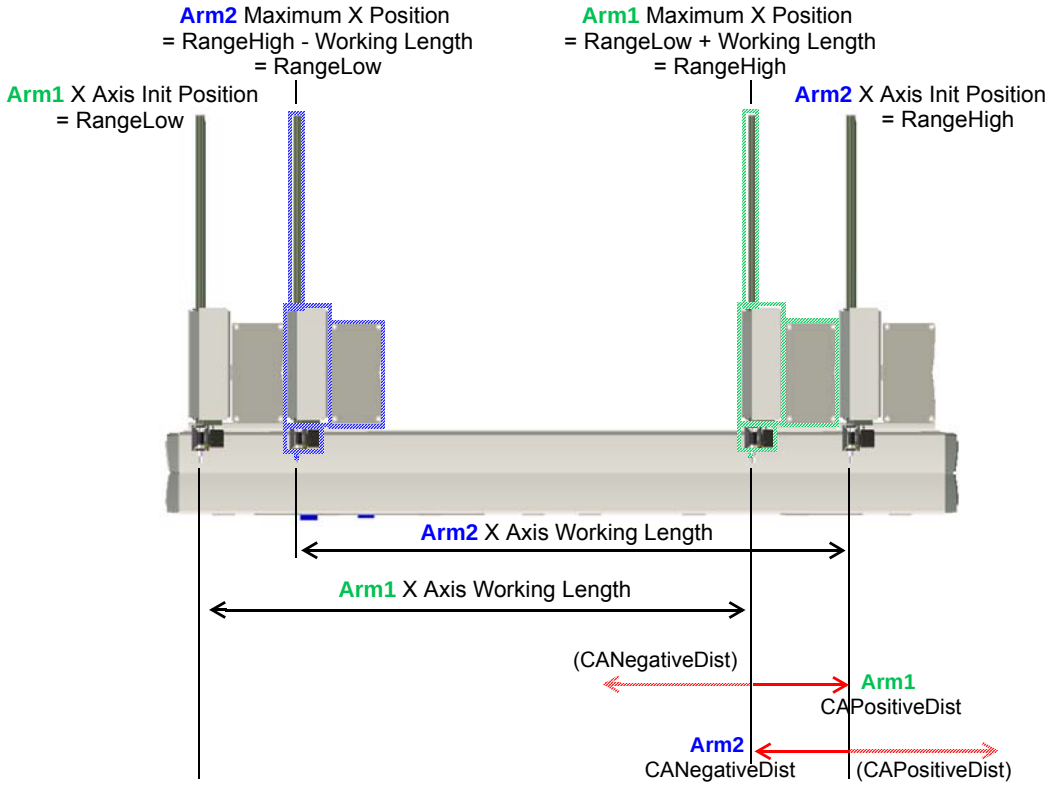


Figure 7-5 *Dual arm Omni Robot X axis positions and parameters*

In this configuration, one arm homes to the left, and the other homes to the right. Each arm has its own settings for *AxisOffset*, *RangeLow* and *RangeHigh*. Note that for Arm1 (the left-homed arm) *AxisOffset* = *RangeLow*, while for Arm2 (the right-homed arm) *AxisOffset* = RangeHigh.

In a dual arm system, both X axes can have their coordinate systems re-mapped to a user-specified reference point. For example, you can map Arm2 to the same reference point that is shown in Figure 7-4 for Arm1, to enable both arms to address the same reference position using the same coordinate address.

When you are calibrating and working with a Dual Z, the rear-most Z must be used to establish a baseline to verify the true offset of the front-most Z (nominally 9 mm or 18 mm in the Y direction).

The Omni allows axes to be configured into arm groups. For the Dual Z it is possible to configure a single arm with 2 Z axes or 2 virtual arms, each as a single XYZ. The same axes can be configured into different arm groups, for example:

- Arm1: axes 1,2,3,7

  or

- Arm1: axes 1,2,3;
  Arm2: axes 1,2,7

In this example:

- Assume the total length of the X axis is 1000 mm, so that the position of Arm2 after initialization is X=1000.
- Assume the Working Length for Arm2 X axis has been set to 800 mm.

The user can manually move Arm2 X axis to position 215, and do a Set,Pos,100 to set the reference X-coordinate to 100.

Once that is done, the system calculates the *AxisOffset*, *RangeLow*, and *RangeHigh* as follows:

- Arm2 X axis *AxisOffset* = distance from the reference position to Arm2 X axis home + value of the reference position new coordinate.

  The physical distance from the reference position to Arm2 X axis home is 1000-215, or 785 mm, so the *AxisOffset* = 785 + 100 = 885.
- *RangeHigh* = Arm2 X axis *AxisOffset* = 885.
- *RangeLow* = Arm2 X axis *AxisOffset* - Arm2 Working Length = 885-800 = 85.

Now, both Arm1 and Arm2 use the same reference position (physical position 215 mm) and refer to it as X=100.

Note: *The Travel Position, or TravelPos, is used to specify a "safe" position to which the Z axis is moved prior to executing a move in order to avoid potential collisions with elements of a test fixture, for example. This is set with the Set,TravelPos command.*

When working with a Dual Z, it is possible to configure an arm or arms which include both of the Zs, an arm which only includes one of the two Zs, or multiple arms which each include one of the two Zs.

WARNING! *Because both of the Zs in the Dual Z are physically attached, when an arm which physically contains one of the Zs moves, both Zs will move even if they are not both included in the arm grouping. Thus, it is necessary to make sure that both the Zs on the Dual Z are located in an appropriate Travel Position before the move begins.*

The simplest way to achieve this is to only set up arms that include both of the Zs on a Dual Z. Alternately, the application can be written to check all axes in a list for safe travel positions, and move them to a safe travel height regardless of whether they are grouped into an arm or not.

7.2.2 Collision Avoidance

In a dual arm configuration, there are additional parameters that are used for collision avoidance: the ID of the neighboring arm (in either the positive or negative direction) and the minimum distance required between the arm and its neighbor to help avoid a collision.

Caution! *The Collision Avoidance parameters should be set only after a system has been calibrated.*

Figure 7-5 illustrates these parameters:

- For Arm1, CAPositiveDist defines the minimum distance required between itself and Arm2. *CANegativeDist* is set to zero, indicating there is no neighbor in the negative direction.
- For Arm2, CANegativeDist defines the minimum distance required between itself and Arm1. *CAPositiveDist* is set to zero, indicating there is no neighbor in the positive direction.

WARNING! *If the collision avoidance parameters are set incorrectly in a dual arm system, and the speeds of the two arms are NOT the same, it would be possible for the faster arm to catch up to the slower arm and collide with it. It is important the speed of both arms be set to the same values. Use the Set,DefSpeed command for each arm to ensure the axes of both arms are set to the same values.*

WARNING! *The possibility of collision exists in a multiple arm system if one arm malfunctions and stops in the middle of a run. It is recommended that the OEM application software be designed such that if an asynchronous Device Error event occurs, the application will immediately terminate motion on all axes. See Section 7.8, "Events" on page 7-88, for more information about asynchronous events.*

7.2.3 Changing Configuration Settings Programmatically

It is recommended that any specific configuration settings required by an application be made programmatically by the application itself, rather than relying on specific power-on default settings. The various Set commands described in Section 7.6, "Parameter Get and Set Commands" can be used for this purpose.

Furthermore, if the application requires that the robot be calibrated to a specific coordinate system, it is strongly recommended that the system calibration be handled under program control, to ensure that the settings are done correctly, rather than relying on the operator to make the settings manually using the Configurator software.

The following example illustrates the commands that could be used within a program to calibrate the robot before beginning the execution of the application's main functions. See Section 7.7, "Action Commands"for more information about the commands in this example.

```
** Initialize the robot **
A0CavroInit
** Disable the three arms so they can be moved manually **
M1Disable
M2Disable
M3Disable
** Communicate with the operator to move the arms to the desired
physical reference position, and indicate when they are in place. **
** Then, set the Omni position coordinates approriately for the
application (shown as <nn> below). **
M1,Set,Pos,<nn>
M2,Set,Pos,<nn>
M3,Set,Pos,<nn>
** Re-enable the axes and reinitialize the robot **
M1Enable
M2Enable
M3Enable
A0CavroInit
```

If this is a dual arm system, repeat this procedure for the second arm.

7.3 Using the Command Set

The Command Processor is software that provides a TCP/IP server for a custom application to communicate with the Omni Robot. The devices attached to the CIB, including axes and pumps, can be queried, controlled, or operated by sending a command string to the TCP/IP server.

Upon receiving a command string, the Command Processor validates the string syntax, target device, command name, and parameter ranges (if any) before translating it to lower-level device commands and forwarding them to the target device via the CIB.

In addition, the Command Processor also allows applications to communicate with other Cavro® RS-485 and Cavro® CAN device options which attach to the Omni robot using Omni data protocol.

7.3.1 Command String Format

The command string is a text string that can be recognized by the Command Processor. Each command string consists of:

- a target device (device type and device ID, if applicable)
- the name of the command sent to the device
- any command parameter(s)
- a delimiter to separate one command string from the next. Each command must end with the delimiter.

Target Device

The **device type** is denoted by one of the following alphabetical characters:

- **A** for arm
- **C** for CIB
- **D** for RS-485 devices
- **M** for axis
- **E** for Cavro® CAN devices

The **device ID** is an integer (0-255) that corresponds to the CAN or RS-485 hardware address setting of the device. The device ID for a logical device, such as an arm, is assigned by the Command Processor during system configuration.

Note: *Device IDs for a type D device are determined by adding 1 to the address of the device. For example, a diluter with RS-485 address 4 will become target device D5.*

Note: *Device IDs for a type E device are determined by the address switch/ jumper. For example, a Cavro® CAN diluter with address 0 will become target device E0.*

Arm and Axis Device IDs can be viewed using the Configurator software. Axis IDs for an arm can be retrieved programmatically using the Arm Get,Axes command.

Command Name

The command name is the name of the command sent to the device. Complete reference information on the Cavro® Omni Robot command set is provided in Section 7.6, ”Parameter Get and Set Commands”, Section 7.7, ”Action Commands”, and Section 7.10, ”Gripper Commands”later in this chapter.

If the device is an RS-485 device (device type D), the device’s native commands will replace the Omni robot command in the command string. The native command name replaces the name of the Omni Robot command in the string.

Parameters

Each command has a unique set of parameters. Parameters are shown in brackets for each command in the command set.

Command strings are delimited by the / character. Multiple commands separated by the delimiter may be concatenated in one string.

Command String Examples

Table 7-1 Omni axis command: Values in the string M1MoveAbs,123.4/

M	**The target device is an axis.**
1	**The axis' CAN address (device ID) is 1.**
MoveAbs	**Use the MoveAbs command, which moves the axis.**
123.4	**The target position of the movement, in millimeters (mm).**
/	**The command string delimiter.**

Table 7-2 Cavro RS-485 command: D5,ZR/, sent to a diluter

D	**The target device is an RS-485 device type.**
5	**The Cavro® RS-485 device address is 4. (Its ID is the RS-485 address plus 1.)**
ZR	**The native diluter command and parameter string.**
/	**The command string delimiter.**

Table 7-3 Cavro CAN command: E5-1,ZR/, sent to a CAN device

E	**The target device is a CAN device type.**
5	**The address of the CAN device.**
-	**The CAN command frame type separator**
1	**The CAN frame type of the command (see the CAN product manual).**
,	**The command string block separator.**
ZR	**The native diluter command and parameter string.**
/	**The command string delimiter.**

7.3.2 Command Execution

During the execution of an action command (that is, a command that moves one or more axes or operates the cLLD system), a device is in a locked state and can accept only the query (Get) command. If an arm is executing an action command, all axes belonging to that arm are locked. The OEM application receives an error response when sending an action command to a locked device.

The exception is the TerminateMotion command, which can be executed by a device when the target axis is executing the *CavroInit* or *MoveAbs,[Pos]* command.

Caution! *Always wait for the completion of a command before sending a subsequent command. If a new command is sent before the previous command was completed, the new command may not be completed properly. Sending too many commands to the CIB in too short a period of time could result in commands being dropped.*

7.3.3 Command Response Format

For every command sent to the CIB, the OEM application receives one reply with the result after the device completes the command. The reply consists of:

- target device
- name of the command that was executed
- result code
- optional error text as determined by the result code
- a string delimiter

For an Omni arm or axis command, a result code of 0 indicates successful command execution.

For a Diluter, 0 means the Diluter has received the command.

Caution! *When a response from a diluter, received by the Command Processor, contains the character '/' it will automatically be replaced by the character '#' to avoid conflicting with the char '/', command string delimiter. The host application should convert this '#' back to '/' wherever applicable.*

Example:

```
--> D1,?23/
<-- D1,0,0'700001 Rev-A 03#2010/    ==> Response received at the
                                        host application from the
                                        Command Processor.
```

The actual response from the diluter is: `700001 Rev-A 03/2010`

Command Response Examples

Omni Axis command response:

The values in the response **M1MoveAbs,7,Device_Not_Initialized/** correspond to Table 7-4.

Table 7-4 M1MoveAbs,7,Device_Not_Initialized/ response values

M	**The target device is an axis.**
1	**The axis' CAN address is 1.**
MoveAbs	**The string is in response to the MoveAbs command.**
7	**The command failed and raised an error of type 7. In the case of multiple errors, the last reported error is displayed here.**
Device_Not_ Initialized	**The error occurred because the device was not initialized**
/	**The string delimiter.**

Arm command with Dual Z response with errors on both Z axes:

The values in the response **A0DetectLiquid,15,cLLD_No_Liquid_Detected(M6-27)(M3-15)** correspond to Table 7-5.

Table 7-5 A0DetectLiquid response values

A	**The target device.**
0	**The arm address.**
DetectLiquid	**The string is in response to the DetectLiquid command.**
15	**The last error.**
cLLD_No_ Liquid_ Detected	**The last error occurred because no liquid was detected.**
(M6-27)(M3-15)	**All errors are listed here; error 27 on M6 device, error 15 (last error) on M3 device.**
/	**The string delimiter.**

Cavro® RS-485 command response:

For Diluter commands, the first parameter of the response is always an error code, followed by the actual reply string from the device. For example the response to the command **D5,ZR/** is shown in Table 7-6.

Table 7-6 Cavro RS-485: D5,0,0@/, response values

D	**The target device is a diluter.**
5	**The diluter's RS-485 device address is 4.**

Table 7-6 Cavro RS-485: D5,0,0@/, response values

0	**Command Processor error code for "No Error." Note: This field was omitted in V1 if the diluter had received the command.**
0@	**The reply string from the diluter (always in HEX. See the specific product documentation to interpret the data.)**
/	**The command string delimiter.**

The application receives unsolicited messages if an error occurs that is unrelated to the execution of a command; this information is transmitted as an event. The event notification consists of the event source, the string Event, the event ID number, and optional parameters as determined by the event ID number.

For more information on events see Section 7.8, "Events" on page 7-88. and the list of event codes in Section 7.8.1, "EventID Values" on page 7-89.

Cavro® CAN command response:

Table 7-7 Cavro CAN response: **E5-1,0, '*/***

E	**The target device is a CAN device type.**
5	**The address of the CAN device**
-	**The CAN command frame type separator**
1	**The CAN frame type of the command (see the CAN product manual).**
,	**The separate device from a command string block.**
'*/*	**The response string from a command string block.**
/	**The command string delimiter.**

Caution! *To enable Cavro® CAN devices to accept "E" commands, the application must send **C0Set,EnableCavroCAN,0** prior to any commands sent to Cavro® CAN devices. For more information about Set,EnableCavroCAN see EnableCavroCAN on page 7-31.*

As a reminder, these are the valid device types supported on the Omni:

- **A** for arm
- **C** for CIB
- **D** for RS-485 device (such as diluter)
- **M** for axis
- **E** for Cavro® CAN (such as ADP)

7.4 Command Syntax Rules

Note: *Type D and E commands are addressed to Cavro® devices, which will execute them. These commands are not transparent to the OMNI software. Refer to the respective Cavro® device manuals for more information about your devices.*

Note: *The ADP is a type E device on the OMNI.*

Note: *Commands are case-sensitive.*

Table 7-8 Formatting conventions

Convention	Description
bold font	Commands and keywords.
<*italic*> font with angle brackets	Arguments that accept values.
italic font (without angle brackets)	Parameters that affect a command but are not arguments to the command.
[ ]	Keywords or arguments that appear within square brackets are optional.
{x | y | z}	A choice of required keywords appears in braces separated by vertical bars. Select one.
`courier` font	Examples of information displayed on the screen.
`bold courier` font	Examples of information entered by the programmer or user

- All optional arguments that are omitted must be replaced with a comma (as a placeholder).

  For example, the command **M3CheckForExit, , , ,50/** specifies that default values are to be used for the first three parameters (*RetractSpeed*, *RetractDist*, and *Limit*) and specifies a value for the *Sensitivity* parameter. See the CheckForExit command for more details about this command.

For more information about the structure of a command and its corresponding reply, see *Section 7.3, "Using the Command Set" on page 7-12**.*

7.5 Parameters

All parameters that are stored in the CIB can be retrieved with the Get command, and many of the parameters can be changed with the Set command.

Table 7-9 lists the parameters and whether or not they can be changed. For more information on the Set and Get commands, see Section 7.6, ”Parameter Get and Set Commands” on page 7-22.

Table 7-9 *Parameters*

Device Type	Parameter	Description	Set?	See Page
A	Axes	Returns ID Numbers of ACBs in a logical arm device	No	7-24
M	AxisDesc	Axis type, including TML script version number	No	7-24
M	AxisOffset	Coordinates at the axis’ home position, in mm	No	7-25
C	BootVersion	Boot code version	No	7-25
C	CANBaudRate	CAN baud rate, fixed at 500 K/s	No	7-25
M	CANegativeAxisID	ID of the neighboring axis in the negative direction	Yes	7-25
M	CANegativeDist	Minimum distance required to avoid collision between an axis and the neighboring axis in the negative direction	Yes	7-26
M	CAPositiveAxisID	ID of the neighboring axis in the positive direction	Yes	7-26
M	CAPositiveDist	Minimum distance required to avoid collision between an axis and the neighboring axis in the positive direction	Yes	7-27
C	ClearInt5	Allows Omni axes to start accepting commands after an Int5 stop	Yes	7-27
M	DefAcceleration	Acceleration of the axis during any move command except the Initialize Axis command	Yes	7-28
M	DefInitAcceleration	Acceleration of the axis during the Initialize Axis command	Yes	7-28
M	DefInitSpeed	Initialization speed of the axis, used when executing the Initialize Axis command	Yes	7-29
M	DefSpeed	Default axis movement speed	Yes	7-29
M	DropForceFactor	Force used to drop the disposable tip	Yes	7-30
M	DropSpeed	Travel speed used by the Standard Z or Dual Z axis to drop a disposable tip	Yes	7-30
C	EnableCavroCAN	Enables Cavro® CAN communication support	Yes	7-31
M	ExitLimit	Acceptance limit for a successful exist signal check	Yes	7-31

Table 7-9 *Parameters (continued)*

Device Type	Parameter	Description	Set?	See Page
M	ExitRetractDist	Travel distance in the Z direction that the Z axis uses to retract and search for an exit signal with the CheckForExit command	Yes	7-31
M	ExitRetractSpeed	Speed that the Z axis uses to retract and search for an exit signal	Yes	7-32
M	FirmwareID	Axis firmware part number and revision	No	7-32
C	FWRevision	Firmware part number and revision number	No	7-32
C	FWVersion	CIB firmware version number	No	7-33
M	HighestTemp	Initial (reset) value of the axis' highest recorded temperature in Celsius	Yes	7-33
M	HighestTempAbs	Axis' highest recorded temperature ever, in Celsius	No	7-33
M	InitAcceptanceWindow	Allowable position difference between the two initializations in a double initialization performed with the Initialize Axis command	Yes	7-34
M	InitForceFactor	Force used to search for a hard stop during execution of the Initialize Axis command	Yes	7-34
M	InitMode	Value that indicates the initialization mode of the module	Yes	7-35
C	Int5Mode	Enables or disables pin Int5 triggering stop of axes in motion	Yes	7-35
C	Input1	Status of Input 1 (J9, pin 13 and J8, pin 10)	No	7-36
C	Input2	Status of Input 2 (J9, pin 14 and J8, pin 12)	No	7-36
C	Input3	Status of Input 3 (J9, pin 15 and J8, pin 14)	No	7-36
C	Input4	Status of Input 4 (J4, pin 6)	No	7-37
C	Input5	Status of Input 5 (J5, pin 6)	No	7-37
C	Input6	Status of Input 6 (J18, pin 5)	No	7-37
C	Input7	Status of Input7/INT5 (J17, pin 4)	No	7-37
C	IO1	Status of I/O1 (J4, pin 5), when I/O1 mode is set to output	Yes	7-38
C	IO1Mode	I/O direction for I/O1 (J4, pin 5)	Yes	7-38
C	IO2	Status of I/O2 (J5, pin 5), when I/O2 mode is set to output	Yes	7-39
C	IO2Mode	I/O direction for I/O2 (J5, pin 5)	Yes	7-39
C	IPAddress	IP address of the CIB	Yes	7-40
M	LLDAcceptanceWindow	Maximum acceptable difference between the two detection positions used in double detection	Yes	7-40
M	LLDDetectionMethod	Value that indicates the detection method of the LLD system	Yes	7-41

Table 7-9 *Parameters (continued)*

Device Type	Parameter	Description	Set?	See Page
M	LLDRetractDist	Retract distance for the Z axis after the first detection of a double detection	Yes	7-41
M	LLDRetractOnError	Value that represents whether or not the Z axis will retract to the start position after a “No Liquid Detected” error	Yes	7-42
M	LLDSearchSpeed	Speed used when searching for liquid, in mm/s	Yes	7-42
M	LLDSensitivity	Sensitivity, in increments, used when searching for liquid	Yes	7-42
M	LLDSubmergeDist	Distance the Z axis extends after a successful detection is performed	Yes	7-43
C	MACAddress	MAC address of the CIB	No	7-44
C	Output1	Status of Output 1 (J9, pin 7 and J8, pin 13)	Yes	7-44
C	Output2	Status of Output 2 (J9, pin 8 and J8, pin 15)	Yes	7-44
C	Output3	Status of Output 3 (J17, pin 3)	Yes	7-45
C	Output4	Status of Output 4 (J18, pin 3)	Yes	7-45
M	PickForceFactor	Force used to engage disposable tip during pickup	Yes	7-45
M	PickSpeed	Travel speed used by the Standard Z, Dual Z, or Universal Z with ADP axis to pick up a disposable tip	Yes	7-46
C	PortNumber	Port number on the host PC that the CIB uses to communicate	Yes	7-46
M	Pos	Position of axis per the device encoder	Yes	7-47
C	PowerOnMinutes	Value of the counter for the cumulative number of minutes	Yes	7-47
C	PowerOnMinutesAbs	The highest number of power on minutes ever recorded	No	7-48
C	PowerUps	Value of the counter for the cumulative number of power ups	Yes	7-48
C	PowerUpsAbs	The highest number of power ups ever recorded	No	7-48
M	PWMDutyCycle	High (on) state interval of the PWM output signal period, in 25ns increments	Yes	7-49
M	PWMFrequency	PWM output signal period, in 25ns increments	Yes	7-50
M	RangeHigh	Highest reachable axis position, in mm	No	7-51
M	RangeLow	Lowest reachable axis position, in mm	No	7-51
C	RS485Config	Selects port settings for RS-485 communications	Yes	7-52
C	SerialNumber	Serial number of the CIB	No	7-52
M	Status	Current status of the axis	No	7-52

Table 7-9 *Parameters (continued)*

Device Type	Parameter	Description	Set?	See Page
C	SWRevision	Command Processor part number and revision number	No	7-53
C	SWVersion	Command Processor version number	No	7-53
M	TipPresence	Value that indicates if a tip is attached to the Z axis	No	7-53
M	TotalMoveCount	Initial (reset) count of number of movements performed by the axis	Yes	7-54
M	TotalMoveCountAbs	Cumulative total number of axis moves	No	7-54
M	TotalTravelDist	Initial (reset) value of the total distance traveled with this axis	Yes	7-54
M	TotalTravelDistAbs	Cumulative total distance traveled by this axis	No	7-55
M	TravelPos	Position to which the axis will first move before it moves to the final position set by the MoveAbs command	Yes	7-55
C	UserData	Up to 256 bytes of user data in the CIB nonvolatile memory	Yes	7-56
C	VoltageFU1	Power supply voltage after FU1 from the CIB	No	7-56
C	VoltageFU2	Power supply voltage after FU2 from the CIB	No	7-57
C	VoltageFU3	Power supply voltage after FU3 from the CIB	No	7-57
C	VoltageFU4	Power supply voltage after FU4 from the CIB	No	7-57
M	WorkingLen	Distance between the axis home position and the maximum reachable axis position	Yes	7-57

7.6 Parameter Get and Set Commands

This section describes the Get and Set commands for working with values of the parameters stored in the Command Processor. All parameters have a Get command that retrieves their value. Only parameters that can be set programmatically have Set commands.

The **Get** command retrieves the value of a parameter from the Command Processor. The reply to the Get command includes the command target's device type, device ID, the result of command execution, and the requested property and value. For example, the reply to the command **M1Get,DefSpeed** is **M1Get,DefSpeed,0,500.0/**.

The **Set** command edits the value of a parameter stored in the Command Processor. The reply to the Set command includes the command target's device type, device ID, and result of command execution. A result of 0 indicates

execution success. For example, the reply to the command **M1Set,DefSpeed,500.0** is **M1Set,0**, indicating success.

7.6.1 Possible Accuracy Loss in Set Commands

The Set command accepts floating point values for distance, position, speed, acceleration, and force related properties. These are stored in the Command Processor as floating point values; they are converted to integer internal device units and downloaded to the device when needed for execution of a motion command. For that reason, the user may expect to lose some accuracy due to conversion truncation.

The units used throughout the system are:

- mm - distance and position
- mm/s - speed
- mm/s^2 - acceleration
- Internal Unit - force

7.6.2 Saving Changes to Parameter Values

When you use the Set command to change the value of a parameter, the changed value is stored in RAM in the Command Processor. When the Command Processor is shut down, any changes made using Set commands are lost, and the value of the parameters reverts to its initial value. This enables a developer to specify a set of default parameter values that will take effect for the duration of a particular application, without impacting the initial power-on defaults.

Except for the **Set,Pos** command, which automatically saves the position permanently, you can use the Configurator software to permanently change the value of a parameter (i.e., to change its power-on default value). Changes made through the Configurator are permanently saved by the CIB. See the Configurator documentation for more information.

7.6.3 Error List

All Get and Set commands can potentially generate the same errors. They are:

Generated Errors:

2 -- Invalid Command
3 -- Invalid Operand - Out of Range
5 -- Device Not Implemented
8 -- Device Busy
12 - Invalid number of operands

7.6.4 List of Parameters and their Get and Set Commands

This section contains an alphabetical list of parameters stored in the Command Processor, along with the Get and Set commands that can be used with each parameter.

WARNING! *Take care to ensure that the values of any parameters you set are within the valid range for your robot hardware (such as axis dimensions and travel length).*

Caution! *Setting parameters to values lower than the recommended minimum or higher than the recommended maximum may affect Omni Robot performance. Operation may become unstable even though the parameter value was accepted.*

Axes

Synopsis: The ID numbers of the axis control boards (ACBs) that make up a logical Arm device

Values: Valid ID number (see description)

Syntax: **`A<address>Get,Axes/`**

Description: This is the only Get command that can be used for an Arm/Axis group (device type A). Axes is a read-only parameter and its value cannot be changed.

Example: Get the ID of the axes that make up the Arm at address 0:

```
A0Get,Axes/
```

A response of: *A0Get-Axes,0,3,2,5/* indicates the Get,Axes command was successfully executed and the Arm has axes with the ID numbers of 3, 2, and 5.

AxisDesc

Synopsis: The axis type, including the TML script version number

Values: Valid axis type

Syntax: **`M<AxisID>Get,AxisDesc/`**

Description: AxisDesc is a read-only parameter and its value cannot be changed.

The form is aabbcccc, where:

- aa = major version
- bb = minor version
- cccc = axis type.

Example: Get the axis type for the axis at address 1:

```
M1Get,AxisDesc/
```

AxisOffset

Synopsis: The coordinates at the axis' home position, in mm

Values: Valid coordinates

Syntax: **M<*AxisID*>Get,AxisOffset/**

Description: AxisOffset is a read-only parameter and its value cannot be changed.

Example: Get the axis offset for the axis at address 1:

```
M1Get,AxisOffset/
```

BootVersion

Synopsis: The boot code version

Values: Valid version number

Syntax: **C0Get,BootVersion/**

Description: BootVersion is a read-only parameter and its value cannot be changed.

Example: Get the version number of the boot code for the CIB:

```
C0Get,BootVersion/
```

CANBaudRate

Synopsis: The CAN baud rate

Values: Valid baud rate

Syntax: **C0Get,CANBaudRate/**

Description: This value is fixed at 500 K/s, and cannot be changed by the user.

Example: Get the baud rate for the CAN:

C0Get,CANBaudRate/

CANegativeAxisID

Synopsis: The ID of the neighboring axis in the negative direction

Values: 0 to 15

Syntax: **M<*AxisID*>Get,CANegativeAxisID/**
M<*AxisID*>Set,CANegativeAxisID,<*ID*>/

Description: Valid values are 0-15. If there is no neighboring axis in the negative direction, the value is zero. Getting this parameter can be used to determine whether there is a neighboring axis.

Example: Get the ID of the neighboring axis in the negative direction:

```
M1Get,CANegativeAxisID/
```

Set the ID of the neighboring axis in the negative direction to 2:

```
M1Set,CANegativeAxisID,2/
```

CANegativeDist

Synopsis: The minimum distance required to avoid collision between an axis and the neighboring axis in the negative direction

Values: 1 to 10,000 mm

Syntax:
```
M<AxisID>Get,CANegativeAxisID/
M<AxisID>Set,CANegativeAxisID,<Distance>/
```

Description: Valid values are from 1 to 10,000 mm. If there is no neighboring axis in the negative direction, set the value to zero. Use Get,CANegativeAxisID to determine if there is a neighbor in the negative direction.

Calibrate the robot *before* setting this parameter.

Example: Get the minimum distance of the neighboring axis in the negative direction:

```
M1Get,CANegativeDist/
```

Set the minimum distance of the neighboring axis in the negative direction to 10 mm:

```
M1Set,CANegativeDist,10/
```

CAPositiveAxisID

Synopsis: The ID of the neighboring axis in the positive direction

Values: 0 to 15

Syntax:
```
M<AxisID>Get,CAPositiveAxisID/
M<AxisID>Set,CAPositiveAxisID,<ID>/
```

Description: Valid values are 0-15. If there is no neighboring axis in the positive direction, the value is zero. Getting this parameter can be used to determine whether there is a neighboring axis.

Example: Get the ID of the neighboring axis in the positive direction:

```
M1Get,CAPositiveAxisID/
```

Set the ID of the neighboring axis in the positive direction to 2:

```
M1Set,CAPositiveAxisID,2/
```

CAPositiveDist

Synopsis: The minimum distance required to avoid collision between an axis and the neighboring axis in the positive direction

Values: 1 to 10,000 mm

Syntax:

```
M<AxisID>Get,CAPositiveAxisID/
M<AxisID>Set,CAPositiveAxisID,<Distance>/
```

Description: Valid values are from 1 to 10,000 mm. If there is no neighboring axis in the positive direction, set the value to zero. Use Get,CAPositiveAxisID to determine if there is a neighbor in the positive direction.

Calibrate the robot *before* setting this parameter.

Example: Get the minimum distance of the neighboring axis in the positive direction:

```
M1Get,CAPositiveDist/
```

Set the minimum distance of the neighboring axis in the positive direction to 10 mm:

```
M1Set,CAPositiveDist,10/
```

ClearInt5

Synopsis: Allows Omni axes to start accepting commands after a stop triggered by Int5.

Values: 0

Syntax:

```
C0Set,ClearInt5,0/
```

Description: When an Int5 event is triggered, the system stops with a controlled shutdown of all ACB axes. This also blocks the ACBS from receiving commands from the CIB.

This command re-enables the Omni axes to receive commands from the CIB. 0 is the only valid value.

Example: Re-enable the Omni axes to receive commands from the CIB:

```
C0Set,ClearInt5,0/
```

DefAcceleration

Synopsis: The acceleration of the axis during any move command, except the CavroInit command. (See the DefInitAcceleration parameter for the CavroInit command.)

Values: 1 to 100,000 mm/s^2

Syntax:
```
M<AxisID>Get,DefAcceleration/
M<AxisID>Set,DefAcceleration,<Accel>/
```

Description: Acceleration values are based on calculations. The actual acceleration of the X axis and Y axis could be lower due to the positioning algorithm depending on travel distance and speed settings. The default values are:

- X axis: 3500 mm/s^2
- Y axis: 3900 mm/s^2
- Standard Z axis: 6000 mm/s^2
- Universal Z axis: 1000 mm/s^2
- Dual Z axis: 6000 mm/s^2

Example: Get the acceleration of axis 1:

```
M1Get,DefAcceleration/
```

Set the acceleration of axis 1 to 1200:

```
M1Set,DefAcceleration,1200/
```

DefInitAcceleration

Synopsis: The acceleration of the axis during the CavroInit command

Values: 1 to 100,000 mm/s^2

Syntax:
```
M<AxisID>Get,DefInitAcceleration/
M<AxisID>Set,DefInitAcceleration,<Accel>/
```

Description: Acceleration values are based on calculations. The actual acceleration of the X axis and Y axis could be lower due to the positioning algorithm depending on travel distance and speed settings. The default values are:

- X axis: 3500 mm/s^2
- Y axis: 3900 mm/s^2
- Standard Z axis: 6000 mm/s^2
- Universal Z axis: 1000 mm/s^2

- Dual Z axis: 6000 mm/s^2

Example: Get the initialization acceleration of axis 1:

```
M1Get,DefInitAcceleration/
```

Set the initialization acceleration of axis 1 to 2000:

```
M1Set,DefInitAcceleration,2000/
```

DefInitSpeed

Synopsis: The initialization speed of the axis, which is used when executing the CavroInit command

Values: 0.1 to 200 mm/s

Syntax:
```
M<AxisID>Get,DefInitSpeed/
M<AxisID>Set,DefInitSpeed,<Speed>/
```

Description: The default values are:

- X axis: 80 mm/s
- Y axis: 80 mm/s
- Z axis: 40 mm/s

Example: Get the initialization speed of axis 1:

```
M1Get,DefInitSpeed/
```

Set the initialization speed of axis 1 to 80:

```
M1Set,DefInitSpeed,80/
```

DefSpeed

Synopsis: The default axis movement speed

Values: 0.1 to 10000 mm/s

Syntax:
```
M<AxisID>Get,DefSpeed/
M<AxisID>Set,DefSpeed,<Speed>/
```

Description: The default values are:

- X axis: 800 mm/s
- Y axis: 600 mm/s
- Standard Z axis: 600 mm/s
- Universal Z axis: 250 mm/s
- Dual Z axis: 600 mm/s

Example: Get the default speed of axis 1:

```
M1Get,Defspeed/
```

Set the default speed of axis 1 to 800:

```
M1Set,DefSpeed,800/
```

DropForceFactor

Synopsis: The force used to drop the disposable tip

Values: 8000 to 22000

Syntax:
M<*AxisID*>Get,DropForceFactor/
M<*AxisID*>Set,DropForceFactor,<*Force*>/

Description: The default value is 17000. The higher the force factor, the stronger the applied force is.

Note: *Because the Drop Tip command is not yet supported for the Universal Z axis, using the DropForceFactor command with the Universal Z axis will have no effect.*

Example: Get the force used to drop the disposable tip:

```
M3Get,DropForceFactor/
```

Set the force used to drop the disposable tip to 20000:

```
M3Set,DropForceFactor,20000/
```

DropSpeed

Synopsis: The travel speed used by the Standard Z or Dual Z axis to drop a disposable tip

Values: 1 to 150 mm/s

Syntax:
M<*AxisID*>Get,DropSpeed/
M<*AxisID*>Set,DropSpeed,<*Speed*>/

Description: The default value is 60 mm/s.

Note: *Because the Drop Tip command is not yet supported for the Universal Z axis, using the DropSpeed command with the Universal Z axis will have no effect.*

Example: Get the travel speed used to drop the disposable tip:

```
M3Get,DropSpeed/
```

Set the travel speed used to drop the disposable tip to 80:

```
M3Set,DropSpeed,80/
```

EnableCavroCAN

Synopsis: Enables Cavro® CAN communication support.

Values: 0

Syntax: `C0Set,EnableCavroCAN,0/`

Description: The command needs to be executed every time the Command Processor driver is reinstantiated. 0 is the only valid value.

Example: Enable Cavro® CAN communication support:

```
C0Set,EnableCavroCAN,0/
```

ExitLimit

Synopsis: The acceptance limit for a successful exit signal check

Values: 0 to 50 mm

Syntax:
`M<AxisID>Get,ExitLimit/`
`M<AxisID>Set,ExitLimit,<Limit>/`

Description: The default value is 5 mm.

Example: Get the acceptance limit for a successful exit signal check:

```
M3Get,ExitLimit/
```

Set the acceptance limit for a successful exit signal check to 50:

```
M3Set,ExitLimit,50/
```

ExitRetractDist

Synopsis: The travel distance in the Z direction that the Z axis will move when searching for an exit signal with the CheckForExit command

Values: 1 to 100 mm

Syntax:
`M<AxisID>Get,ExitRetractDist/`
`M<AxisID>Set,ExitRetractDist,<RetractDist>/`

Description: The default value is 10.

Example: Get the retract travel distance:

```
M3Get,ExitRetractDist/
```

Set the retract travel distance to 40:

```
M3Set,ExitRetractDist,40/
```

ExitRetractSpeed

Synopsis: The speed that the Z axis will use when retracting and searching for an exit signal

Values: 1 to 150 mm/s

Syntax:

```
M<AxisID>Get,ExitRetractSpeed/
M<AxisID>Set,ExitRetractSpeed,<Speed>/
```

Description: The default value is 60.

Example: Get the retract travel speed:

```
M3Get,ExitRetractSpeed/
```

Set the retract travel speed to 35:

```
M3Set,ExitRetractSpeed,35/
```

FirmwareID

Synopsis: The part number and revision number of the firmware in the Axis PCB

Values: Valid axis firmware part and revision number

Syntax:

```
M<AxisID>Get,FirmwareID/
```

Description: FirmwareID is a read-only parameter and its value cannot be changed.

Example: Get axis 1 firmware part number and revision:

```
M1Get,FirmwareID/
```

FWRevision

Synopsis: The CIB firmware part number and revision number

Values: Valid firmware part and revision number

Syntax:

```
C0Get,FWRevision/
```

Description: FWRevision is a read-only parameter and its value cannot be changed.

Example: Get the firmware part number and revision:

```
C0Get,FWRevision/
```

FWVersion

Synopsis: The CIB firmware version number

Values: Valid CIB firmware version number

Syntax: `C0Get,FWVersion/`

Description: FWVersion is a read-only parameter and its value cannot be changed.

Example: Get the CIB firmware version number:

```
C0Get,FWVersion/
```

HighestTemp

Synopsis: The initial (reset) value of the axis' highest recorded temperature in Celsius

Values: Set: 0 only; Get: any valid temperature in Celsius

Syntax: `M<AxisID>Get,HighestTemp/`
`M<AxisID>Set,HighestTemp,0/`

Description: The Set command can only be used to change the value of the parameter to 0. The Get command returns the highest recorded temperature since the most recent Set command.

Example: Get the highest recorded temperature in Celsius:

```
M1Get,HighestTemp/
```

Set the highest recorded temperature to zero:

```
M1Set,HighestTemp,0/
```

HighestTempAbs

Synopsis: The axis' highest recorded temperature ever, in Celsius

Values: Any valid temperature in Celsius

Syntax: `M<AxisID>Get,HighestTempAbs/`

Description: HighestTempAbs is a read-only parameter and its value cannot be changed.

Example: Get the highest ever recorded temperature in Celsius:

```
M1Get,HighestTempAbs/
```

InitAcceptanceWindow

Synopsis: The allowable position difference between the two initializations in a double initialization performed with the CavroInit command

Values: 0 to 10 mm

Syntax:
```
M<AxisID>Get,InitAcceptanceWindow/
M<AxisID>Set,InitAcceptanceWindow,<Window>/
```

Description: Applies to Z axis only. The default value is 0.5 mm.

Example: Get the allowable position difference between two initializations:

```
M3Get,InitAcceptanceWindow/
```

Set the allowable position difference between two initializations to 5.5 mm:

```
M3Set,InitAcceptanceWindow,5.5/
```

InitForceFactor

Synopsis: The force that is used to search for a hard stop during the execution of the CavroInit command

Values: 6000 to 22000

Syntax:
```
M<AxisID>Get,InitForceFactor/
M<AxisID>Set,InitForceFactor,<Force>/
```

Description: Applies to Z axes only. The higher the force factor, the stronger the applied force is. Default values are:

- Standard Z axis: 8000
- Universal Z axis: 22000
- Dual Z axis: 8000

Example: Get the force used to search for a hard stop:

```
M3Get,InitForceFactor/
```

Set the force used to search for a hard stop to 20000:

```
M3Set,InitForceFactor,20000/
```

InitMode

Synopsis: The value that indicates the initialization mode of the module

Values: 0 (double init) or 1 (single init)

Syntax: **M<*AxisID*>Get,InitMode/**
M<*AxisID*>Set,InitMode,<*Mode*>/

Description: Applies to Z axis only. The default value is 0 (double init).

Caution! *Because the axis uses a hard-stop for initialization, using single init increases the risk of the axis initializing at an incorrect position due to mechanical interference.*

Example: Get the initialization mode of the module:

```
M3Get,InitMode/
```

Set the initialization mode of the module to single init:

```
M3Set,InitMode,1/
```

Int5Mode

Synopsis: The value that indicates what action is performed when Int5 is triggered

Values: 0 or 1

Syntax: **C0Set,Int5Mode,<*Value*>/**

Description: Enables or disables the use of input from pin Int5 to trigger stopping of axes in motion:

- 0: No action performed when Int5 triggered.
- 1: All axis motor movements are stopped when Int5 is triggered.

Note: It is strongly recommended to exit the application and cycle the CIB power after a stop.

Objects not driven by the axis motor, such as gripper fingers or diluters are not stopped.

Stoppage of axes is not instant but should be less than one second. Stoppage depends on four factors:

- the time it takes for the CIB to receive the INT5 signal
- the time it takes for the tml script to send the stop command
- the speed of the axis when the stop command is received
- the acceleration of the axis when the stop command is received

The stop function is dependent upon communication between the axes and the CIB. If a communication breakdown should happen during the stop, it is possible that not all devices will be stopped.

Example: Set the action performed when Int5 is triggered to no action:

```
C0Set,Int5Mode,0/
```

Input1

Synopsis: Status of Input 1 (J9, pin 13, and J8, pin 10)

Values: 0 or 1

Syntax: `C0Get,Input1/`

Description: Input1 is a read-only parameter and its value cannot be changed.

Example: Get the status of Input1:

```
C0Get,Input1/
```

Input2

Synopsis: Status of Input 2 (J9, pin 14, and J8, pin 12)

Values: 0 or 1

Syntax: `C0Get,Input2/`

Description: Input2 is a read-only parameter and its value cannot be changed.

Example: Get the status of Input2:

```
C0Get,Input2/
```

Input3

Synopsis: Status of Input 3 (J9, pin 15, and J8, pin 14)

Values: 0 or 1

Syntax: `C0Get,Input3/`

Description: Input3 is a read-only parameter and its value cannot be changed.

Example: Get the status of Input3:

```
C0Get,Input3/
```

Input4

Synopsis: Status of Input 4 (J4, pin 6)

Values: 0 or 1

Syntax: `C0Get,Input4/`

Description: Input4 is a read-only parameter and its value cannot be changed.

Example: Get the status of Input4:

```
C0Get,Input4/
```

Input5

Synopsis: Status of Input 5 (J5, pin 6)

Values: 0 or 1

Syntax: `C0Get,Input5/`

Description: Input5 is a read-only parameter and its value cannot be changed.

Example: Get the status of Input5:

```
C0Get,Input5/
```

Input6

Synopsis: Status of Input 6 (J18, pin 5)

Values: 0 or 1

Syntax: `C0Get,Input6/`

Description: Input6 is a read-only parameter and its value cannot be changed.

Example: Get the status of Input6:

```
C0Get,Input6/
```

Input7

Synopsis: Status of Input 7/INT5 (J17, pin 4)

Values: 0 or 1

Syntax: `C0Get,Input7/`

Description: Input7 is a read-only parameter and its value cannot be changed. Low to high transition will trigger event ID INT5.

Example: Get the status of Input7:

```
C0Get,Input7/
```

IO1

Synopsis: The status of I/O1 (J4, pin 5), when I/O1 mode is set to output

Values: 0 or 1

Syntax:
```
C0Get,IO1/
C0Set,IO1,<Value>/
```

Description: If I/O1 is acting as output (*I/O1Mode* = 1), valid values are 0 (Low) or 1 (High).

If I/O1 is acting as input (*I/O1Mode* = 0), setting I/O1 has no effect until the mode is changed to output.

Example: Get the I/O 1 status:

```
C0Get,IO1/
```

Set the I/O 1 status to Low:

```
C0Set,IO1,0/
```

Set the I/O 1 status to High:

```
C0Set,IO1,1/
```

IO1Mode

Synopsis: I/O direction for I/O1 (J4, pin 5)

Values: Numbers that represent Input or Output mode

Syntax:
```
C0Get,IO1Mode/
C0Set,IO1Mode,<Mode>/
```

Description: I/O1 is bidirectional, and can be set as Input or Output. Default is Input. Valid values are:

- 0: Input
- 1: Output

Example: Get the I/O 1 direction:

```
C0Get,IO1Mode/
```

Set the I/O 1 direction to Input:

```
C0Set,IO1Mode,0/
```

Set the I/O 1 direction to Output:

```
C0Set,IO1Mode,1/
```

IO2

Synopsis: The status of I/O2 (J5, pin 5), when I/O2 mode is set to output

Values: 0 or 1

Syntax:
```
C0Get,IO2/
C0Set,IO2,<Value>/
```

Description: If I/O2 is acting as output (*I/O1Mode* = 1), valid values are 0 (Low) or 1 (High).

If I/O2 is acting as input (*I/O1Mode* = 0), setting I/O1 has no effect until the mode is changed to output.

Example: Get the I/O 2 status:

```
C0Get,IO2/
```

Set the I/O 2 status to Low:

```
C0Set,IO2,0/
```

Set the I/O 2 status to High:

```
C0Set,IO2,1/
```

IO2Mode

Synopsis: I/O direction for I/O2 (J5, pin 5)

Values: Numbers that represent Input or Output mode

Syntax:
```
C0Get,IO2Mode/
C0Set,IO2Mode,<Mode>/
```

Description: I/O2 is bidirectional, and can be set as Input or Output. Default is Input. Valid values are:

- 0: Input
- 1: Output

Example: Get the I/O 2 direction:

```
C0Get,IO2Mode/
```

Set the I/O 2 direction to Input:

```
C0Set,IO2Mode,0/
```

Set the I/O 2 direction to Output:

```
C0Set,IO2Mode,1/
```

IPAddress

Synopsis: The IP address of the CIB

Values: A valid IP address

Syntax:
```
C0Get,IPAddress/
C0Set,IPAddress,<Address>/
```

Description: The address must be a valid IP address in the form *nnn.nnn.nnn.nnn*, where nnn is a value ranging from 0 to 255. The CIB subnet mask is 255.255.255.0; the host PC must be configured to the same subnet. The default value is 192.168.0.198.

Example: Get the IP address:

```
C0Get,IPAddress/
```

Set the IP address to 192.168.0.198:

```
C0Set,IPAddress,192.168.0.198/
```

LLDAcceptanceWindow

Synopsis: The maximum acceptable difference between the two detection positions used in double detection

Values: 0.5 to 50 mm

Syntax:
```
M<AxisID>Get,LLDAcceptanceWindow/
M<AxisID>Set,LLDAcceptanceWindow,<Window>/
```

Description: This setting applies to both positive and negative differences. The default value is 5.

Example: Get the maximum acceptable difference between two detection positions:

```
M3Get,LLDAcceptanceWindow/
```

Set the maximum acceptable difference between two detection positions to 10 mm:

```
M3Set,LLDAcceptanceWindow,10/
```

LLDDetectionMethod

Synopsis: The value that indicates the detection method of the LLD system

Values: 0 or 1

Syntax:
M<*AxisID*>Get,LLDDetectionMethod/
M<*AxisID*>Set,LLDDetectionMethod,<*Method*>/

Description: Valid values are 0 (for double detection) and 1 (for single detection); the default value is 0.

Example: Get the detection method:

```
M3Get,LLDDetectionMethod/
```

Set the detection method to single detection:

```
M3Set,LLDDetectionMethod,1/
```

LLDRetractDist

Synopsis: The retract distance for the Z axis after the first detection of a double detection

Values: *LLDAcceptanceWindow* to 100 mm

Syntax:
M<*AxisID*>Get,LLDRetractDist/
M<*AxisID*>Set,LLDRetractDist,<*RetractDist*>/

Description: Valid values are between the current value of *LLDAcceptanceWindow* and 100 mm. The default value is 10 mm.

Example: Get the retract distance:

```
M3Get,LLDRetractDist/
```

Set the retract distance to 5mm:

```
M3Set,LLDRetractDist,5/
```

LLDRetractOnError

Synopsis: The value that represents whether or not the Z axis will retract to the start position after a “No Liquid Detected” error

Values: 0 or 1

Syntax: **M<*AxisID*>Get,LLDRetractOnError/**
M<*AxisID*>Set,LLDRetractOnError,<*Retract*>/

Description: Valid values are 0 (retract) and 1 (do not retract); the default value is 0.

Example: Get the Z axis retract-on-error value:

```
M3Get,LLDRetractOnError/
```

Set the Z axis to retract to the start position after a “No Liquid Detected” error:

```
M3Set,LLDRetractOnError,0/
```

LLDSearchSpeed

Synopsis: The speed to be used when searching for liquid, in mm/s

Values: 1 to 150 mm/s

Syntax: **M<*AxisID*>Get,LLDSearchSpeed/**
M<*AxisID*>Set,LLDSearchSpeed,<*SearchSpeed*>/

Description: The default search speed value is 60 mm/s.

Example: Get the search speed:

```
M3Get,LLDSearchSpeed/
```

Set the search speed to 50 mm/s:

```
M3Set,LLDSearchSpeed,50/
```

LLDSensitivity

Synopsis: The sensitivity, in increments, to be used when searching for liquid

Values: 3 (most sensitive) to 120 (least sensitive)

Syntax: **M<*AxisID*>Get,LLDSensitivity/**
M<*AxisID*>Set,LLDSensitivity,<*LLDSensitivity*>/

Description: Valid values are from 3 (highest sensitivity) to 120 (lowest sensitivity); the default value is 63.

The *LLDSensitivity* setting results in a PWM duty cycle setting on the ACB J7 pin 1.

Note that in addition to increasing the liquid sensitivity, setting the *LLDSensitivity* parameter to a more sensitive setting makes the cLLD system more susceptible to outside interference. Conversely, setting the cLLD system to a less sensitive setting makes it less susceptible to interference.

See Section C.10, "Preventing Interference Between Two Robots" on page C-38 and Section C.11, "Preventing Z Axis cLLD Interference in Robots with Multiple Z Axes" on page C-39 for restrictions on the sensitivity settings.

Caution! *An inappropriate sensitivity setting may cause false liquid level detection.*

Example: Get the sensitivity level:

```
M3Get,LLDSensitivity/
```

Set the sensitivity level to 90:

```
M3Set,LLDSensitivity,90/
```

LLDSubmergeDist

Synopsis: The distance that the Z axis will extend after a successful detection is performed

Values: 0 to 100 mm

Syntax:

```
M<AxisID>Get,LLDSubmergeDist/
M<AxisID>Set,LLDSubmergeDist,<SubmergeDist>/
```

Description: The default value is 0.

Example: Get the submerge distance:

```
M3Get,LLDSubmergeDist/
```

Set the submerge distance to 5 mm:

```
M3Set,LLDSubmergeDist,5/
```

MACAddress

Synopsis: The MAC address of the CIB

Values: A valid MAC address

Syntax: **`C0Get,MACAddress/`**

Description: MACAddress is a read-only parameter and its value cannot be changed.

Example: Get the MAC address:

```
C0Get,MACAddress/
```

Output1

Synopsis: The status of Output 1 (J9, pin 7 and J8, pin 13)

Values: 0 or 1

Syntax: **`C0Get,Output1/`**
`C0Set,Output1,<Value>/`

Description: Valid values are 0 (Low) and 1 (High). Default is Low (0).

Example: Get the Output1 status:

```
C0Get,Output1/
```

Set the Output1 status to high:

```
C0Set,Output1,1/
```

Output2

Synopsis: The status of Output 2 (J9, pin 8 and J8, pin 15)

Values: 0 or 1

Syntax: **`C0Get,Output2/`**
`C0Set,Output2,<Value>/`

Description: Valid values are 0 (Low) and 1 (High). Default is Low (0).

Example: Get the Output2 status:

```
C0Get,Output2/
```

Set the Output2 status to high:

```
C0Set,Output2,1/
```

Output3

Synopsis: The status of Output 3 (J17, pin 3)

Values: 0 or 1

Syntax:
```
C0Get,Output3/
C0Set,Output3,<Value>/
```

Description: Valid values are 0 (Low) and 1 (High). Default is Low (0).

Example: Get the Output3 status:

```
C0Get,Output3/
```

Set the Output3 status to high:

```
C0Set,Output3,1/
```

Output4

Synopsis: The status of Output 4 (J18, pin 3)

Values: 0 or 1

Syntax:
```
C0Get,Output4/
C0Set,Output4,<Value>/
```

Description: Open drain power output: valid values are 0 (Low impedance to ground) and 1 (High impedance). Initial value is 0 (Low impedance to ground).

Example: Get the Output4 status:

```
C0Get,Output4/
```

Set the Output4 status to high:

```
C0Set,Output4,1/
```

PickForceFactor

Synopsis: The force used to engage the disposable tip during pickup

Values: 1000 to 20000

Syntax:
```
M<AxisID>Get,PickForceFactor/
M<AxisID>Set,PickForceFactor,<Force>/
```

Description: This force can be set to allow for variances in disposable tips. The higher the force factor, the stronger the applied force is. When using Tecan disposable tips, do not change this value from the default setting. Default values are:

- Standard Z axis: 11800
- Universal Z axis: 7000
- Dual Z axis: 11800

Example: Get the force factor:

```
M3Get,PickForceFactor/
```

Set the force factor to 3000:

```
M3Set,PickForceFactor,3000/
```

PickSpeed

Synopsis: The travel speed used by the Standard Z, Dual Z, and Universal Z with ADP axis to pick up a disposable tip

Values: 1 to 150 mm/s

Syntax:
```
M<AxisID>Get,PickSpeed/
M<AxisID>Set,PickSpeed,<Speed>/
```

Description: This speed can be set to allow for variances in disposable tips. When using Tecan disposable tips, do not change this value from the default setting. The default value is 60 mm/s.

Example: Get the travel speed:

```
M3Get,PickSpeed/
```

Set the travel speed to 80 mm/s:

```
M3Set,PickSpeed,80/
```

PortNumber

Synopsis: The port number on the host PC that the CIB uses for communications

Values: 1000 to 20000

Syntax:
```
C0Get,PortNumber/
C0Set,PortNumber,<Number>/
```

Description: The default value is 9898.

Example: Get the port number:

```
C0Get,PortNumber/
```

Set the port number to 9898:

```
C0Set,PortNumber,9898/
```

Pos

Synopsis: The position of the axis per the device encoder

Values: Axis coordinates

Syntax:
M<*AxisID*>Get,Pos/
M<*AxisID*>Set,Pos,<*Pos*>/

Description: **Get,Pos** returns the actual (physical) position per encoder feedback.

Set,Pos can be used to calibrate axis coordinates to worktable coordinates. Once set, the system automatically re-calculates the axis offset, *RangeLow* and *RangeHigh* parameters.

Caution! *The* ***Set,Pos*** *command will not change the TravelPos setting for any axis automatically. Make sure that the TravelPos setting for each axis is appropriate, after execution of the* ***Set,Pos*** *command.*

WARNING! *Using the Set command will change the Omni Robot coordinate system and the behavior of the system!*

Example: Get the physical position of axis 1:

```
M1Get,Pos/
```

Set the physical position of axis 1 to 100.5:

```
M1Set,Pos,100.5/
```

PowerOnMinutes

Synopsis: The value of the counter for the cumulative number of power on minutes

Values: Set: 0 only; Get: 0 to 4294967295 (0xFFFFFFFF)

Syntax:
C0Get,PowerOnMinutes/
C0Set,PowerOnMinutes,0/

Description: The Set command can only be used to change the value of the parameter to 0. The Get command returns the number of power on minutes since the most recent Set command.

Example: Get the power on minutes:

```
C0Get,PowerOnMinutes/
```

Set the power on minutes to 0:

```
C0Set,PowerOnMinutes,0/
```

PowerOnMinutesAbs

Synopsis: The highest number of power on minutes ever recorded.

Values: 0 to 4294967295 (0xFFFFFFFF)

Syntax: `C0Get,PowerOnMinutesAbs/`

Description: PowerOnMinutesAbs is a read-only parameter and its value cannot be changed.

Example: Get the highest recorded power on minutes:

```
C0Get,PowerOnMinutesAbs/
```

PowerUps

Synopsis: The value of the counter for the cumulative number of power ups

Values: Set: 0 only; Get: 0 to 65535

Syntax:
```
C0Get,PowerUps/
C0Set,PowerUps,0/
```

Description: The Set command can only be used to change the value of the parameter to 0. The Get command returns the number of power ups since the most recent Set command.

Example: Get the number of power ups:

```
C0Get,PowerUps/
```

Set the number of power ups to 0:

```
C0Set,PowerUps,0/
```

PowerUpsAbs

Synopsis: The highest number of power ups ever recorded

Values: 0 to 65535

Syntax: `C0Get,PowerUpsAbs/`

Description: PowerUpsAbs is a read-only parameter and its value cannot be changed.

Example: Get the highest recorded power on minutes:

```
C0Get,PowerUpsAbs/
```

PWMDutyCycle

Synopsis: The high (on) state interval of the PWM output signal period for all Z axis types, in 25ns increments. Not available for X and Y axes. Output signal frequency on J7 pin 1 of the ACB board.

Values: Valid range: 0 (output signal steady at low state) to (PWMFrequency-1); the default value is 63.

Syntax:

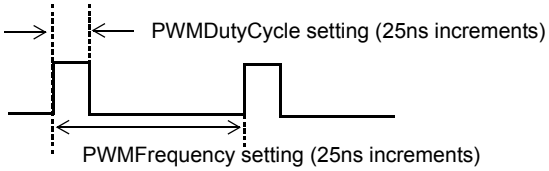
```
M<AxisID>Get,PWMDutyCycle/
M<AxisID>Set,PWMDutyCycle,<Value>/
```

Description: This command sets the On (= high) state time, while the Off (= low) state time is calculated to maintain the period of the output signal as set by the **Set,PWMFrequency** command.

A special case represents the value of 0 (zero), which disables the toggling of the output, leaving the output permanently at its low state. Changing the setting from 0 to a different value will cause a delay in restoring the cLLD functionality. The required time for the cLLD to be fully operable depends on the new sensitivity setting, for example:

- ≥ 50 ms for sensitivity 63 (default), or
- ≥ 500 ms for sensitivity 3 (worst case)

This applies to all commands which include the LLDSensitivity parameter (see also **Set,LLDSensitivity, cLLDSelfTest, CheckForExit, DetectLiquid)**.



The PWM duty cycle is also set through the **LLDSensitivity** command. However, the LLDSensitivity setting becomes effective only at execution of the **DetectLiquid** command or the **CheckForExit** command.

The **PWMDutyCycle** command makes the PWM capabilities of the ACB board available for general purpose unrestricted use, while the **LLDSensitivity** command. allows a limited range only. See the **Set,LLDSensitivity** command for information on its setting range.

The Set,PWMDutyCycle command applies the new settings on the ACB immediately. The setting remains valid until one of the liquid detection commands is sent, when the duty cycle set by LLDSensitivity takes effect.

Example: Get the interval of the PWM output signal period:

```
M3Get,PWMDutyCycle/
```

Set the interval of the PWM output signal period to 63:

```
M3Set,PWMDutyCycle,63/
```

PWMFrequency

Synopsis: The PWM output signal period for all Z-axis types, in 25ns increments. Not available for X and Y axes. Output signal frequency on J7 pin 1 of the ACB board.

Values: (PWMFrequency+1) to 65535; the value is initialized at system startup for Z axes starting from 127 with an increment of 2 to avoid conflict.

Syntax: Get: **M<*AxisID*>Get,PWMFrequency**
Set: **M<*AxisID*>Set,PWMFrequency,<*Value*>/**

Description: Setting PWMFrequency overwrites the frequency for operating the cLLD board that is set automatically by the Command Processor at start up. It is used to separate the frequencies in configurations with multiple Z axes to prevent cLLD interference.

The Command Processor sets the start up frequencies based on the address sequence of all applicable axes types (Z-axes types only) starting with 127 and incrementing by 2 as shown here:

X axis Arm1:	Address 1	--> N/A
Y axis Arm1:	Address 2	--> N/A
Z axis Arm1:	Address 3	--> PWMFrequency = 127
X axis Arm2:	Address 4	--> N/A
Y axis Arm2:	Address 5	--> N/A
Z axis Arm2:	Address 6	--> PWMFrequency = 129
2nd Z axis Arm2:	Address 8	--> PWMFrequency = 131

When setting the PWMFrequency, it is the user's responsibility to ensure that each Z axis ACB has a unique PWMFrequency setting that maintains at least a 2 increment difference from any other Z axis on an Omni robot. The setting range for ACB boards connected to a cLLD board should be limited to 127 – 141.

This command sets the total period of the output signal; the high (On) state is controlled by the **PWMDutyCycle** command; the low state is calculated as the *PWMFrequency –PWMDutyCycle*.

For example, if *PWMDutyCycle* is set to 30, setting *PWMFrequency* to 127 will generate an output signal with an On time of 30 x 25ns = 750ns,a period of

(127+1) x 25ns = 3200ns, and an Off time of (127 – On time) x 25ns = 98 x 25ns = 2450ns.

```
Actual Frequency
= 1/{ (PWMFrequency + 1) * 25e-09 s} Hz
= 1/{ (127 + 1) * 25e-09 s} Hz
= 1/{3200e-09 s} Hz
= 312.5kHz
```

Caution! *An inappropriate PWMFrequency setting may cause false liquid level detection. See Section C.10, "Preventing Interference Between Two Robots" on page C-38 and Section C.11, "Preventing Z Axis cLLD Interference in Robots with Multiple Z Axes" on page C-39 for more details.*

Example: Get the *PWMFrequency* settings:

```
C0Get,PWMFrequency/
```

Set the *PWMFrequency* to 127 on the Z axis (Address 3).

```
M3Set,PWMFrequency,127/
```

RangeHigh

Synopsis: The highest reachable axis position, in mm

Values: X axis left: **AxisOffset + WorkingLen**
X axis right: **AxisOffset**
Y axis: **AxisOffset + WorkingLen**
Z axis: **AxisOffset**

Syntax: **M<*AxisID*>Get,RangeHigh/**

Description: RangeHigh is calculated; it cannot be set.

Example: Get the highest reachable axis position:

```
M1Get,RangeHigh/
```

RangeLow

Synopsis: The lowest reachable axis position, in mm

Values: X axis left: **AxisOffset**
X axis right: **AxisOffset - WorkingLen**
Y axis: **AxisOffset**
Z axis: **AxisOffset - WorkingLen**

Syntax: **M<*AxisID*>Get,RangeLow/**

Description: RangeLow is calculated; it cannot be set.

Example: Get the lowest reachable axis position:

```
M1Get,RangeLow/
```

RS485Config

Synopsis: Selects port settings for RS-485 communications.

Values: 0 or 1

Syntax:
```
C0Get,RS485Config/
C0Set,RS485Config,<Mode>/
```

Description: The values represent:

- 0: 9600 baud, parity none, 8 data bits, 1 stop bit.
- 1: 38400 baud, parity none, 8 data bits, 1 stop bit.

The factory default is 1.

Example: Get the port settings:

```
C0Get,RS485Config/
```

Set the port settings to the 0 configuration:

```
C0Set,RS485Config,0/
```

SerialNumber

Synopsis: The serial number of the CIB

Values: A valid CIB serial number

Syntax:
```
C0Get,SerialNumber/
```

Description: SerialNumber is a read-only parameter and its value cannot be changed.

Example: Get the CIB serial number:

```
C0Get,SerialNumber/
```

Status

Synopsis: The current status of the axis

Values: Initialized or Not Initialized. Also may include one or more additional messages (DeviceError, Disabled, or Busy).

Syntax:
```
M<AxisID>Get,Status/
```

Description: Status is a read-only parameter and its value cannot be changed.

Example: Get the current status of axis 1:

```
M1Get,Status/
```

SWRevision

Synopsis: The Command Processor part number and revision number

Values: Valid part and revision number

Syntax: **C0Get,SWRevision/**

Description: SWRevision is a read-only parameter and its value cannot be changed.

Example: Get the Command Processor part number and revision number:

```
C0Get,SWRevision/
```

SWVersion

Synopsis: The Command Processor version number

Values: Valid version number

Syntax: **C0Get,SWVersion/**

Description: SWVersion is a read-only parameter and its value cannot be changed.

Example: Get the Command Processor version number:

```
C0Get,SWVersion/
```

TipPresence

Synopsis: The value that indicates if a tip is attached to the Z axis

Values: 0 (not present) or 1 (present)

Syntax: **M<*AxisID*>Get,TipPresence/**

Description: TipPresence is a read-only parameter and its value cannot be changed.

Example: Get the status of the tip:

```
M3Get,TipPresence/
```

TotalMoveCount

Synopsis: The number of movements performed by the axis since the last reset

Values: Set: 0 only; Get: any whole number

Syntax:
M<*AxisID*>Get,TotalMoveCount/
M<*AxisID*>Set,TotalMoveCount,0/

Description: The Set command can only be used to reset the value of the parameter to 0. The Get command returns the cumulative number of movements performed by the axis since the last reset. The move count is recorded per axis ID, so if the axis ID is changed the data is lost.

Example: Get the number of movements of axis 1:

```
M1Get,TotalMoveCount/
```

Reset the number of movements of axis 1 to 0:

```
M1Set,TotalMoveCount,0/
```

TotalMoveCountAbs

Synopsis: The total cumulative number of moves by the axis over its entire lifetime

Values: Any whole number

Syntax: **M<*AxisID*>Get,TotalMoveCountAbs/**

Description: The Get command returns the returns the highest number of moves ever recorded for this axis. The move count is recorded per axis ID, so if the axis ID is changed the data is lost.

Example: Get the number of movements:

```
M1Get,TotalMoveCountAbs/
```

TotalTravelDist

Synopsis: The cumulative travel distance completed by the axis in mm since the last reset

Values: Set: 0 only; Get: any number of mm

Syntax:
M<*AxisID*>Get,TotalTravelDist/
M<*AxisID*>Set,TotalTravelDist,0/

Description: The Set command can only be used to reset the value of the parameter to 0. The Get command returns the cumulative travel distance completed by the axis since the last reset. The travel distance is recorded per axis ID, so if the axis ID is changed the data is lost.

Example: Get the number of movements:

```
M1Get,TotalTravelDist/
```

Reset the number of movements to 0:

```
M1Set,TotalTravelDist,0/
```

TotalTravelDistAbs

Synopsis: The total cumulative travel distance completed by the axis over its entire lifetime

Values: Any valid number of mm

Syntax: **M<*AxisID*>Get,TotalTravelDistAbs/**

Description: The Get command returns the returns the longest cumulative travel distance ever recorded for this axis. The travel distance is recorded per axis ID, so if the axis ID is changed the data is lost.

Example: Get the total travel distance:

```
M1Get,TotalTravelDistAbs/
```

TravelPos

Synopsis: The position to which the axis will first move before it moves to the final position set by the MoveAbs command

Values: ***RangeLow*** to ***RangeHigh***

Syntax: **M<*AxisID*>Get,TravelPos/**
M<*AxisID*>Set,TravelPos,<*Pos*>/

Description: If the value of *TravelPos* is 0, no special treatment will occur. During the arm initialization, the axes with *TravelPos* set to non-zero will be initialized first.

See the MoveAbs and Initialize Arm commands for more information.

Note: The Travel Position, or TravelPos, is used to specify a “safe” position to which the Z axis is moved prior to executing a move, in order to avoid potential collisions with elements of a fixture, for example.

When working with a Dual Z, it is possible to configure an arm or arms which include both of the Zs, an arm which only includes one of the two Zs, or multiple arms which each include one of the two Zs.

WARNING! *Because both of the Zs in the Dual Z are physically attached, when an arm which physically contains one of the Zs moves, both Zs will move even if they are not both included in the arm grouping. Thus, it is necessary to make sure that both the Zs on the Dual Z are located in an appropriate Travel Position before the move begins.*

The simplest way to achieve this is to only set up arms that include both of the Zs on a Dual Z. Alternately, the application can be written to check all axes in a list for safe travel positions and move them to a safe travel height regardless of whether they are grouped into an arm or not.

Example: Get the position to which axis 3 will first move before it moves to the final position set by the MoveAbs command:

```
M3Get,TravelPos/
```

Set the position to which axis 3 will first move before it moves to the final position set by the MoveAbs command to 210:

```
M3Set,TravelPos,210/
```

UserData

Synopsis: Up to 256 bytes of user data in the CIB nonvolatile memory

Values: Any valid ASCII string that does not exceed 256 bytes or contain the characters "/" or ","

Syntax:
```
C0Get,UserData/
C0Set,UserData,<Data>/
```

Description: The data can include any ASCII character except the system delimiter character "/" or the separator ","

Example: Get the user data:

```
C0Get,UserData/
```

Set the user data to 1234:

```
C0Set,UserData,1234/
```

VoltageFU1

Synopsis: The power supply voltage after FU1 from the CIB

Values: Valid voltage

Syntax:
```
C0Get,VoltageFU1/
```

Description: VoltageFU1 is a read-only parameter and its value cannot be changed.

Example: Get the power supply voltage after FU1:

```
C0Get,VoltageFU1/
```

VoltageFU2

Synopsis: The power supply voltage after FU2 from the CIB

Values: Valid voltage

Syntax: `C0Get,VoltageFU2/`

Description: VoltageFU2 is a read-only parameter and its value cannot be changed.

Example: Get the power supply voltage after FU2:

```
C0Get,VoltageFU2/
```

VoltageFU3

Synopsis: The power supply voltage after FU3 from the CIB

Values: Valid voltage

Syntax: `C0Get,VoltageFU3/`

Description: VoltageFU3 is a read-only parameter and its value cannot be changed.

Example: Get the power supply voltage after FU3:

```
C0Get,VoltageFU3/
```

VoltageFU4

Synopsis: The power supply voltage after FU4 from the CIB

Values: Valid voltage

Syntax: `C0Get,VoltageFU4/`

Description: VoltageFU4 is a read-only parameter and its value cannot be changed.

Example: Get the power supply voltage after FU4:

```
C0Get,VoltageFU4/
```

WorkingLen

Synopsis: The distance between the axis home position and the maximum reachable axis position

Values: 10 or greater

Syntax: `M<AxisID>Get,WorkingLen/`
`M<AxisID>Set,WorkingLen,<Len>/`

Description: This value is used to calculate *RangeLow* and *RangeHigh*.

Example: Get the distance between axis 1 home position and the maximum reachable axis position:

```
M1Get,WorkingLen/
```

Set the distance between axis 1 home position and the maximum reachable axis position to 1250:

```
M1Set,WorkingLen,1250/
```

7.7 Action Commands

The following table summarizes the Cavro® Omni Robot action commands, and indicates whether they can be executed on an axis (type M), or both an axis and an arm device (type A).

Table 7-10 *Action commands*

Device Type	Description	Command (Arm or Axis)	See Page
M	Performs a diagnostic brake test	BrakeTest	7-62
A, M	**Initialize Arm** or **Initialize Axis**: Initializes device (arm or axis). Arm version uses default initialization sequence based on the *TravelPos* definitions.	CavroInit (Arm) CavroInit (Axis)	7-59 7-63
A, M	Retracts tip and checks for exit from liquid.	CheckForExit	7-69
M	Clears and resets status of the axis.	ClearError	7-64
M	Activates the cLLD self test for the Z axis.	cLLDSelfTest	7-83
A, M	Moves A axis to detect the presence of liquid.	DetectLiquid	7-72
M	Disables the current/drive for the axis. Disabling must be done per axis.	Disable	7-64
A, M	Drops a disposable tip.	DropTip	7-76
M	Enables the current/drive for the axis. Enabling must be done per axis.	Enable	7-65
A, M	**Move Arm Absolute** or **Move Axis Absolute:** Moves arm or axis to specified coordinates.	MoveAbs (Arm) MoveAbs (Axis)	7-60 7-65

Table 7-10 Action commands (continued)

Device Type	Description	Command (Arm or Axis)	See Page
M	Moves single axis to an absolute position with higher ending current.	MoveAbsHiForce	7-66
A, M	Picks up a disposable tip.	PickTip	7-79
M	Moves the specified axis to a relative axis offset position using a fixed speed of 25 mm/s. This command will only be accepted when the axis is ***not*** initialized. This command only works with axis X or Y.	PreInitMoveRel	7-67
A, M	**Terminate Arm Motion** or **Terminate Axis Motion**: Immediately terminates arm or axis motion.	TerminateMotion (Arm) TerminateMotion (Axis)	7-61 7-68

7.7.1 Arm Motion Commands (Device Type A)

The following sections describe the Command Processor commands associated with the operation of a single arm.

For equivalent Command Processor commands that operate on a single axis, see Axis Action Commands (Device Type M).

WARNING! *The Omni Robot contains pointed tips and other sharp-edged elements, which might cause injuries when you reach into the working area.*

- *Always be aware of mechanical hazards. Do not move your head or hands into the path of moving parts during deployment; a risk of injury, blunt force trauma, or piercing exists.*
- *Wear proper laboratory apparel (such as rubber gloves and safety goggles) as appropriate for your deployment.*
- *Personnel who are not operating the Omni Robot module should take extra care when standing near the module, as an unanticipated movement by one or more axes could cause injury to a bystander. This is especially important if the Omni Robot is being operated remotely through the Ethernet connection.*

Initialize Arm

Synopsis: Initializes a robotic arm.

Syntax: **`<ArmID>CavroInit/`**

Description: This command initializes a defined robotic arm. Axes with a defined travel position (defined by the Set,TravelPos command, see page 7-55) will be moved first; if the travel position of all axes is 0, they will start initializing at the same time.

Default initialization force and speeds are used. If different initialization sequences are needed for individual axes, use the Initialize Axis command.

Note: *If the axis has a brake attached, the brake will be mechanically disengaged once the command is issued and remain disengaged.*

Generated Errors:
1 -- Initialization Error (not applicable for Z axis)
5 -- Device Not Implemented
8 -- Device Busy
9 -- Device Error (Hardware Error)
10 - Command aborted
11 -- Double initialization acceptance window exceeded (Z axis only)
12 -- Invalid number of operands
13 -- Device disabled

Example: Initialize Arm1:

```
A1CavroInit/
```

Move Arm Absolute

Synopsis: Moves a robotic arm to an absolute position.

Syntax: `<ArmID>MoveAbs,<Pos1>,<Pos2>,...<Pos15>/`

Description: This command moves the axes of the arm to the given coordinate positions. The parameter sequence corresponds to the sequence of axes defined in the arm configuration. Axes with a defined travel position (that is, a non-zero position assigned with the Set,TravelPos command) will be moved first to their travel positions, followed by the remaining axes to their target positions, and then moving the first group of axes to their target positions.

[*pos1*]...[*pos15*] The coordinate of each axis, definable by user, in mm. The valid range is from *RangeLow* to *RangeHigh*.
For more information on the values of *RangeLow* and *RangeHigh*, see the RangeLow and RangeHigh parameters.
The number of provided parameters will depend on the number of axes associated with an arm. An arm here is an abstract entity and can have up to 15 axes associated with it. All parameters have to be provided according to the number of axes on the arm.

Generated Errors:
3 -- Invalid Operand - Out of Range
5 -- Device Not Implemented
6 -- Arm Collision Avoided
7 -- Device Not Initialized
8 -- Device Busy
9 -- Device Error (Hardware Error)
10 -- Command Aborted
12 -- Invalid number of operands
13 -- Device Disabled
14 -- Invalid Travel Pos

Example: Move the X axis of Arm1 to position 200, the Y axis of Arm1 to position 100, and the Z axis of Arm1 to position 150:

```
A1MoveAbs,200,100,150/
```

Terminate Arm Motion

Synopsis: Terminates the arm motion immediately.

Syntax: ***<ArmID>*TerminateMotion/**

Description: This command terminates the motion of all axes on the arm. All motors remain enabled after ramping down according to the most recent acceleration parameters. This command is accepted only during execution of the commands MoveAbs or CavroInit.

If the command is sent to an arm in an idle state, no action occurs and no error message will be returned.

Generated Errors:
5 -- Device Not Implemented
8 -- Device Busy
9 -- Device Error (Hardware Error)
12 -- Invalid number of operands
13 -- Device Disabled

Example: Terminate motion of Arm1:

```
A1TerminateMotion/
```

7.7.2 Axis Action Commands (Device Type M)

The commands included in this section operate a single axis.

WARNING! *The Omni Robot contains pointed tips and other sharp-edged elements, which might cause injuries when you reach into the working area.*

- *Always be aware of mechanical hazards. Do not move your head or hands into the path of moving parts during deployment; a risk of injury, blunt force trauma, or piercing exists.*
- *Wear proper laboratory apparel (such as rubber gloves and safety goggles) as appropriate for your deployment.*
- *Personnel who are not operating the Omni Robot module should take extra care when standing near the module, as an unanticipated movement by one or more axes could cause injury to a bystander. This is especially important if the Omni Robot is being operated remotely through the Ethernet connection.*

Brake Test

Synopsis: Performs a diagnostic brake test.

Syntax: **`<AxisID>BrakeTest/`**

Description: This command moves the Universal Z downward twice from the current position to 40 mm away. The first time it moves with the brake disengaged, monitors the current consumption during the motion, and moves back to the original position. The second time it moves with brake engaged, monitors the current consumption, then moves back to the original position. The difference of the average current consumption at cruising speed of the two tests will determine if the brake is in a good state. Otherwise, maintenance or part replacement is required.

Generated Errors:
0 -- No Error (brake is in a good state)
3 -- Invalid Operand
5 -- Device Not Implemented
7 -- Device Not Initialized
8 -- Device Busy
9 -- Device Error (Hardware Error)
12 -- Invalid number of operands
13 -- Device Disabled
60 -- Brake test failed

The following parameters have an effect on the behavior of this command:

DefSpeed — The default axis movement speed. Valid values are 0.1 to 10000 mm/s. The default value is:

- Universal Z axis: 250 mm/s

It can be changed with the Set DefSpeed command.

DefAcceleration — The acceleration of the axis during any move command, except the Initialize Axis command. Valid values are 1 to 100,000 mm/s^2. The default value is:

- Universal Z axis: 1000 mm/s^2

It can be changed with the Set DefAcceleration command.

Example: Test the brake on axis 3:

```
M3BrakeTest/
```

Initialize Axis

Synopsis: Initializes a single axis of motion.

Syntax: **<*AxisID*>CavroInit/**

Description: This command initializes a single axis. An axis will not perform any action commands without being initialized.

For the X axis and Y axis, the CavroInit command causes the axis to search for the home sensor using DefInitSpeed (defined by the Set,DefInitSpeed command) and DefInitAcceleration (defined by the Set,DefInitAcceleration command).

For Z axes, the CavroInit command causes the axis to search for the mechanical hard stop using DefInitSpeed, DefInitAcceleration, and the force factor defined by the Set,InitForceFactor command.

If the initialization mode (defined by Set,InitMode) is set to double initialization, the CavroInit command will perform two initializations. The first initialization uses a fraction of InitForceFactor, and the second uses InitForceFactor. The command is successful if the position difference between the two initializations is within the acceptance window defined by Set,InitAcceptanceWindow.

Double initialization is the default mode for Z axes.

If the axis has a brake attached, the brake will be mechanically disengaged once the command is issued and remain disengaged.

WARNING! *Before initializing the Z axis, be sure the probe tip is not inside a cap. Initializing the Z axis while the probe is still inside a cap can result in contamination or damage to labware.*

Generated Errors:
1 -- Initialization Error (not applicable for Z axes)
5 -- Device Not Implemented
8 -- Device Busy
9 -- Device Error (Hardware Error)
10 - Command aborted
11 -- Double initialization acceptance window exceeded (for Z axes only)
12 -- Invalid number of operands
13 -- Device disabled

Example: Initialize axis 1:

```
M1CavroInit/
```

Clear Error

Synopsis: Clears device errors.

Syntax: **<*AxisID*>ClearError/**

Description: This commands clears all errors that prevent action commands from executing on the axis, and resets the error status.

Example: Clear errors from axis 1

```
M1ClearError/
```

Disable

Synopsis: Disables motor current/drive.

Syntax: **<*AxisID*>Disable/**

Description: This command disables the motor current/drive. There is a 3-second delay from the time the Disable command is issued until the axis is disabled and the axis brake (if present) is disengaged.

The Disable command cannot be sent while the axis or the arm to which this axis belongs is in motion.

WARNING! *For an axis with a brake, this command may trigger the axis to drop.*

Caution! *There is a 3-second delay from the time the Disable command is issued until the axis brake is disengaged and the axis falls.*

Generated Errors: 5 -- Device Not Implemented
12 -- Invalid number of operands

Example: Disable axis 1:

```
M1Disable/
```

Enable

Synopsis: Enables motor current/drive.

Syntax: **<*AxisID*>Enable/**

Description: This command enables the motor current/drive. The Enable command should not be executed while the axis is in motion (that is, being moved by hand). If the axis has a brake attached, the brake will be mechanically disengaged once the command is issued.

Caution! *If the axis is moved between the time the command is issued and the motor is energized, there may be an unexpected movement when the axis returns to the position it held when the command was issued.*

Generated Errors: 5 -- Device Not Implemented
8 -- Device Busy
12 -- Invalid number of operands

Example: Enable axis 1:

```
M1Enable/
```

Move Axis to Absolute Position

Synopsis: Moves a single axis to an absolute position.

Syntax: **<*AxisID*>MoveAbs,<*Pos*>/**

Description: This command moves the axis to an absolute position, using the speed (*DefSpeed*) assigned with the Set,DefSpeed command. The axes with an initial position from which the motion will begin (that is, a non-zero position assigned with the Set,TravelPos command) will not be retracted.

<Pos>	The coordinate of the axis, in mm. The valid range is from RangeLow to RangeHigh. For more information on the values of RangeLow and RangeHigh, see RangeLow and RangeHigh, respectively.
Generated Errors:	3 -- Invalid Operand - Out of Range 5 -- Device Not Implemented 6 -- Arm Collision Avoided 7 -- Device Not Initialized 8 -- Device Busy 9 -- Device Error (Hardware Error) 10 -- Command aborted 12 -- Invalid number of operands 13 -- Device Disabled

Example: Move axis 1 to absolute position 100.54

```
M1MoveAbs,100.54/
```

Move Axis to Absolute Position with High Force

Synopsis: Moves the specified axis to an absolute axis coordinate position using specified speed parameter and `DefAcceleration`.

Syntax: ***<AxisID>*MoveAbsHiForce,*<Pos>*,[*Speed*]/**

Description: This command moves the axes of the arm to the given coordinate positions. The parameter sequence corresponds to the sequence of axes defined in the arm configuration. Axes with a defined travel position (that is, a non-zero position assigned with the Set,TravelPos command) will be moved first to their travel positions, followed by the remaining axes to their target positions, and then moving the first group of axes to their target positions.

<Pos>	The coordinate of each axis, definable by user, in mm. The valid range is from *RangeLow* to *RangeHigh*. For more information on the values of *RangeLow* and *RangeHigh*, see the RangeLow and RangeHigh parameters.
[Speed]	The movement speed. Valid values are from 0.1 to 10000 mm/sec; the default value is 40 mm/sec.
Generated Errors:	3 -- Invalid Operand - Out of Range 5 -- Device Not Implemented 7 -- Device Not Initialized 8 -- Device Busy 9 -- Device Error (Hardware Error) 10 -- Command Aborted (when command is terminated by TerminateMotion command) 12 -- Invalid number of operands 13 -- Device Disabled

Example: Move the axis to position 50.5:

```
M3MoveAbsHiForce,50.5/
```

Move the axis to position 50.5 with a speed of 100:

```
M3MoveAbsHiForce,50.5,100/
```

Move Axis to Relative Position Before Initialization

Synopsis: Moves a single axis to a relative position before initialization.

Syntax: **<AxisID>PreInitMoveRel,<offset>/**

Description: This command moves the specified axis to a relative axis offset position at a fixed speed of 25 mm/s. Motion is stopped if a hard-stop is hit. This command will only be accepted when the axis is ***not*** initialized. This command only works with X and Y axes.

Caution! *Because the* ***PreInitMoveRel*** *movement offset value can not be range limited by the software, the user is responsible for selecting a value which will not result in a collision with obstacles on the work table or the axis hard-stops.*

<Offset>	The relative offset to current position of each axis, definable by user. Valid values are numeric units in mm.

Generated Errors:
3 -- Invalid Operand - Out of Range
5 -- Device Not Implemented
8 -- Device Busy
9 -- Device Error (Hardware Error)
12 -- Invalid number of operands
13 -- Device disabled
101- PreInitMoveRel move blocked
102 - PreInitMoveRel not allowed at current state
103 - PreInitMoveRel encoder error

Example: Move axis 1 to relative position -2000

```
M1PreInitMoveRel,-2000/
```

Terminate Axis Motion

Synopsis: Terminates the axis motion immediately.

Syntax: **<*AxisID*>TerminateMotion**

Description: This command terminates the motion of the axis. The motor remains enabled after ramping down according to the most recent acceleration parameters. This command is accepted only during execution of the axis commands MoveAbs or CavroInit; no error is returned and no action takes place if the command is sent to an axis in an idle state or if the axis is not initialized.

Generated Errors:
5 -- Device Not Implemented
8 -- Device Busy
12 -- Invalid number of operands
13 -- Device Disabled

Example: Terminate motion of axis 1:

```
M1TerminateMotion/
```

7.7.3 Liquid Handling Commands (Device Type A)

Check For Exit

Synopsis: Checks for exit signal using the cLLD system.

Syntax: **<*ArmID*>CheckForExit,<*RetractSpeed*>,<*RetractDist*>,<Limit>, <Sensitivity>/**

Description: This command is illustrated step-by-step in Figure 7-6 on page 7-71.

This command moves the tip out of the liquid starting from the current position and checks for an exit signal using the specified *RetractSpeed* and *Sensitivity*. The exit detection is successful if the exit signal (low to high transition) appears between start of the move and the position set in the specified *Limit*.

This command is specific to Z axes with cLLD and arm groups containing that axis.

For the Dual Z axis, each axis has its own *RetractSpeed*, *RetractDist*, *Limit*, and *Sensitivity*. If you set them here, these settings overwrite any parameters set for the Dual Z elsewhere. If you do not set them here, the individual settings are used.

This command replies with an error code 0 (no error) if all Z-drives executed successfully; otherwise, the last error encountered is returned. For errors 9, 18, 19, 20, or 27, the application can parse the “comment” field of the arm command to report the errors. Axis errors are reported in pairs, for example (Ma-b)(Mc-d) where “a” and “c” are axis IDs and “b” and “d” are error codes.

<*RetractSpeed*>	The retract speed. Valid values are from 1 to 150 mm/s; the default is the value of *ExitRetractSpeed*, which can be changed with the Set,ExitRetractSpeed command.
<*RetractDist*>	The travel distance. Valid values are from 1 to 100 mm; the default is the value of *ExitRetractDist*, which can be changed with the Set,ExitRetractDist command.
<*Limit*>	The acceptance limit for a successful exit signal check. Valid values are from 0 to 50 mm; the default is the value of *ExitLimit*, which can be changed with the Set,ExitLimit command.
<*Sensitivity*>	The sensitivity in increments during exit search. Valid values are from 3 (highest sensitivity) to 120 (lowest sensitivity); the default is the value of *LLDSensitivity*, which can be changed with the Set,LLDSensitivity command.

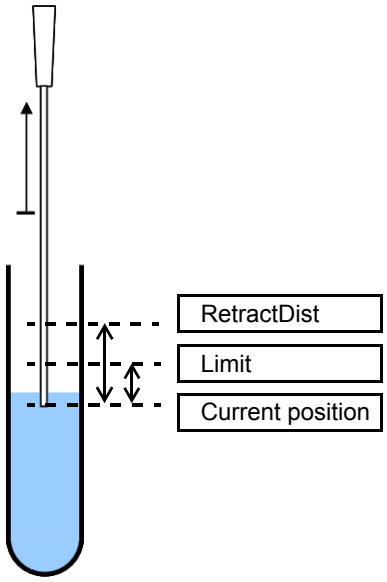
Generated Errors:

3 -- Invalid Operand - Out of Range
5 -- Device Not Implemented
7 -- Device Not Initialized
8 -- Device Busy
9 -- Device Error (Hardware Error)
12 -- Invalid number of operands
13 -- Device Disabled
18 -- cLLD Exit Limit Exceeded at Exit Detection (if the exit signal appears after the position set by *ExitLimit*)
19 -- cLLD No Exit Signal at Exit Detection
20 -- cLLD Not Ready
27 -- DiTi - No Tip Mounted

Example: Check for exit signal of Arm1 with the default retract speed, a retract distance of 20 mm, the default limit, and a sensitivity of 63:

```
A1CheckForExit,,20,,63/
```

The following diagram (Figure 7-6) illustrates the specific actions that are taken to execute the CheckForExit command.



Actions	Move from current position by *RetractDist* using *RetractSpeed*. Checks for exit signal during entire motion. Does not stop motion upon exit signal.
Related Error Messages	**Error 20**: cLLD Not Ready (if output is already high at search begin). **Error 27**: DiTi - No tip mounted. **Error 18**: cLLD exit limit exceeded at exit detection (If the Exit signal appears after the position set by limit). **Error 19**: No exit signal at exit detection.
Notes	*RetractDist* and *Limit* are relative positions (based on current position)

Figure 7-6 *Actions taken to execute a CheckForExit command*

Detect Liquid

Synopsis: Performs liquid detection using the cLLD.

Syntax: **`<ArmID>DetectLiquid,<StartPos>,<SearchLim>,<SubmergeDist>,<Sensitivity>,<DetectionMethod>,<RetractOnError>/`**

Description: *StartPos*, *SearchLim*, and *SubmergeDist* are required parameters and can only be set here. *Sensitivity*, *DetectionMethod*, and *RetractOnError* can be set either here or on the axis and depend on the value of each Axis.

The steps described in the next paragraphs are illustrated in the diagrams (Figure 7-7 and Figure 7-8) on pages 7-74 and 7-75.

When the DetectLiquid command is issued, the Z-drive moves to the start position (*StartPos*) using the default move parameters (step A in Figure 7-7), and starts searching for liquid from that position using the speed and search sensitivity specified by *LLDSearchSpeed* and *Sensitivity* (step B). When the liquid surface is detected by the tip, the Z-move immediately stops (step C, Figure 7-8).

If performing a single detection (that is, *DetectionMethod* = 1), the detection is considered successful if the detection position minus the given submerge distance (*SubmergeDist*) is higher than or equal to the value of *SearchLim*. In this case, steps D though F in Figure 7-8 are skipped.

If performing a double detection (that is, *DetectionMethod* = 0), the tip retracts the default distance defined by Set,LLDRetractDist (step D in Figure 7-8) using the given search speed (*LLDSearchSpeed*) and starts the liquid search again (step E). It is considered successful if the second detection position minus the given submerge distance (*SubmergeDist*) is higher than or equal to the value of *SearchLim,* and the difference between the two detection positions is less than or equal to the acceptance window (step F). The acceptance window is defined by the Set,LLDAcceptanceWindow command.

If the liquid detection is successful, a subsequent down move is performed to submerge the tip, using the speed specified by LLD*SearchSpeed* (step G in Figure 7-8).

An error is generated if the limit specified in *SearchLim* is reached without detecting the liquid. This command replies with an error code 0 (no error) if all Z-drives executed successfully; otherwise, the last error encountered is returned. For errors 9, 15, 16, 17, 20, or 27, the application can parse the “comment” field of the arm command to report the errors. Axis errors are reported in pairs, for example (Ma-b)(Mc-d) where “a” and “c” are axis IDs and “b” and “d” are error codes.

The *RetractOnError* parameter determines if the tip remains at the current position (*SearchLim*) or if it retracts to the start position using LLD*SearchSpeed* after an error is generated.

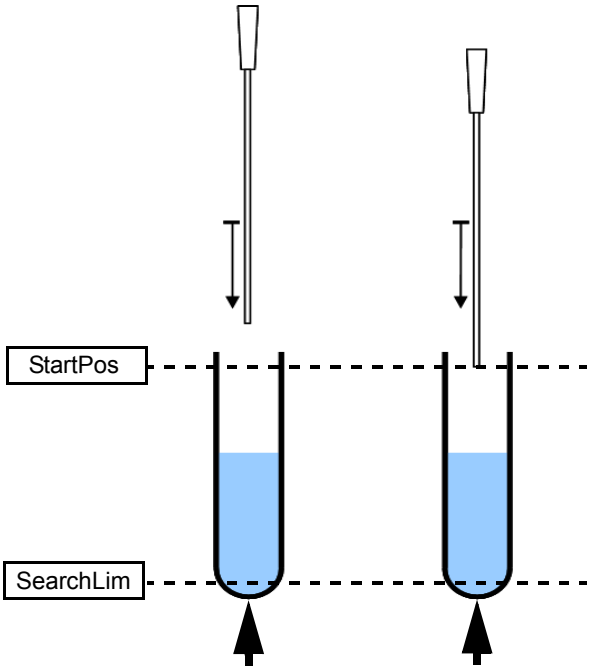
This command is specific to Z axes with cLLD and arm groups containing that axis.

<*StartPos*>	Start position for the liquid search. Valid values are from *RangeLow* to *RangeHigh*. If you are using disposable tips, the highest position should be no higher than *RangeHigh* –10 mm. For more information on the values of *RangeLow* and *RangeHigh*, see the RangeLow and RangeHigh parameters.
<*SearchLim*>	The position limit to search for liquid, starting from the position given in *StartPos*. If liquid is not found during this entire travel, an error is returned. Valid values are from *StartPos* to *RangeLow*. For more information on *RangeLow*, see the RangeLow parameter.
<*SubmergeDist*>	The submerge distance after successful detection. Valid values are from 0 to 100 mm; the default value is the value of *LLDSubmergeDist*, which can be changed with the Set,LLDSubmergeDist command.
<*Sensitivity*>	The sensitivity in increments during liquid search. Valid values are from 3 (highest sensitivity) to 120 (lowest sensitivity); the default value is the value of *LLDSensitivity*, which can be changed with the Set,LLDSensitivity command.
<*DetectionMethod*>	0 -- Double detection (the default, unless overridden by the Set,LLDDetectionMethod command). 1 -- Single detection.
<*RetractOnError*>	0 -- Retract to start position upon error (no liquid detected) (the default, unless overridden by the Set,LLDRetractOnError command). 1 -- Do not retract upon error (no liquid detected).
Generated Errors:	3 -- Invalid Operand - Out of Range 5 -- Device Not Implemented 7 -- Device Not Initialized 8 -- Device Busy 9 -- Device Error (Hardware Error) 12 -- Invalid number of operands 13 -- Device Disabled 15 -- cLLD - No Liquid Detected 16 -- cLLD - Not Enough Liquid Detected 17 -- cLLD - Liquid Detection Failure - Double Detection Acceptance Window Exceeded 20 -- cLLD Not Ready 27 -- DiTi - No Tip Mounted

Example: Detect liquid for Arm1 with a start position of 50 mm, a search limit of 20 mm, the default submerge distance, default sensitivity, default detection method, and do not retract upon error:

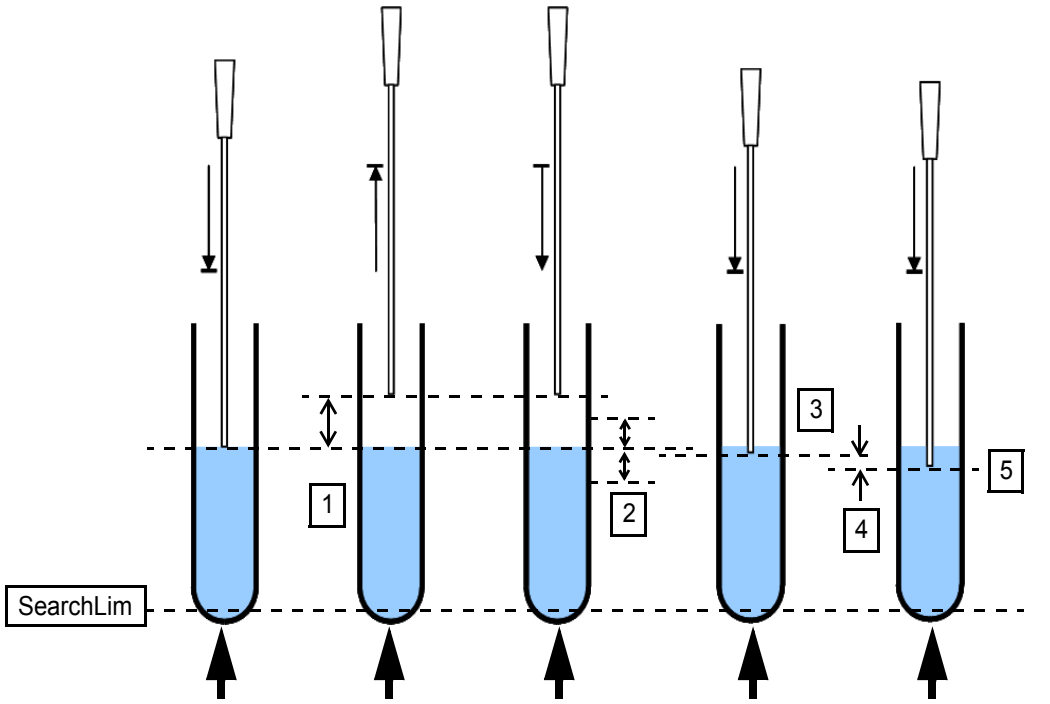
```
A1DetectLiquid,50,20/
```

The following diagrams (Figure 7-7 and Figure 7-8), show the individual actions taken to execute the DetectLiquid command. Each column in the table describes the movement illustrated in the diagram directly above it.



Actions	A) Move from current position to *StartPos* with default speed parameters	B) Start liquid search from *StartPos* with *LLDSearchSpeed* towards *SearchLim*
Related Error Messages		**Error 20**: cLLD Not Ready (if output is already high at search begin). **Error 27**: DiTi - No tip mounted.

Figure 7-7 *Actions to execute the DetectLiquid command, part 1*



Actions	C) Move stops when the tip touches the liquid surface.	D) Upwards move relative to current position (*RetractDist* **[1]**)	E) Re-start liquid search from current position with *LLDSearch-Speed* towards *SearchLim*	F) Move stops when the tip touches the liquid surface. Checks if detection position **[3]** is within the Acceptance Window **[2]**	G) Additional downward move by the amount of *LLDSubmergeDist* **[4]**
Related Error Messages	**Error 15**: cLLD - No liquid detected (if *SearchLim* was reached without detection).			**Error 15**: cLLD - No liquid detected (if *SearchLim* was reached without detection). **Error 17**: cLLD - Liquid Detection Failure - double detection acceptance window exceeded.	**Error 16**: cLLD - Not enough liquid detected (if calculated final position **[5]** would be lower than *SearchLim*.
Notes		This step is skipped if *LLDDetection-Method* = 1	This step is skipped if *LLDDetection-Method* = 1	This step is skipped if *LLDDetection-Method* = 1	

Figure 7-8 *Actions to execute the DetectLiquid command, continued (part 2)*

Drop Tip

Synopsis: Drops a disposable tip.

Syntax: **`<ArmID>DropTip,<StartPos>,<Limit>,<AcceptanceLimit>,<Speed>,<ForceFactor>/`**

Description: This command drops a disposable tip from the current position.

The steps described in the next paragraphs are illustrated in the diagram (Figure 7-9) on page 7-82.

The tip presence signal of the cLLD board is checked for low state prior to any further action. The Z-drive moves to the start position (*StartPos*) using regular move parameters (Step A in Figure 7-9). Then it starts the search function from the current XY position (Step B) by moving up at the given speed and force (*Speed* and *ForceFactor*) until the drive is stopped by hitting the hard stop (Step C). An error is generated if the Z-drive does not reach the stop position limit (specified by *AcceptanceLimit*) or if it reaches the drop position limit (specified by *Limit*). The Z-drive retracts to *StartPos* using regular move parameters and checks for high state of the tip presence signal.

This command replies with an error code 0 (no error) if all Z-drives executed successfully; otherwise, the last error encountered is returned. For errors 9, 22, 23, 24, or 27, the application can parse the “comment” field of the arm command to report the errors. Axis errors are reported in pairs, for example (Ma-b)(Mc-d) where “a” and “c” are axis IDs and “b” and “d” are error codes.

For the Dual Z axis, each axis has its own *RetractSpeed*, *RetractDist*, *Limit*, and *Sensitivity*. If you set them here, these settings overwrite any parameters set for the Dual Z elsewhere. If you do not set them here, the individual settings are used.

StartPos, *Limit*, and *AcceptanceLimit* are required parameters and are command-specific. *Speed* and *ForceFactor* are optional and axis-specific. They can be set elsewhere and depend on the value of each Axis.

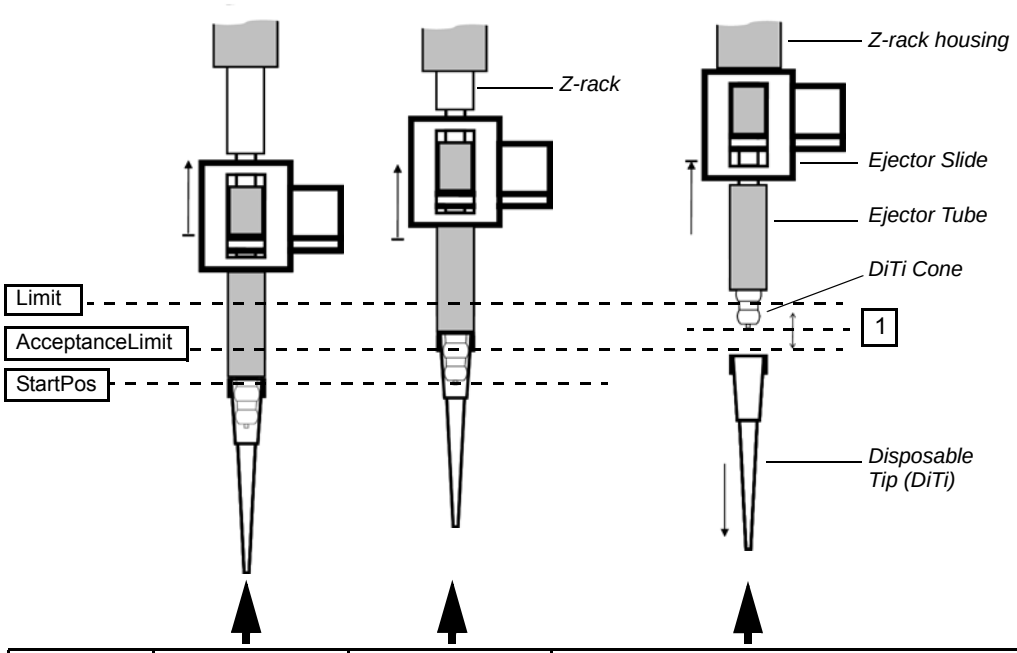
This command is specific to Z axes with cLLD and arm groups containing that axis. The Drop Tip command is not yet supported for the Universal Z axis.

<*StartPos*>	The Z-drive position, in mm, for the beginning of the drop tip action. Valid values are from *RangeLow* to (*RangeHigh* – 10 mm); the default value is *AxisOffset* – 20 mm. For more information on the values of *RangeLow*, *RangeHigh*, and *AxisOffset*, see the RangeLow, RangeHigh, and AxisOffset parameters.
<*Limit*>	The position limit for the drop search. Valid values are from *StartPos* to (*RangeHigh* + 50 mm); the default value is *AxisOffset* + 20 mm. For more information on the value of *RangeHigh*, see the RangeHigh parameter.
<*AcceptanceLimit*>	The limit for a valid stop position. Valid values are from *StartPos* to the value of *Limit*; the default is the value of *AxisOffset*.
<*Speed*>	The drop speed. Valid values are from 1 to 150 mm/sec; the default value is the value of *DropSpeed*, which can be changed with the Set,DropSpeed command.
<*ForceFactor*>	The force used to drop the disposable tip. Valid values are from 8000 to 22000; the default is the value of *DropForceFactor*, which can be changed with the Set,DropForceFactor command.
Generated Errors:	3 -- Invalid Operand - Out of Range 5 -- Device Not Implemented 7 -- Device Not Initialized 8 -- Device Busy 9 -- Device Error (Hardware Error) 12 -- Invalid number of operands 13 -- Device Disabled 22 -- DiTi - Tip Not Dropped (If the Tip Presence Signal State Remains High) 23 -- DiTi - Tip Drop Error (If the Stop Occurs Before the *AcceptanceLimit*) 24 -- DiTi - Hard Stop Not Found (If Limit is Reached) 27 -- DiTi - No Tip Mounted

Example: Drop a disposable tip from Arm1 with a start position of 200 mm, a search limit of 220 mm, an acceptance limit of 210, the default speed, and a force factor of 15000:

```
A1DropTip,200,220,210,,15000/
```

Figure 7-9 shows the specific actions taken to execute the Drop Tip command. Each step (column) in the table describes the movement illustrated in the diagram directly above it.



Intended Actions	A) Move from current position to *StartPos* with default speed parameters.	B) Start hard stop search from *StartPos* with *Speed* towards *Limit.*	C) The ejector slide hits the z-rack housing and continues the move. The ejector slide is pushed down with the force set by *ForceFactor*. The force is transmitted through the ejector tube and results in tip ejection. The axis is forced to stop when the ejector slide is pushed all the way down. The stop position **[1]** should occur between *AcceptanceLimit* and *Limit.*
Related Error Messages	**Error 27**: DiTi - No Tip mounted (if the tip presence signal state is high prior to the search).		**Error 23**: DiTi - Tip Drop error (if the stop occurs below *AcceptanceLimit*. **Error 22**: DiTi - Tip not dropped (if the tip presence signal state remains high after the axis has stopped). **Error 24**: DiTi - Hard stop not found (if *Limit* is reached).

Figure 7-9 *Actions taken to execute the Drop Tip command*

Pick Tip

Synopsis: Picks up a disposable tip.

Syntax: **`<ArmID>PickTip,<StartPos>,<Limit>,<AcceptanceLimit>,<Speed>,<ForceFactor>/`**

Description: This command picks up a disposable tip.

The steps described in the next paragraph are illustrated Figure 7-10 on page 7-81. Error conditions that may occur are shown in Figure 7-11.

This command picks up a disposable tip. The tip presence signal of the cLLD board is checked for high state prior to any further action. The Z-drive moves to the start position (*StartPos*) using regular move parameters (Step A in Figure 7-10). Then it starts the pick function from the current XY position (Step B) by moving down with the given speed and force (specified in *Speed* and *ForceFactor*) until the drive is stopped by hitting the tip (Step C). An error is generated if the Z-drive does not reach the stop position limit (specified by *AcceptanceLimit*) or if it reaches the drop position limit (specified by *Limit*). The Z-drive retracts to *StartPos* using regular move parameters and checks for the low state of the tip presence signal (Step D).

This command replies with an error code 0 (no error) if all Z-drives executed successfully; otherwise, the last error encountered is returned. For errors 9, 25, 26, 28, 29, or 30, the application can parse the “comment” field of the arm command to report the errors. Axis errors are reported in pairs, for example (Ma-b)(Mc-d) where “a” and “c” are axis IDs and “b” and “d” are error codes.

This command is specific to Z axes with cLLD and arm groups containing that axis. For the Dual Z axis, each axis has its own *RetractSpeed*, *RetractDist*, *Limit*, and *Sensitivity*. If you set them here, these settings overwrite any parameters set for the Dual Z elsewhere. If you do not set them here, the individual settings are used.

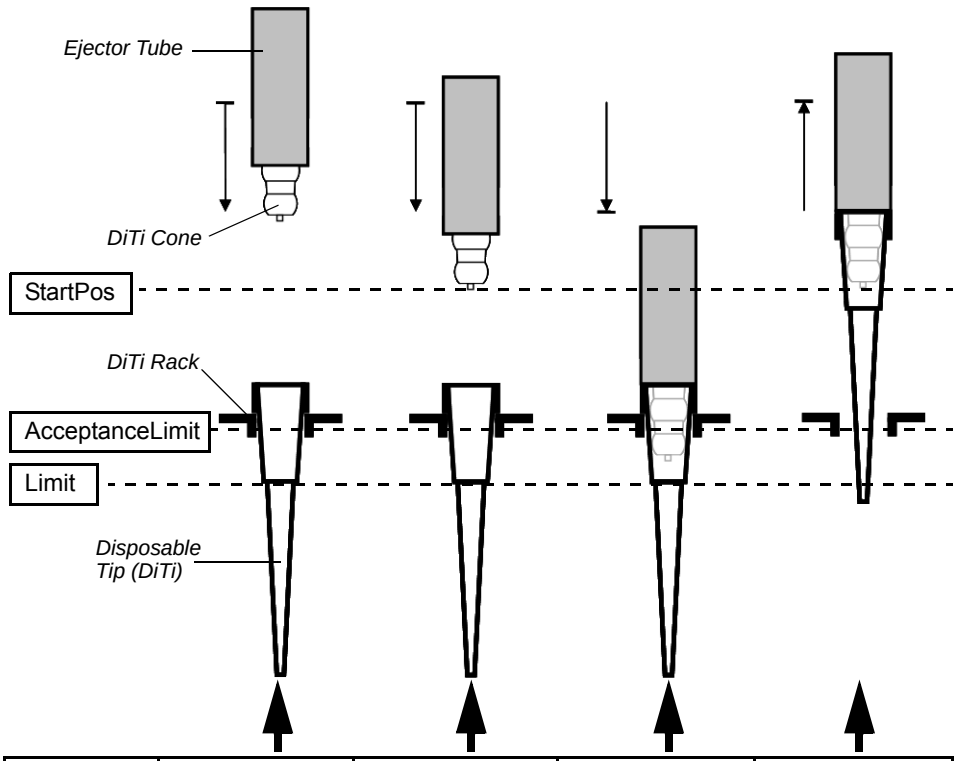
Caution! *The default gap time between each axis’* ***Pick Tip*** *start is set to 0.5 seconds. This gap time cannot be changed in applications. If the application sets the speed parameter for the second Z higher than for the first Z, this may overcome the programmed gap and cause problems with tip picking.*

<StartPos>	The Z-drive position, in mm, for the beginning of the pick tip action. Valid values are from *RangeLow* to (*RangeHigh* – 10 mm). For more information on the values of *RangeLow* and *RangeHigh*, see the RangeLow and RangeHigh parameters.
<Limit>	The Z distance to travel, in mm, when searching for a disposable tip before reporting an error. Valid values are from *<StartPos>* to *RangeLow*.
<AcceptanceLimit>	The limit for a valid stop position. Valid values are from *<StartPos>* to *<Limit.>*
<Speed>	The pick speed. Valid values are from 1 to 150 mm/sec; the default value is the value of *PickSpeed*, which can be changed with the Set,PickSpeed command. The default time gap between each axis' **Pick Tip** at start position is 0.5 seconds.
<ForceFactor>	The force used to engage the disposable tip. Valid values are from 1000 to 20000; the default value is the value of *PickForceFactor*, which can be changed with the Set,PickForceFactor command.
Generated Errors:	3 -- Invalid Operand - Out of Range 5 -- Device Not Implemented 7 -- Device Not Initialized 8 -- Device Busy 9 -- Device Error (Hardware Error) 12 -- Invalid number of operands 13 -- Device Disabled 25 -- DiTi - Tip Not Fetched 26 -- DiTi - Tip Crash (If the Stop occurs before AcceptanceLimit) 28 -- DiTi - Tip Already Mounted (If the tip presence signal state is low prior to the search) 29 -- DiTi - Tip Not Found 30 -- DiTi - Tip Not Mounted Properly

Example: Pick up a disposable tip from Arm1 with a start position of 50 mm, a search limit of 20 mm, an acceptance limit of 30, the default pick speed, and the default force factor:

```
A1PickTip,50,20,30/
```

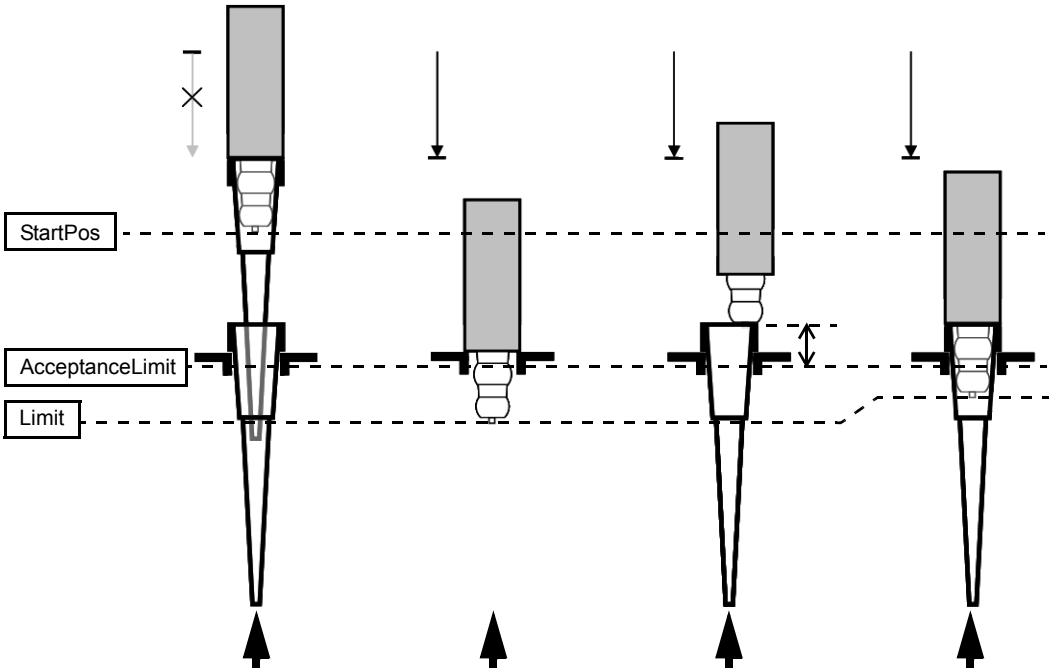
Figure 7-7 shows the specific actions taken to execute the Pick Tip command. Each step (column) in the table describes the movement illustrated in the diagram directly above it.



Actions	A) Move from current position to *StartPos* with default speed parameters.	B) Start tip search from *StartPos* with *Speed* towards *Limit.*	C) DiTi cone engages in the tip with a force set by *ForceFactor.* The Z axis move is stopped at a position below *AcceptanceLimit* but above *Limit.*	D) Upwards move back to *StartPos*.
Related Error Messages		**Error 28**: DiTi - Tip already mounted (if the tip presence signal state is low prior to the search).	**Error 26**: DiTi - Tip crash (if the stop occurs above *AcceptanceLimit*. **Error 29**: DiTi - Tip not found (if the Z axis has reached *Limit*.	**Error 30**: DiTi - Tip not properly mounted (if the Z axis reached *Limit* in step C) and the tip got mounted).

Figure 7-10 *Actions taken to execute the Pick Tip command, normal operation*

Figure 7-11 describes the error conditions that may occur with the Pick Tip command. The steps refer to the actions described in Figure 7-10.



Intended Actions	Start tip search from *StartPos* with *Speed* towards *Limit* (step B).	DiTi cone engages in the tip with a force set by *ForceFactor*. The Z axis move is stopped below *AcceptanceLimit* but above *Limit* (step C).	DiTi cone engages in the tip with a force set by *ForceFactor*. The Z axis move is stopped below *AcceptanceLimit* but above *Limit* (step C).	DiTi cone engages in the tip with a force set by *ForceFactor*. The Z axis move is stopped below *AcceptanceLimit* but above *Limit* (steps C/D).
Error Condition	A tip was already mounted and not discarded prior to this operation.	The axis reaches *Limit* because there is no tip at the current position.	Inappropriate X/Y position results in crash above *AcceptanceLimit*.	*Limit* is reached during tip search due to inappropriate *Limit* (too high). The tip can be mounted or not, depending on the pick force.
Resulting Action	The axis does not start the search move, and responds with **Error 28**: DiTi - Tip already mounted.	Axis retracts to *StartPos* and responds with **Error 29:** DiTi - Tip not found.	Axis stops, retracts to StartPos, and responds with **Error 26**: DiTi - Tip crash.	Axis stops, retracts to *StartPos*, and responds with either **Error 29**: DiTi - Tip not found, or **Error 30**: Tip not properly mounted (tip is mounted.

Figure 7-11 *The Pick Tip command, error conditions*

7.7.4 Liquid Handling Commands (Device Type M)

Check For Exit

Synopsis: Checks for exit signal using the cLLD system.

Syntax: **<*AxisID*>CheckForExit,<*RetractSpeed*>,<*RetractDist*>,<Limit>,<Sensitivity>/**

Description: This command moves the tip out of the liquid starting from the current position and checks for an exit signal.

This command is specific to Z axes with cLLD.

For a detailed description of this command, generated errors, and a definition of its parameters see the CheckForExit command on page 7-69.

See Figure 7-6 on page 7-71 for a step-by-step illustration of this command.

Check for exit signal of Axis 3 with the default retract speed, a retract distance of 20 mm, the default limit, and a sensitivity of 63:

Example: M3CheckForExit,,20,,63/

cLLDSelfTest

Synopsis: Activates the cLLD self test.

Syntax: **<*AxisID*>cLLDSelfTest,<*TestLevel*>,<*LLDSensitivity*>/**

Description: Activates the cLLD self test.

This command is specific to Z axes with cLLD.

Note: *After the Universal Z with cLLD executes this command, the axis state becomes uninitialized.*

This command activates the cLLD self test (Testlevel 0 or 1) using the given *LLDSensitivity* and reports the cLLD output states.

It responds with the status of the Dip-in and Dip-out signal when the self test capacitance is turned on and off. The response is a four-digit number in the following binary format:

1	0	0	1
Dip-in signal response at capacitance increase	Dip-out signal response at capacitance increase	Dip-in signal response at capacitance decrease	Dip-out signal response at capacitance decrease

For each digit, 0 indicates no detection signal was produced and 1 indicates that a detection signal was produced.

<*Testlevel*>	The capacitance test level. Valid values are 0 (high capacitance test level, ΔC ~ 0.75pF) and 1 (low capacitance test level, ΔC ~ 0.15pF); the default value is 0.
<*LLDSensitivity*>	The sensitivity in increments during liquid search. Valid values are from 3 (highest sensitivity) to 120 (lowest sensitivity); the preset default value is 63, but the default can be changed with the Set,LLDSensitivity command.
Generated Errors:	3 -- Invalid Operand - Out of Range 12 -- invalid number of operands 20 -- cLLD Not Ready 27 -- DiTi - No Tip Mounted

Examples: Perform self test, high capacitance test level, with lowest sensitivity:

Command: **M3cLLDSelfTest,0,120/**
Response: **M3cLLDSelfTest,0,0000/**
(no detection signal was produced)

Perform self test, high capacitance test level, with sensitivity set to 50:

Command: **M3cLLDSelfTest,0,50/**
Response: **M3cLLDSelfTest,0,1001/**
(dip in signal produced at capacitance increase; dip out signal produced at capacitance decrease)

Detect Liquid

Synopsis: Performs liquid detection using the cLLD.

Syntax: **<*AxisID*>DetectLiquid,<*StartPos*>,<*SearchLim*>,<*SubmergeDist*>,<*Sensitivity*>,<*DetectionMethod*>,<*RetractOnError*>/**

Description: For a detailed description of the actions of this command, generated errors, and a definition of its parameters, see the DetectLiquid command on page 7-72.

Note: *Special considerations need to be kept in mind when using an 8-channel pipetting head. See Section 7.7.5, "Liquid Level Detection with 8-Channel Pipetting Head" on page 7-86.*

A step-by-step illustration of the actions of this command are shown in Figure 7-7 and Figure 7-8 on pages 7-74 and 7-75.

This command is specific to Z axes with cLLD.

Example: Detect liquid for Axis 3 with a start position of 50 mm, a search limit of 20 mm, the default submerge distance, default sensitivity, default detection method, and do not retract upon error:

Axis: `M3DetectLiquid,50,20,,,,1/`

Drop Tip

Synopsis: Drops a disposable tip.

Syntax: `<AxisID>DropTip,<StartPos>,<Limit>,<AcceptanceLimit>,<Speed>,<ForceFactor>/`

Description: This command drops a disposable tip from the current position.

For a detailed description of the actions of this command and a definition of its parameters, see the Drop Tip command on page 7-76.

See Figure 7-9 on page 7-78 for a step-by-step illustration of this command.

This command is specific to Z axes with cLLD and arm groups containing that axis. The Drop Tip command is not yet supported for the Universal Z axis.

Example: Drop a disposable tip from Axis 3 with a start position of 200 mm, a search limit of 220 mm, an acceptance limit of 210, the default speed, and the default force factor:

Axis: `M3DropTip,200,220,210/`

Pick Tip

Synopsis: Picks up a disposable tip.

Syntax: `<AxisID>PickTip,<StartPos>,<Limit>,<AcceptanceLimit>,<Speed>,<ForceFactor>/`

Description: This command picks up a disposable tip.

For a detailed description of the actions of this command and its parameters, see the Pick Tip command on page 7-79.

See Figure 7-10 on page 7-81 for a step-by-step illustration of this command.

This command is specific to Z axes with cLLD.

Example: Pick up a disposable tip from Axis 3 with a start position of 50 mm, a search limit of 20 mm, an acceptance limit of 30, the default pick speed, and the default force factor:

Axis: `M3PickTip,50,20,30/`

7.7.5 Liquid Level Detection with 8-Channel Pipetting Head

Only applications with equal fill levels qualify for 8-channel head liquid level detection. It is the user's / programmer's responsibility to set the submerge distance appropriately to overcome minor differences between the individual fill levels and / or tolerances of the 8-channel head.

Caution! *When using the 8-channel pipetting head, users should be aware that the first probe making contact with liquid will cause the Z axis to stop moving. If the containers being used have uneven liquid fill levels, liquid level detection may have unexpected results.*

In addition, other factors may affect the ability of the 8-channel pipetting head to accurately detect liquid levels. For best results, check the following factors:

Levelness of Robotics

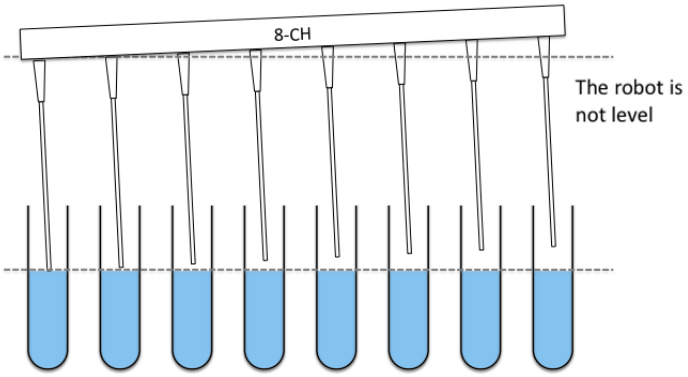


Figure 7-12 *Impact of robot not leveled properly*

Levelness of Labware

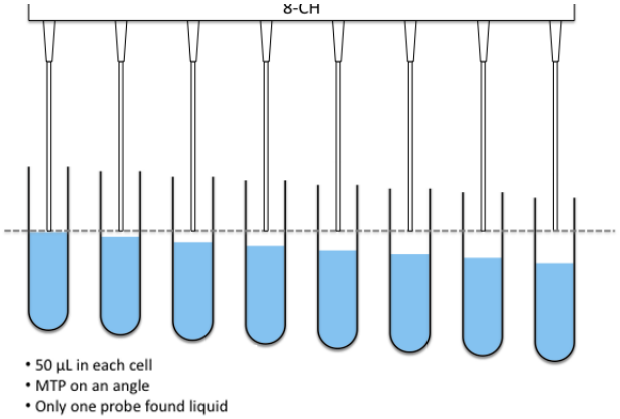


Figure 7-13 *Impact of uneven labware*

Uneven Liquid Levels

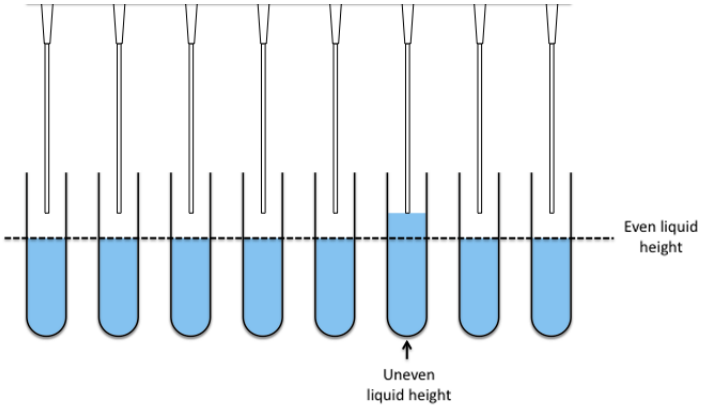


Figure 7-14 *Impact of uneven liquid levels*

7.8 Events

All events are sent from the CIB to the PC, so devices can actively send information back to the host.

Omni Robot Events use the following syntax:

Event, <EventID>, <Source>, <Comment>

Reports an event as it occurs on a device

This event reports an event as it occurs on a device connected to the Omni Robot. The format of event messages generated by the CIB follows a convention that includes the *DeviceType*, *DeviceID*, *EventID*, and any additional parameters.

<EventID>	Number identifying the event. See Section 7.8.1, "EventID Values" on page 7-89 for a list of *EventID*s and their parameters.
<Source>	The device type and ID of the event source. For example: - W0 - Command Processor - C0 - CIB - M1 - axis with ID number 1
<Comment>	Any additional text related to the event.
Examples:	*M4Event,106,M4,tip_loss/* -- tip lost at axis 4 *M2Event,101,M2,device_error,MER=0x11/* -- device error

WARNING! *The possibility of collision exists in a multiple arm system if one arm malfunctions and stops in the middle of a run. It is recommended that the OEM application software be designed such that if an asynchronous Device Error event occurs, the application will immediately terminate motion on all axes.*

7.8.1 EventID Values

The following tables list the possible values for event IDs and axis status.Table 7-11 shows the possible values of the EventID parameter, which is returned by an event.

Table 7-11 EventID values

EventID	Description	Event Parameter
101	Device Error	See Table 7-12.
102	TCP/IP Disconnected	N/A
104	CAN Buffer Overflow	N/A
105	CIB Voltage Unstable	N/A
106	Tip Loss*	N/A
107	CAN Bus Error	N/A
108	INT5 (This event is triggered by a low to high transition at Input 7. See the Input7 parameter.)	N/A
109	CIB check sum error	N/A
110	System unexpected error	N/A
111	Write log file error	N/A
112	Device type E detected	N/A
113	Gripper object dropped	N/A
114	Gripper unexpected signal change	N/A
115	Undefined event type	N/A
116	CIB internal buffer full	N/A

* The ADP U3R command must be issued to enable tip loss events. Refer to the ADP Manual for detailed information.

Table 7-12 displays the possible values of the axis error status register. The axis error status register is reported with the DeviceError event ID 101.

Table 7-12 Axis error status register

Bit	Error	Actual Response
0	CAN bus error	0x0001
1	Short-circuit	0x0002
2	Invalid setup data	0x0004
3	Control error	0x0008
4	Serial comm. error	0x0010
5	Position wrap around	0x0020
6	LSP (limit +) active	0x0040
7	LSN (limit -) active	0x0080
8	Over current	0x0100
9	I^2t	0x0200
10	Over temp - motor	0x0400
11	Over temp - drive	0x0800
12	Over voltage	0x1000
13	Under voltage	0x2000
14	Command error	0x4000
15	Enable input is inactive	0x8000

7.9 Error Codes

System and Command Error Codes are described in Appendix E, “Command Processor Command and Error Quick Reference Guide”.

7.10 Gripper Commands

This section provides reference information for Omni Robot software commands for the gripper. For a complete list of Omni Robot software commands, see Chapter 7, “Operating in Command Processor Mode”. For documentation about how to use the command set, see Section 7.3, ”Using the Command Set” on page 7-12. For an example of how to use a command in conjunction with a C# program, see Section 7.11.2, ”Omni API Usage Examples” on page 7-105.

Caution! *Don't send commands to the gripper which are not intended for execution with a gripper, because this could cause unintended motion of the gripper.*

WARNING! *There is a potential risk of collision during the initialization sequence. Take precautions to avoid collisions.*

WARNING! *Input a collision avoidance number larger than the size of the largest gripper with its maximum load, based on your system configuration and/or gripper finger type/orientation.*

WARNING! *If there are two arms and two grippers, there is a risk of collision. The collision avoidance number needs to be larger than the size of the larger gripper with its maximum load.*

WARNING! *Confirm that the gripper is mounted in the correct orientation and with the correct position offset. If you remove the gripper and replace it in a different mounting position, you must tell the software about the new position. The software cannot detect changes in gripper mounting position.*

GripperInit <InitMode>

Synopsis: Initializes the gripper.

Syntax: **`<AxisID>GripperInit,[InitMode]/`**

Description: The <*InitMode*> parameter specifies the initialization modes. Under normal init, the function will check for object presence prior to init and if object is detected, the function will not perform an initialization. The function will also not init if the CAM is in an intermediate position (neither fully closed nor opened). Alternative init will not perform the checking. Closed or open specifies the gripper fingers' position after the initialization is performed. The default <*InitMode*> is 0, normal init closed.

Note: *Initializing the gripper must be done by calling this command. Any other means of initializing the axis, ex. M3CavroInit, will not initialize the gripper.*

[*InitMode*]	Optional parameter. If not specified, defaults to 0. Possible values: 0 - Normal init, Closed 1 - Alternative init, Closed 2 - Normal init, Open 3 - Alternative init, Open
Generated Errors:	3 -- Invalid Operand 5 -- Device Not Implemented 8 -- Device Busy 9 -- Device Error (Hardware Error) 12 -- Invalid number of operands 41-- Gripper not attached 43 -- Gripper move error 44 -- Gripper CAM at intermediate starting position (Normal Init mode only) 45 -- Gripper object detected 48 -- Gripper sensor error

Example: To initialize the gripper with an alternative init, open:

```
M3GripperInit,3/
```

To initialize the gripper with no arguments:

```
M3GripperInit/
```

GripperClose

Synopsis: Closes the gripper.

Syntax: **<*AxisID*>GripperClose/**

Description: Performs the gripping operation. If an object is NOT detected after close operation is performed, an error code 46 - Gripper no object detected will be returned.

Generated Errors:	5 -- Device Not Implemented 8 -- Device Busy 9 -- Device Error (Hardware Error) 12 -- Invalid number of operands 41-- Gripper not attached 42 -- Gripper not initialized 43 -- Gripper move error 46 -- Gripper no object detected

Example: To close the gripper:

```
M3GripperClose/
```

GripperOpen

Synopsis: Opens the gripper.

Syntax: **<*AxisID*>GripperOpen/**

Description: Opens the gripper fingers.

Generated Errors:
5 -- Device Not Implemented
8 -- Device Busy
9 -- Device Error (Hardware Error)
12 -- Invalid number of operands
41-- Gripper not attached
42 - Gripper not initialized
43 -- Gripper move error

Example: To open the gripper:

```
M3GripperOpen/
```

GripperState

Synopsis: Checks the state of the gripper.

Syntax: **<*AxisID*>GripperState/**

Description: Determines the state of the gripper and returns gripper state information.

States:
0 - Gripper Open
1 - Gripper Not Attached
2 - Gripper Move Error
3 - Gripper Not Initialized
4 - Gripper Object Detected
5 - Gripper Closed

Example: To check the state of the gripper:

```
M3GripperState/
```

7.10.1 Gripper Events

Table 7-11 shows the possible values of the gripper EventID parameter, which is returned by an event.

Table 7-13 EventID values

EventID	Description	Event Parameter
113	Gripper Object dropped	N/A
114	Gripper unexpected signal change	N/A

7.10.2 Gripper Error Codes

The following table lists command error codes for the gripper. See Table E-7, “Command error codes,” on page E-14 for a complete list of all Omni error codes, including error codes for the gripper.

Table 7-14 *Command error codes*

Error No.	Code	Description
41	Gripper_not_attached	Gripper not attached
42	Gripper_not_initialized	Gripper not initialized
43	Gripper_move_error	Gripper move error
44	Gripper_CAM_at_intermediate_ starting_position	Gripper CAM at intermediate starting position (Normal Initmode only)
45	Gripper_object_detected	Gripper object detected
46	Gripper_no_object_detected	Gripper no object detected
48	Gripper_sensor_error	Gripper sensor error

7.10.3 Returning to Home Position With the Gripper Attached

Under normal circumstances, the program controlling the Omni should return the gripper to home position (or another chosen safe position) when it is not in use. The choice of using home position or another designated “Safe” position is at the programmer’s discretion.

If there has been an event such as a power failure, the program controlling the gripper may not be able to return the gripper to home position because the Omni may no longer have information about the size and location of attached fingers on

the gripper, or the gripper payload. It is also possible when power is lost that the gripper might move to a position from which it is blocked from moving to a safe position during the normal initialization process. In either case, the gripper can be returned to home using a different sequence of commands that allows the program to more slowly feel its way back to a safe position moving in smaller increments. Once the gripper has returned to a safe position, proper initialization (which is required for operation) can occur.

The following example illustrates how to help the Omni return to home position when the gripper is attached, with eccentric (L-shaped) fingers mounted pointing toward the X axis. In this example, the gripper fingers are located underneath the X axis, preventing normal initialization by mechanical interference, as shown in Figure 7-15.

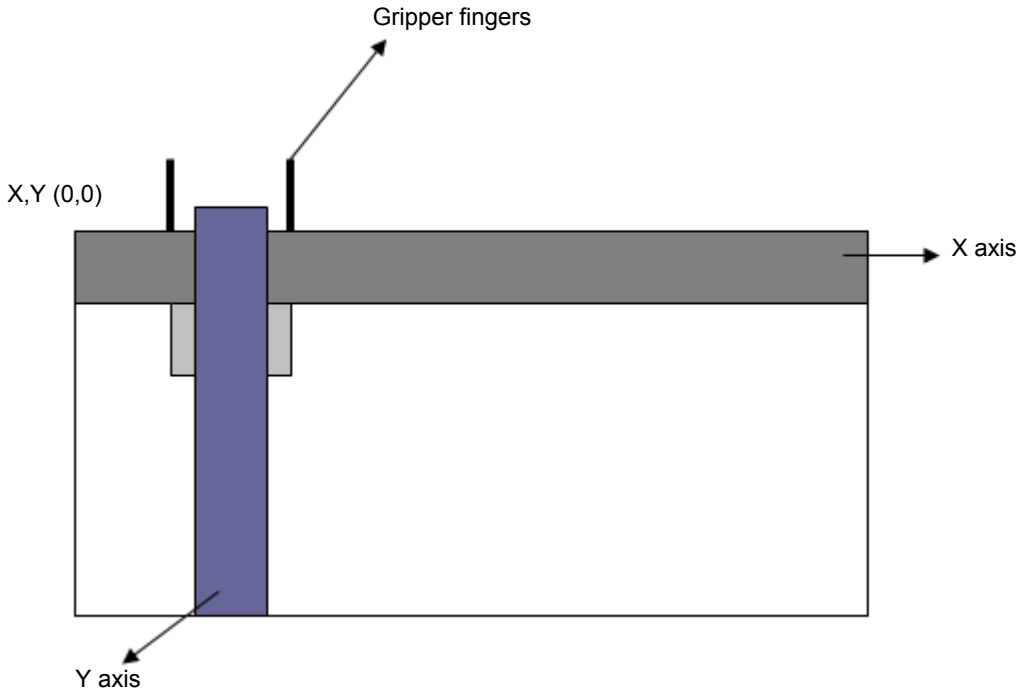


Figure 7-15 *Gripper with fingers underneath X axis*

7.10.4 Script Excerpt Example: Initialize arm procedure

Note: *This example script does not handle possible errors that could occur for each of the commands. See Section E, "Command Processor Command and Error Quick Reference Guide" for a list of possible errors for each command. If an error occurs, it is returned when the command is executed.*

Note: *This is not a complete script. It uses script excerpts to illustrate the steps in this procedure.*

Note: *The move parameter values shown in these examples are for illustration only. They should be replaced with values appropriate for each application.*

Note: *The* **PreInitMoveRel** *command operates at a slow fixed speed and is used to move one X or Y axis at a time. Choose the appropriate axis to start with for your scenario.*

Note: *For the re-homing in this sample script, one axis at a time is initialized. This maximizes the ability to perform each initialization safely.*

1 Use the **PreInitMoveRel** command to guide axes out to a safe position where the fingers are clear of the X axis and all axes can return home freely.

2 Move the gripper on the Y axis to a safe position where the Universal Z axis can initialize freely. (See Figure 7-16)

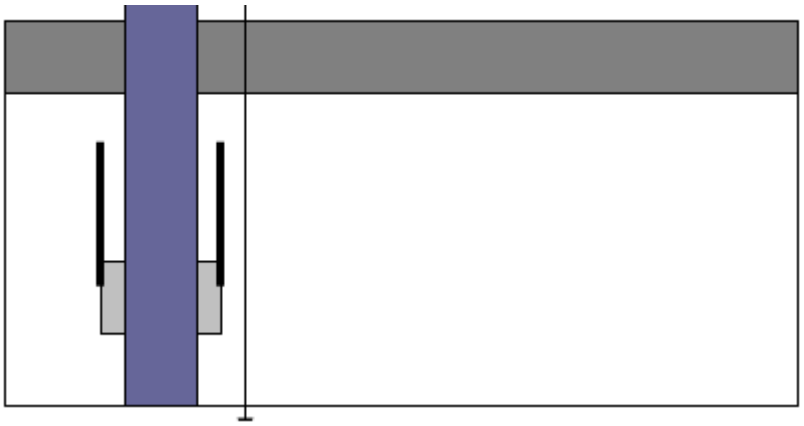


Figure 7-16 *Gripper moved to safe position*

```
//Relative move, 150 mm away from current position in positive
//direction
//Assume that 150 mm clears the finger without colliding with
//the X-extrusion
 reply = omniSvr.SendCmd("M2PreInitMoveRel,150");
 if (reply.ErrorCode != OmniErrorCode.No_Error)
 {
 //encounter error exit
 return;
 }
```

3 Initialize the Universal Z axis, retracting upward.

```
reply = omniSvr.SendCmd("M3CavroInit");
```

4 Initialize the X axis. (See Figure 7-17)

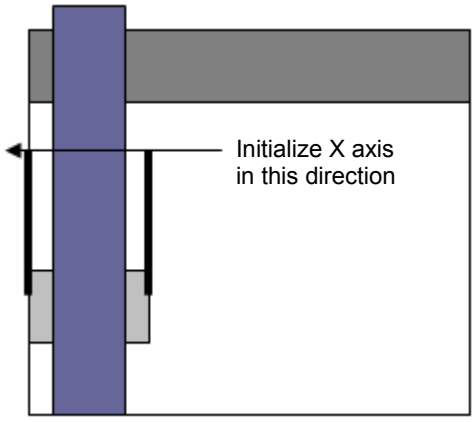


Figure 7-17 *Initialize X axis*

```
reply = omniSvr.SendCmd("M1CavroInit");
```

5 Move the Universal Z axis down to a safe position where it will not collide with the X axis body when the Y axis initializes.

```
//Assuming 150 mm will bring the gripper to safe position
//for initialization on Y axis.
 reply = omniSvr.SendCmd("M3MoveAbs,150");
```

6 Initialize theY axis. (See Figure 7-18)

```
reply = omniSvr.SendCmd("M2CavroInit");
```

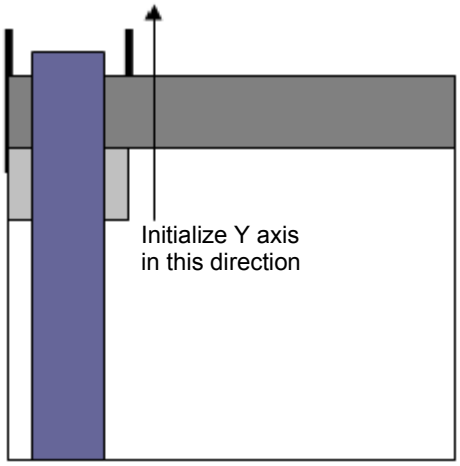


Figure 7-18 *Initialize Y axis*

Note: *The X axis and Y axis are now at home position, but the Universal Z axis is not. It has to move down away from home position so that the Y axis can return to home position. The important thing is that all axes now have been initialized and are ready for use.*

7 The application program can now initialize the gripper and other system peripherals.

7.10.5 Script Excerpt Example: Pick up and drop a plate

The following script excerpt example illustrates the commands that control picking up and dropping a plate:

```
private void PickDropAPlate()
{
       OmniReplyArgs reply = new OmniReplyArgs();

       //Goto position where a plate is located and where the
       //fingers can grab the object
       //Assuming X/Y/Z coords 300,150,150 is the position

       //Go to X/Y coord
       reply = omniSvr.SendCmd("M1MoveAbs,300");
       reply = omniSvr.SendCmd("M2MoveAbs,150");

       //Open the gripper
       reply = omniSvr.SendCmd("M3GripperOpen");

      //Go down to Z-coord where the gripper can grab the plate
       reply = omniSvr.SendCmd("M3MoveAbs, 150");
       //Grabbing plate
       reply = omniSvr.SendCmd("M3GripperClose");

       //Retracting Universal Z
       reply = omniSvr.SendCmd("M3MoveAbs, 200");

       //Universal Z goes down to a position where a plate
       //can be safely placed.
       reply = omniSvr.SendCmd("M3MoveAbs, 150");

       //Placing a plate by releasing the gripper
       reply = omniSvr.SendCmd("M3GripperOpen");

       //Retracting Universal Z
       reply = omniSvr.SendCmd("M3MoveAbs, 200");
}


private void ProcessTypedEvent(object sender, OmniEventArgs arg)
{
```

```
if(arg.EventCode == OmniEventCode.Gripper_object_dropped)
                MessageBox.Show("Event occurs: Gripper drops object");
}
```

7.11 Using the Omni Command Processor APIs

The Cavro® Omni Robot Command Processor (CP) is designed to run under the Microsoft .NET framework and is shipped as a Dynamic Link Library (DLL). The library, **`NrcCmdProc.dll`**, provides functions to enable the Omni Robot application developer to send commands to the Omni Robot programmatically.

Because the Command Processor is a Microsoft .NET-based component, the developer must choose a programming language that is capable of using this library. These languages may include C#, VB.NET, LabVIEW, and IronRuby.

In Version 2 and newer releases of the Omni Robot software, a set of APIs (V2 API) has been added to the **`NrcCmdProc.dll`** library. The V2 APIs encapsulate the TCP/IP communication used for sending commands to the Omni Robot, eliminating the need for the developer to deal with TCP/IP I/O directly. This greatly simplifies the interface to the Omni Robot for the developer. For detailed information about the features available in different versions of the Omni Robot software, see Table 1-1, “Support for Omni features by software version number,” on page 1-6.

The V1 API continues to be supported along with the V2 API. However, because the V2 API is built on the V1 API, the developer must choose to instantiate either

- V1: NrcCmdProc.CmdProcMain class

or

- V2: NrcCmdProc.V2.OMNIApi class

An application can only activate one set of APIs, and will not have access to the Omni Robot’s TCP/IP port when using the V2 API.

Note: *When using a different version of* ***NrcCmdProc.dll*** *in an existing application, Tecan recommends that you verify and validate your application again after making this change.*

7.11.1 The Cavro Omni Robot V2 API

The Omni V2 API allow applications to be in complete control of the Cavro® Omni Robot by calling the functions in the APIs. The V2 API handles all communication with the Omni Robot—the application will not have access to the Omni Robot's TCP port when using the V2 API.

The classes for V1 and V2 APIs are organized as follows:

Table 7-15 Omni V1/V2 API class descriptions

Name Space	Classes	Description
`NrcCmdProc`	`CmdProcMain`	Main V1 API class
	`OmniErrorCode`	All Omni errors including both versions of APIs and the Command Set.
	`OmniEventCode`	All Omni events including both versions of APIs and the Command Set.
	`OmniEventArgs`	All Omni event arguments
	`OmniReplyArgs`	All command reply arguments
`NrcCmdProc.V2`	`OmniAsyncResult`	Synchronization information
	`OmniExecResult`	Contains the results of a command execution
	`OmniApi`	Main V2 class

CmdProcMain Class

The CmdProcMain class provides the methods to establish a TCP/IP connection to the Omni by connecting the TCP/IP server hosted inside the Command Processor to the application (see Figure 7-1, "Communication flow diagrams, V1 vs. V2 Command Processors" on page 7-2). All data transmission between the application and the Omni robot uses the Omni data protocol. See Section 7.3, "Using the Command Set" for more information on how to format outgoing messages (events) and how to parse and interpret incoming messages (responses).

Table 7-16 CmdProcMain Class Methods

public CmdProcMain ( )
Class constructor. The log file will be written to the `Log` *folder under* `Application.UserAppDataPath`.
public CmdProcMain (string logpath)
Class constructor. The log file will be written to the `log` folder under **logpath**.
public StartCmdProc (string *cibAddr*, int *cibPortNum*, int *cpListenerPort*, string *binAddrString*, int *bindPort*)
Activates the Command Processor and attempts to establish communication between the PC and the Omni Robot.
public OmniErrorCode StartCmdProc(string *cibAddr*, int *cibPortNum*, int *cpListenerPort*, string *binAddrString*, int *bindPort*)
Overload method that allows host application to specify IP address of the local net card to physically connect to the Omni and the specific port. (If the port is in use, the Omni Command Processor will search for the first free port.
public void ShutdownCmdProc()
Deactivates the Command Processor and closes the connection to the Robot.

OmniApi Class

The OmniApi class provides the methods to send commands to the robot and catch the events from the robot.

Table 7-17 OmniApi class methods

public OmniApi ( )
Class constructor. The log file will be written to the `..\Log` directory under the system-specific user directory, for example, if the system-specific user directory is `C:\Documents and Settings`, then the log file will be in `C:\Documents and Settings\Log`.
public OmniApi (string logPath)
Class constructor. The log file will be written to the directory specified as the `[logPath]`.
public OmniErrorCode InitOmniApi (string *cibAddr*, int *cibPortNum*)

Table 7-17 OmniApi class methods (continued)

Activates the Command Processor and attempts to establish communication between the PC and the Omni Robot. An application can use other methods in this class after a successful call to this method. Internally, InitOmniApi calls *NrcCmdProc.CmdprocMain.StartCmdProc()*.
public OmniErrorCode InitOmniApi (string *cibAddr*, int *cibPortNum*, string *binAddrString*, int *bindStartPortNum*)
Overload method that allows host application to specify IP address of the local net card to physically connect to the Omni and the specific port. (If the port is in use, the Omni Command Processor will search for the first free port.
public void TerminateOmniApi ()
Deactivates the Command Processor and closes the connection to the Robot. This function calls *NrcCmdProc.CmdprocMain.ShutdownCmdProc()*.
public bool AddEventHandler (EventHandler<OmniEventArgs> *handler*)

Subscribe to Omni Robot event. The handler must have the following form:
`void handler(Object sender, OmniEventArgs arg)`
`{`
`  // code to process the received event`
`  // the event code is in arg.EventNumber`
`}`

public bool DeleteEventHandler (EventHandler<OmniEventArgs> handler)
Remove handler from Omni event subscription list.
public OmniReplyArgs SendCmd (string cmd)

Send an Omni command synchronously.
The function will block the calling thread until the command execution has completed by the robot. For non-native commands such as “D” commands, SendCmd will return to the caller as soon as it receives a reply from the target device.
Examples:
Device type M
`SendCmd(“M1CavroInit”);`
Function returns when the axis completes axis initialization.
Device type D
`SendCmd(“D1,ZR”);`
Function returns as soon as the device receives the command, and waits for status. The application needs to poll the device state for command execution completion.

Table 7-17 OmniApi class methods (continued)

public OmniReplyArgs SendCmd (string *cmd*, int *waitTime*)
Sends an Omni command synchronously with a timeout. The function will block the calling thread until the command execution completes or the function times out. *waitTime* is in milliseconds. Set *waitTime* to -1 or *Timeout.Infinite* (.NET System.Threading) for an infinite wait time.
public OmniAsyncResult BeginSendCmd (string *cmd*, int *waitTime*)
Sends an Omni command asynchronously with a timeout. A subsequent call to *EndSendCmd()* will block the calling thread until the command execution completes or timeout. *waitTime* is in milliseconds. Set *waitTime* to -1 or *Timeout.Infinite* (.NET System.Threading) for an infinite wait time.
public OmniAsyncResult BeginSendCmd (string *cmd*, int *waitTime*, AsyncCallback *callback*, Object *state*)

Sends an Omni command asynchronously with a timeout and a callback. A subsequent call to *EndSendCmd()* will block the calling thread until the command execution completes or timeout. The callback will receive the “state” object when the function gets called.
The callback function is as follows:
`void callback(IAsyncResult ar)`
`{`
`  OmniAsyncResult result = (OmniAsyncResult result)ar;`
`}`

public OmniReplyArgs EndSendCmd (OmniAsyncResult *asyncResult*)
Blocks the calling thread until *asyncResult* completes the wait.
public event EventHandler OmniEventUntyped

Allow the application to use a generic handler to subscribe to Omni events.
Example:
`void handler(Object sender, EventArgs arg)`
`{`
`  OmniEventArgs omniEvent = (OmniEventArgs)args`
`  ...`
`}`

OmniAsyncResult Class

This class implements the *IAsyncResult* interface for *BeginSendCmd()*.

OmniReplyArgs Class

The example shows the parsed reply string: **M1Get-Pos,0,12.3**

Table 7-18 OmniReplyArgs class members

Type	Name	Description	Example
`OmniErrorCode`	`ErrorCode`	See OmniErrorCode*	`No_Error`
`int`	`ErrorNumber`	See OmniErrorCode*	`0`
`string`	`Comment`	Comment section of the reply string	`12.3`
`double`	`FloatValue`	Double value of the Comment section, null if conversion fails.	`12.3`
`int`	`IntValue`	Integer value of the Comment section, null if conversion fails.	`null`
`char`	`DeviceType`	Sender device type.	`M`
`ushort`	`DeviceID`	Sender device ID	`1`
`string`	`ReplyString`	The original reply string	`M1Get-Pos,0,12.3`

* *See Section 7.8.1, "EventID Values" on page 7-89 for a list and definitions of the error codes.*

OmniEventArgs Class

The class properties contain parsed event information. The example shows the parsed event string: **M1Event,101,Device_error,code=0x0002**

Public members:

Table 7-19 OmniEventArgs class

Type	Name	Description	Example
`OmniEventCode`	`EventCode`	See the section OmniEventCode	`Device_error`
`int`	`EventNumber`	See the section OmniEventCode	`101`
`string`	`Comment`	The comment part of the event string	`Device_error, code=0x0002`

Table 7-19 OmniEventArgs class (continued)

char	**DeviceType**	The DeviceType of the event source.	**M**
ushort	**DeviceID**	The device ID of the event source.	**1**
string	**EventString**	The original event string sent by the device	**M1Event,101, Device_error, code=0x0002**

OmniEventCode

See Section 7.8.1, ”EventID Values” on page 7-89 for a list and definitions of the event codes.

7.11.2 Omni API Usage Examples

The following illustrates the usage of the Omni V2 API. Note that these are not complete examples, rather just snippets illustrating the usage of the main methods defined above.

You must include both the V1 and V2 name spaces:

```
using NrcCmdProc;
using NrcCmdProc.V2;
```

Make sure these are added to the project References list.

The following lines show the basic structure of code that starts the Command Processor, sends some commands, and then shuts down the Command Processor.

```
////////////////////////////////////////////////////////////
   // Create an Omni API object
   NrcCmdProc.V2.OmniApi omni = new OmniApi();

   // initiate Command Processor
   omni.InitOmniApi("192.168.0.198", 9898);

   // initalized arm 0
   omni.SendCmd("A0CavroInit");

   // Move arm 0 to location 100, 100, 100
   omni.SendCmd("A0MoveAbs,100,100,100");

   // terminate omni
   omni.TerminateOmniApi();

////////////////////////////////////////////////////////////
```

In reality, an application would also do some error checking. The following lines illustrate this: the code sends a Move command only if the CavroInit command has succeeded, and prints a message if the Move command is successful.

```
OmniReplyArgs reply;
OmniErrorCode code;

NrcCmdProc.V2.OmniApi omni = new OmniApi();
code = omni.InitOmniApi("192.168.0.198", 9898);
if (code == OmniErrorCode.No_Error)
{
    reply = omni.SendCmd("A0CavroInit");
    if (reply.ErrorCode == OmniErrorCode.No_Error)
    {
        reply = omni.SendCmd("A0MoveAbs,100,100,50");
        if (reply.ErrorCode == OmniErrorCode.No_Error)
        {
            MessageBox.Show("A0 Moved");
        }
    }
}
// terminate omni
omni.TerminateOmniApi();
```

The API provides methods to subscribe to and unsubscribe from Omni events (the `AddEventHandler()` and `DeleteEventHandler()` methods). However, the API also supports the use of the compound operators `+=` and `-=` for subscribing and unsubscribing. For example:

```
Omni.AddEventHandler(ProcessTypedEvent);
```

and

```
Omni.OmniEventUntyped += ProcessUntypedEvent;
```

are functionally the same.

Additional examples written in C# are provided with the software distribution, and can be found in the **`OMNISampleCode`** folder in the root directory of the Installation CD.

7.12 Basic Omni Hardware Control

For examples of Command Process command sequences, see Section G.3, ”Solutions” in Appendix G.

7.13 Resolving Performance Issues

When operating the Omni Robot, your PC could perform poorly if the TcpAckFrequency in the PC registry is not set to 1. To resolve this issue, see the following web pages:

- http://support.microsoft.com/kb/328890
- http://support.microsoft.com/kb/815230

This page has been left intentionally blank.

8 Cavro Air Displacement Pipettor Mounting and Integration

The Cavro® Air Displacement Pipettor (ADP) is a compact OEM pump module designed to automate hand pipetting applications in the <5 µL to 1 mL range. It provides excellent pipetting performance in an easily integrated, flexible, and highly reliable platform.

This chapter describes the interface between the ADP and the Cavro® Omni Robot. This chapter is organized into the following sections:

For more information about the ADP, refer to the *Cavro® Air Displacement Pipettor Operating Manual*.

8.1 Introduction

Figure 8-1 shows the main components of the ADP.

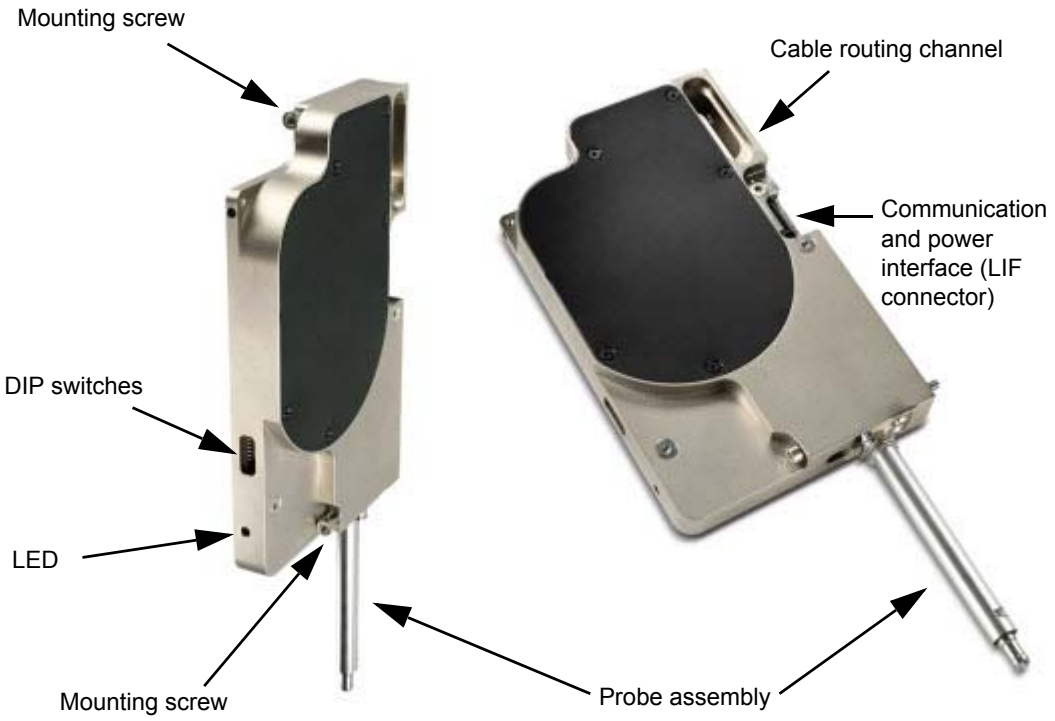


Figure 8-1 *ADP components*

8.2 Operating Temperature Range

Caution! *If an ADP is mounted on the Omni, the operating temperature range is 15-35°C, per ADP operating temperature specifications. (Without the ADP, the operating temperature range for the Omni is 10-40°C.)*

8.3 Installing the ADP

Tools needed:

- 2.5 mm hex Allen wrench

Steps:

1 Remove the cover from the Universal Z axis by pulling it straight off, as shown in Figure 8-2.

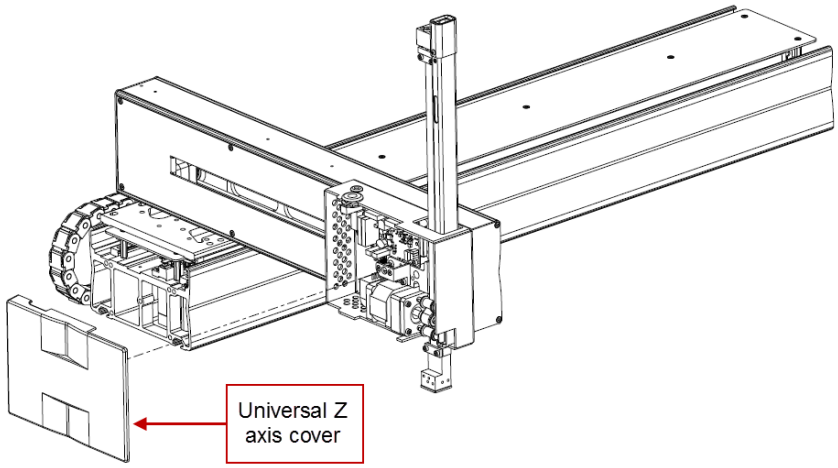


Figure 8-2 *Removing Universal Z axis cover*

2 Verify that the brake PCBA connections are set to factory specifications as shown in Figure 8-3.

Caution! *Ensure that the FFC cable is perpendicular to the printed circuit board and fully inserted into the connector to avoid possible damage.*

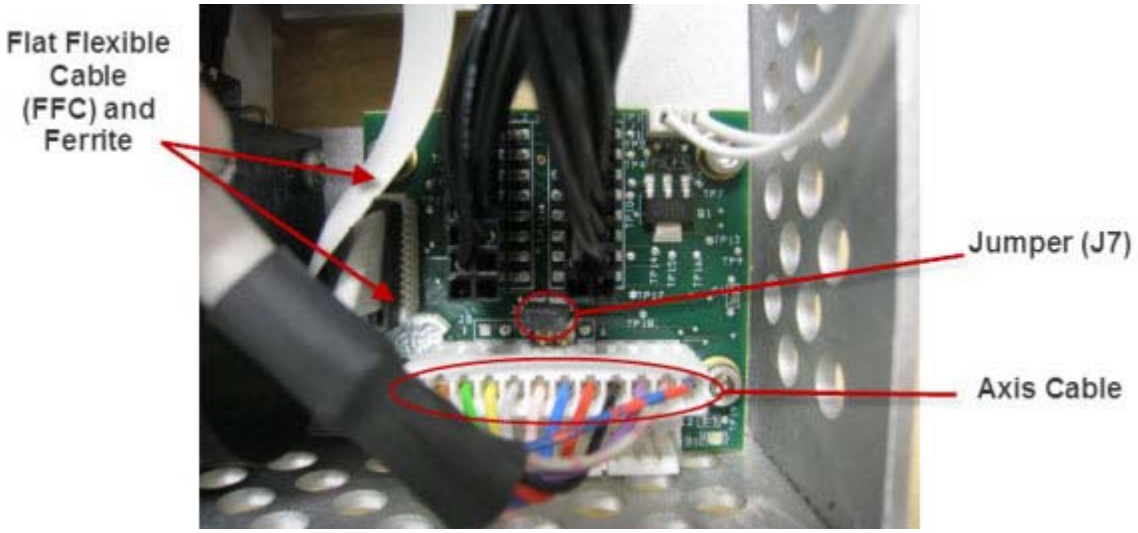


Figure 8-3 *Brake PCBA connections*

Note: *The axis cable supplies power and communication to the ADP.*

3 Route the FFC and cLLD coax cables from the tubing guide as shown in Figure 8-4.

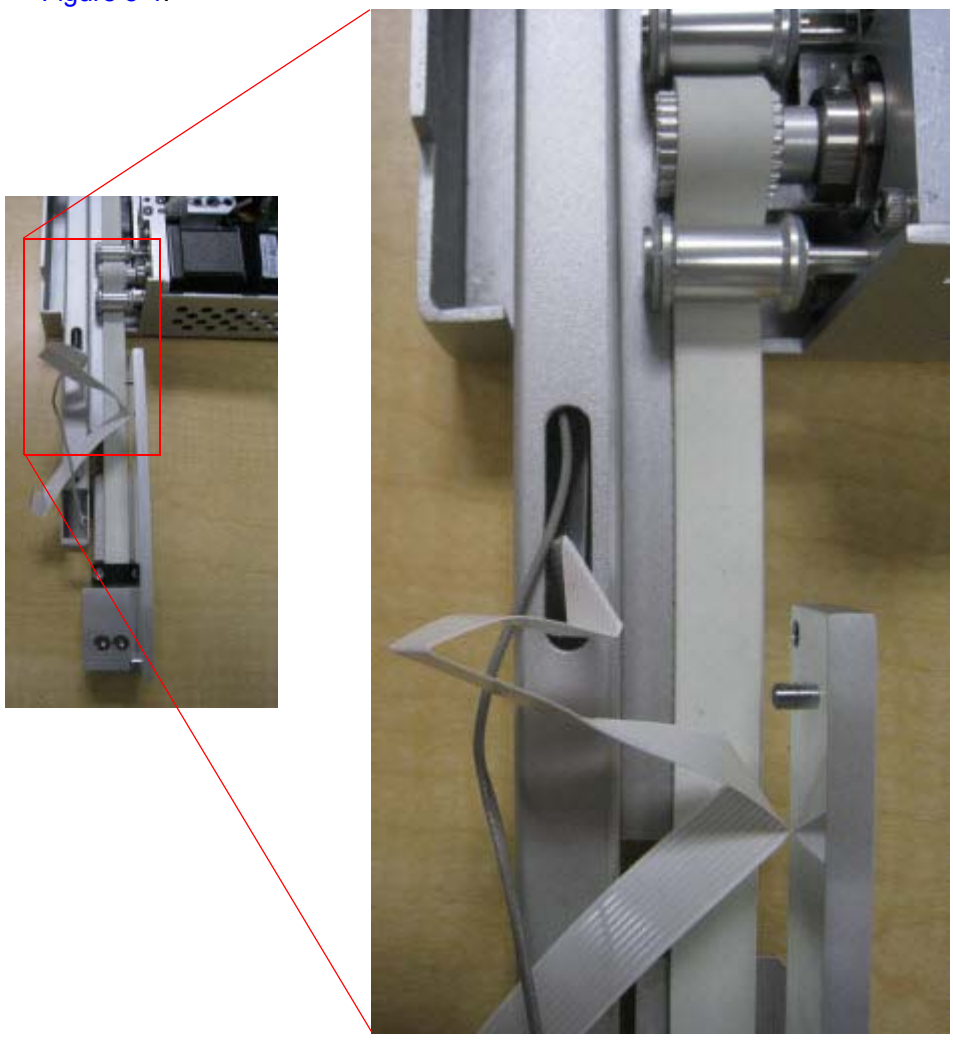


Figure 8-4 *FFC and cLLD coax coming out of the tubing guide*

4 Plug the FFC and cLLD coax cables into the ADP. The cable connectors on the ADP are shown in Figure 8-5 and the connected cables are shown in Figure 8-6.

Caution! *Ensure that the FFC cable is perpendicular to the printed circuit board and fully inserted into the connector to avoid possible damage.*

Caution! *The FFC is pre-folded on both ends of an Omni Universal Z with ADP. Do not change the cable folds, because this can damage the cable.*

Note: *Spare FFCs are folded on one end. Refer to the instructions in document 30065978,* "Customer Work Instruction for Replacement of FFC" for how to install and fold the other end.

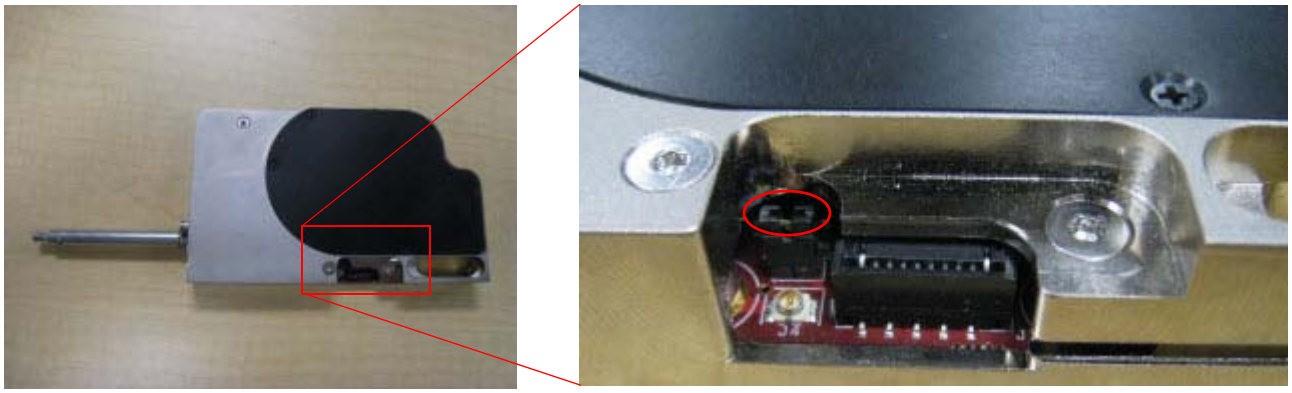


Figure 8-5 *ADP cable connectors*

Note: *The jumper circled in Figure 8-5 should be removed when an ADP is used on the Omni. If the jumper is left in place, the ADP will not be able to perform cLLD.*

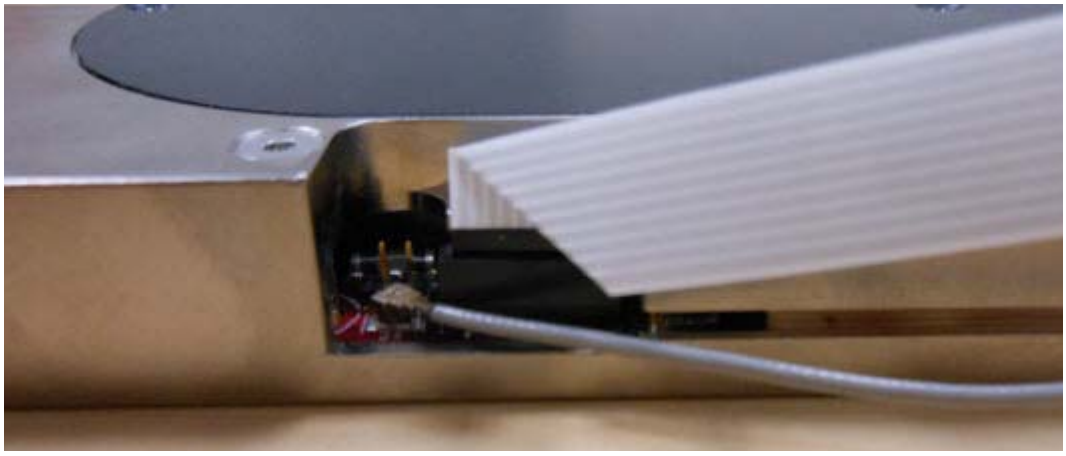


Figure 8-6 *FFC and cLLD cables connected to ADP*

5 Place the FFC and cLLD cables into ADP frame cable guides as shown in Figure 8-7 and Figure 8-8.

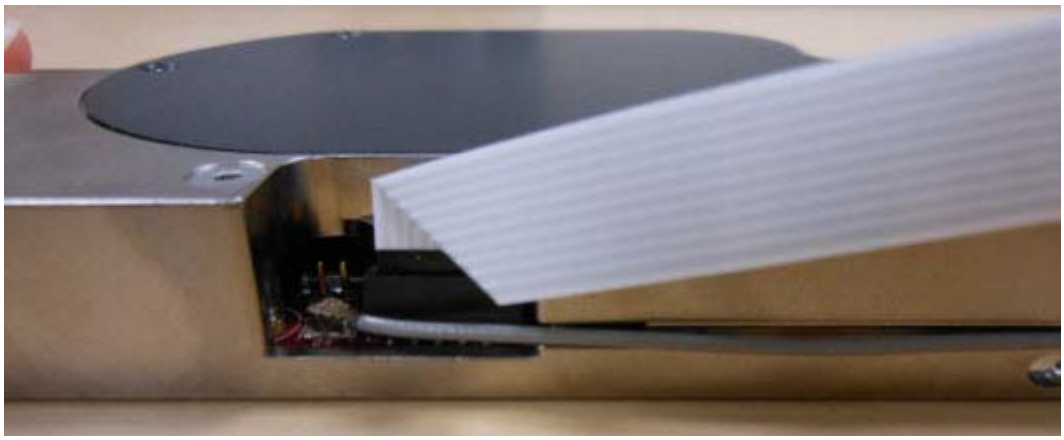


Figure 8-7 *The cLLD coax cable positioned along the cable guide on the ADP*

Caution! *Use care when positioning the cLLD coax cable. It is possible to accidentally damage the flat ribbon cable.*

Figure 8-8 *The FFC positioned over the cLLD coax along the cable guide on the ADP*

6 Place the ADP onto the mounting bracket, using the dowel pins for location and alignment as shown in Figure 8-9 and Figure 8-10. Be careful not to let the cables move out of the guide features, or the ADP will not sit properly on the bracket. Improper seating of the ADP in the bracket might cause interference, cable damage, etc.

In Figure 8-9, the picture on the left shows moving the ADP into position to align with the dowel pins, and the picture on the right shows the ADP positioned on the Universal Z.

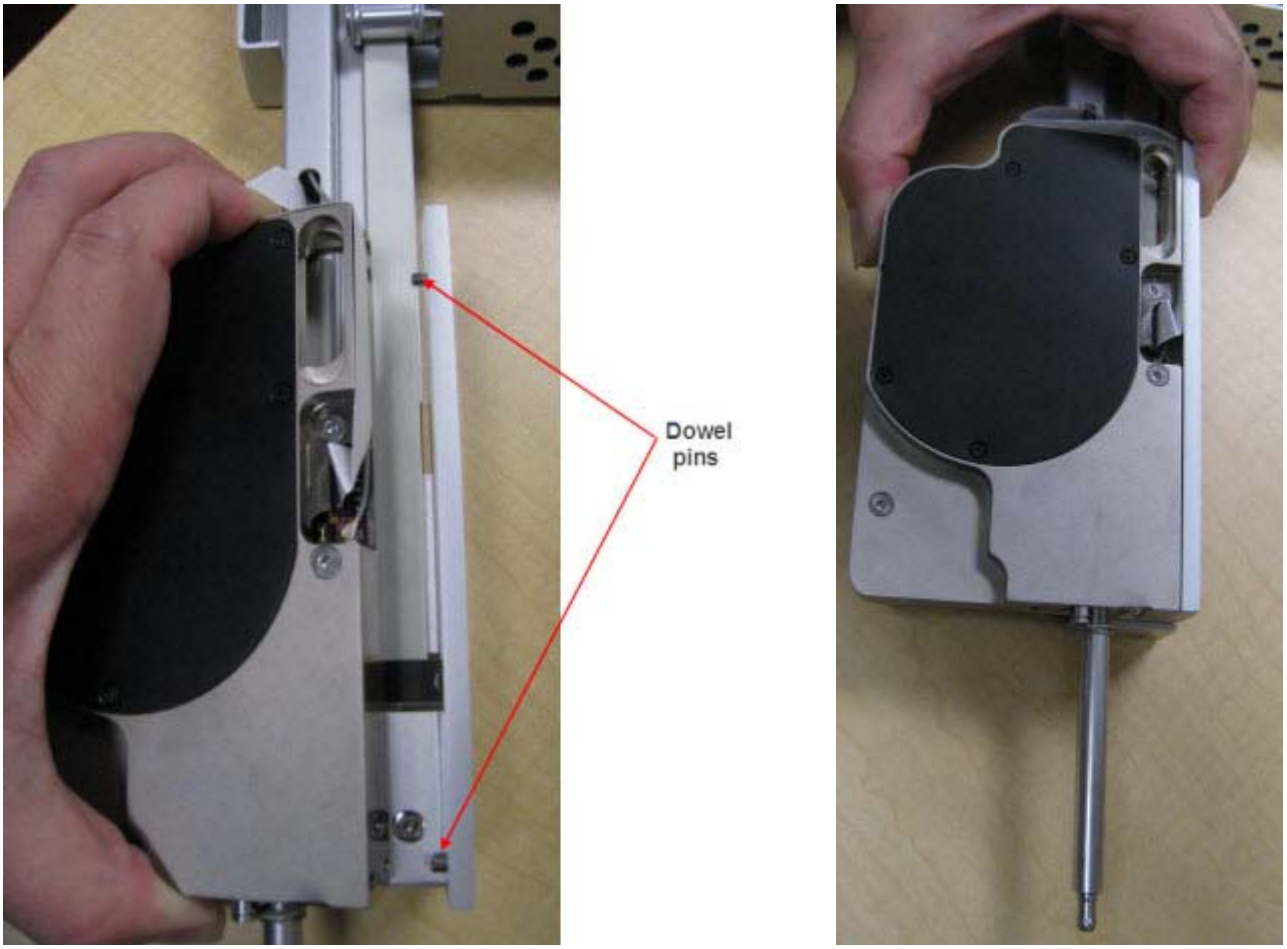


Figure 8-9 *Mounting the ADP on the Universal Z axis*

Figure 8-10 *Close-ups of dowel pins and mating hole/slot*

7 Install the lower mounting screw as shown in Figure 8-11.

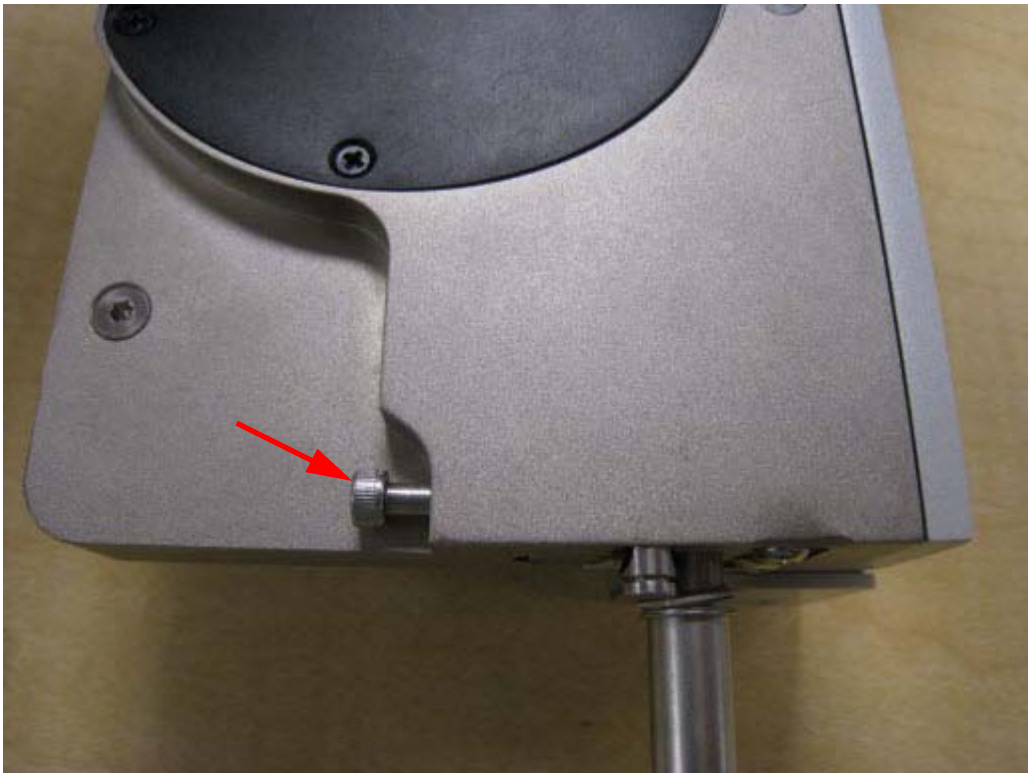


Figure 8-11 *Lower mounting screw*

8 Install the FFC clamp bracket and upper mounting screw. The bracket and mounting screw are shown before installation in Figure 8-12 and after installation in Figure 8-13.

Figure 8-12 *FFC clamp bracket and upper mounting screw before installation*

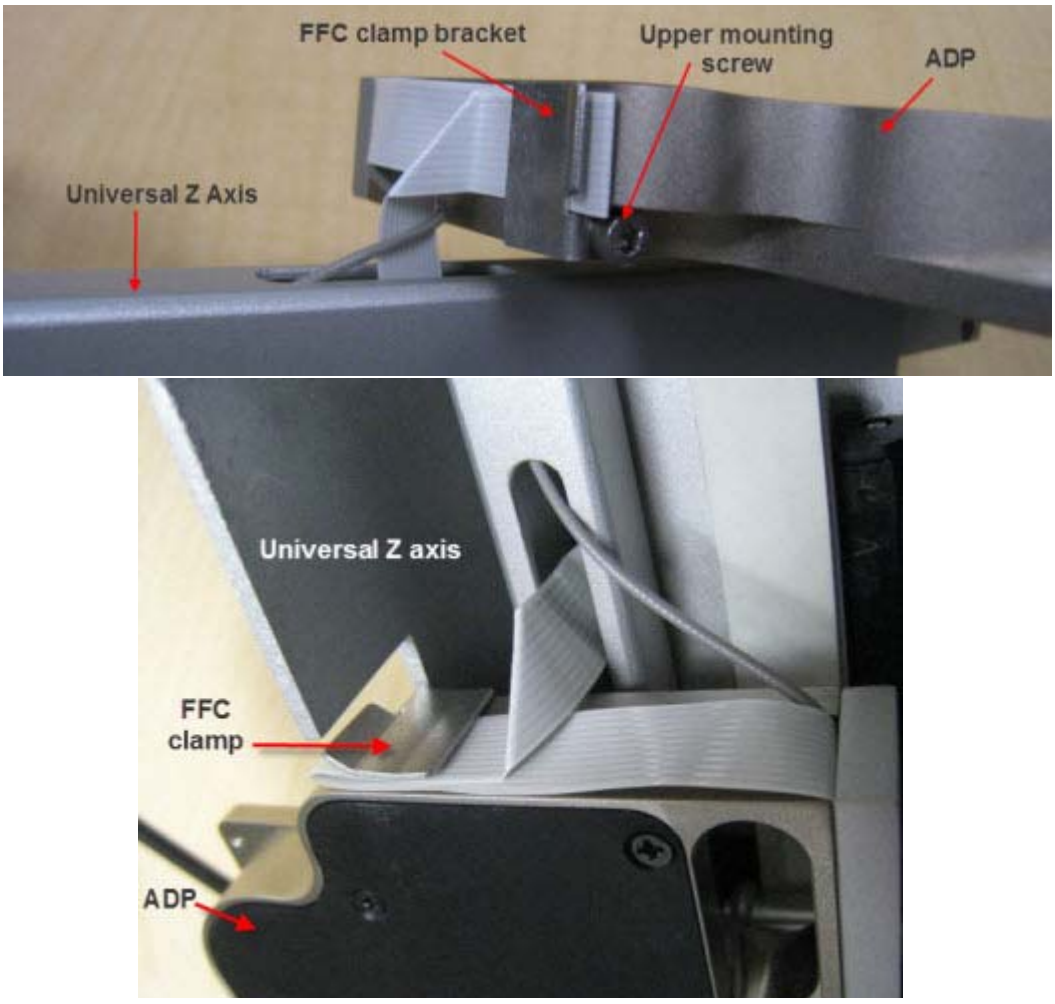


Figure 8-13 *Two views of the clamp bracket and upper mounting screw attached to the ADP*

9 Verify that the DIP switch settings on the ADP match those shown in Figure 8-14.

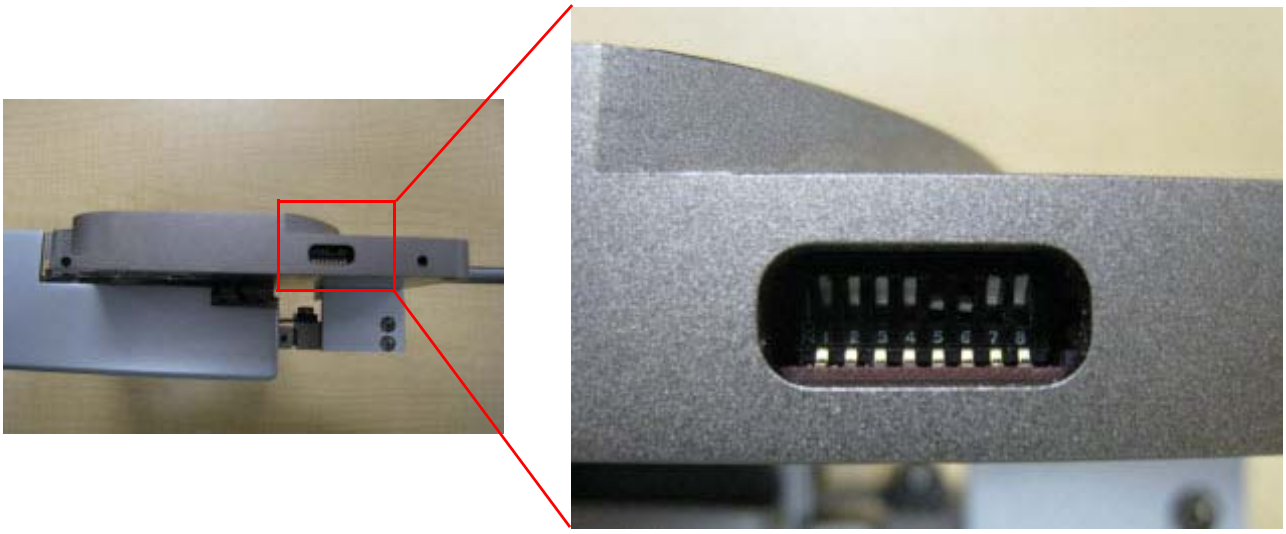


Figure 8-14 *DIP switch settings*

DIP switch positions (up or down) can be changed using a small screwdriver or similar tool. Changes to the ADP DIP switch settings require the ADP to be power cycled. They do not take effect until the ADP is power cycled.

Use the following table to ensure the correct settings for your application.

Table 8-1 DIP switch settings

Switch	Function	Switch Down	Switch Up
1	Address0	on	off (default)
2	Address1	on	off (default)
3	Address2	on	off (default)
4	Diagnostics (self-test)	enabled	normal operation (default)
5	RS-485 termination	true (default)	false
6	CAN/RS-485 termination	true (default)	false
7	Communication Mode	RS-485	CAN (default)
8	Baud rate (programmable)	- 9600 (RS-485) - auto-detect (RS-485) - 100K (CAN)	- 9600 (RS-485) - 38400 (RS-485) - 115200 (RS-485 default) - 100K (CAN) - 125K (CAN) - 250K (CAN) - 500K (CAN default) - 1M (CAN)

The default DIP switch settings for ADP set the termination jumper to ON for ADP. The termination jumper on the Z-ACB should be removed. This applies for both single and dual ADP (on a dual arm Omni) systems. For more information about DIP Switch settings, refer to the *Cavro® Air Displacement Pipettor Operating Manual*.

8.4 ADP and Omni Command Processor Integration

All ADP data streaming is transmitted through the Command Processor to the control application in the form of events. The host application should subscribe to these events and handle this data according to the specified settings. See the *Cavro® Air Displacement Pipettor Operating Manual* for more information about the data streaming settings.

The purpose of the sample code in this section is to show briefly how to control and intercept the ADP data streaming, for the complete solution, see the complete sample application code located in the `..\\Demo` directory where Omni V4.0 is installed.

8.4.1 Integration Sequences

Note: *This sequence assumes that the ADP is already installed correctly, as described in Section 8.2, "Operating Temperature Range"*

1 Start the Command Processor using the InitOMNIApi method. See Table 7-17, OmniApi class methods for more information.

```
OmniApi api = new OmniApi();
OmniErrorCode code = api.InitOmniApi("192.168.0.198", 9898);
```

2 Subscribe to the Command Processor to receive ADP data streaming in the form of Events.

Note: *See Section 7.8, "Events" and Section 6.6, "Status Codes, Error Codes, and Events" for more information about Events. See Table 7-17, OmniApi class methods for more information about* `AddEventHandler`.

```
//Adding event handler
api.AddEventHandler(OmniEventHandler);

//Event Handler Implementation
void OmniEventHandler(Object sender, OmniEventArgs arg)
{
//event coming back from E CAN devices
   if (arg.DeviceType == 'E')
   {
      //Data pressure streaming
      if ((arg.Comment[0] == '3' && arg.Comment[1] <= '3' &&
         arg.Comment[1] >= '1')
      {
         //See sample code for complete data parsing solution
      }
      //cLLDevent signal
      else if (arg.Comment[0] == '4' && arg.Comment[1] == 'C')
      {
         //See sample code for complete data parsing solution
      }
   }
}
```

3 Handle ADP events.

Note: *See the Cavro® Air Displacement Pipettor Operating Manual for a list of all possible returned event IDs and data.*

```
void OmniEventHandler(Object sender, OmniEventArgs arg)
{
   //event coming back from E CAN devices
   if (arg.DeviceType == 'E')
```

Caution! *This is a code snippet. It is* ***not*** *the complete data parsing solution. Refer to the ADP sample code that is installed in the OMNI Configurator application directory.*

```
{
//Data pressure streaming
      if ((arg.Comment[0] == '3' && arg.Comment[1] <= '3' &&
         arg.Comment[1] >= '1')
      {
         //See sample code for complete data parsing solution
      }
      //cLLD event signal
      else if (arg.Comment[0] == '4' && arg.Comment[1] == 'C')
      {
         //See sample code for complete data parsing solution
      }
      else if (arg.Comment[0] == '4' && arg.Comment[1] == '3')
      {
         // event type 43H, 'C' cLLD low - > high
      }
      else if (arg.Comment[0] == '6' && arg.Comment[1] == '3')
      {
         // event type 63H, 'c' cLLD high - > low
      }
      else if (arg.Comment[0] == '5' && arg.Comment[1] == '4')
      {
         // event type 54H, 'T' Tip loss event
      }
   }
}
```

Caution! *Assuming that the host software and hardware are capable of handling all the data streams at the selected rate and does not cause a bottleneck in reading the data from CIB:*

- *The Omni Command Processor can support up to 4 pressure data streams at minimum of 3 milliseconds interval between event messages for moderate network traffic (Ethernet and CAN).*
- *For higher network traffic, it is recommended to lower the number of the pressure streams or increase the time interval between the event messages.*

4 General Settings:

– U3R enables RS-485-Ch 1 to be used for a tip loss event. See the "Commands" section in the *Cavro® Air Displacement Pipettor Operating Manual* for more information.
– !R performs a soft reset of the ADP device

Set data sampling interval and data points per package, for example:

– ADP: p6,20R (set sampling interval in ms)
– ADP: p7,1R (set data point per package)

Set data streaming events over CAN, for example:

– ADP: o6,1R (enable LLD event see ADP manual)
– ADP: o5,1R (enable cLLD high -> low event)
– ADP: o4,1R (enable cLLD low -> high event)
– ADP: o1,1R (enable tip loss event)

The following code snippet enables the ADP to generate a tip loss event:

```
api.SendCmd("E0-1,o1,1R");
```

5 System Initialization:

– OMNI: CavroInit (initialize OMNI system)
– DP: WR (initialize ADP)

Caution! *Only one mode can be active at a time. Whichever mode is set last will be the active mode.*

6 Set Detection Mode

ADP: cLLD mode:

– ADP: U71R (note: enable cLLD mode)

ADP: pLLD mode:

– ADP: U70R (note: enable pLLD mode)

ADP: hLLD mode:

– ADP: U72R (note: enable hLLD mode)

Note: *pLLD works by using slow constant aspiration to detect blocks to aspiration that changed pressure (for example, encountering liquid). pLLD will not work correctly if the plunger is positioned too high at the start of pLLD. Make sure the plunger is in the proper position for pLLD data streaming before continuing.*

7 Start Data Streaming:

– ADP: o0,1R

8 ADP/OMNI liquid Detection and Aspirate/Dispense sequences:

Note: *There are two things that have to terminate:*

- *First, the motion of the OMNI. It will begin at start search position and search till it either finds liquid or reaches search limit.*
- *Second, the ADP is independent of the OMNI. The ADP will terminate search when it finds liquid.*

Note: *The ADP will not terminate if no liquid is found unless the B1R command is terminated when the ADP reaches the search limit.*

```
// detect liquid from current plunger pos, B1 - start plunger
// from current location
// The following two commands must be executed simultaneously. This
// is done with the api.BeginSendCmd function

OmniAsyncResult result1 = api.BeginSendCmd("E0-1,B1R", -1);

OmniAsyncResult result2 = api.BeginSendCmd("M3DetectLiquid, 150,
                                           100", -1);

OmniReplyArgs reply2 = api.EndSendCmd(result2);

//Terminating ADP B1R command
api.SendCmd("E0-2,4",-1);

//Aspirate
api.SendCmd("E0-1,P50,1R", -1);

//Dispense
//go to dispense position
api.SendCmd("A0MoveAbs,X,Y,Z",-1);

//ADP dispense
api.SendCmd("E0-1,D80,1R", -1);
```

9 Stop Data Streaming:

– ADP: o0,0R

8.5 ADP and Omni Embedded Integration

In Embedded Mode, ADP data streaming is only available through TCP/IP port 9897. The host will receive ADP data streaming as event packages. Section 6.5.6, ”Event Responses” for detailed information about the event package format. Also please refer to Section 6.13.2, ”Basic Omni Hardware Control” for ADP/OE liquid detection and aspirate/dispense sequences.

8.6 Checking for proper ADP function after a collision

If a collision occurs, it is important to take the following steps before using the ADP with a payload.

1. Initialize the Omni.
2. Pick up a tip with the ADP.
3. Perform a full aspirate and dispense sequence.
4. Eject the tip from the ADP.
5. Examine the probe assembly for visual scratches that might affect performance.

This page has been left intentionally blank.

9 Gripper Mounting and Integration

The Omni gripper module is a robust, flexible, plate-and-tube, gripper device. Depending on the type of fingers installed, the gripper can grip standard microtiter plates (MTP) that are approximately 85.5 mm wide, or tubes ranging from 10-16 mm in diameter. This chapter is organized into the following sections:

9.1 Theory of Operation

The gripper uses a cam-driven actuator to open the gripping fingers. It uses a tension spring to grip an object. This design allows for safe recovery even after a power loss condition.

When the gripper receives a command from the Omni to open, the cam moves each finger 10 mm apart, opening the grip by a total of 20 mm. Once the cam reaches the final position, a sensor trips to indicate that the cam has reached its fully opened state. When the gripper is given the command to close, the cam rotates to the closed position, allowing the fingers to come together under the force of the spring. If an object such as a plate is present, the fingers are not able to close all the way, and the fully-closed sensor is not activated. This indicates to the system that the gripper has an object.

As shown in Figure 9-1, a gripper equipped with the plate finger has a default movement range of 76 - 96 mm. This allows the gripper to pick up and detect plates from 82 - 91 mm in size. Using the cam adjustment screw, the grip range can be adjusted by 5 mm, shifting the gripping range to 81 - 101 mm. This adjustment allows the gripper to grab and detect objects from 87 - 96 mm.

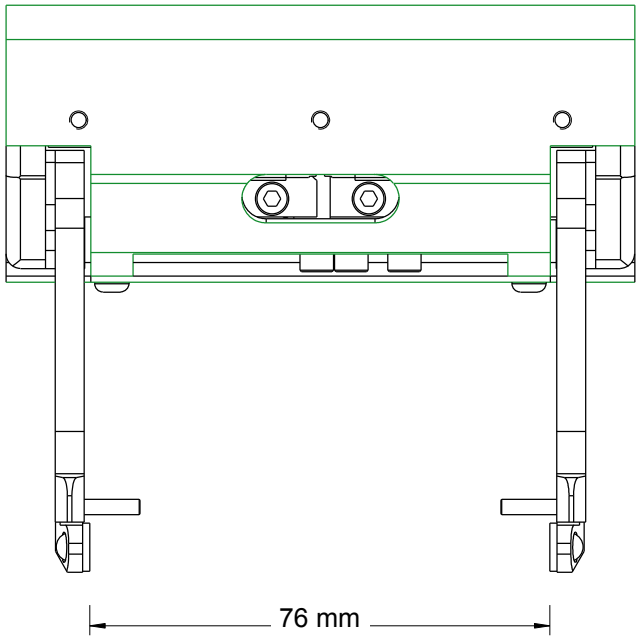


Figure 9-1 *Gripper in closed position; grip size shown in mm*

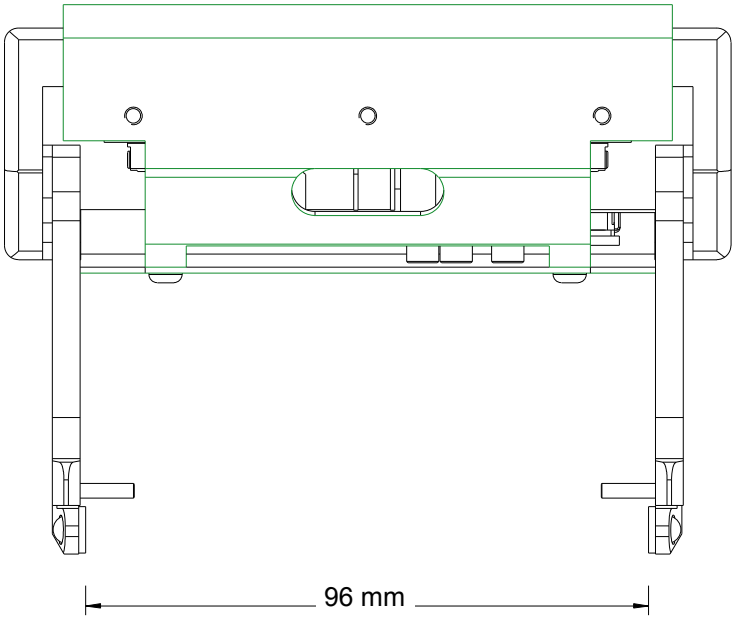


Figure 9-2 *Gripper in open position; grip size shown in mm*

All Cavro® Omni Robots that are equipped with a gripper come pre-configured with a Universal Z axis, and gripper-compatible firmware.

Figure 9-3 shows the gripper with eccentric fingers installed.

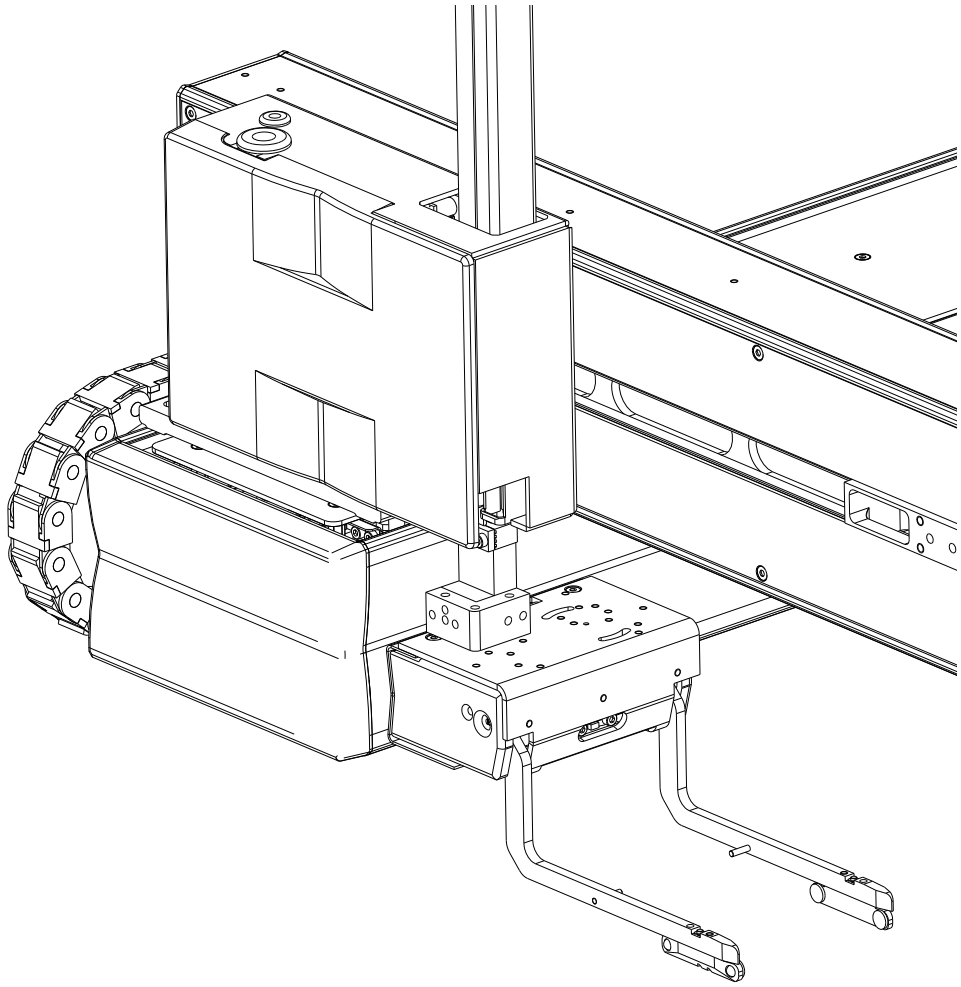


Figure 9-3 *Gripper*

9.2 Gripper Parts List

When you open your Gripper box, you will see the following parts:

- Omni Gripper Module Body
- Mounting bracket and screws
- High-tension spring (low-tension spring comes pre-installed)
- Grounding strap and screws
- Fingers (optional)

9.3 Gripper Fingers

There are three types of gripper fingers that you can order:

- Concentric: Picks up a plate (approximately 85.5 mm wide) directly below the gripper and moves it to another location where there is sufficient vertical clearance to accommodate the gripper. See Figure 9-4.
- Eccentric (L-shaped): Picks up a plate (approximately 85.5 mm wide) in front of the gripper, allowing for pick up and drop off into a space without sufficient vertical clearance to accommodate the gripper, such as a hotel or incubator. See Figure 9-5.
- Tube: Picks up a single test tube (10-16 mm in diameter) directly below the gripper. See Figure 9-6.

All three types of fingers are equipped with rubber pads that create friction between the fingers and their payload. These pads are made of EPDM for chemical compatibility and longevity. Figure 9-4, Figure 9-5, and Figure 9-6 show these three types of fingers.

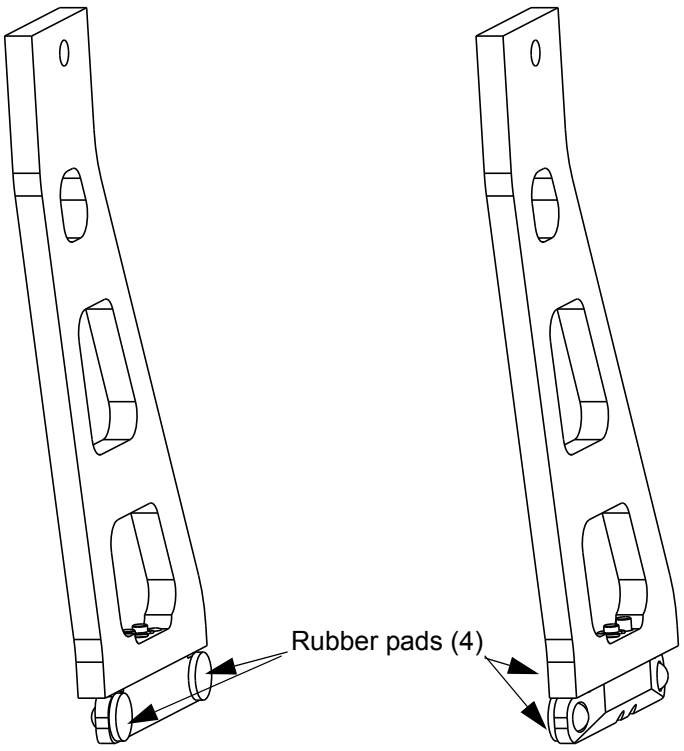


Figure 9-4 *Concentric (straight down) fingers*

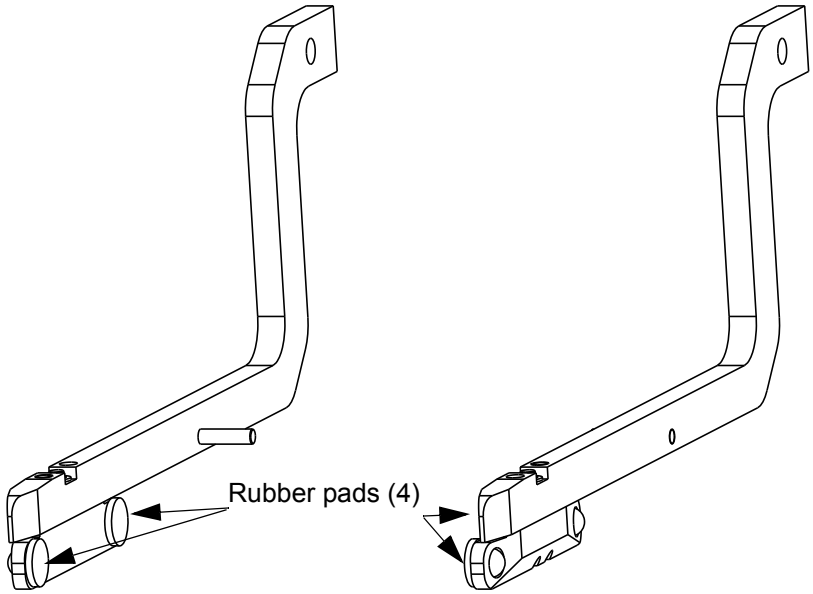


Figure 9-5 *Eccentric (L-shaped) fingers*

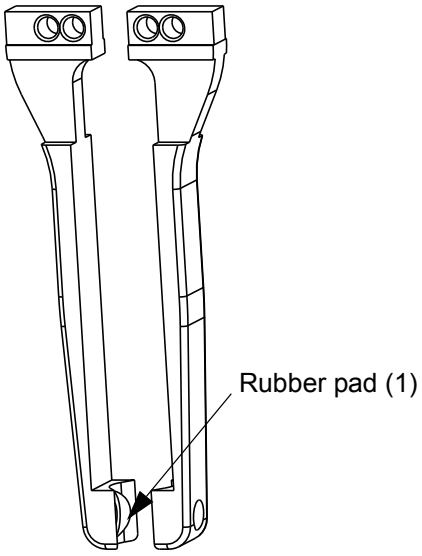


Figure 9-6 *Tube fingers*

9.3.1 Custom Fingers

Some OEM customers may supply their own fingers or order custom fingers from Tecan. For assistance with designing and/or building your custom fingers, contact Tecan. If you have already designed custom fingers and only need information about attaching them to the gripper, see Figure 9-7 and Table 9-2, Gripper specifications, for information about gripper dimensions.

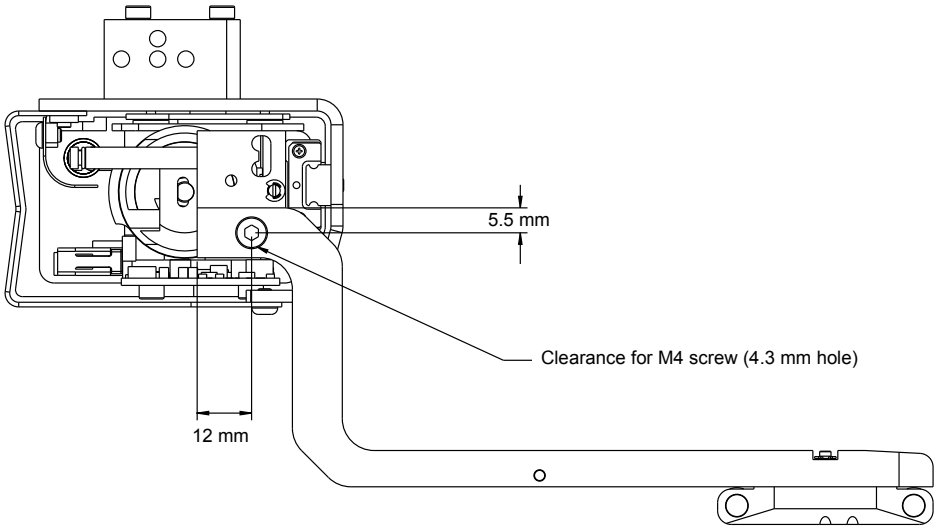


Figure 9-7 *Placement of screw hole for attaching side-mounted fingers*

9.4 Gripper Status Lights

There are three LEDs on left side of gripper that are visible when the left side cover is removed, as shown in Figure 9-8. The Omni software can detect the state of these lights to tell whether the gripper is open, closed, or holding an object. When lit, these three LEDs indicate (Left to Right):

- Cam Open Light (Left): Cam actuator is in the open position, which allows the fingers to open fully. There may or may not be an object in the gripper.
- Cam Closed Light (Middle): Cam actuator is in the closed position, which places the fingers under spring force for closure. There may or may not be an object in the gripper.
- Fingers Closed Light (Right): Fingers are fully closed. There is no object in the gripper.

Table 9-1 *Possible end-state light conditions*

Condition	Gripper closed light	Cam closed light	Cam open light
Gripper commanded to close with object	Off	On	Off
Gripper commanded to close without object	On	On	Off
Gripper commanded to open with object	Off	Off	On
Gripper commanded to open without object	Off	Off	On

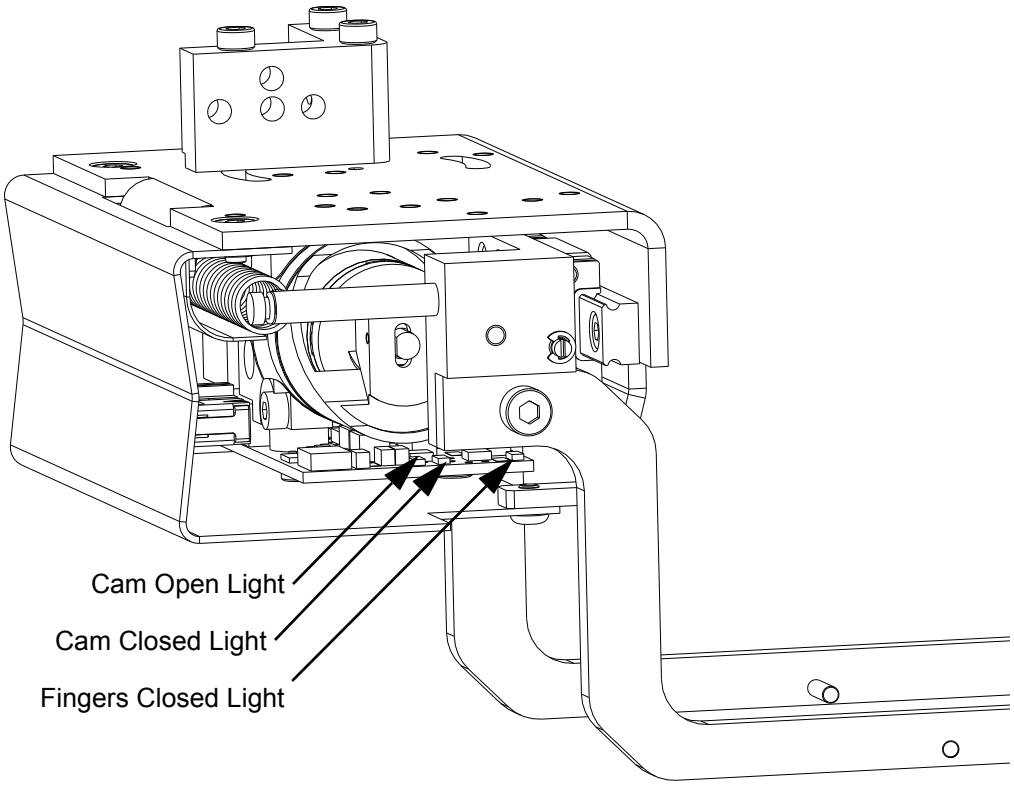


Figure 9-8 *Gripper status lights, visible when side cover is removed*

9.5 Installing the Gripper

There are several steps required to install the gripper on the Universal Z axis. Each of these steps is described in detail in the sections that follow:

- Preparing the Universal Z Axis
- Installing the Cable in the Universal Z Axis
- Mounting Options
- Gripper Mounting Procedure

9.5.1 Preparing the Universal Z Axis

To prepare a Universal Z axis for mounting the gripper, make sure that the jumper settings match those shown in Figure 9-9. If you purchased a new Omni at the same time you purchased the gripper, the jumper settings should already be set correctly. If you are adding a gripper to an existing Omni, you may need to adjust the jumper settings.

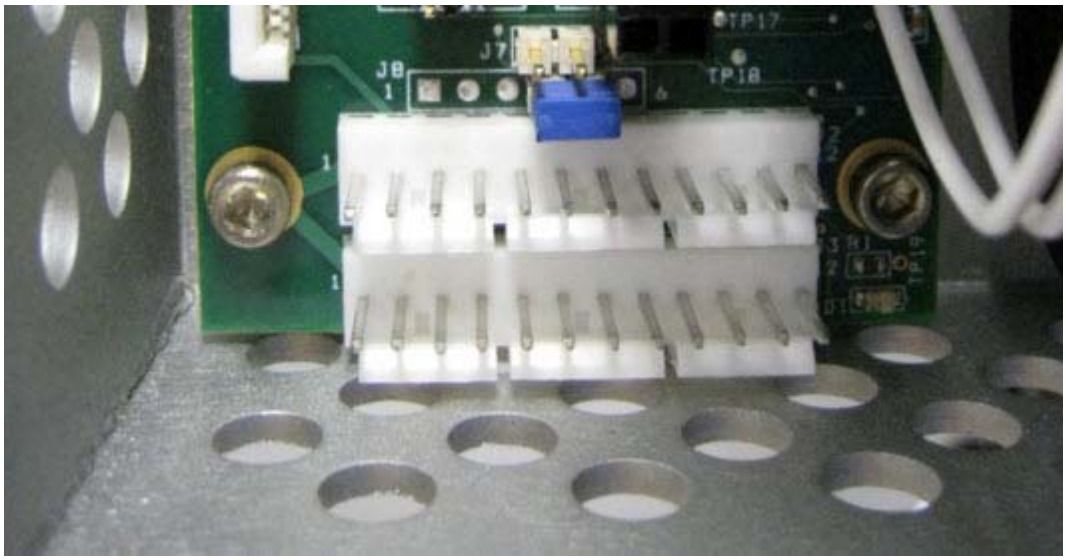


Figure 9-9 *Jumper settings for Universal Z axis*

9.5.2 Installing the Cable in the Universal Z Axis

Tools needed:

- 2 mm hex Allen wrench

Follow these steps to install the cable:

1. Make sure that power to the Omni is turned off and the Z axis cover has been removed.
2. Remove the top piece of the cable tubing guide with one screw.

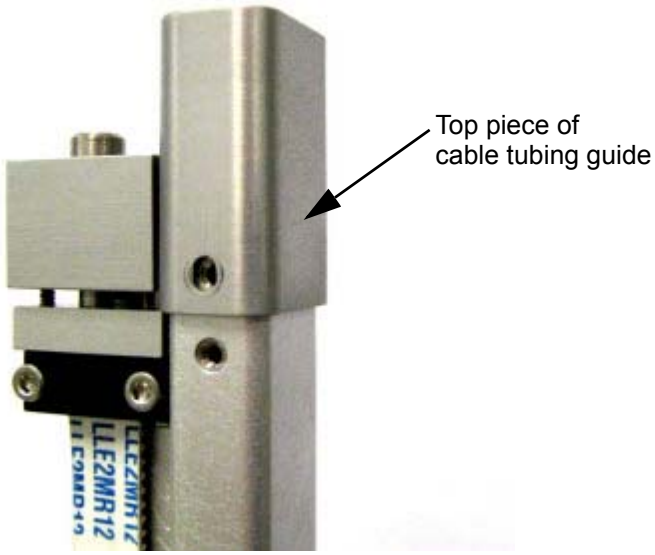


Figure 9-10 *Cable tubing guide with top piece on*

3. Thread the cable through the top piece of the cable tubing guide.

4 Thread the cable through the hole in the Universal Z housing.

Figure 9-11 *Cable threaded through hole in the Universal Z housing*

5 Slide the cable partway into the top of the cable tubing guide at an angle, through the slot in the side. Figure 9-12 shows the slot in the side of the tubing guide and the cable angled through the slot.

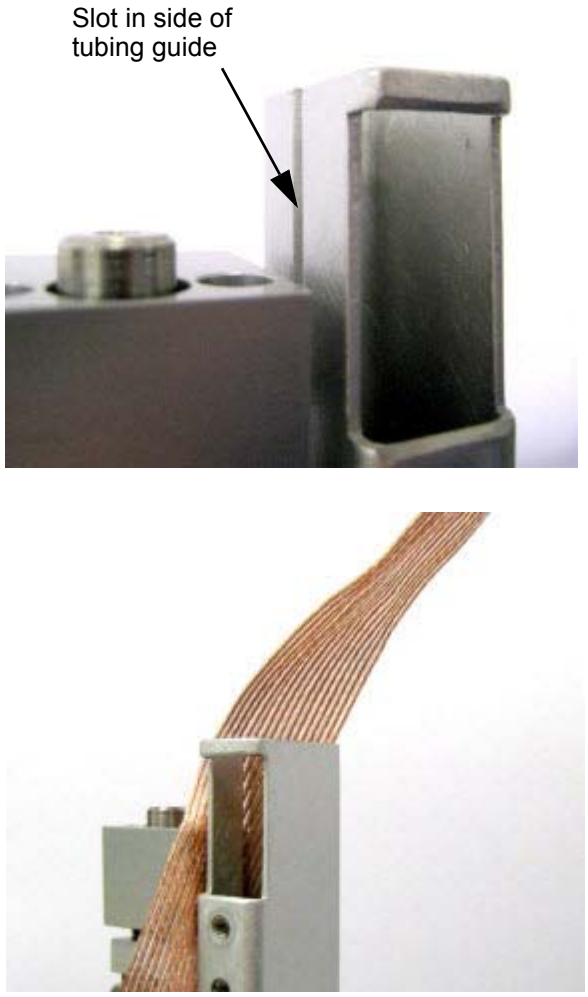


Figure 9-12 *Tubing guide with cable angled through slot*

6 Using a gentle back-and-forth motion, slide the cable all the way into the tubing guide, moving it back and forth until the cable is completely inside the tube.

7 When the cable is entirely inside the tube, pull down gently to draw through the full length of the tube.

8 Line up the screw holes on the tubing guide and top piece.

9 Attach the top with one screw.

10 Place the cable's round black grommet into the notch in the Universal Z housing.

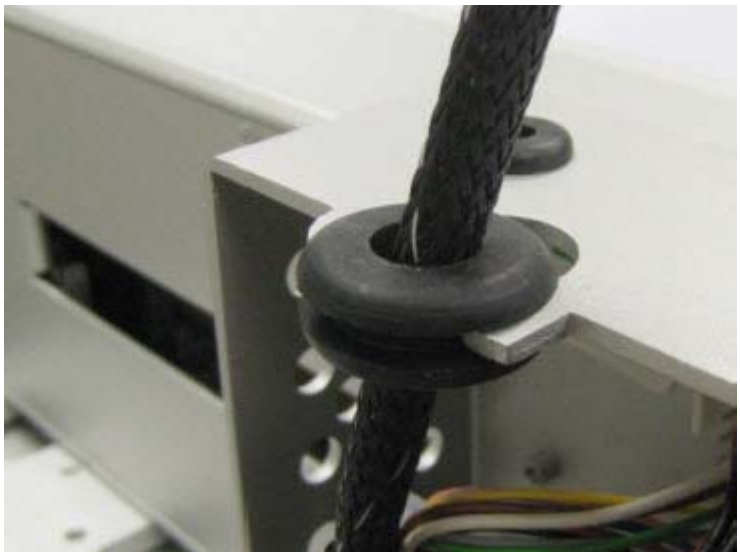


Figure 9-13 *Grommet placed in notch in the Universal Z housing*

11 Plug the cable into the brake PCBA.

WARNING! *Make sure that both ends of the cable connectors are firmly seated. If a connector becomes loose, there is a danger that the Universal Z might drop suddenly and crash into the work surface.*

9.5.3 Mounting Options

The gripper body comes with eight sets of gripper body mounting holes that allow the mounting bracket to be attached in your choice of four directional orientations on either the right or left side, for a total of eight unique mounting positions. Figure 9-14 shows the mounting bracket attached in each of the four directional orientations on the right side of the gripper.

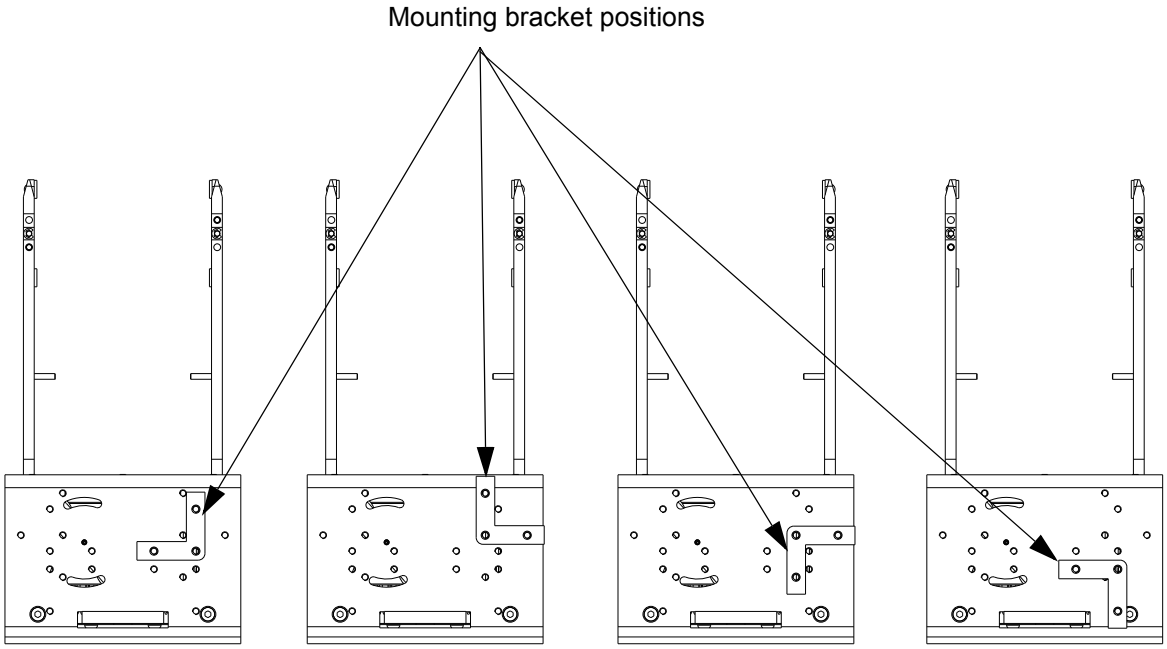


Figure 9-14 *Gripper mounting positions (right side)*

Your choice of which mounting position to use depends on the configuration of your workspace. You may need to try different mounting positions to find the one that works best.

The mounting bracket is attached with three screws, as shown in Figure 9-15.

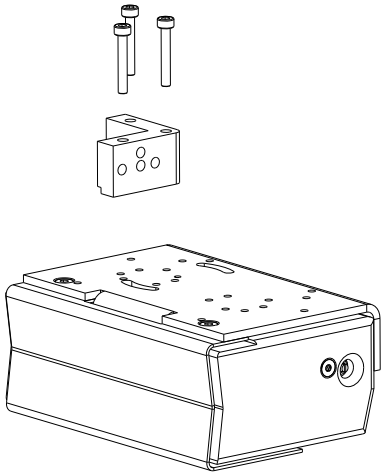


Figure 9-15 *Mounting bracket and screws*

9.5.4 Gripper Mounting Procedure

Complete the mechanical integration, electrical integration, and software setup and integration described in Chapter 3, Chapter 4, and Chapter 5 before mounting the gripper.

Note: *Before mounting the gripper on the Universal Z axis, make sure that you have the correct spring installed for your application. See Section 9.7, "Tension Springs" for more information.*

Tools needed:

- 2 mm, 2.5 mm, 3 mm hex Allen wrenches

Follow these steps to mount the gripper to the Omni Robot:

1. Make sure that power to the Omni is turned off.
2. Remove both side covers by using an Allen wrench to remove one screw from each side.

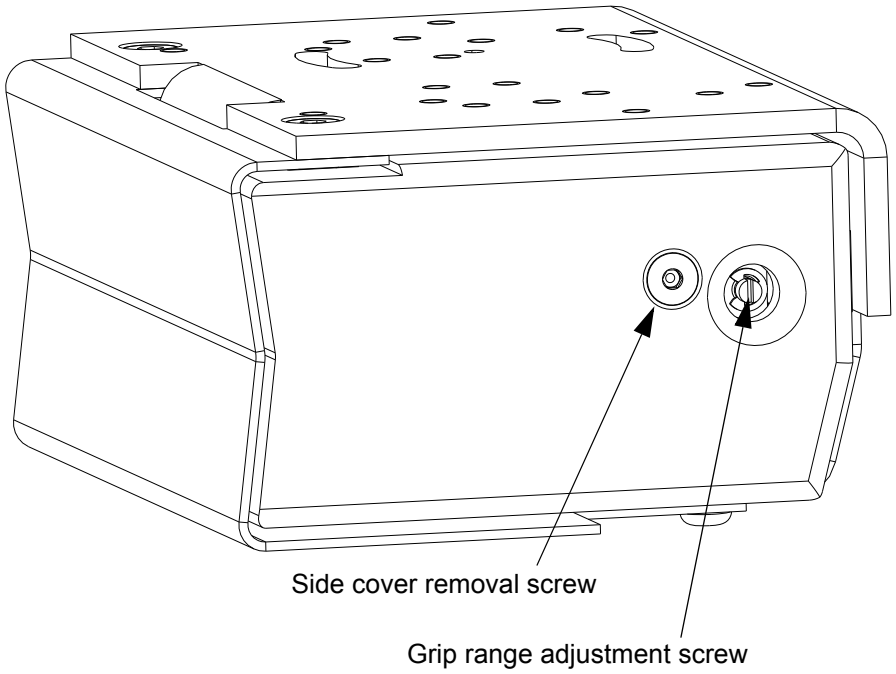


Figure 9-16 *Gripper showing side cover removal screw*

3 Remove rear cover using an Allen wrench.

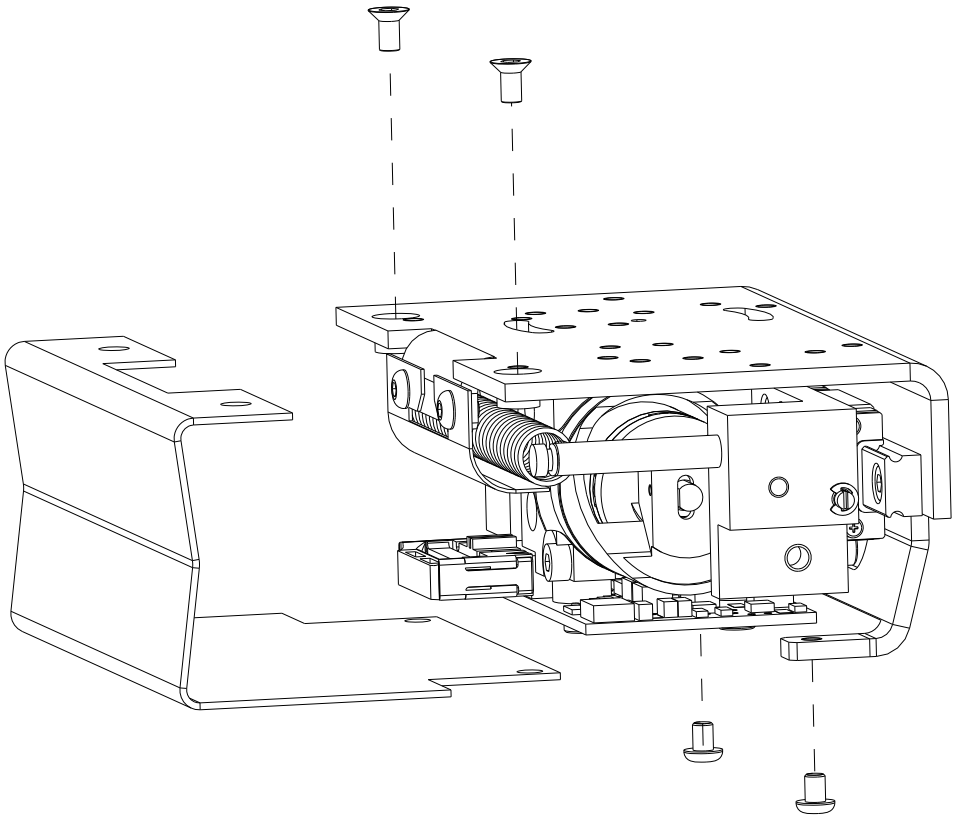


Figure 9-17 *Removing the gripper from the Universal Z axis*

4 Remove the cable holder using an Allen wrench.

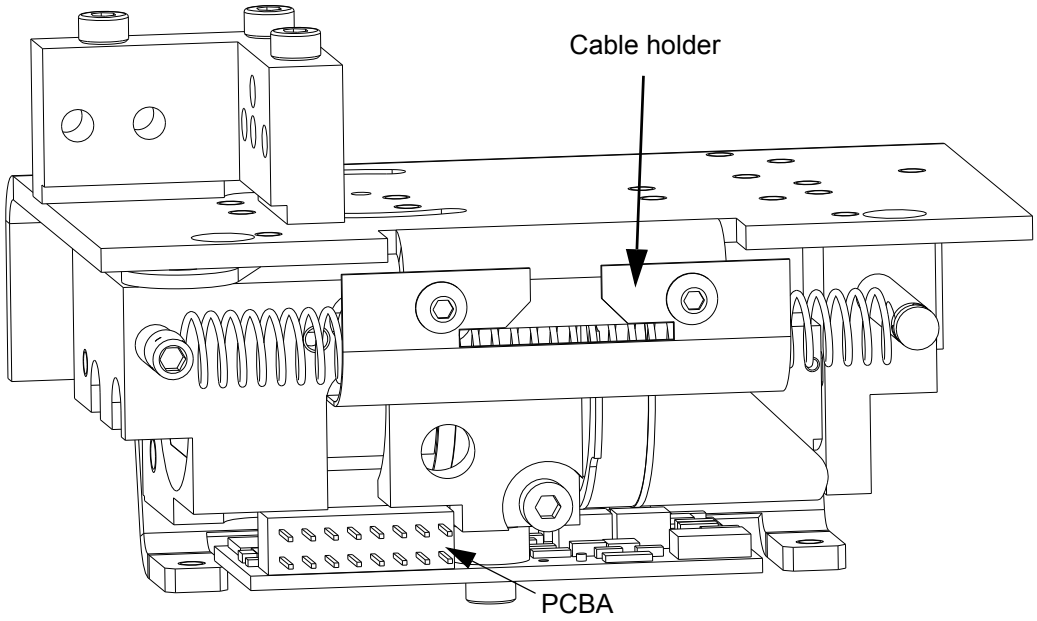


Figure 9-18 *Gripper showing cable holder and PCBA*

5 If necessary, adjust the spring and/or spring post position as described in Section 9.7, ”Tension Springs”.

6 Connect the cable to the gripper PCBA.

WARNING! *Make sure that both ends of the cable connectors are firmly seated. If a connector becomes loose, there is a danger that the Universal Z might drop suddenly and crash into the work surface.*

7 Make an S-shaped fold in the cable. First, bend the beginning of the cable toward the tension spring from the PCBA, as shown in Figure 9-19.

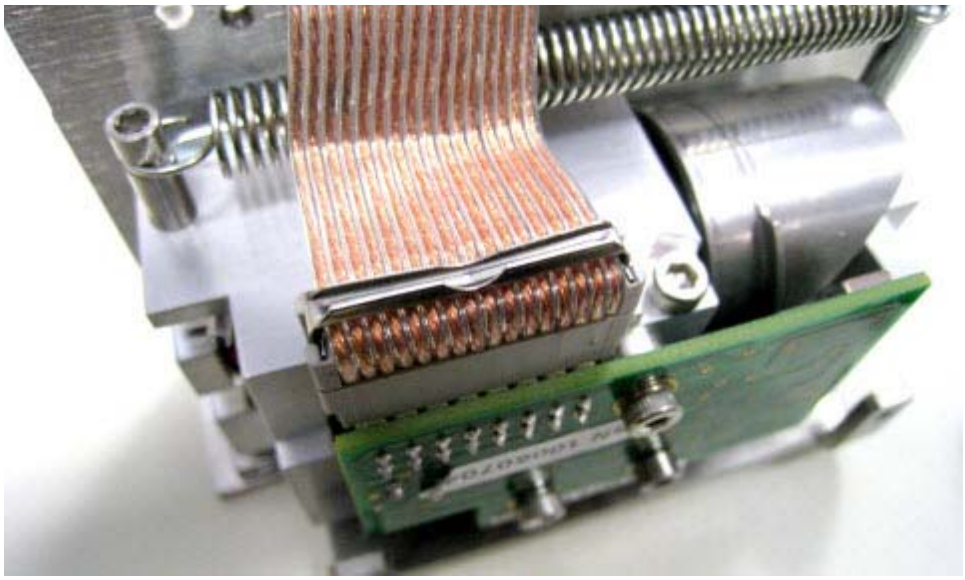


Figure 9-19 *First bend in cable, toward the tension spring*

8 Next, bend the cable back toward the PCBA with a light crease (2 - 3 mm wide across the curve of the crease), approximately 17 mm from the beginning of the cable, as shown in Figure 9-20.

Figure 9-20 *Correct position of first cable crease*

9 Finish the S-shaped fold by bending the cable back toward the tension spring at an angle, so that the cable lies across the cable guide, between the

mounting screws for the cable holder. Make sure that the bent edge of the cable does not extend past the edge of the board, as shown in Figure 9-21.

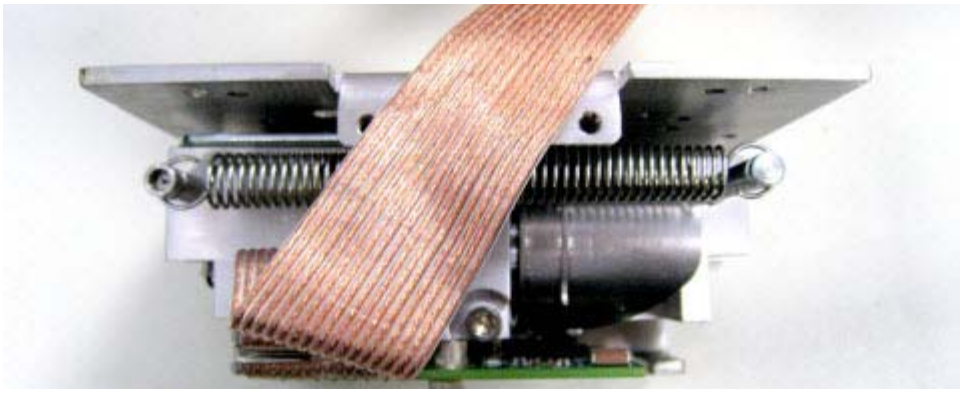


Figure 9-21 *Correct second cable crease position*

10 Mount cable holder using two screws, as shown in Figure 9-22. Hand-tighten the screws to prevent pinching the cable.

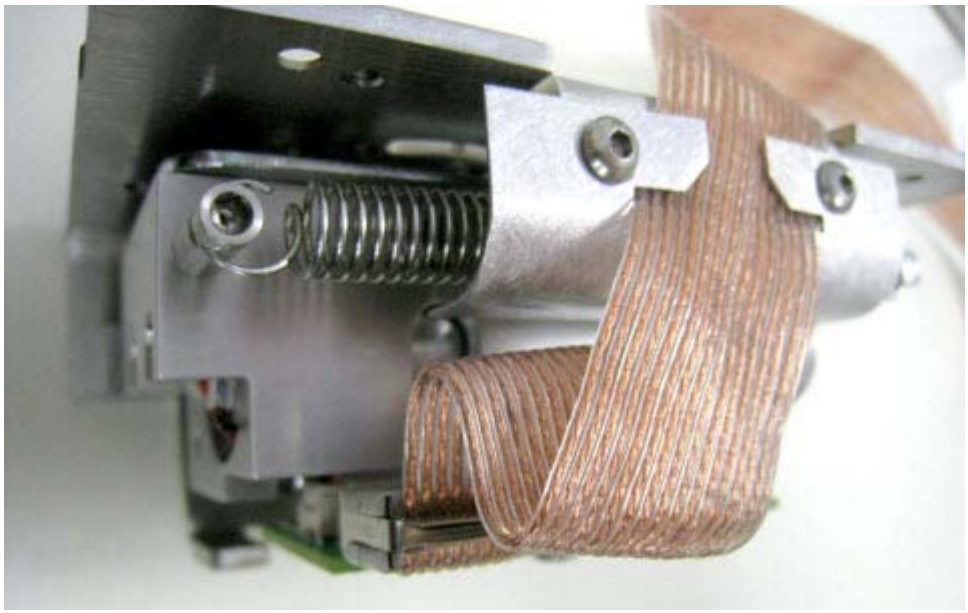


Figure 9-22 *S-fold in gripper cable with cable holder mounted*

11 Replace the rear cover of the gripper using four screws.

12 Make sure that the cable does not touch the bottom of the inside of the gripper cover. It is okay if the cable touches the back of the inside of the gripper cover. Figure 9-23 shows the correct position.

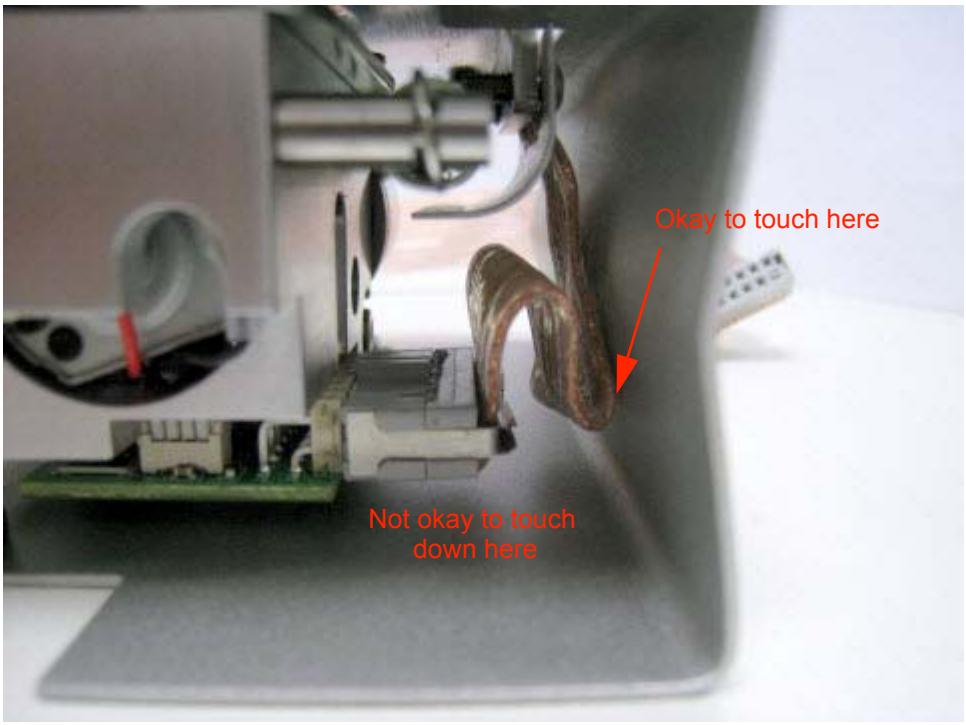


Figure 9-23 *Correct position of cable inside gripper case*

13 Attach gripper fingers with screws using an Allen wrench, being careful not to pinch the red and black wires inside the gripper.

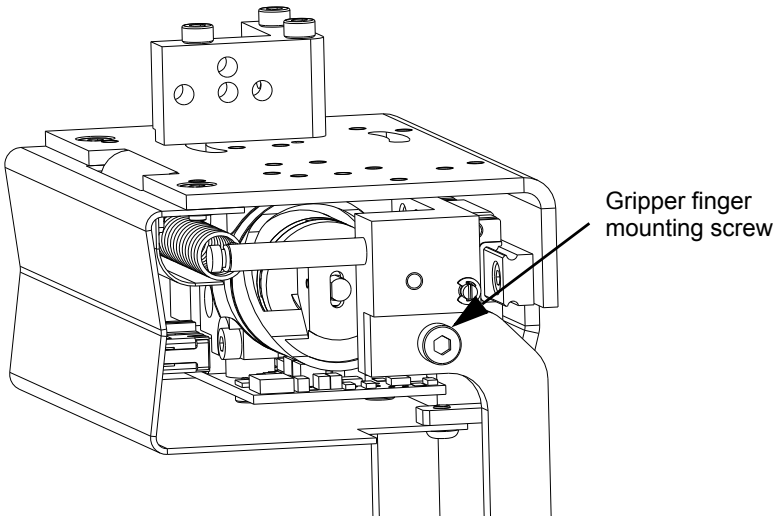


Figure 9-24 *Gripper finger mount for concentric/eccentric fingers*

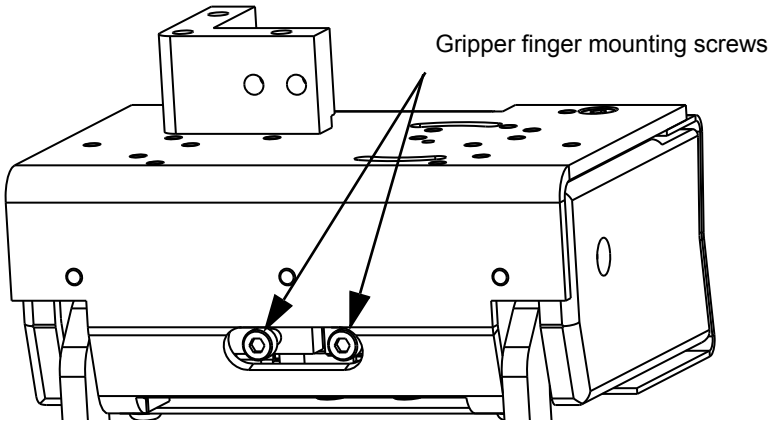


Figure 9-25 *Gripper finger mount for tube fingers*

Note: *For tube fingers, you can use any combination of the mounting holes in the fingers to get the grip range you need. You may need to try all three combinations of mounting holes (both inner holes, both outer holes, one inner and one outer hole) to find the size that works best (see Figure 9-26). You may also need to adjust the grip range as described in Section 9.6.1, "Grip Sensor and Range Fine Tuning".*

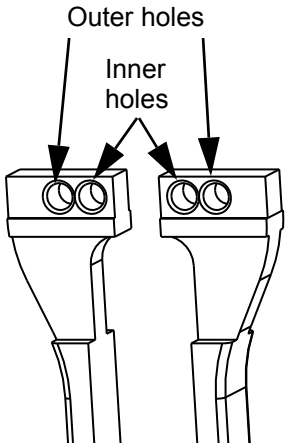


Figure 9-26 *Tube finger mounting holes*

14 Replace the side covers using one spacer and one screw per side.

15 Gently pull the gripper fingers apart to test whether there is sufficient slack in the cable.

16 Attach the mounting bracket in your chosen position, using the 3 longer screws and an Allen wrench. See Figure 9-14 for illustrations of mounting bracket positions to choose from.

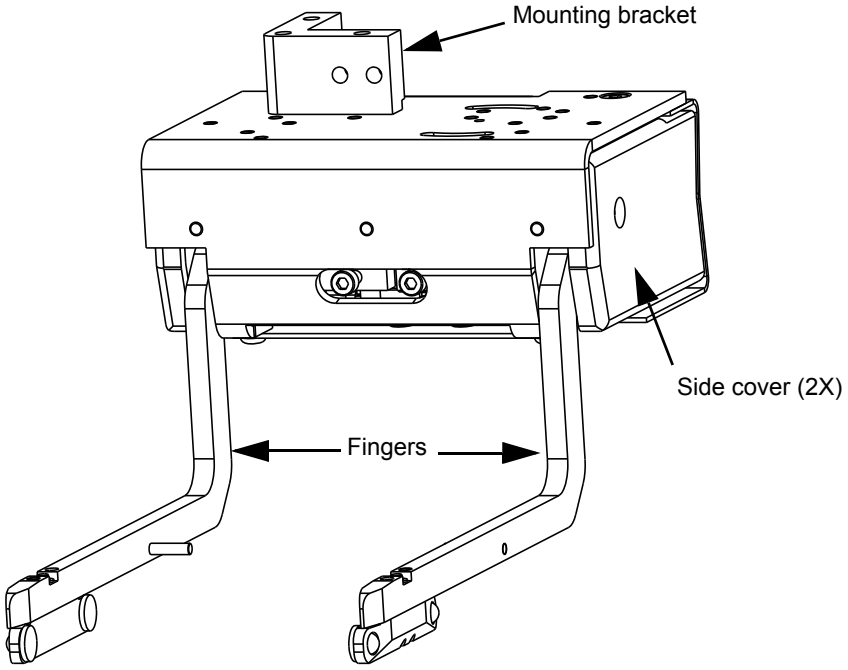


Figure 9-27 *Gripper showing side cover*

17 Place the bracket on the Universal Z axis, using the lip to balance as shown in Figure 9-28, and attach using two of the four shorter screws (one on the side and one on the back) and an Allen wrench.

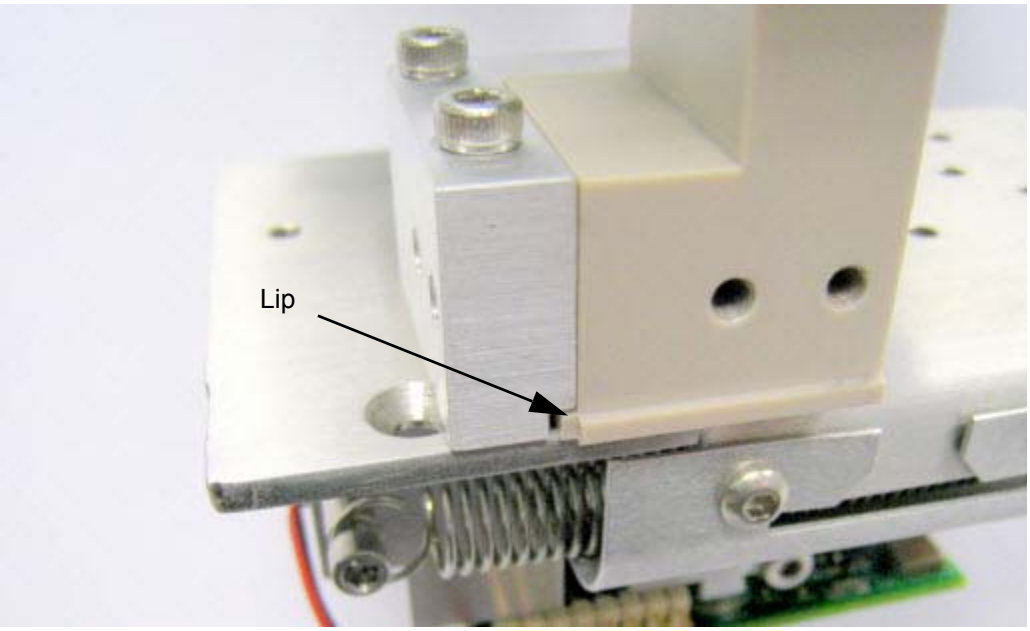


Figure 9-28 *Positioning the gripper on the Universal Z lip*

18 Attach the grounding strap and tighten both screws.

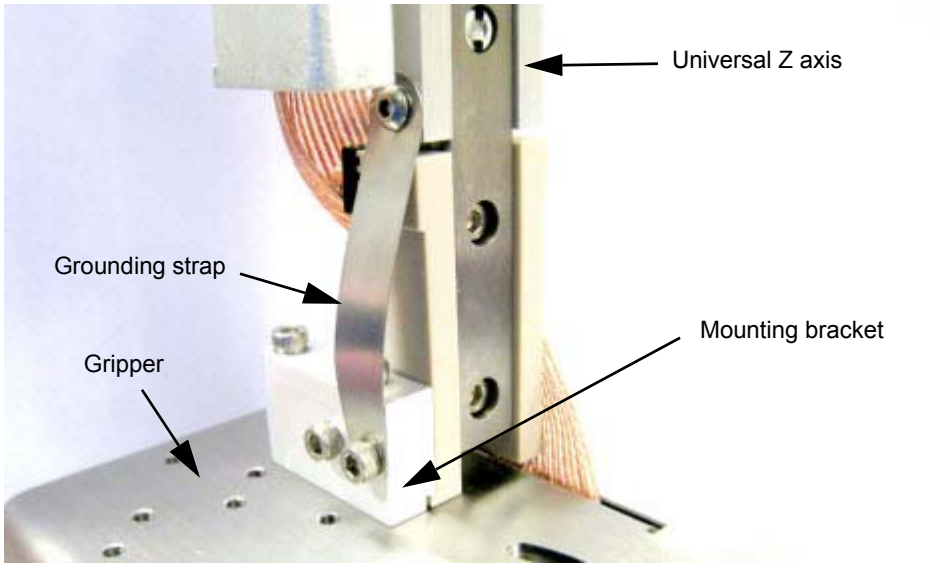


Figure 9-29 *Grounding strap attached with 2 screws*

19 Mount the remaining two Universal Z axis mounting screws.

20 Pull gently on the cable at the top of the Universal Z axis to remove excess slack.

21 Level the gripper fingers as described in Section 9.9, ”Leveling the Gripper Fingers”.

22 Turn on power to the Omni robot.

9.6 Testing the Grip Sensor

Note: *The procedures in this section do not specify commands specific to either Command Processor mode or Embedded mode. For more information about gripper commands, see:*

- *Command Processor: Section 7.10, ”Gripper Commands”*
- *Embedded: Section 6.10, ”Gripper Commands”*

After the gripper is mounted, it is important to test the grip, and adjust it if necessary, before using it. To test and adjust the grip, follow these steps:

1 Remove one side cover as shown in Figure 9-8 to see the gripper status lights.

2 Use the close command to close the gripper without an object between the fingers. Observe that the Finger Closed Light is on (see Figure 9-8). If the Finger Closed Light is not on, proceed to Section 9.6.1, ”Grip Sensor and Range Fine Tuning” step 1.

3 Place a plate or tube in the fingers, and the gripper full closed light should not come on. See Section 9.4, ”Gripper Status Lights”and Figure 9-8. If you are not satisfied with the grip, see Section 9.6.1, ”Grip Sensor and Range Fine Tuning”.

Note: *The gripper comes from the factory adjusted to fit a standard microtiter plate. Plate sizes vary slightly from vendor to vendor, so minor adjustment of the grip may occasionally be required.*

Caution! *Test your labware during mounting and integration to ensure that it operates correctly with your Omni gripper fingers.*

WARNING! *Possible crush risk. Test to make sure that gripping forces on labware are sufficient for secure handling, but not excessive.*

Caution! *Adjust the finger offset for:*

- *adequate clearance of nearby objects, and*
- *adequate clearance and secure grip on labware.*

WARNING! *There is a potential risk of collision during the initialization sequence, and also if there are two arms with grippers. Input a collision avoidance number larger than the size of the largest gripper with its maximum load, based on your system configuration and/or gripper finger type/orientation.*

WARNING! *Do not exceed the maximum allowable payload of .75 lb.*

9.6.1 Grip Sensor and Range Fine Tuning

The gripper is factory preset for a standard microtiter plate (approximately 85.5 mm wide). However, there can be small differences in size between various plate types and manufacturers, so it may be necessary to fine-tune the gripper.

Tools needed:

- 2.5mm flat head screwdriver

To fine-tune the gripper, follow these steps:

1. Use the close command to close the gripper without an object between the fingers. Observe that the Finger Closed Light is on (see Figure 9-8). If the Finger Closed Light does not come on, proceed to step 6.
2. Pull the fingers apart and place a plate between the fingers.
3. Observe the Finger Closed Light. If it is turned off, proceed to step 5.
4. While observing the Finger Closed Light, use a small flat-head screwdriver to turn the grip range adjustment screw (see Figure 9-30) clockwise until the Finger Closed Light (see Figure 9-8) turns off.

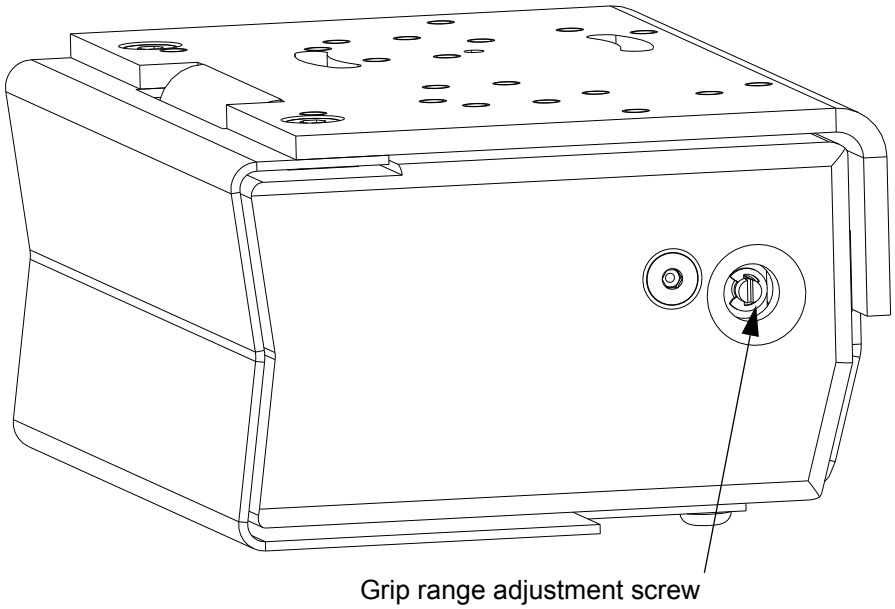


Figure 9-30 *Grip range adjustment screw*

5 When the light turns off, make 3-5 hand turns clockwise with the screwdriver. The distance when the fingers are empty should be at least 1-2 mm less than the width of the target object.

6 Remove any object from within the Gripper. Use the close command to close the Gripper.

7 While observing the Finger Closed Light, use a small flat-head screwdriver to turn the grip range adjustment screw counter-clockwise until the Finger Closed Light is on.

8 Turn the screw 2-3 more turns in the counter-clockwise direction.

9 To check that the adjustments in steps 6-8 did not over adjust, verify steps 1-4 again.

Note: *Each full turn of the screw moves the grip range by approximately 0.5 mm.*

10 Replace the cover on the gripper.

9.7 Tension Springs

The gripper comes from the factory with the standard tension spring pre-adjusted to fit a standard microtiter plate. A high-tension spring also comes with the gripper for use with target objects that require a firmer grip. There are two post positions available for each spring, for a total of four possible gripping forces:

- Longer spring, outer position: approx. 2 lb gripping force
- Longer spring, inner position: approx. 1 lb gripping force
- Shorter spring, outer position: approx. 2.7 lb gripping force
- Shorter spring, inner position: approx. 1.7 lb gripping force

WARNING! *Be sure to use the correct spring for your labware. Excessive gripping force can cause damage to labware. If you need a spring tension that is not listed here, please contact Tecan.*

9.7.1 Changing Tension Springs and Spring Post Position

If gripper has already been mounted, start with step 1. If the gripper has not yet been mounted, start with step 3.

1 Turn power off to the Omni.

2 Remove the gripper from Universal Z axis and gripper mounting block to gain access to the rear cover of the gripper.

3 Remove the rear cover using an Allen wrench as shown in Figure 9-17.

4 Remove the cable holder and disconnect the cable from the PCBA as shown in Figure 9-18.

5 Carefully unhook one end of the tension spring from its spring post, then the other end. Spring posts are shown in Figure 9-31.

6 Using an Allen wrench, adjust the position of the removable spring post if needed to fit the spring. Spring post positions are shown in Figure 9-31.

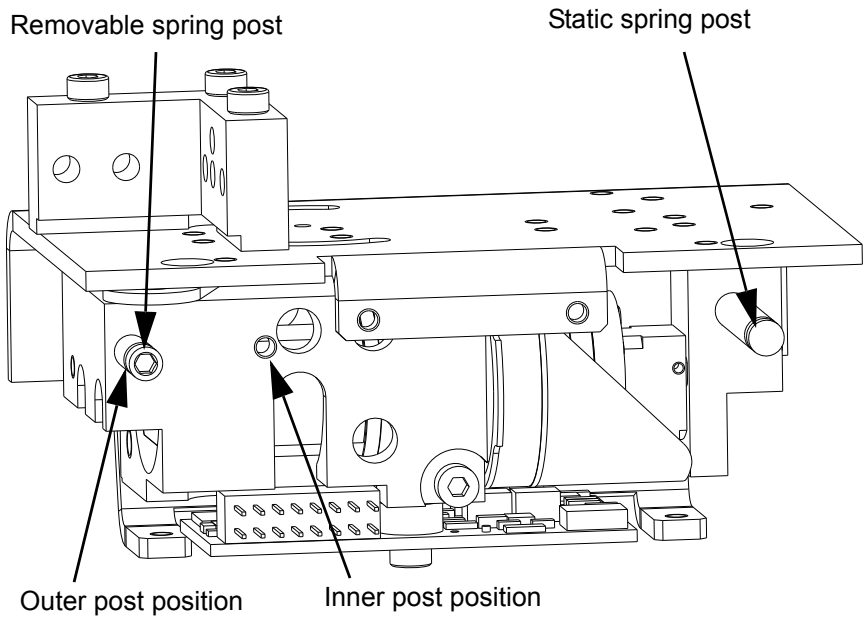


Figure 9-31 *Spring post positions*

7 Hook the new tension spring in place; first one end, then the other.

8 Replace the cable, making sure to leave sufficient slack for the gripper fingers to move back and forth. For detailed information, see Section 9.5.4, ”Gripper Mounting Procedure”.

9 Pull the gripper fingers apart to test whether there is sufficient slack in the cable.

10 Replace the rear cover.

11 If you removed the mounting bracket, replace the mounting bracket and reattach it to the Universal Z. For detailed information, see Section 9.5.4, ”Gripper Mounting Procedure”.

12 Turn on power.

9.8 Changing Fingers

The following steps describe the procedure for changing gripper fingers after the gripper has been mounted and installed.

1. Turn off power to the Omni.
2. Remove both side covers by using an Allen wrench to remove one screw from each side. See Figure 9-16.

Note: *You do not need to remove the side covers to change tube fingers unless you are going to replace them with concentric or eccentric fingers.*

3. Unscrew each finger with an Allen wrench.
4. Install and level the new fingers as described in Section 9.9, ”Leveling the Gripper Fingers”.
5. Replace the side covers on the gripper.
6. Turn on power.

9.9 Leveling the Gripper Fingers

Fingers may not be perfectly level after installation or may go out of level in the event of a crash. To adjust the level of the fingers, follow these steps:

1. Make sure that power to the Omni is turned off.
2. Remove both side covers by using an Allen wrench to remove one screw from each side.
3. Loosen the screws that hold the gripper fingers.
4. Manually move the gripper down until the fingers touch the work surface.
5. Use the work surface as a guide to help you position the fingers parallel to the work surface.
6. Tighten the screws that hold the gripper fingers.
7. Replace the side covers on the gripper.
8. Turn on power.

9.10 How to Replace the Gripper Cable

Follow these steps to replace the gripper cable:

1. Make sure that power to the Omni is turned off and the Z axis cover has been removed.
2. Remove the gripper from Universal Z axis and gripper mounting block to gain access to the rear cover of the gripper.
3. Remove the rear cover using an Allen wrench as shown in Figure 9-17.
4. Remove the cable holder and disconnect the cable from the PCBA as shown in Figure 9-18.
5. Remove and replace the cable from the Universal Z as described in Section 9.5.2, ”Installing the Cable in the Universal Z Axis”.
6. Reconnect the cable to the gripper PCBA and reinstall the gripper as described in Section 9.5.4, ”Gripper Mounting Procedure”.
7. Reinstall the Universal Z axis cover.
8. Turn on power.

9.11 Maintenance

Under normal use, the only maintenance that gripper should require is occasional cleaning.

9.11.1 How to Clean the Gripper

The gripper can be cleaned in the same way as other parts of the Omni. For information about cleaning the Omni, see:

- Section 2.5, ”Component Decontamination” on page 2-4
- Appendix A, “Maintenance Schedule”

9.11.2 Spare Parts

If you need to replace a part on your gripper, see Table B-11, Gripper for a list of replacement parts.

9.12 Specifications

Table 9-2 lists specifications for the gripper. Gripper measurements are illustrated in Figure 9-32, Figure 9-33, Figure 9-34, Figure 9-35, and Figure 9-36.

Table 9-2 *Gripper specifications*

Spec	Value
Maximum payload weight	◆ Approx. .75 lb
Gripper	◆ Width (side to side): 104 mm ◆ Depth (front to back): 74.5 mm ◆ Height (top to bottom): 47.5 mm
Tube fingers (for gripping tubes)	◆ Width from outside of fingers in closed position: 21 mm ◆ Gripping surface length: 10 mm ◆ Minimum grippable tube size: 10 mm diameter ◆ Maximum grippable tube size: 16 mm diameter ◆ Distance from front of gripper to center of grip: 9.9 mm ◆ Height (with gripper): 119.4 mm ◆ Height (without gripper): 71.9 mm
Concentric fingers (straight down)	◆ Height (with gripper): 146.3 mm ◆ Height (without gripper): 98.8 mm ◆ Maximum width (with adjustment): 96 mm ◆ Minimum width (default): 82 mm ◆ Gripping surface length: 47.5 mm ◆ Longer tension spring, outer position: approx. 2 lb gripping force ◆ Longer tension spring, inner position: approx. 1 lb gripping force ◆ Shorter tension spring, outer position: approx. 2.7 lb gripping force ◆ Shorter tension spring, inner position: approx. 1.7 lb gripping force ◆ Distance from front of gripping surface to front of gripper: 22.4 mm

Table 9-2 *Gripper specifications (continued)*

Spec	Value
Eccentric fingers (L-shaped)	◆ Finger length: 130 mm ◆ Finger height (with gripper): 94 mm ◆ Finger height (without gripper): 46.5 mm ◆ Finger width: 5.7 mm ◆ Gripping surface height (without arm): 7.8 mm ◆ Gripping surface height (with arm): 16.3 mm ◆ Maximum width (with adjustment): 96 mm ◆ Minimum width (default): 82 mm ◆ Gripping surface length: 8.3 mm ◆ Longer tension spring, outer position: approx. 2 lb gripping force ◆ Longer tension spring, inner position: approx. 1 lb gripping force ◆ Shorter tension spring, outer position: approx. 2.7 lb gripping force ◆ Shorter tension spring, inner position: approx. 1.7 lb gripping force
Custom finger attachment dimensions	◆ Distance from end of side-mounted finger to middle of mounting screw: 12 mm ◆ Distance from top of side-mounted finger to middle of mounting screw: 5.5 mm ◆ Size of mounting screw: M4 (4.3 mm)

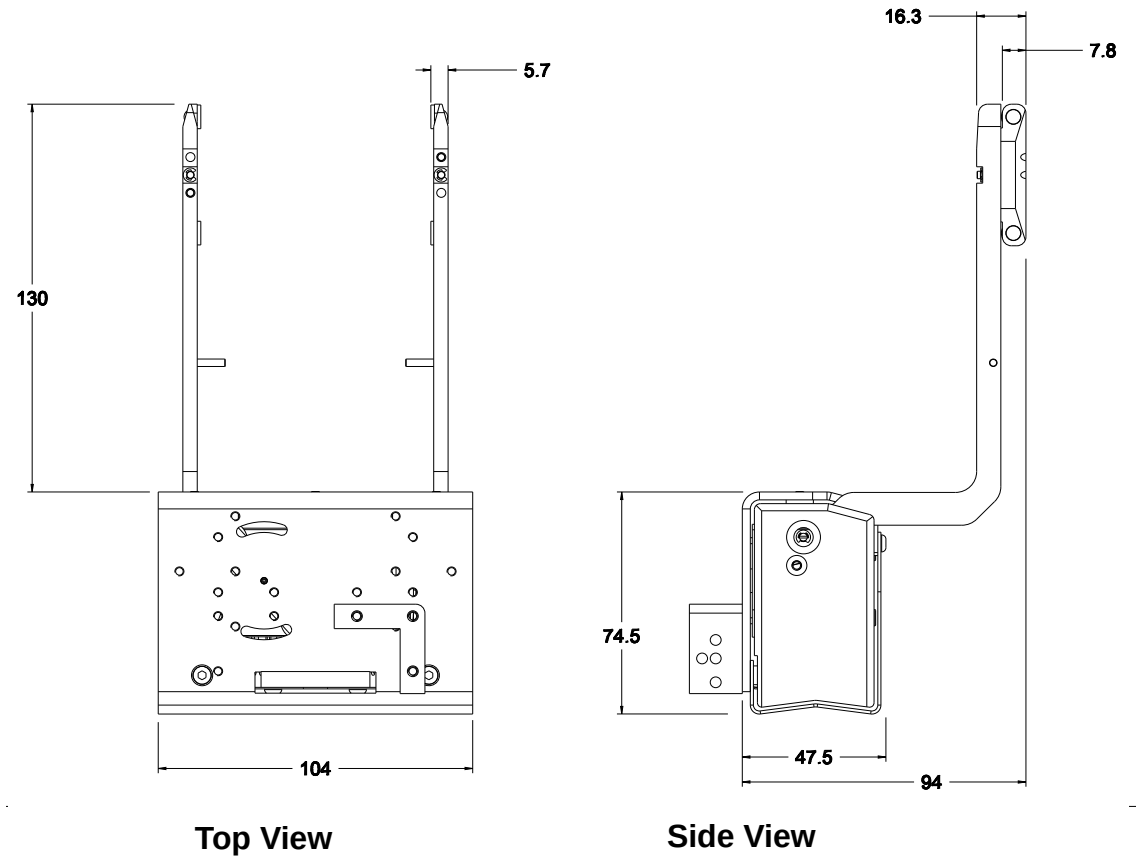


Figure 9-32 *Eccentric gripper finger dimensions in mm*

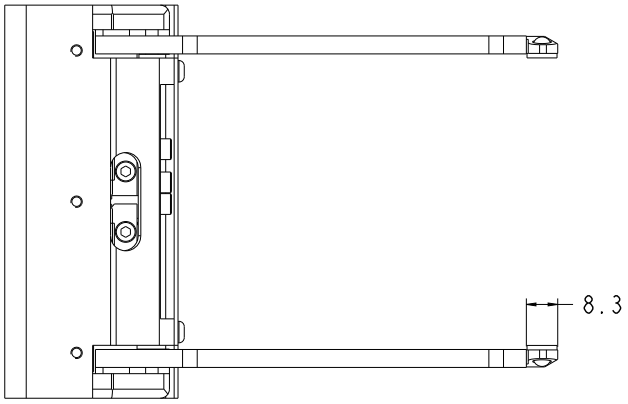


Figure 9-33 *Eccentric gripper finger dimensions in mm (bottom view)*

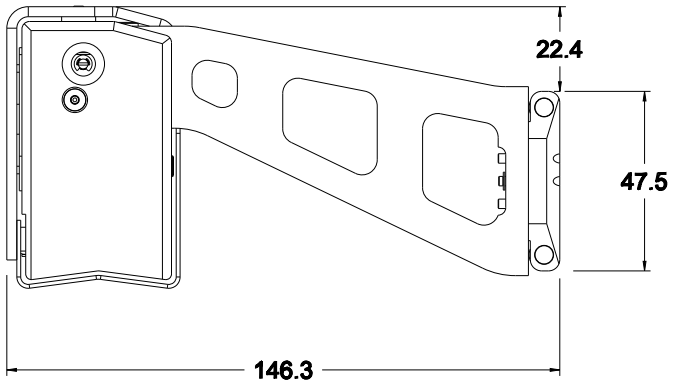


Figure 9-34 *Concentric gripper finger dimensions in mm (side view)*

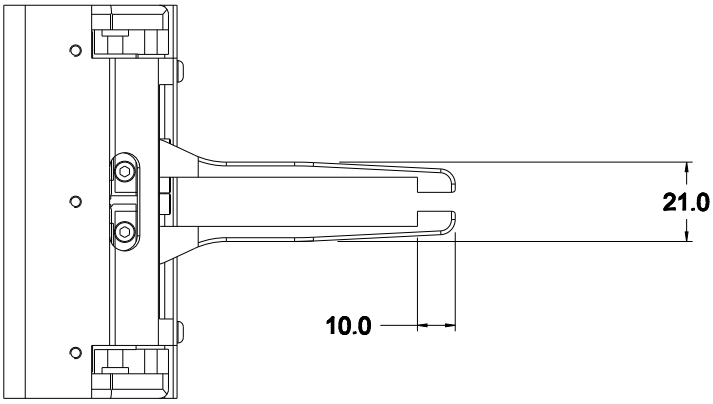


Figure 9-35 *Tube gripper finger dimensions in mm (bottom view)*

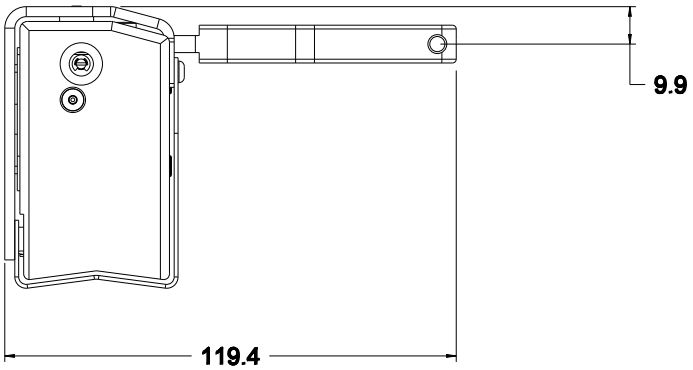


Figure 9-36 *Tube gripper finger dimensions in mm (side view)*

9.13 Optimization

Any device driven by a motor may encounter situations in which motion is accompanied by vibration. This situation is influenced by the mass of the payload, the position of the payload on the arm, and the speeds and accelerations in use. If unacceptable vibration occurs, these factors can be adjusted to optimize the motions required.

Any adjustment should be done with care. Adjustments may improve motion, have no effect, or make vibration worse. Adjustments should be made one at a time to assess their effect before adding additional adjustments:

- First, check that payload is not excessive.
- Second, if possible, change the speed. If the vibration is on only one axis, adjust that axis speed until vibration ceases.
- Third, If possible, change the mounting position of the gripper to a different position. This will change the center of mass, which can affect vibration.

For example, if you are experiencing vibration with these values:

- Payload: 0.75 lb
- X axis speed: 300 mm/second
- Y axis speed: 300 mm/second
- Z axis speed: 200 mm/second
- Gripper mounted facing away from the X axis

Try increasing the speed along the X and Y axes to 600 mm/second. If that does not help, try a different size payload, or changing the mounting position to face away from the Y axis. Keep experimenting until the Omni moves smoothly.

Note: *This is an example only, and is not intended to imply that the example circumstance is known to be a problem. Also, the suggested adjustment is not intended to be a specific recommendation for the example circumstance.*

9.14 Choosing Labware for the Gripper

Use labware that meets the specifications described in Table 9-2, and test it with a representative sample payload to make sure that the Omni grips and moves it properly.

9.15 Troubleshooting

- If none of the gripper status lights is on, the gripper is not receiving power. Make sure all cables are connected properly.
- If all three of the gripper status lights are on, there is a problem with the gripper. If this happens, call Tecan for assistance.
- If the gripper grabs something that is smaller than the size range it has been programmed to expect, it will report an error even though it is gripping something.

A Maintenance Schedule

The following tables provide a schedule for the maintenance and cleaning of the Omni Robot and its related components. For specific instructions on performing the maintenance tasks, see Appendix B, “Maintenance Tasks”.

Note: *Cleaning or decontaminating the Omni Robot with special cleaning agents is omitted from the maintenance schedule. Determining how often the module must be cleaned or decontaminated, and with what agents, is the responsibility of the OEM user.*

Table A-1 *Daily maintenance tasks*

Component	Maintenance Task	Reference
Liquid system	Check for leakage	Section B.1, ”Liquid System Maintenance” on page B-1
	Tighten the tubing connections	Diluter or syringe documentation
	Flush	Diluter or syringe documentation
	Check for air bubbles	
Diluters and Syringes (optional)	Tighten syringes and plunger lock screws	Diluter or syringe documentation
Probes	Clean	Section B.2.2, ”Cleaning a Standard Probe” on page B-4
	Check for damage	Section B.2.3, ”Checking a Probe for Damage” on page B-4
Disposable tip (DiTi) cones	Clean	Section B.3.2, ”Cleaning a DiTi Pickup Mechanism and Performing Daily Maintenance” on page B-9
	Check for deposits	
	Tighten	

Table A-2 *Quarterly maintenance tasks*

Component	Maintenance Task	Reference
Liquid system	Replace the inner tubing	Section B.2, ”Probes” on page B-2

Table A-3 *Semi-yearly tasks (every six months)*

Component	Maintenance Task	Reference
Disposable tip (DiTi) cones	Replace critical components	Section B.3.3, ”Removing the DiTi Pickup Mechanism” on page B-9

Table A-4 *Yearly tasks*

Component	Maintenance Task	Reference
X axis, Y axis, and Universal Z axis	System check: • Check belts for wear • Conduct performance test per OEM recommendation	Note that the Standard Z axis and Dual Z axis require no scheduled maintenance.
Module exterior and interior	Wipe down extrusions	Use clean room wipes and remove any accumulated dust or debris. Note that this task may be required more often if the module is operated without covers or in an open environment. The final determination of the maintenance schedule is the responsibility of the OEM user. Clean the exterior painted surfaces gently to avoid damaging the finish. Section 2.5, ”Component Decontamination” on page 2-4
Brake	Lubrication	Section B.6.5, ”Lubrication Instructions for the Universal Z Axis Brake Plunger Mechanism” on page B-18

B Maintenance Tasks

This appendix describes the maintenance tasks needed for the portions of the Omni Robot that could require servicing over the lifetime of the module.

Except where specifically noted otherwise, these maintenance tasks can be performed by the end user; the OEM user is responsible for passing the maintenance instructions on to the end user.

Note: *After performing maintenance on the Omni Robot, perform the recommended OEM system tests, including a tip pickup test, a cLLD test, and a brake test, if applicable. This ensures that the Omni Robot is operating to the desired specifications.*

B.1 Liquid System Maintenance

WARNING! *Liquid hazards*

- *Potential hazards to personnel may exist from the liquids being handled by the Cavro® Omni Robot.*
- *Infectious clinical samples, toxic or corrosive chemicals, or radioactive chemicals may be present.*
- *Appropriate hand, eye, and clothing protection must be worn while operating the module.*

WARNING! *Always disconnect the Cavro® Omni Robot power source before performing any maintenance tasks. Never clean the Cavro® Omni Robot when it is plugged in.*

Caution! *Contamination inside the tip adapter can affect the liquid level detection function.*

- *Always drain probes and liquid tubing before executing maintenance tasks on the liquid system.*
- *Always clean and dry the liquid tubing and the probe on the outside before installing or removing.*

B.1.1 Finding and Repairing Leaks

Caution! *A leaking liquid system causes pipetting inaccuracy and possible carry over.*

- *Never operate the Omni Robot module if the liquid system is leaky.*
- *Dripping can be caused by air bubbles in the lines or loose fittings.*
- *System liquid should be kept at room temperature and degassed.*

A sign that the liquid system is leaking is when liquid droplets hang from the fixed probe or DiTi cone before the Omni Robot is switched on. Leakage can be caused by aggressive liquids. When using aggressive liquids, take into account the chemical resistance of the tubing material. FEP tubing is recommended.

Perform the following tasks if the liquid system is leaking:

1. Tighten the lock nut or DiTi cone (as described in Section B.2, ”Probes” on page B-2 or Section B.3, ”Disposable Tip (DiTi) Pickup Mechanism and Tips” on page B-7, respectively).
2. If a diluter is connected to the module, tighten any connections from the diluter to the Omni Robot, including tubing connections.
3. Flush the liquid system as described in the documentation for your diluter. During flushing, carefully observe the tubing. If necessary, gently move the tubing to make sure all air bubbles are removed.
4. Observe the probe or DiTi cone for one minute. If no droplets form, the leakage is repaired.
5. If the system still leaks, call a Tecan authorized field service engineer.

B.2 Probes

Caution! *Electric discharge can damage the liquid detector.*

- *Discharge yourself electrically through contact with a grounded object before touching the probe.*

WARNING! *Pipetting tubing and probes can be contaminated.*

- *Decontaminate the module and assure appropriate safety measures.*

WARNING! *Pipetting probes can cause injuries.*

- *Avoid contact with the probe and contact with aerosols when accessing the worktable, by wearing adequate protective clothing.*

Caution! *Handle probes with extreme caution at all times.*

- *If a probe is to be replaced, do not remove the lock nut from the probe.*
- *Always hold the probe at its upper end, avoiding contact with the coated surface whenever possible.*

In general, all Z axis probes are maintained in the same way.

The Universal Z axis 8-channel pipetting head comes with probes pre-installed.

B.2.1 Installing a Standard Z Axis or Dual Z Axis Probe

Figure B-1 shows the parts involved in installing a standard probe.

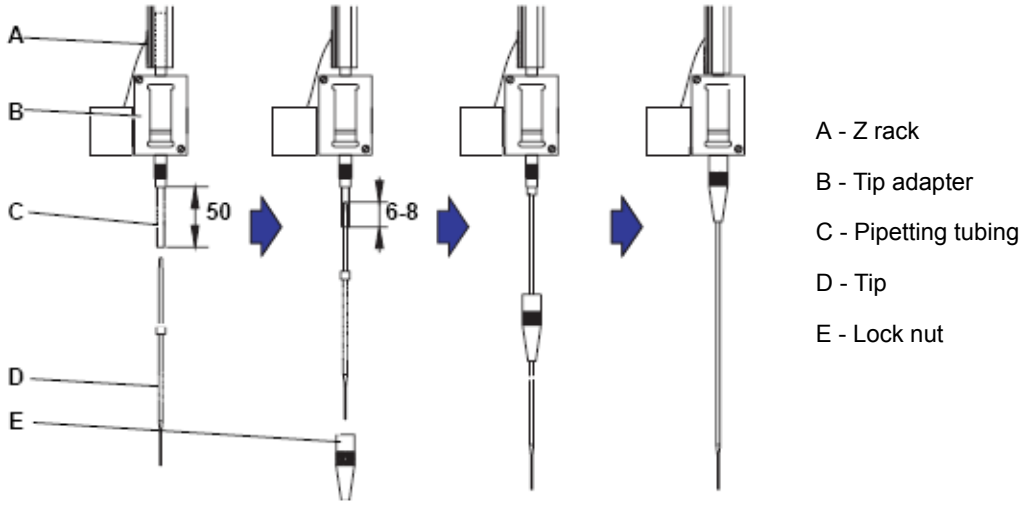


Figure B-1 *Standard probe installation*

To attach a fixed probe (adjustable or nonadjustable):

1. Drain the system fluid from the tubing and the probe.
2. Manually move the Z rack up until it lightly touches the stop.
3. Move the Z rack to the front of the workspace.
4. Carefully pull the pipetting tubing approximately 50 mm out of the tip adapter.
5. Wrap the tube near its end with a small piece of emery cloth to have a better grip on the tube.

Note: *Do not use emery cloth on probes, as it would abrade the delicate probe coating. Use a dry piece of emery cloth for an improved grip on tubing only; wet sandpaper could leave tiny particles and thus clog inside the tubing and/or probe.*

6. While holding the tube end wrapped in emery cloth, insert the blank, conical end of the probe 6 to 8 mm into the tubing along its axis.

7 For an adjustable probe, loosen the four adjustment screws sufficiently to obtain enough room for the positioning of the sealing washers. Make sure the lock nut contains no washers.

8 For an adjustable probe, put two sealing washers around the probe:
   – First, a white Teflon washer.
   – Second, a black plastic sealing washer, which will sit on the lock nut inner bottom.

9 Position the lock nut around the probe end, and pull it up over the probe and (for an adjustable probe) over the two washers, avoiding contact with the delicate end of the probe and its coating.

10 Insert the probe and tubing into the adapter, and tighten the lock nut, screwing it to the probe holder. In the case of an adjustable tip, when the lock nut is fairly tight, stop tightening when its adjustment screws are at a 45 degree angle to the worktable's X/Y coordinate system.

11 In the case of an adjustable probe, tighten the four adjustment screws.

12 Clean the probe using a lint-free tissue soaked in ethanol (70%) or isopropanol. Make sure not to damage the tip coating.

13 In the case of an adjustable probe, use the adjustment procedure provided to complete the installation.

B.2.2 Cleaning a Standard Probe

Before switching on the module, use a lint-free tissue soaked in ethanol (70%) or isopropanol to clean the fixed tip. Make sure not to damage the tip coating.

B.2.3 Checking a Probe for Damage

Caution! *Bent or damaged probes can cause pipetting inaccuracy and liquid detection errors.*

- *Do not operate the Omni Robot with damaged or bent probes.*

Visually inspect the tip coating before using the Omni Robot. Use a mirror for proper inspection of the tip outlet. Make sure that the tip is not bent. If the tip coating is damaged or the tip is bent, the tip must be replaced as described in Section B.2.4, "Replacing a Standard Z Axis or Dual Z Axis Probe".

B.2.4 Replacing a Standard Z Axis or Dual Z Axis Probe

To replace a fixed probe (adjustable or nonadjustable):

1 Drain the system fluid from the tubing and the probe.

2 Manually move the Z rack up until it lightly touches the stop.

3 Move the Z rack to the front of the workspace.

4 If an adjustable tip is installed, loosen the four tip adjusting screws.

5 Unscrew the lock nut, holding the tip immediately below the lock nut with your other hand.

6 When the lock nut is freed, let go of the tip and slide off the lock nut along the tip axis, avoiding contact between the lock nut and the tip coating.

7 If the tip is adjustable, move to a suitable area and turn the lock nut upside down to remove both sealing washers. Make sure the sealing washers are no longer inside the lock nut.

8 Holding the tip at its upper end, carefully pull the pipetting tubing approximately 50 mm out of the tip adapter.

9 Pull the tip out of the tubing, holding the tubing with your other hand. You may use a dry piece of emery cloth for an improved grip on the tube. Be careful not to kink the tubing.

Note: *Do not use emery cloth on probes, as it would abrade the delicate probe coating. Use a dry piece of emery cloth for an improved grip on tubing only; wet sandpaper could leave tiny particles and thus clog inside the tubing and/or probe.*

10 Cut off approximately 5 mm of the tubing, using a sharp knife to obtain a straight cut.

11 Install the new probe as described in Section B.2.1, ”Installing a Standard Z Axis or Dual Z Axis Probe” on page B-3.

12 Dispose of a used tip as a sharp.

B.2.5 Replacing an 8-Channel Pipetting Head

See Section 3.9.2, ”8-Channel Pipetting Head” on page 3-23 for step by step instructions for installing a new 8-channel pipetting head.

B.2.6 Replacing 8-Channel Pipetting Head Probes

To replace a probe on an 8-channel pipetting head:

1 Drain the system fluid from the tubing and the probes.

2 Manually move the Universal Z axis up and toward the front of the workspace for easier access to the 8-channel head.

3 Pull the tubing off the tips. Wrap the tube near its end with a small piece of emery cloth to have a better grip on the tube. Be careful not to kink the tubing.

Note: *Do not use emery cloth on probes, as it would abrade the delicate probe coating. Use a dry piece of emery cloth for an improved grip on tubing only; wet sandpaper could leave tiny particles and thus clog inside the tubing and/or probe.*

4 Loosen the two screws that secure the Delrin probe support to the 8-channel head block. Loosen the screws evenly to avoid bending the probes; too much deflection will cause the probes to be permanently bent. See Figure B-2.

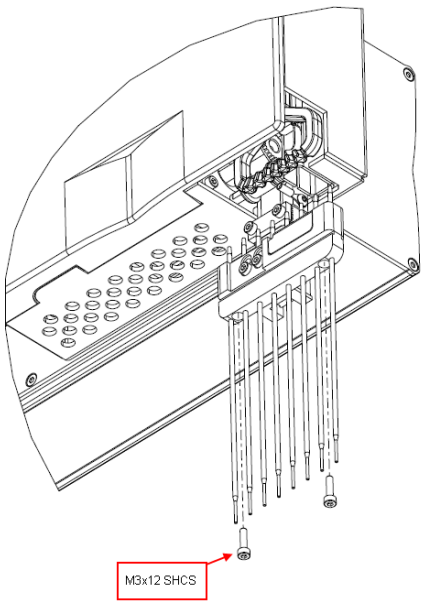


Figure B-2 *Removing 8-channel pipetting head probe support from block*

5 Remove the Delrin block with springs and probes as shown in Figure B-3.

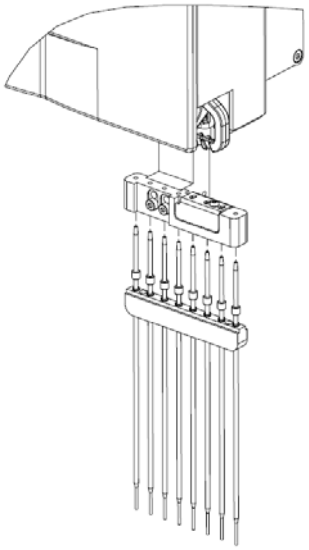


Figure B-3 *Removing 8-channel head probes*

6 Remove and replace the probes within the Delrin block. The springs are held in place by a slight interference fit; technicians should be careful to not lose springs if they come out of the Delrin block when removing the probes.

Caution! *Missing springs can impact cLLD performance.*

7 Reinstall the Delrin block with springs and new probes, tightening the screws evenly (alternating from screw to screw as they thread into the main body of the 8-channel head), to avoid causing too much deflection of the probes, which can result in a probe being permanently bent.

8 Cut off approximately 5 mm of the tubing, using a sharp knife to obtain a straight cut.

9 Attach the tubing to the blank, conical end of the probe 6 to 8 mm along its axis. (A dry piece of emery cloth may be used to improve grip on the tubing only.)

B.3 Disposable Tip (DiTi) Pickup Mechanism and Tips

Caution! *Before loading disposable tip trays into the rack and onto the worktable, make sure that the DiTis are faultless and clean:*

- *Ensure that only regular and straight Tecan disposable tips are being used.*
- *Inspect the DiTi box for traces of microbial contamination.*

WARNING! *Pipetting tips can cause injuries.*

- *Avoid contact with the pipetting tips and contact with aerosols when accessing the worktable by wearing adequate protective clothing.*

WARNING! *Possible contamination. Disposable tips can be contaminated.*

- *Assure appropriate safety measures (for example, wear rubber gloves).*
- *Dispose of used DiTis properly and safely according to your local regulations.*

B.3.1 Installing a DiTi Pickup Mechanism on a Standard Z Axis or Dual Z Axis

Figure B-4 shows the parts involved in installing the DiTi pickup mechanism.

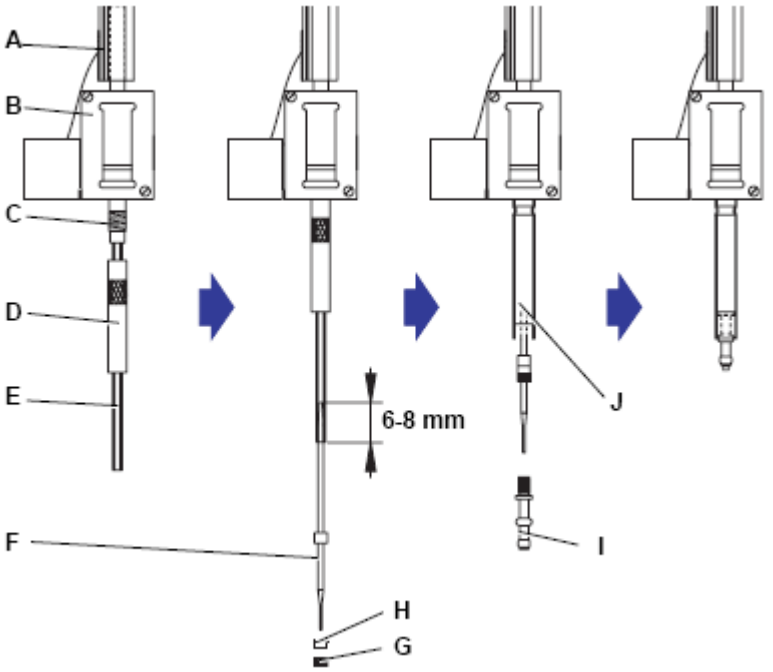


Figure B-4 *DiTi pickup mechanism installation*

Perform the following tasks to install the DiTi pickup mechanism:

1. Drain the system fluid from the tubing and the tubing extension.
2. Manually move the Z rack up until it lightly touches the stop.
3. Move the Z rack to the front of the workspace.
4. Carefully pull the pipetting tubing approximately 50 mm out of the tip adapter.
5. Screw the adapter cylinder onto the tip adapter.
6. Tighten the adapter cylinder slightly.
7. Wrap the tube near its end with a small piece of emery cloth to have a better grip on the tube.

Note: *Do not use emery cloth on probes because it can abrade the delicate probe coating. Use a dry piece of emery cloth for an improved grip on tubing only; wet sandpaper could leave tiny particles that can cause a clog inside the tubing and/or probe.*

8. While holding the tube end wrapped in emery cloth, insert the conical part of the tubing extension 6 to 8 mm into the tubing. If needed, pull the tubing further out of the adapter cylinder to install the tubing extension.

9 Insert the tubing extension into the adapter cylinder.

10 Slide separator ring and O-ring onto the tubing extension.

11 Guide the tip ejector tube, outer rim pointing upwards, over the adapter cylinder and fasten with the DiTi cone.

12 Tighten carefully, using the supplied cone wrench.

B.3.2 Cleaning a DiTi Pickup Mechanism and Performing Daily Maintenance

WARNING! *Possible contamination. The space between disposable tip cones and the tubing extension can become contaminated and thus create a contamination risk. Decontaminate the module on a regular basis to mitigate against the contamination risk.*

WARNING! *Possible malfunction of the DiTi adapter.*

- *Disposable tip cones can become wet or contaminated, and in time, deposits on the cone can result in disposable tip pick-up problems and can clog the tubing extension.*

Perform the following tasks on a daily basis:

1 Clean the DiTi cone with a lint-free tissue soaked in ethanol (70%) or isopropanol.

2 Visually check that the DiTi cone and the protruding tip, i.e., tubing extension, are clean and free of all deposits.

3 If deposits are visible, remove and thoroughly clean the disposable tip adapter.

B.3.3 Removing the DiTi Pickup Mechanism

Perform the following tasks to remove the DiTi pickup mechanism:

1 Drain the system fluid from the tubing and the tubing extension.

2 Hold the tip adapter while unscrewing the DiTi cone, using the supplied cone wrench.

3 Remove the Tip Ejector Tube.

4 Free approximately 50 mm of tubing by pulling down the tubing extension.

5 Pull the tubing off the tubing extension and unscrew the adapter cylinder.

B.4 Replacing Liquid Tubing

WARNING! *Pipetting tubing and probes can be contaminated.*

- *Decontaminate the module and assure appropriate safety measures.*

Caution! *To avoid the possibility of leaks leading to spills or contamination, check for leaks after changing tubing. Refer to section Section B.1.1, "Finding and Repairing Leaks".*

B.4.1 Replacing Standard Z Axis and Dual Z Axis Tubing

WARNING! *Even though the outer tubing of a liquid system does not require any maintenance, the liquid tubing must be replaced periodically.*

Caution! *When replacing liquid tubing in a Dual Z Axis, make sure to connect the new tubing to the same tip and pump as the old tubing. It is recommended that one liquid tube be replaced at a time to avoid possible confusion.*

To replace the liquid tubing:

1. Drain the system fluid from the tubing and the probe.
2. Remove the cover from the Z axis by pulling it straight off as shown in Figure 3-12, "Installation of liquid tubing showing tubing path, Standard Z axis" on page 3-16.
3. Remove the probe or DiTi cone from the tip adapter as described in Section B.2.4, "Replacing a Standard Z Axis or Dual Z Axis Probe" on page B-4 and Section B.3.3, "Removing the DiTi Pickup Mechanism" on page B-9, respectively.
4. Disconnect the tubing from the probe or DiTi adapter. Wrap the tube near its end with a small piece of emery cloth to have a better grip on the tube.

Note: *Do not use emery cloth on probes, as it would abrade the delicate probe coating. Use a dry piece of emery cloth for an improved grip on tubing only; wet sandpaper could leave tiny particles and thus clog inside the tubing and/or probe.*

5. Pull the tubing out of the sheath tubing and out of the back of the robot.
6. Install liquid tubing as described in Section 3.8.1, "Standard and Dual Z Axis Tubing".

After replacing tubing, use the OEM software to perform a liquid level detection test.

B.4.2 Replacing Universal Z Axis Tubing

Note: *For easier replacement of liquid tubing with the 8-channel head, users can detach the 8-channel head prior to changing the liquid tubing. This is not a requirement, but if users choose to do so, they should dismount both the main body of the 8-channel head and the cables, using the installation procedures in* *Section 3.9.2, "8-Channel Pipetting Head" on page 3-23**.*

To replace the liquid tubing for the Universal Z axis with 8-channel pipetting head:

1. Drain the system fluid from the tubing and the probes.
2. Manually move the Universal Z axis up and toward the front of the workspace for easier access to the 8-channel head.
3. Remove the cover from the Universal Z axis by pulling it straight off, as shown in Figure 3-13.
4. Disconnect the tubing from the probes. Wrap the tube near its end with a small piece of emery cloth to have a better grip on the tube.

Note: *Do not use emery cloth on probes, as it would abrade the delicate probe coating. Use a dry piece of emery cloth for an improved grip on tubing only; wet sandpaper could leave tiny particles and thus clog inside the tubing and/or probe.*

5. Pull the tubing out from the e-chain in the rear of the robot.
6. Remove the tubing guide entrance piece and tubing guide cover from the Y axis (see Figure B-5, "Removing Universal Z axis tubing guide elements" on page B-11).

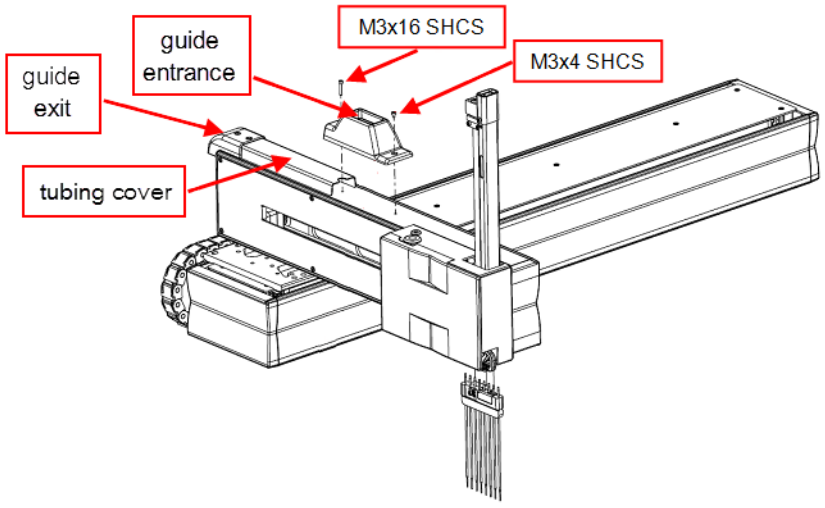


Figure B-5 *Removing Universal Z axis tubing guide elements*

7. Install liquid tubing as described in Section 3.8.1, "Standard and Dual Z Axis Tubing".

B.5 Replacing the CIB Fuses

If an Omni Robot that was previously working properly suddenly stops responding to commands, you might need to replace the fuses on the CIB.

WARNING! *The Omni Robot uses high amperage fuses. Failure to follow the proper procedure to remove and replace the fuses could lead to serious injury or equipment damage.*

For the rating of the fuses, see Section C.8, ”Electrical Specifications” on page C-22.

To replace the fuses on the CIB:

1. Unplug the Omni Robot power supply.
2. Remove the four screws that secure the CIB mounting plate to the underside of the X axis and lower it down to access the CIB.
3. Locate the fuses in the upper-left corner on the front side of the CIB (see Figure B-6).

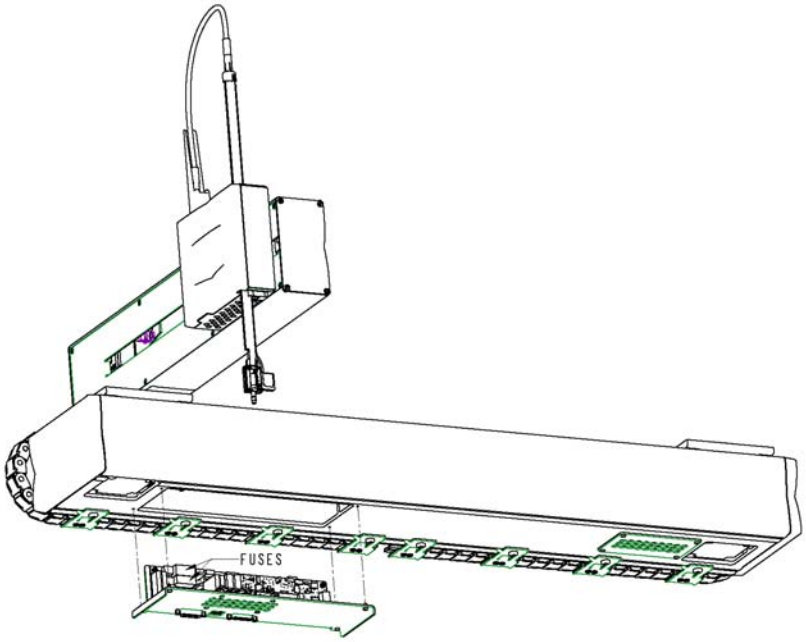


Figure B-6 *Location of fuses on CIB*

4. Using a fuse removal tool, remove and replace the old fuse(s).
5. Reattach the CIB mounting plate to the underside of the X axis using four screws, making sure all cables are back in the extrusion.
6. Plug in the Omni Robot power supply.

B.6 Replacing the cLLD Cable

The cLLD system receives signals through a coaxial cable that connects the Z axis and the tip adapter. Maintenance of the cLLD cable differs between a Standard Z or Dual Z axis and a Universal Z axis.

B.6.1 cLLD Cable Replacement on Standard Z Axis

To replace the cLLD cable on a Standard Z axis:

1. Unplug the Omni Robot power supply.
2. Remove the Z axis cover (number **1** in Figure B-7).

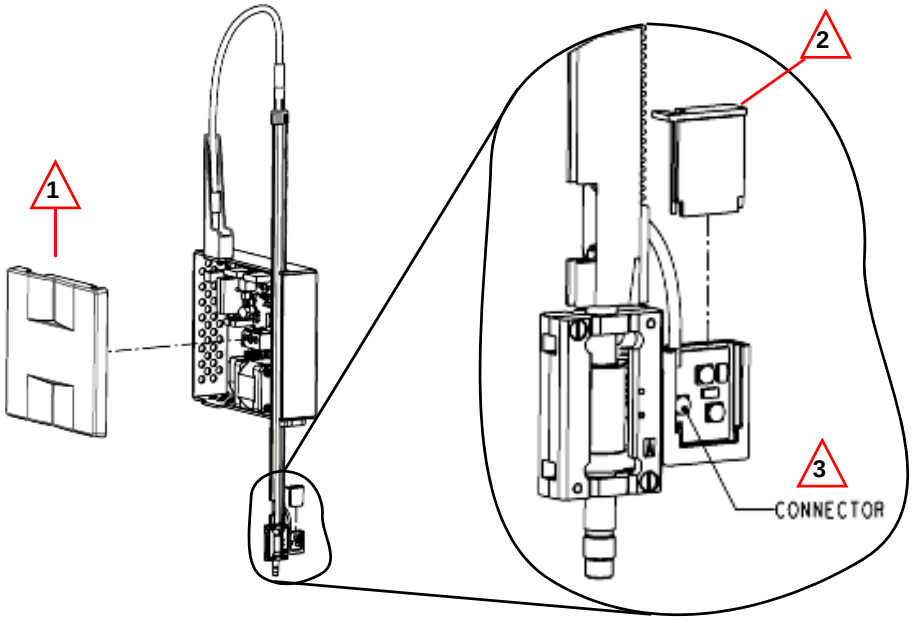


Figure B-7 *Cable connector location on tip adapter*

3. Remove the small cover on the tip adapter by sliding it upwards (number **2** in Figure B-7).
4. Disconnect the tip adapter (number **3** in Figure B-7) and disconnect the cable from the Z axis (number **4** in Figure B-8).
5. Remove the cable from the Z axis end (not the tip adapter end).

6 Insert the new cable from the Z axis end (not the tip adapter end).

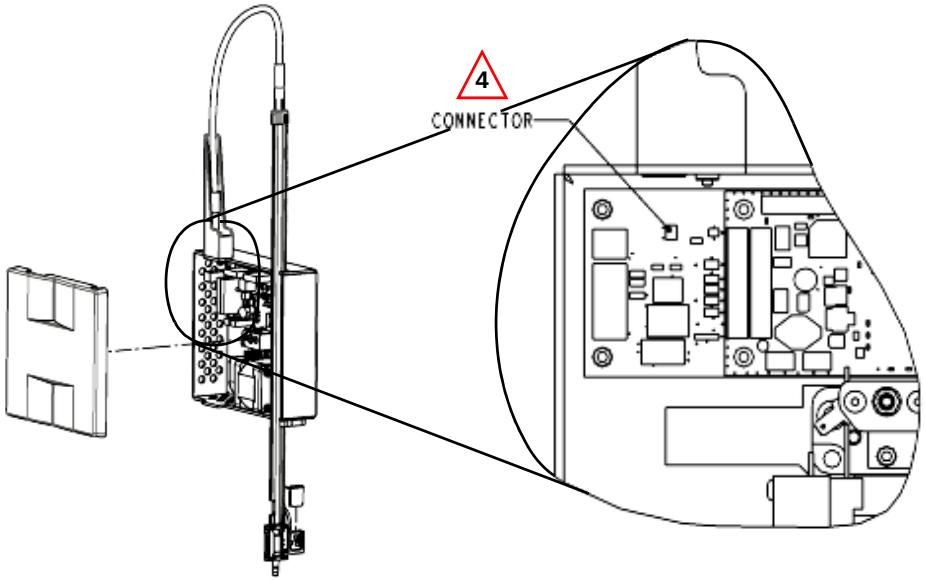


Figure B-8 *cLLD cable connector location on cLLD PCBA*

7 Connect the cable to the tip adapter at J1 on the PCBA (number **3** in Figure B-7), and then to the Z axis (number **4** in Figure B-8).

8 Replace the covers on the Z axis and tip adapter (numbers **1** and **2** in Figure B-7).

9 Plug in the Omni Robot power supply.

B.6.2 cLLD Cable Replacement on Dual Z Axis

To replace the cLLD cable on a Dual Z axis:

1 Unplug the Omni Robot power supply.

2 Remove the Z axis cover (number **1** in Figure B-7, “Cable connector location on tip adapter” on page B-13).

3 Remove the small cover on the tip adapter by sliding it upwards (number **2** in Figure B-7).

4 Disconnect the tip adapter (number **3** in Figure B-7) and disconnect the cable from the Z axis (number **4** in Figure B-8).

5 Remove the cable from the Z axis end (not the tip adapter end).

6 Insert the new cables from the Z axis end (not the tip adapter end).

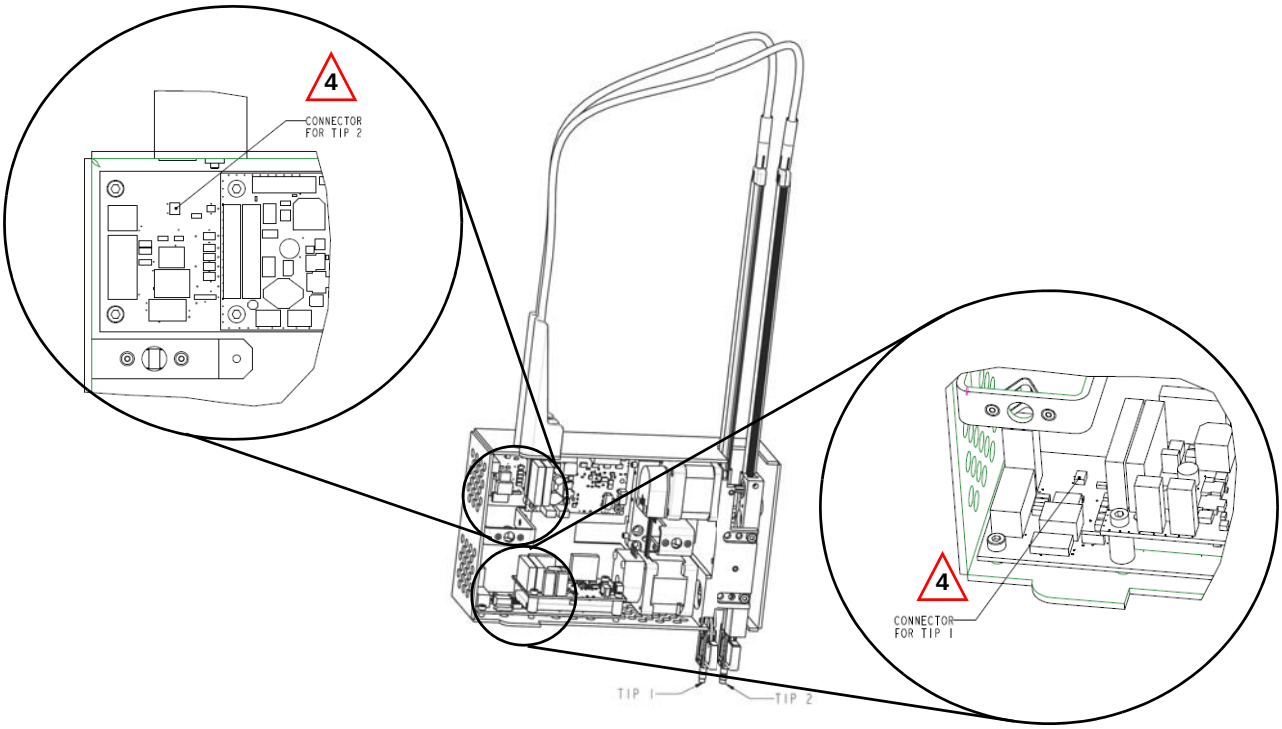


Figure B-9 *cLLD cable connector location on cLLD PCBA, Dual Z axis*

7 Connect the cable to the tip adapter at J1 on the PCBA (number **3** in Figure B-7), and then to the Z axis (number **4** in Figure B-8).

Note: *The lower PCBA connects to tip 1 and the upper PCBA connects to tip 2 as shown in* Figure B-9.

Note: *For the Dual Z axis, the mesh closest to the cover connects to the front-most probe or DiTi adapter.*

8 Replace the covers on the Z axis and tip adapter (numbers **1** and **2** in Figure B-7).

9 Plug in the Omni Robot power supply.

B.6.3 cLLD Cable Replacement on Universal Z Axis with 8-Channel Pipetting Head

To replace the cLLD cable on a Universal Z axis with an 8-channel pipetting head:

1 Unplug the Omni Robot power supply.

2 Drain the system fluid from the tubing and the probes.

3 Manually move the Universal Z axis up and toward the front of the workspace for easier access to the 8-channel head.

4 Remove the cover from the Universal Z axis by pulling it straight off as shown in Figure 3-13.

5 Disconnect the tubing from the probes. Wrap the tube near its end with a small piece of emery cloth to have a better grip on the tube.

Note: *Do not use emery cloth on probes, as it would abrade the delicate probe coating. Use a dry piece of emery cloth for an improved grip on tubing only; wet sandpaper could leave tiny particles and thus clog inside the tubing and/or probe.*

6 Dismount the 8-channel pipetting head from the Universal Z axis and remove the sheet metal PCBA cover (see installation instructions in Section 3.9.2, ”8-Channel Pipetting Head” on page 3-23).

7 Disconnect the cable from the 8-channel head PCBA.

8 Dismount the tubing guide ramp (see Figure 3-14, “Mounting tubing guide components” on page 3-18).

9 Disconnect the cable from the cLLD board in the Universal Z axis.

10 Remove cable and braided sleeve from the system by pulling it out the top of the Z axis tubing guide and Z axis housing grommet.

11 Remove cable from the braided sleeve and install the new cable.

12 Reinstall cable and braided sleeve into the Z axis housing grommet so that it is held in place (a few mm of protrusion of the sleeve into the Z axis is sufficient).

13 Connect the cable to the cLLD board (see Figure B-8, “cLLD cable connector location on cLLD PCBA” on page B-14).

14 Feed cable down through the Z axis tubing guide and out the bottom.

15 Reinstall the tubing guide ramp (see Figure 3-14, “Mounting tubing guide components” on page 3-18 and Figure 3-15, “8-channel head tubing guide orientation” on page 3-19).

16 Reinstall the 8-channel head as described in Section 3.9.2, ”8-Channel Pipetting Head” on page 3-23.

17 Reinstall the Universal Z cover.

18 Plug in the Omni Robot power supply.

B.6.4 cLLD Cable Replacement on Universal Z Axis with ADP

To replace the cLLD cable on a Universal Z axis with ADP:

1 Unplug the Omni Robot power supply.

2 Remove the Universal Z axis cover (see Figure B-11, “Removing the Universal Z axis cover” on page B-18).

3 Dismount the ADP and disconnect cables from the Universal Z axis (see installation instructions in Section 8.3, ”Installing the ADP”).

4 Disconnect the cable from the cLLD board in the Universal Z axis.

5 Remove cable and braided sleeve from the system by pulling it out the top of the Z axis tubing guide and Z axis housing grommet.

6 Remove cable from the braided sleeve and install the new cable.

7 Reconnect and reattach the ADP (see installation instructions in Section 8.3, ”Installing the ADP”).

8 Ensure the braided sleeve is pushed all the way forward on the cLLD coax until it is visible in the opening of the Z axis tubing guide where the cables come out to connect to the ADP, then reinstall the cable and braided sleeve into the Z-axis housing grommet so that it is held in place (approximately 6-12 mm of protrusion).

9 Gently coil the cLLD coax approximately as shown in Figure B-10.

Figure B-10 *cLLD coax cable coiled for installation*

10 Connect the cable to the cLLD board (see Figure B-8, “cLLD cable connector location on cLLD PCBA” on page B-14).

11 Reinstall the Universal Z cover.

12 Plug in the Omni Robot power supply.

B.6.5 Lubrication Instructions for the Universal Z Axis Brake Plunger Mechanism

To lubricate the brake plunger mechanism on the Universal Z axis:

1. Unplug the Omni Robot power supply.
2. Remove the Universal Z axis cover (see Figure B-11).

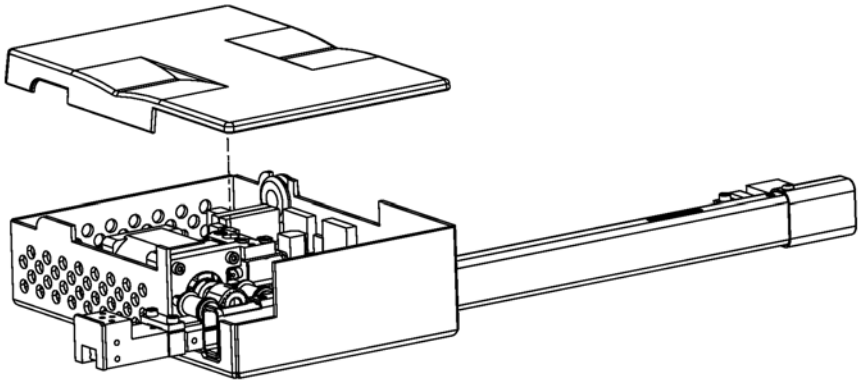


Figure B-11 *Removing the Universal Z axis cover*

3. Unplug the brake solenoid (see Figure B-12).

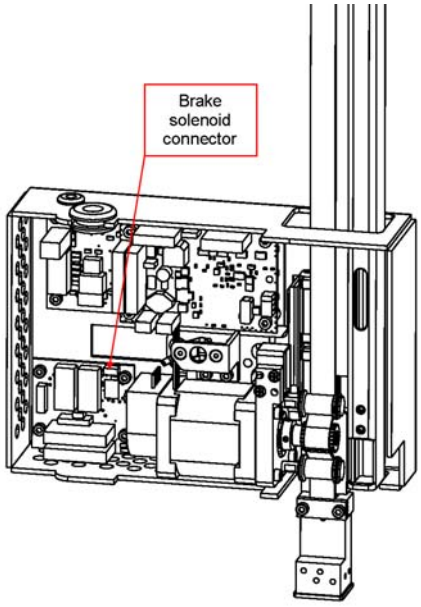


Figure B-12 *The brake solenoid connector*

4 Remove the solenoid mounting screws (see Figure B-13).

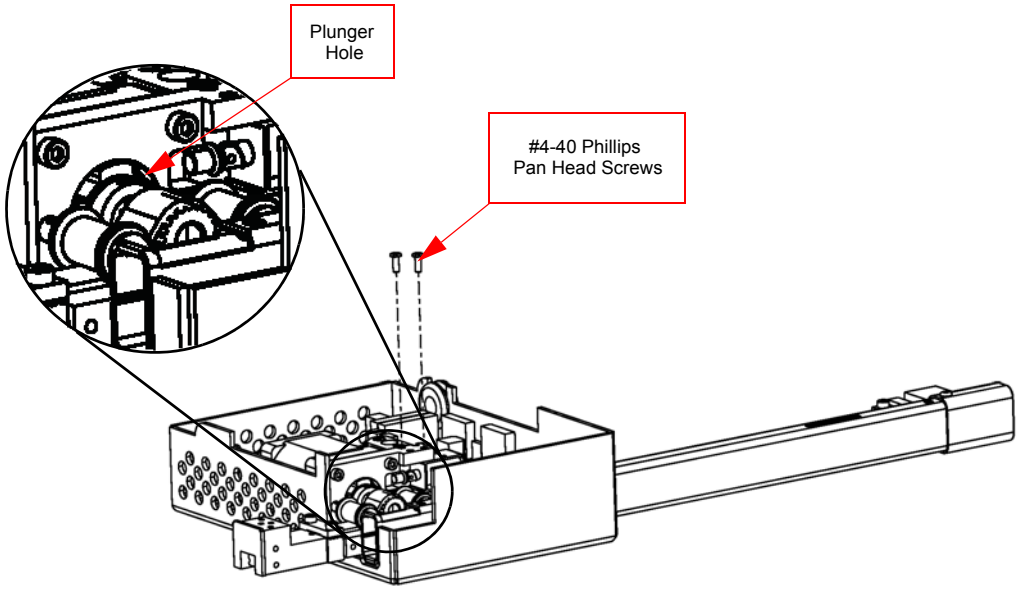


Figure B-13 *Removing the solenoid mounting screws and solenoid*

5 Remove the solenoid and plunger from the system (see Figure B-13).

Caution! *Be careful to not lose the plunger spring. Brake functionality may be compromised if the spring is lost.*

6 Using a cotton swab, wipe away grease from the plunger and the plunger hole in the mounting block (see Figure B-13 and Figure B-14, “Solenoid and plunger assembly surface to be greased” on page B-20).

7 Apply a thin coat of grease to the plunger. The recommended grease is T-SY P/N 30037875 (see Table B-12, “Other field-replaceable parts,” on page B-26).

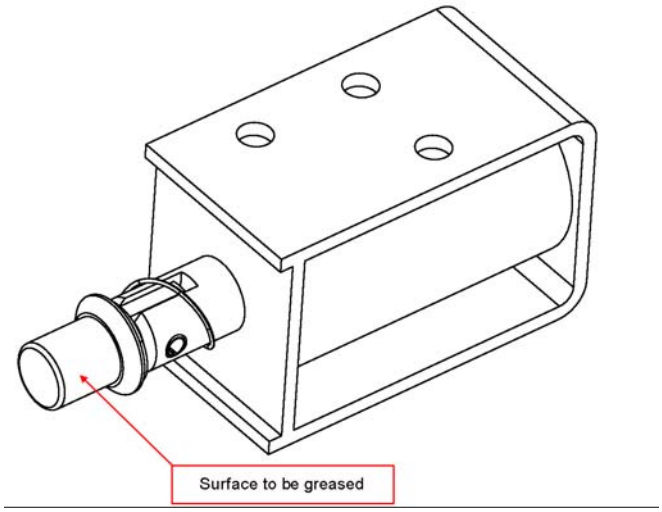


Figure B-14 *Solenoid and plunger assembly surface to be greased*

8 Reinstall the plunger and solenoid.
9 Reinsert the mounting screws and reconnect the cable to the brake PCBA.
10 Reinstall the Universal Z cover.
11 Plug in the Omni Robot power supply.
12 Run the brake self-test to ensure proper brake function.

B.7 Field Replaceable Parts

The following tables list the Omni Robot parts that can be replaced in the field.

Table B-1 *PCB assemblies*

Part Number	Description
30072452	PCBA ACB
30037796	PCBA CIB
30037798	PCBA Hall sensor
30037800	PCBA cLLD
30055188	PCBA 8-Channel liquid detection
30060830	PCBA Omni Universal Z brake
30082709	Cavro® Omni Hub
30082710	Axis Termination Board

Table B-2 *Axis cables*

Part Number	Description
30038590	Axis cable for 500X Axis cable for Cavro® Omni Hub
30037806	Axis cable for 750X
30038592	Axis cable for 1250X
30038594	Axis cable for 150Y
30037804	Axis cable for 300Y
30037802	CIB to ACB axis cable; Universal Z ACB to brake axis cable (Universal Z with ADP only); ACB to ACB axis cable (Dual Z only)
30055181	CIB to ACB axis cable (Dual X) 500X
30055183	CIB to ACB axis cable (Dual X) 750X
30055186	CIB to ACB axis cable (Dual X) 1250X

Table B-3 *Other cables*

Part Number	Description
30045965	X axis encoder cable
30045967	Y axis encoder cable
30037780	Z axis encoder cable
30037810	cLLD to ACB cable Universal Z ACB to brake cable
30037851	Ethernet cable
30072454	Liquid detection coax cable (1100 mm)
30039363	Left-oriented Universal Z with ADP flat flexible cable (FFC)
30039365	Right-oriented Universal Z with ADP flat flexible cable (FFC)
30084748	CIB RS-232 communication cable
30084751	CIB CAN communication cable

Table B-4 *Flex chains and support brackets*

Part Number	Description
30037877	Flex chain for 500X
30037879	Flex chain for 750X
30037881	Flex chain for 1250X
30037883	Flex chain for 150Y
30037885	Flex chain for 300Y
30055885	Flex chain for 500X, wide chain
30055887	Flex chain for 750X, wide chain
30055889	Flex chain for 1250X, wide chain
30062662	Standard flex chain support bracket (pack of 4)
30062664	Large flex chain support bracket (pack of 4)

Table B-5 *Motors and motor assemblies*

Part Number	Description
30037777	X axis motor assembly
30037774	Left-oriented Y motor assembly
30045810	Right-oriented Y motor assembly
30037808	Left-oriented Standard Z motor; Left-oriented Dual Z motor (lower)
30045814	Right-oriented Standard Z motor; Right-oriented Dual Z motor (lower)
30055881	Left-oriented Universal Z motor assembly
30055883	Right-oriented Universal Z motor assembly
30072456	Left-oriented Dual Z motor (upper)
30072458	Right-oriented Dual Z motor (upper)

Table B-6 *Timing belts*

Part Number	Description
30037784	Timing belt for 500X
30037786	Timing belt for 750X
30037788	Timing belt for 1250X
30037790	Timing belt for 150Y
30037792	Timing belt for 300Y
30045812	Timing belt for 210 Universal Z

Table B-7 *Tubing*

Part Number	Description
30037856	Sheath tubing for 500X
30037858	Sheath tubing for 750X
30037860	Sheath tubing for 1250X
30049478	FEP M6, .059” ID Tubing, 120” long
30041398	FEP 1/4-28, .059” ID Tubing, 120” long
30082735	FEP M6, .059” ID Tubing, 120” long (pack of 8)
30082737	FEP 1/4-28, .059” ID Tubing, 120” long (pack of 8)

Table B-8 *Z axis components*

Part Number	Description
30037771	Z rack and housing assembly
30037812	Adapter from tubing mesh to Z box
30037868	Z axis tubing and coax support mesh
30038596	Adapter from tubing mesh to Z rack (Standard Z and Dual Z rear rack)
30072460	Adapter from tubing mesh to Z rack (Dual Z front rack)
30055891	Standard Z axis cover
30055893	Universal Z axis cover
30072462	Dual Z axis cover
30073921	Dual Z rack spacer tool
30060832	Brake solenoid
30070195	Omni tip adapter and probe holder

Table B-9 *Probes*

Part Number	Description
30055737	DiTi cone option
30061013	DiTi cone wrench (5 pcs)
30055731	Tip, standard 96-well with lock nut
30055733	Tip, ceramic 96-well with lock nut
30084741	Tip, low volume with lock nut
30055739	Lock nut, standard without probe

Table B-10 *8-Channel pipetting head option*

Part Number	Description
30045276	8-channel head kit
30055895	Replacement probes (pack of 8)
30055897	Replacement 8-channel tubing separator and guide
30056512	Tubing guide kit for 150Y
30056510	Tubing guide kit for 300Y

Table B-11 *Gripper*

Part Number	Description
30060980	Concentric fingers (straight down)
30060965	Eccentric fingers (L-shaped)
30060982	Tube fingers (for gripping tubes)
30062669	Spring set: Low-tension spring, high-tension spring, extra spring post
30062672	EPDM finger pads
30062667	Cable kit: Cable, ground strap, screw
30062667	Gripper cable

Table B-12 *Other field-replaceable parts*

Part Number	Description
30037854	Fuse for CIB
30082739	Power connector parts kit
30037782	Home sensor for X and Y axes
30037871	T-nuts (pack of 10)
30037875	Grease kit

Table B-13 *Spare axes*

Part Number	Description
30072464	Standard left Z axis
30072928	Standard right Z axis
30060760	Universal Z left axis with cLLD
30060762	Universal Z right axis with cLLD
30060764	Universal Z left axis without cLLD
30060766	Universal Z right axis without cLLD
30072941	Universal Z left axis for ADP
30072943	Universal Z right axis for ADP
30072444	Dual Z left axis, 9 mm spacing
30072446	Dual Z right axis, 9 mm spacing
30072448	Dual Z left axis, 18 mm spacing
30072450	Dual Z right axis, 18 mm spacing
30084720	Single arm X axis, 500 mm travel
30084722	Single arm X axis, 750 mm travel
30084724	Single arm X axis, 1250 mm travel
30084726	Dual arm X axis, 500 mm travel
30084728	Dual arm X axis, 750 mm travel
30084730	Dual arm X axis, 1250 mm travel

Table B-13 *Spare axes (continued)*

Part Number	Description
30084732	Y axis, 150 mm travel, left
30084734	Y axis, 150 mm travel, right
30084736	Y axis, 300 mm travel, left
30084738	Y axis, 300 mm travel, right

This page has been left intentionally blank.

C Specifications

The following drawings and tables contain the mechanical and electrical specifications of the Omni.

C.1 Mechanical Drawings

The following figures show the mounting hole and overall size dimensions of the available Omni Robot axes, and XYZ robots configured with a Standard Z axis, Dual Z axis, and Universal Z axis.

If you require an axis of a length that is not listed in this section, contact your Tecan representative for more information about custom lengths.

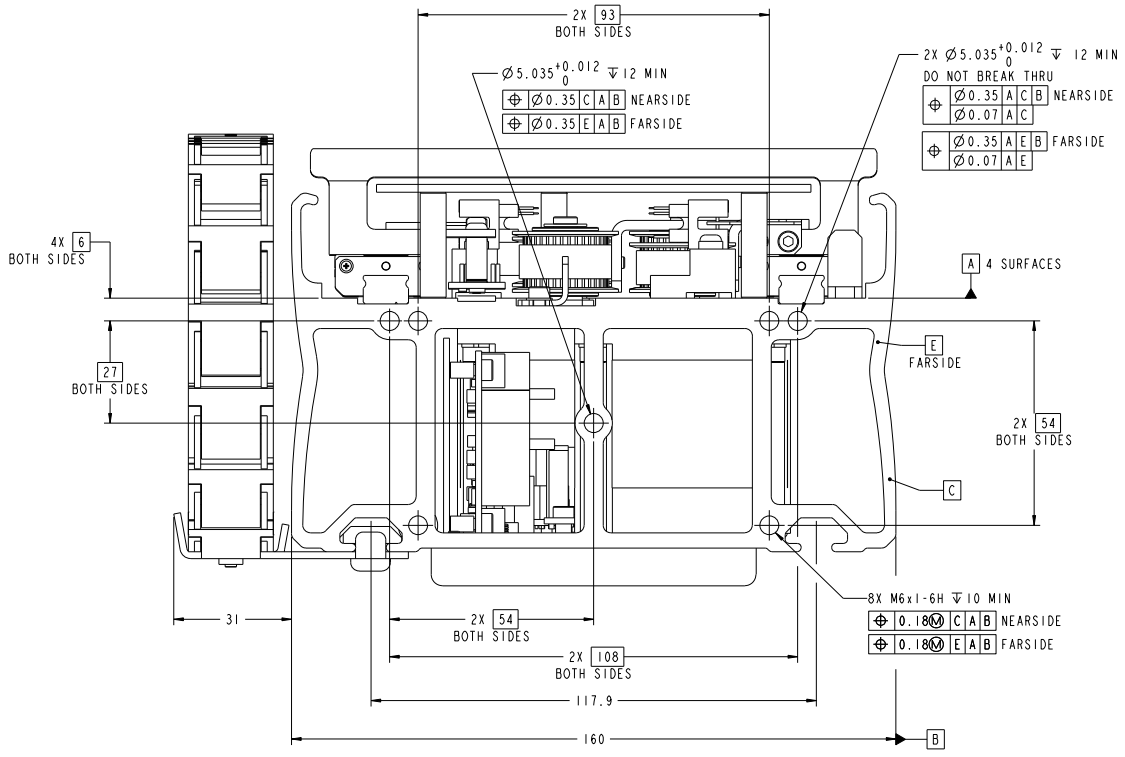


Figure C-1 *X axis mounting hole dimensions*

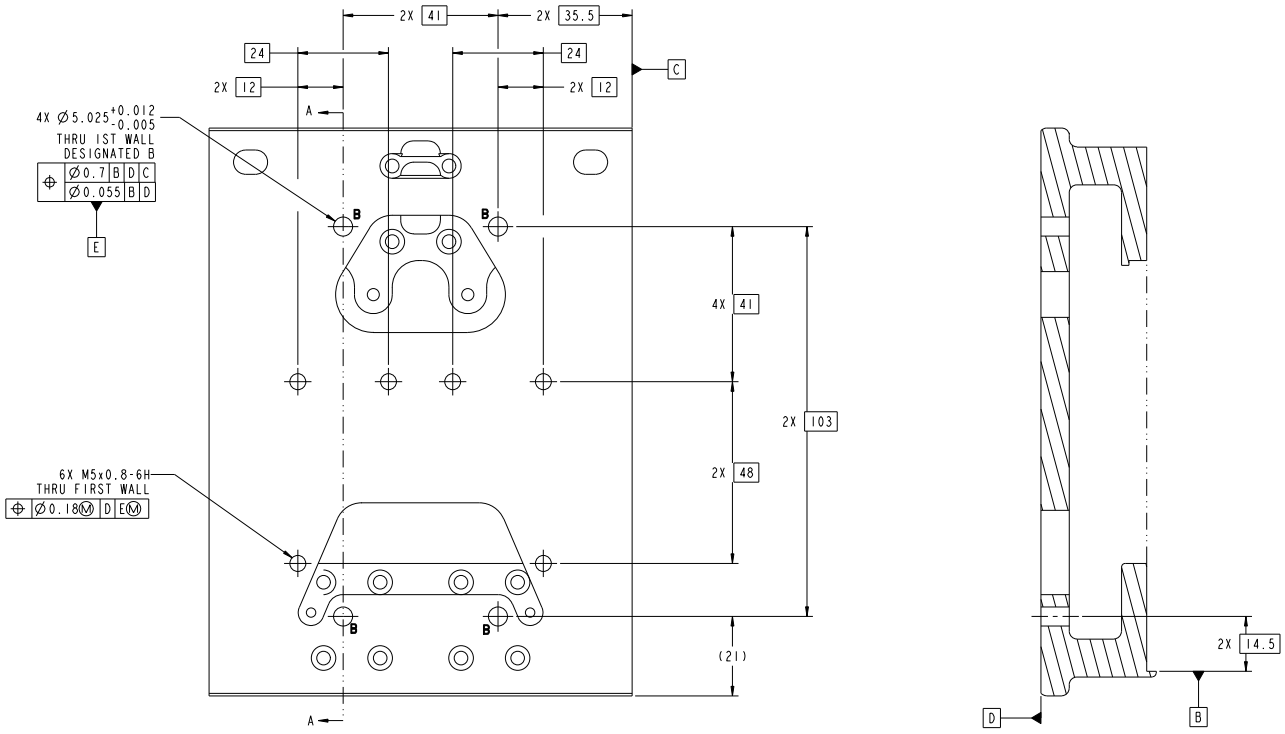


Figure C-2 *X carriage mounting hole dimensions*

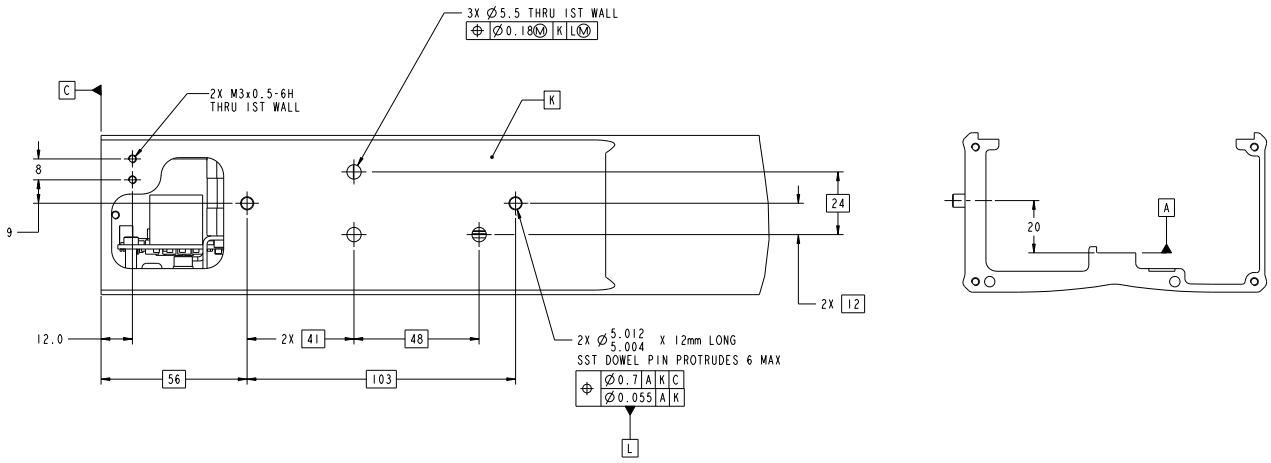


Figure C-3 *Left-oriented Y axis mounting hole dimensions*

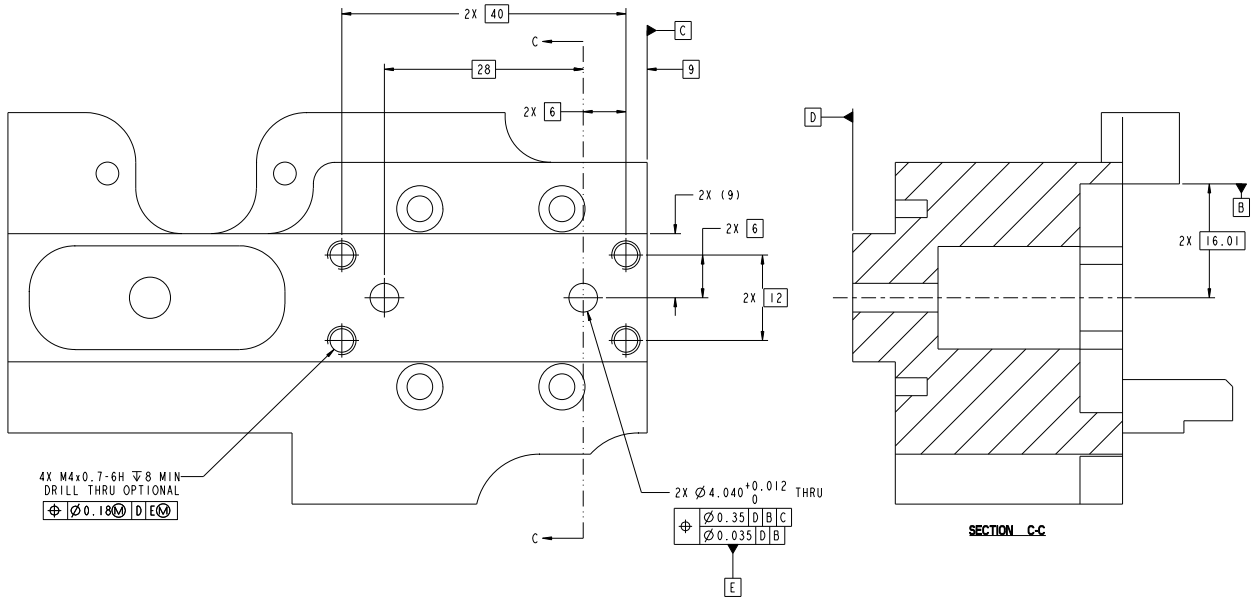


Figure C-4 *Left-oriented Y carriage mounting hole dimensions*

Note: *A right-oriented Y axis and Y carriage are mirror images of Figure C-3 and Figure C-4.*

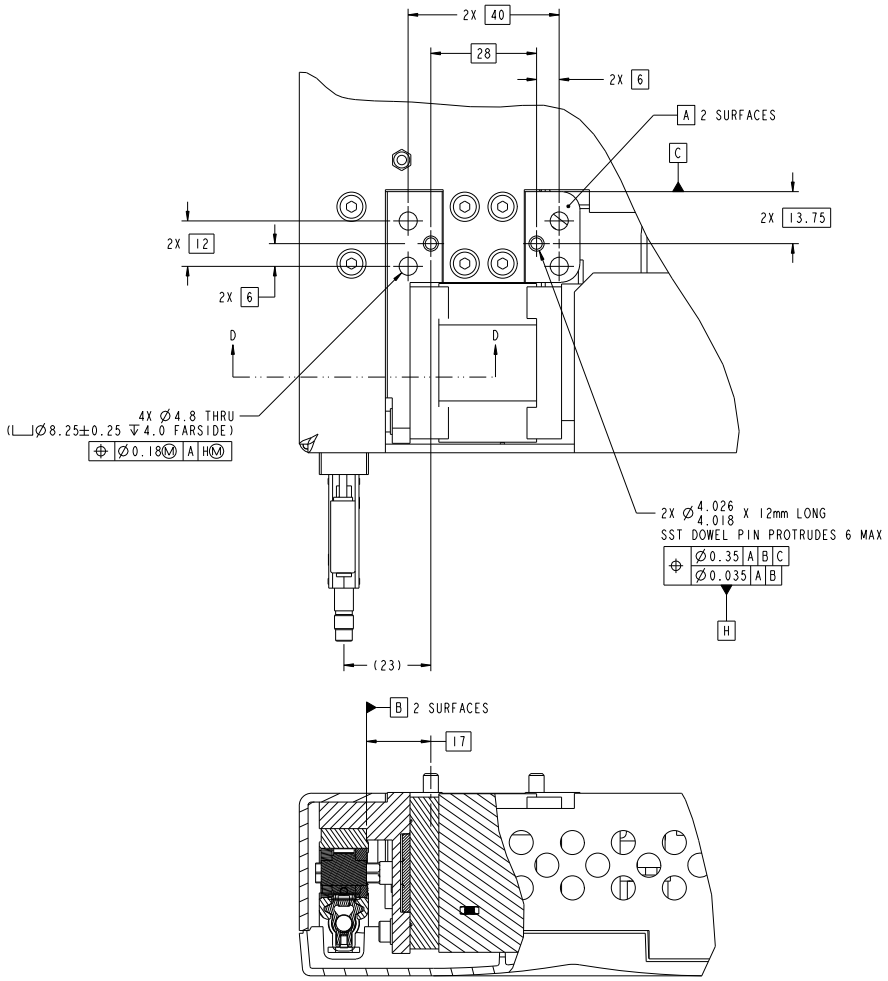


Figure C-5 *Left-oriented Z axis mounting hole dimensions*

Note: *A right-oriented Z axis is the mirror image of Figure C-5. The mounting holes and dowel pins are the same for all Z axis types.*

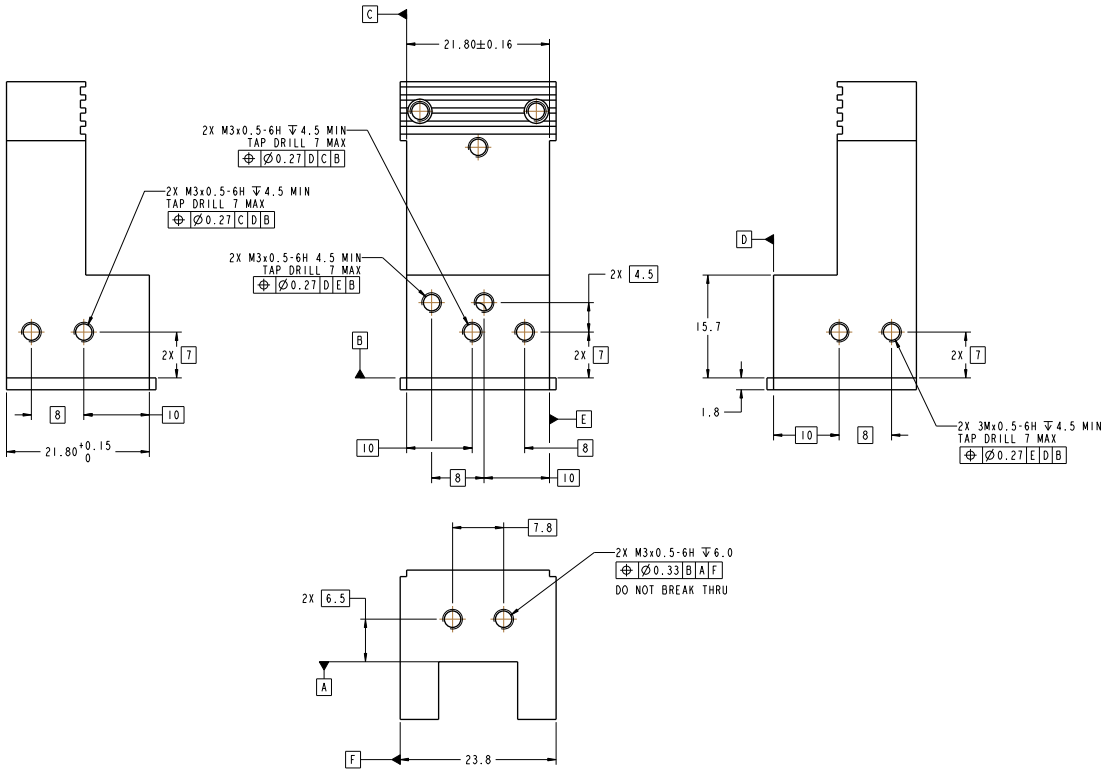


Figure C-6 *Universal Z mounting hole dimensions*

C.1.1 Cavro Omni Robot with Standard Z Axis

Note: *The dimensions for the Omni with Standard Z axis include pipetting tubing and the capacitive liquid level detection (cLLD) cable guide.*

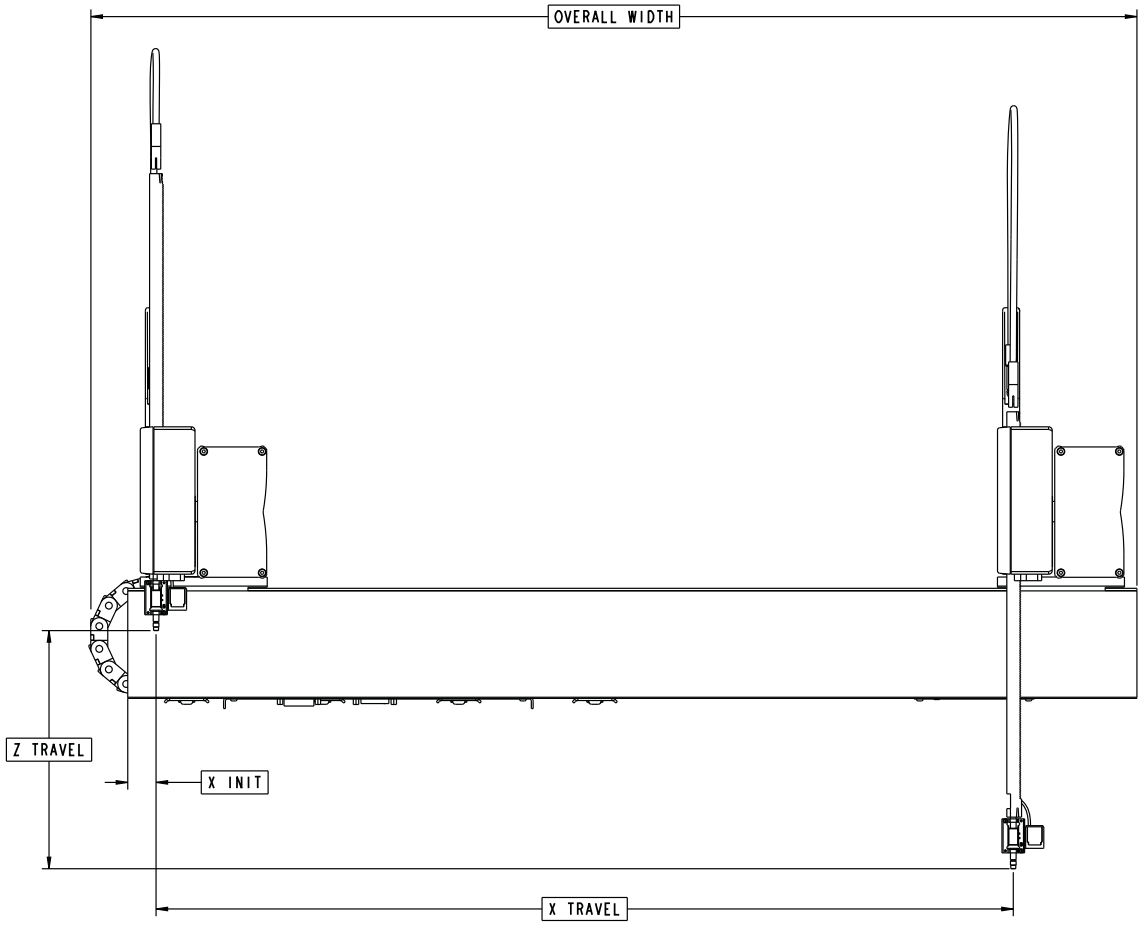


Figure C-7 *Omni Robot dimensions with Standard Z axis (front view)*

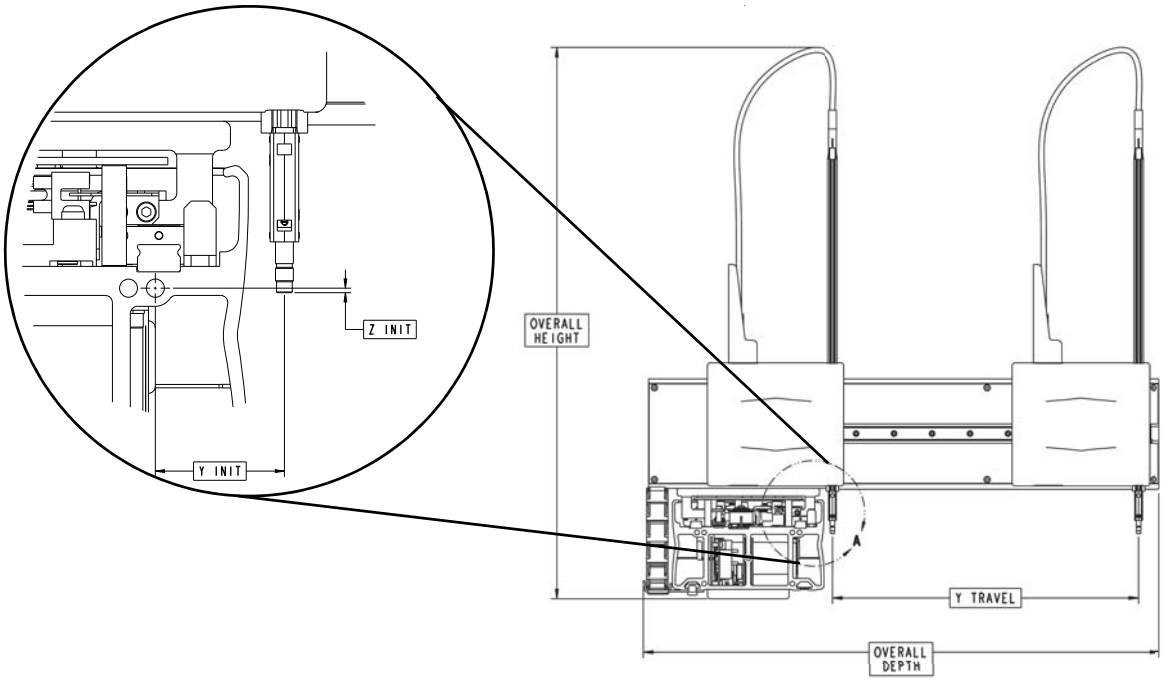


Figure C-8 *Omni Robot dimensions with Standard Z axis (end view)*

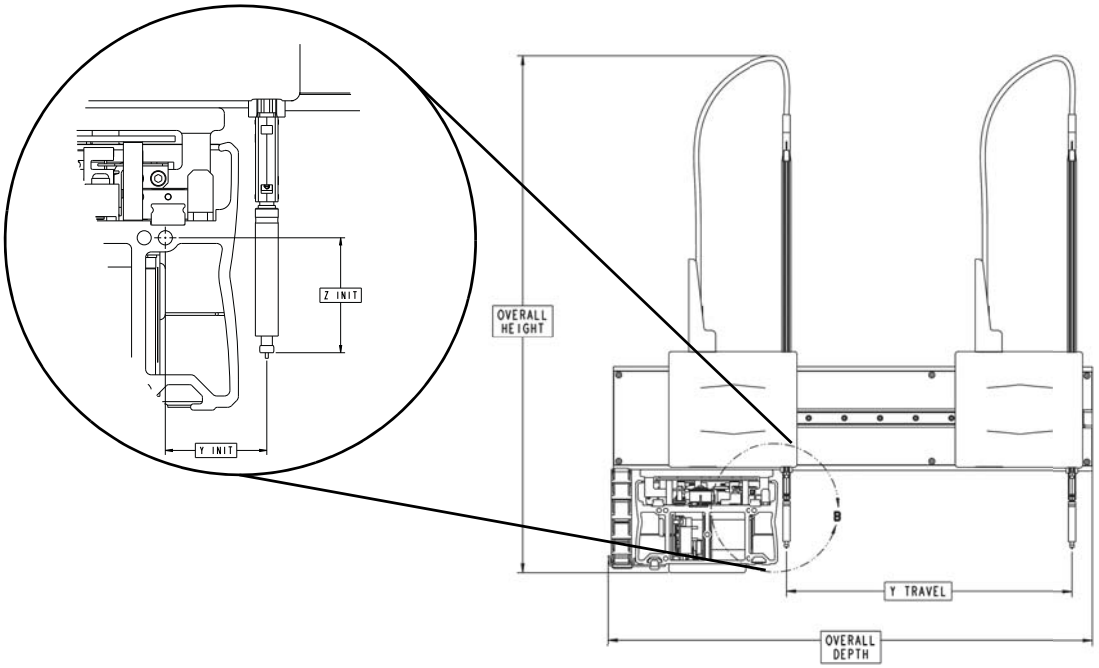


Figure C-9 *Omni Robot dimensions with Standard Z axis and DiTi (end view)*

C.1.2 Cavro Omni Robot with Dual Z Axis

Note: *The dimensions for the Omni with Dual Z axis include pipetting tubing and the capacitive liquid level detection (cLLD) cable guide.*

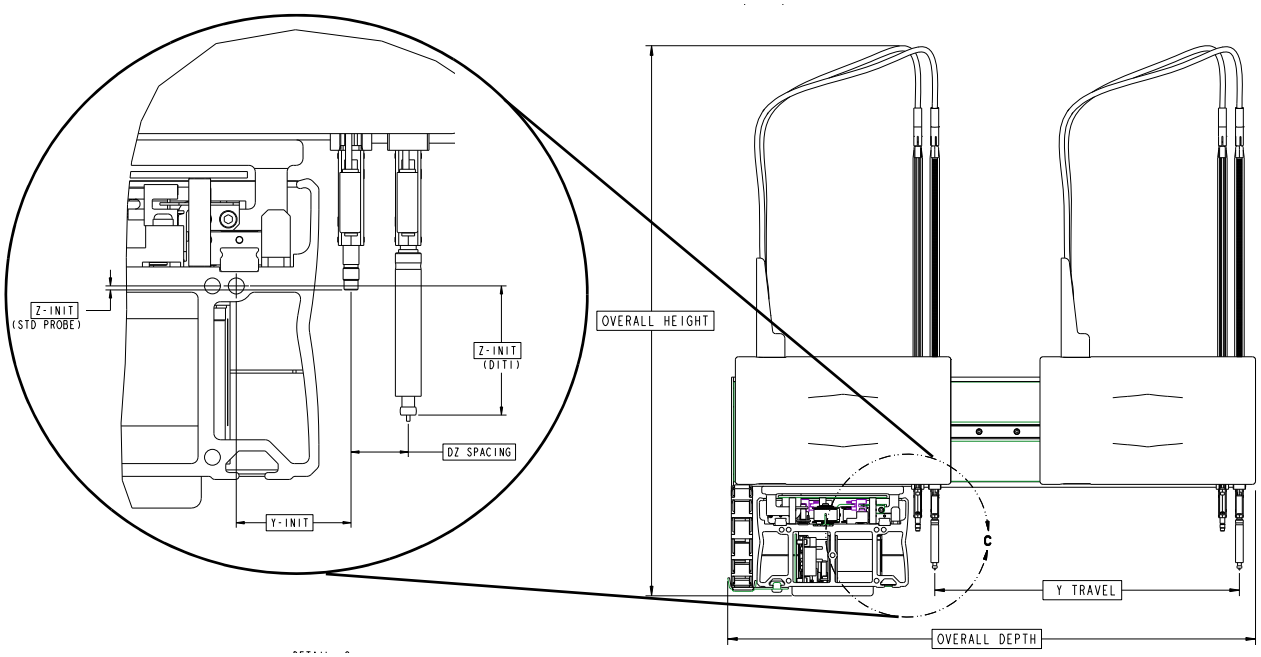


Figure C-10 *Omni Robot dimensions with Dual Z axis and DiTi (end view)*

C.1.3 Cavro Omni Robot with Universal Z Axis

Note: *The dimensions with Universal Z axis do NOT include pipetting tubing or cables.*

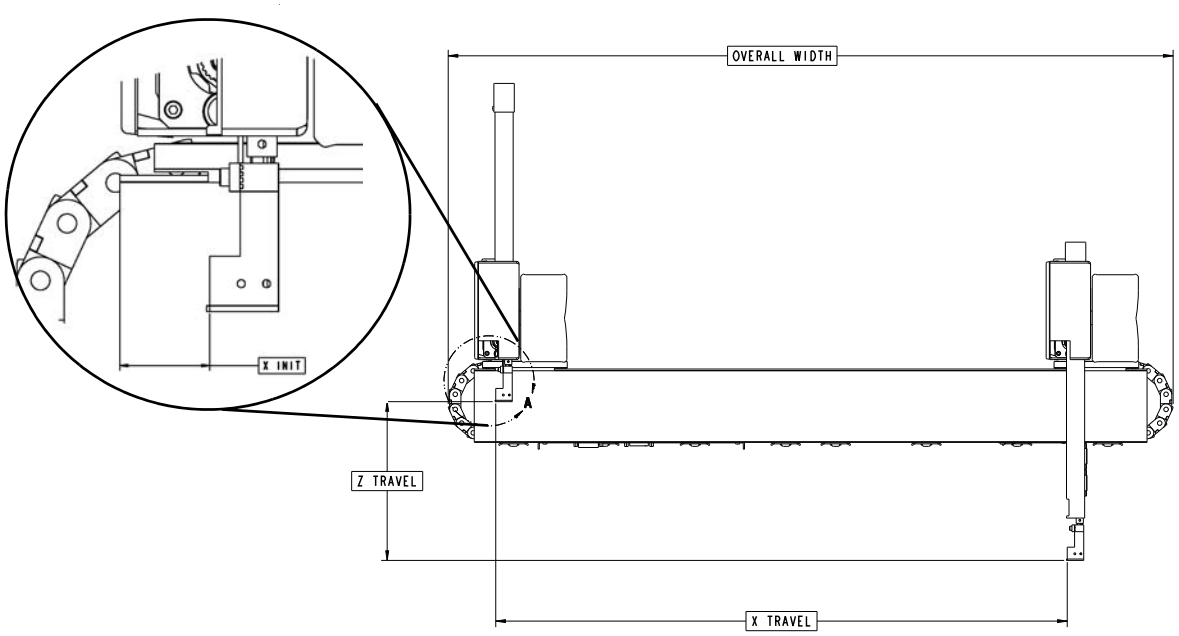


Figure C-11 *Omni Robot dimensions with Universal Z axis (top view)*

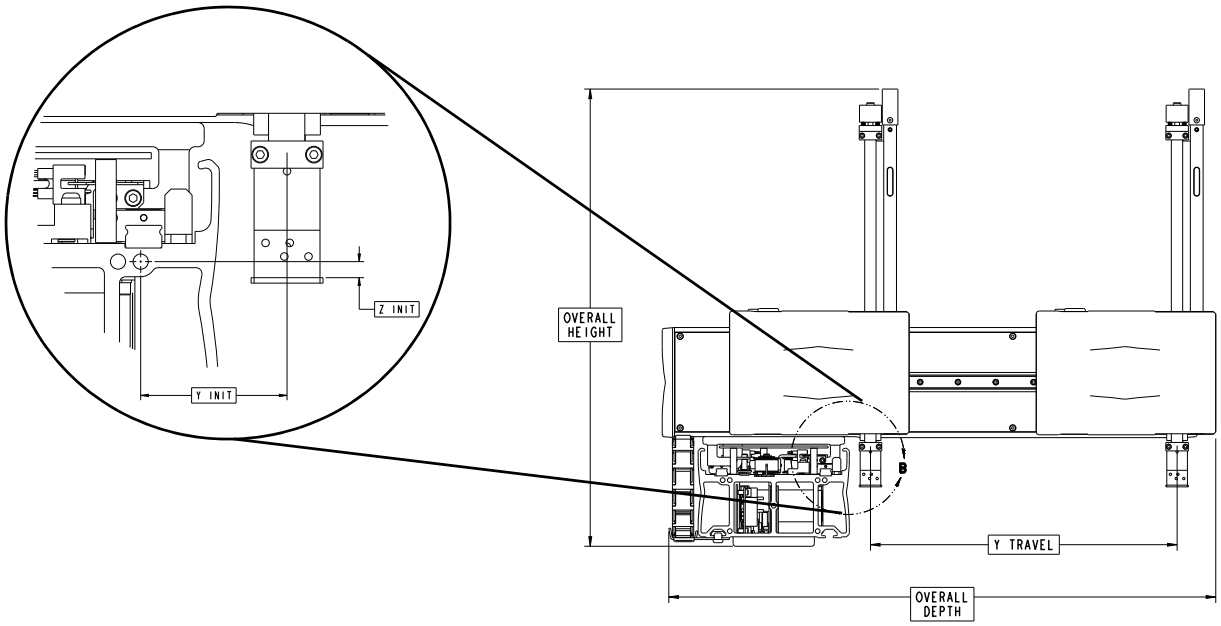


Figure C-12 *Omni Robot dimensions with Universal Z axis (end view)*

C.2 Mechanical Specifications

Table C-1 *X axis dimensions*

Maximum overall width	single arm configuration, excluding end caps	700 mm/950 mm/1450 mm
	single arm configuration, including end caps	715 mm/965 mm/1465 mm
	dual arm configuration	740 mm/990 mm/1490 mm
Nominal initial position for first arm	Standard Z axis and Dual Z axis (left-oriented)	27.1 mm
	Universal Z axis (left-oriented) (surface of Universal Z mount with 4 tapped holes)	30.6 mm (rear-most probe of 8-channel head in landscape will be at 27.1 mm; 15.6 mm for ADP probe)
	Standard Z axis and Dual Z axis (right-oriented)	111.2 mm
	Universal Z axis (right-oriented) (surface of Universal Z mount with 4 tapped holes)	107.7 mm (rear-most probe of 8-channel head in landscape will be at 111.2 mm; 122.7 for ADP probe)
Travel length	single arm	500 mm/750 mm/1250 mm
	dual arm (two Standard Z axes, two Dual Z axes, or two Universal Z axes oriented for maximum accessible area)	387 mm/637 mm/1137 mm (see Note)
	dual arm (two Universal Z axes oriented for maximum shared area)	365 mm/615 mm/1115 mm (see Note)
	dual arm (two Dual Z axes oriented for maximum shared area)	335 mm/585 mm/1085 mm (see Note)

Note: *In a dual arm configuration, some travel length is lost in the X direction due to the presence of the second arm.*

Note: *A Dual Z may protrude past the ends of the X axis extrusion depending on the orientation of the arm and the travel position. Any protrusion is always above the X carriage.*

Table C-2 *Y axis dimensions*

Maximum overall depth	Standard Z axis (including front and rear cover plates)	405 mm/555 mm
	Standard Z axis (including rear cover plate and front molded cover)	416 mm/566 mm
	Dual Z axis (unaffected by covers)	416 mm/566 mm
	Universal Z axis (unaffected by cover plate)	435 mm/585 mm
Nominal initial position	Standard Z axis and Dual Z axis	39.1 mm
	Universal Z axis (center plane of universal Z mount)	53.5 mm (rear-most probe of 8-channel head in landscape position will be at 39.1 mm; 66.6 mm for ADP probe)
Travel length	any Z axis	150 mm/300 mm
Dual Z spacing	axis center to center spacing	9 mm/18 mm

Note: *For systems with large/wide flex chains on the X axis, the overall Y depth will increase by 30 mm.*

Table C-3 *Z axis dimensions*

Maximum overall height	Standard Z axis and Dual Z axis	655 mm
	Universal Z axis (excludes cabling and tubing)	490 mm
Nominal initial position	Standard Z axis and Dual Z axis (with standard probe)	1.75 mm (surface that contacts the probe collar with probe installed)
	Standard Z axis and Dual Z axis (with DiTi)	41.2 mm (bottom surface of DiTi cone)
	Universal Z axis	5.8 mm
	Universal Z axis with 8-channel head (rear-most probe of 8-channel head in landscape position)	1.75 mm (surface at bottom of counter bore that contacts the probe collar)
	Universal Z with ADP	83.9 mm (bottom surface of the ADP probe DiTi cone)
Travel length	All Z axes	210 mm

C.3 8-Channel Pipetting Head

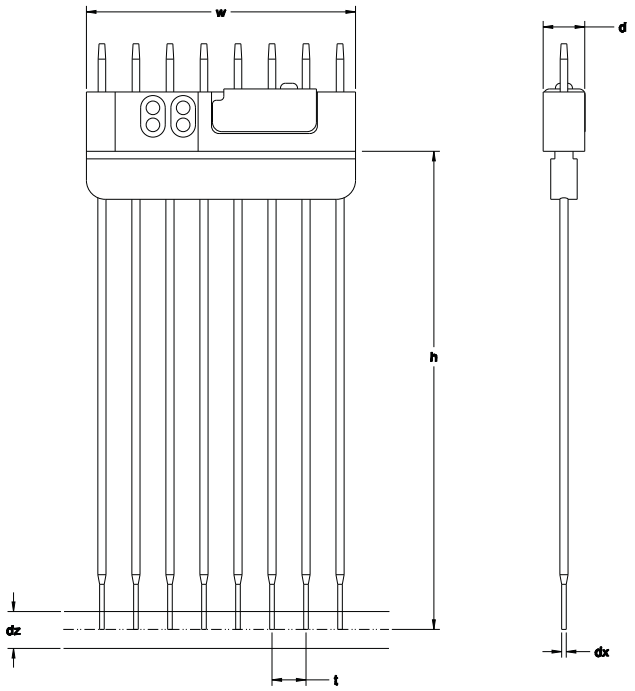


Figure C-13 *8-channel pipetting head on Universal Z axis*

Table C-4 8-Channel pipetting head specifications

Maximum overall width (w)	72 mm
Maximum overall depth excluding sheet-metal cover (d)	13 mm
Maximum tip height from end effector mounting surface (h)	127.5 mm
Minimum exposed probe length (l)	112 mm

Note: *Overall dimensions do not include routing of liquid detection coax cable or liquid signal PCBA ground wire.*

Table C-5 8-Channel head tip relative positions

Distance from tip c to c (t) (measured at tubing connection)	9 mm +/- 0.25 mm
Z height variation (d_z) (measured at the tip)	≤0.37 mm

Note: *Bent probes can affect tip height variation measurements. Probes must all be straight and parallel for specifications to apply.*

Note: *Variation measured on all 8 probe tips.*

C.4 Cavro Omni Hub

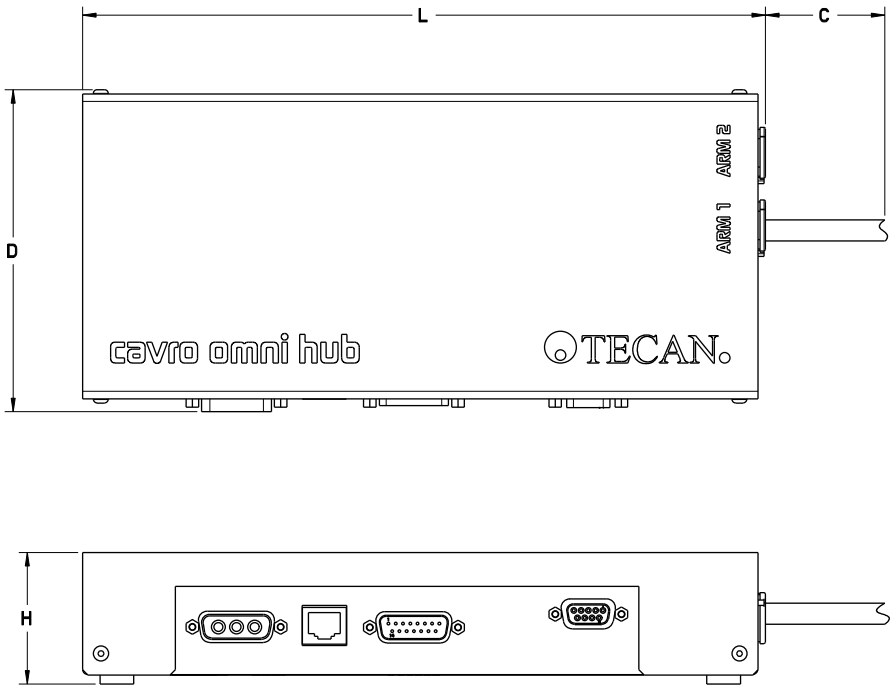


Figure C-14 *Cavro Omni Hub*

Table C-6 Cavro Omni Hub size specifications

Maximum length (L)	260 mm
Maximum overall depth (D)	125 mm
Maximum overall height (H)	53 mm
Minimum cable length (C)	750 mm

C.5 Dual Z Axis

The Omni Dual Z axis module provides two Z racks (in a single Z axis module) spaced either 9 mm or 18 mm center-to-center in the Y direction (as viewed from the front of a complete XYZ robot) parallel to each other and each capable of independent vertical motion.

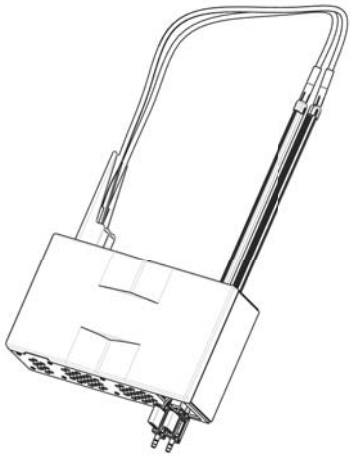


Figure C-15 *Dual Z axis module*

C.5.1 Simultaneous Access of MTP Wells

Simultaneous access refers to movement of both probes or DiTis mounted on a Dual Z axis that results in both axes descending into the wells of an MTP at the same time. This section describes restrictions on simultaneous access with various kinds of microtiter plates (MTPs).

1536-Well MTP

Either Z axis can access a 1536-well MTP individually, but probes must be individually calibrated to the plate.

Caution! *Attempting simultaneous access of wells in a 1536-well MTP is not advisable even when using low volume probes. Due to manufacturing and assembly tolerances, there is insufficient clearance for the second probe to simultaneously access a well.*

384-Well MTP

Simultaneous access of wells in a 384-well MTP with standard probes is possible when each 384-well MTP is calibrated to one of the probes. Either Z axis can access a 384-well MTP individually, but probes must be individually calibrated to the worktable.

Caution! *It is not advisable to attempt simultaneous access of wells in a 384-well MTP if the robot calibration point is external to the plate. Due to manufacturing and assembly tolerances, there is insufficient clearance for the second probe to simultaneously access a well.*

96-Well MTP

Simultaneous access of wells in a 96-well MTP with standard probes is possible without calibrating each plate individually.

96-Well MTP with a 1mL DiTi

Simultaneous access of wells in a standard 96-well MTP with a 1mL DiTi is possible without calibrating each plate individually.

Caution! *Use of V-bottom MTPs may result in slight contact between the DiTi and the well wall due to the smaller well diameter than flat and U-bottom plates.*

Deep Well 96-Well MTP with 1mL DiTi

Caution! *Simultaneous access of wells in a deep-well 96-well MTP with a 1mL DiTi is not advisable. There is insufficient clearance between the well and the DiTi to simultaneously access both wells without contacting the walls.*

C.5.2 Tip Adapter Installation on a Dual Z

Install the tip adapter on a Dual Z oriented so that the PCBA housing points in the direction of the molded cover. It is important to maintain this installation orientation if tip adapters are removed or replaced in order to minimize alignment variation between tips. Manufacturing of the tip adapter assembly sometimes imparts a slight angle to the component in a consistent direction which translates to the tip.

Caution! *Installing the tip adapters in other orientations may affect the ability of the Omni robot to simultaneously access wells of a 384-well MTP.*

C.5.3 cLLD Sensitivity Settings

Caution! *The cLLD sensitivity setting value should be >=63. Lower setting values could lead to interference between the two Zs in a Dual Z module depending on the application, tip spacing, and tip type used. This may require integration testing to verify the required performance. For more information about cLLD sensitivity settings, see* *Section C.10, "Preventing Interference Between Two Robots" on page C-38* *and* *Section C.11, "Preventing Z Axis cLLD Interference in Robots with Multiple Z Axes" on page C-39**.*

C.6 Performance Specifications

The performance specifications in the following tables were calculated for a robot with three axes (X, Y, and Z) operating at ambient 20-25°C with axes mounted such that the tip of the Z axis was pointing down.

WARNING! *If the OEM system causes the Omni to exceed these performance specifications, you may see unpredictable results or behavior.*

Note: *For best accuracy and repeatability, a short pause before lowering the Z axis to the target may be helpful when approaching small targets with a Universal Z, to ensure that the mechanical system settles after X or Y movements.*

Note: *The radiated immunity tests were performed with the default settings for liquid detection.*

C.6.1 Payload

Table C-7 shows the maximum payload of the Omni axes. The maximum payload on the X axis is the total moment load based on a 3.4 kg distributed load of a Y axis with a 300 mm travel length, plus an additional load of 3.2 kg placed on the Y axis carriage at a Y-travel of 300 mm.

Note: *An offset or cantilevered payload can affect performance such as smoothness of operation or life of axis. Smoothness of operation can be optimized by changing speed and/or acceleration parameters.*

Table C-7 Payload

Max load on X axis	Less than or equal to 6.6 kg @ 230 mm from the center of the X rails in the Y direction
Max load on Y axis	Less than or equal to 3.2 kg @ 54 mm from the surface of the Y slide (vertical position, standard Y configuration)
Max load on Standard and Dual Z axes	Capable of using tip options provided by Tecan; do not use any other tip option
Max load on Universal Z axis	Less than or equal to 1.5 kg

C.6.2 Speed and Acceleration

Table C-8 shows the operating speed of the Omni Robot, determined using a full three-axis robot with a standard 300 mm travel Y axis and a 210 mm travel Z axis mounted on an X axis.

Table C-8 Speed and acceleration ranges

Axis	Speed	Acceleration
X axis	0.005-0.8 m/s	0.2-3.0 m/s^2
Y axis	0.005-0.6 m/s	0.2-3.0 m/s^2
Z axis (Standard or Dual)	0.0005-0.6 m/s	0.2-6.0 m/s^2
Universal Z axis	0.0005-0.6 m/s	0.2-2.0 m/s^2

Note: *Higher payloads require slower X and Y axis accelerations and speeds. The mass limit for the Y axis is 3.2 kg. The maximum suggested speed is 0.30 m/s and the maximum suggested acceleration is 1.0 m/s^2. This may require further adjustment depending on the distribution of the mass in the payload.*

Note: *At higher accelerations, there may be a discrepancy between the actual and commanded acceleration due to the positioning algorithm.*

C.6.3 Accuracy

Table C-9 shows the accuracy along each axis of the Omni Robot over the entire travel range for individual X or Y axes. The measurement for the X axis was taken at the surface of the X axis carriage, centered between the rails. The measurement for the Y axis was taken at the surface of the Y axis carriage, centered on the rail.

Note: *Accuracy values are bidirectional.*

Table C-9 XY single axis accuracy

X axis	±0.2 mm
Y axis	±0.2 mm

Table C-10 shows the accuracy along each axis of the Omni Robot over the entire travel range for three-axis systems. The measurements for the X, Y, and Z axes were taken at the end of a standard probe. The measurement for the Universal Z axis was taken from the end effector mounting surface.

Table C-10 XYZ single axis accuracy

X axis	±0.3 mm (up to 1 meter) ±0.4 mm (up to 1.25 meters)
Y axis	±0.3 mm (up to 0.3 meters)
Z axis (Standard or Dual)	±0.4 mm (up to 0.21 meters)
Universal Z axis	±0.25 mm (up to 0.21 meters)

C.6.4 Repeatability

Table C-11 shows the repeatability along each axis of the Omni Robot over the entire travel range for individual X or Y axes. The measurement for the X axis was taken at the surface of the X axis carriage, centered between the rails. The measurement for the Y axis was taken at the surface of the Y axis carriage, centered on the rail.

Note: *Repeatability values are bidirectional.*

Table C-11 XY single axis repeatability

X axis	less than or equal to 0.12 mm
Y axis	less than or equal to 0.12 mm

Table C-12 shows the repeatability along each axis of the Omni Robot over the entire travel range for three-axis systems. The measurements for the X, Y, and Z axes were taken at the end of a standard probe. The measurement for the Universal Z axis was taken from the end effector mounting surface.

Table C-12 XYZ single axis repeatability

X axis	less than or equal to 0.2 mm
Y axis	less than or equal to 0.2 mm
Z axis (Standard or Dual)	less than or equal to 0.4 mm
Universal Z axis	less than or equal to 0.15 mm

Table C-13 Robot life specifications, distance traveled

Omni Configuration	Travel Distance	Full Travel Move Equivalents
Omni with 1.5 kg payload	500 km	1,000,000 X moves of 500 mm, or 3,333,000 Y moves of 150 mm, or 2,381,000 Z moves of 210 mm
Omni without payload	2500 km	5,000,000 X moves of 500 mm, or 16,667,000 Y moves of 150 mm, or 11,905,000 Z moves of 210 mm

C.7 Capacitive Liquid Level Detection

50 µl of test solution (saline: 0.05% NaCl) can be detected in a 96-well flat-bottom microplate, e.g., Nunc 269787 or NWR 62409-120.

C.8 Electrical Specifications

Figure C-16 shows an electrical diagram of the Cavro® Omni Robot.

Figure C-17 shows an electrical diagram of the XYZ axis group with Standard Z axis.

Figure C-18 shows an electrical diagram of the XYZ axis group with the Universal Z axis and an 8-channel head.

Figure C-19 shows an electrical diagram of the XYZ axis group with the Universal Z axis and Gripper.

Figure C-20 shows an electrical diagram of the XYZ axis group with the Universal Z axis and ADP.

Figure C-21 shows an electrical diagram of the XYZ axis group with the Dual Z axis.

Figure C-22 shows an electrical diagram of an Omni axis group without a Z axis.

Note: *this is a general example and multiple variations of axis group are possible with the axis termination board. The CIB could be located in an X axis or in a Cavro® Omni Hub adding to the possibilities but the general connection flow remains the same.*

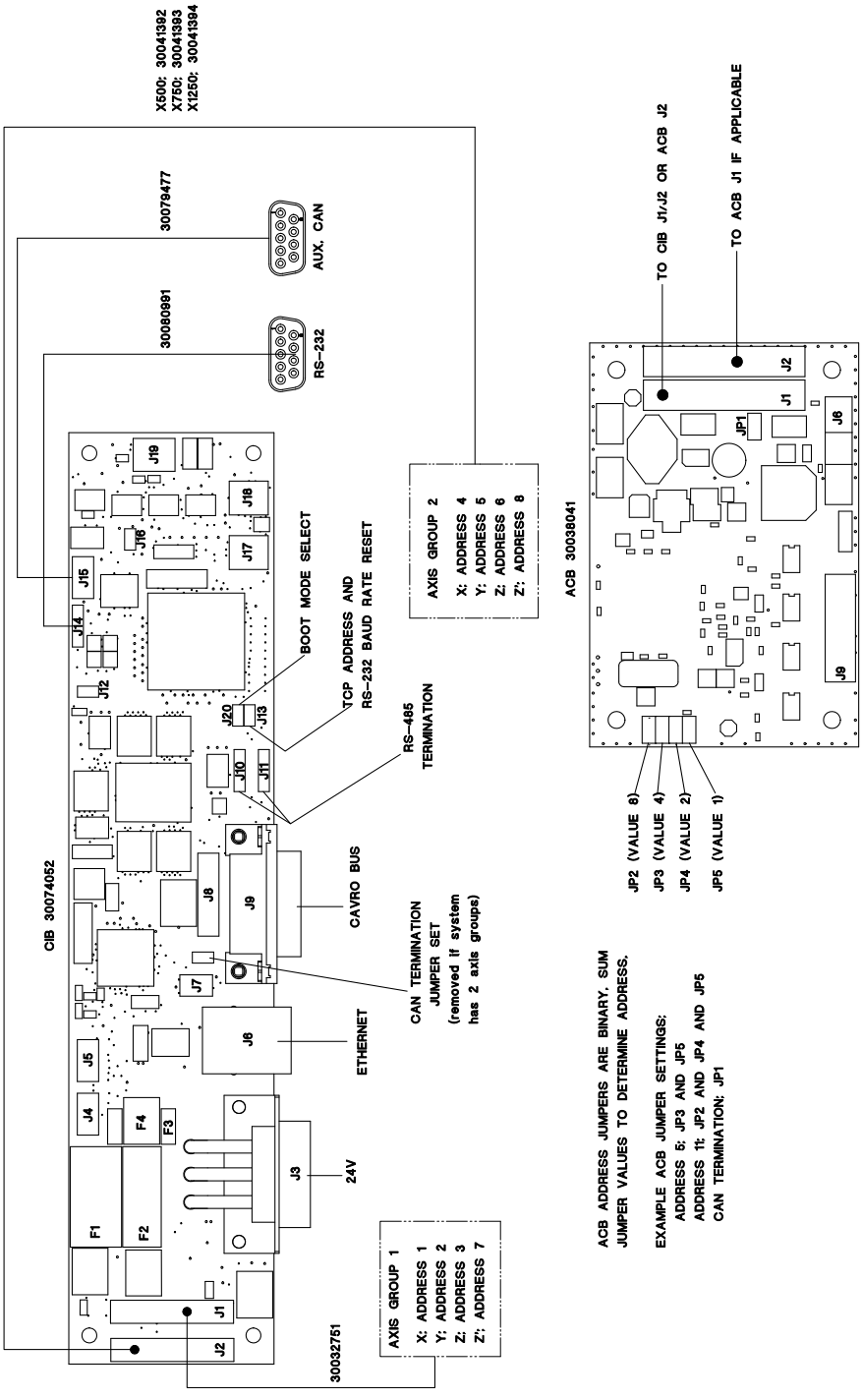


Figure C-16 *Omni Robot electrical diagram*

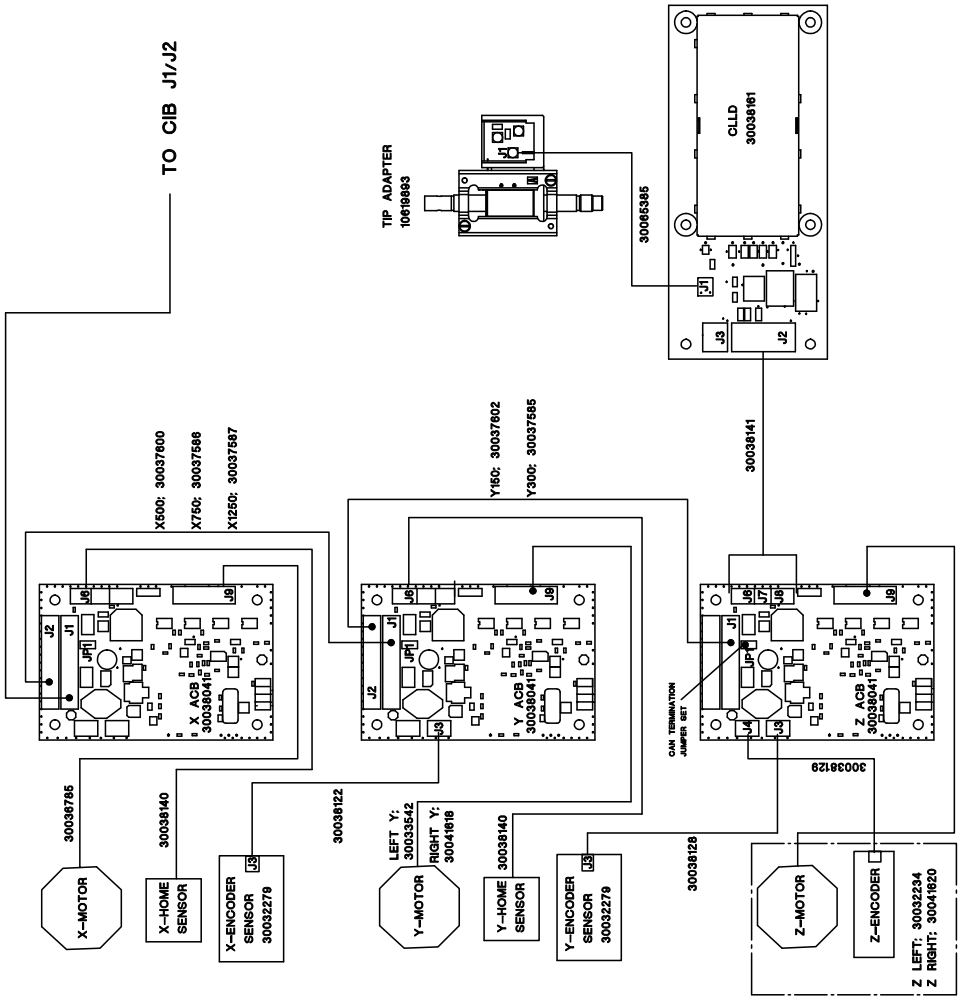


Figure C-17 *XYZ axis group with Standard Z axis*

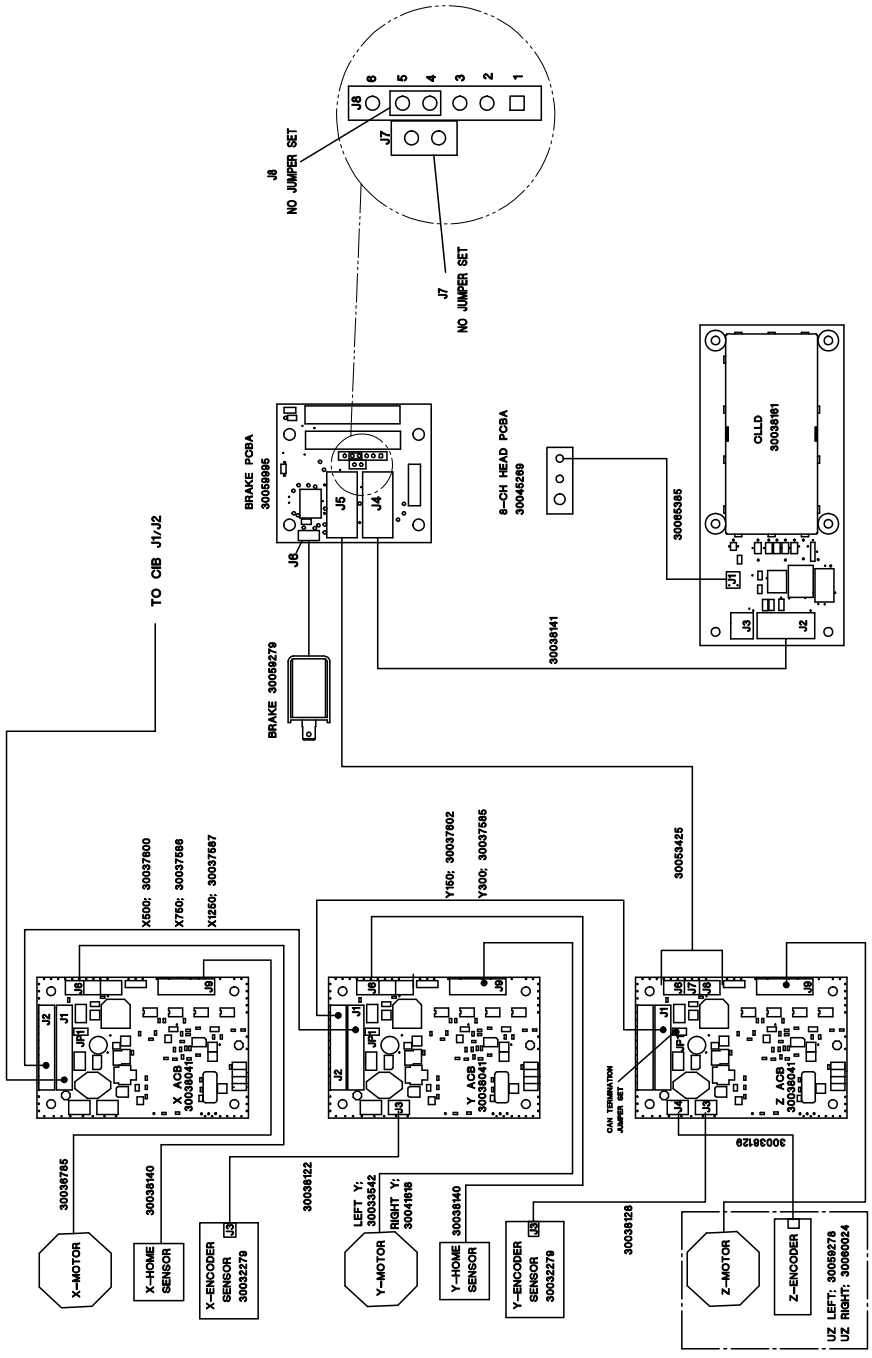


Figure C-18 *XYZ axis group with Universal Z axis and 8-channel head*

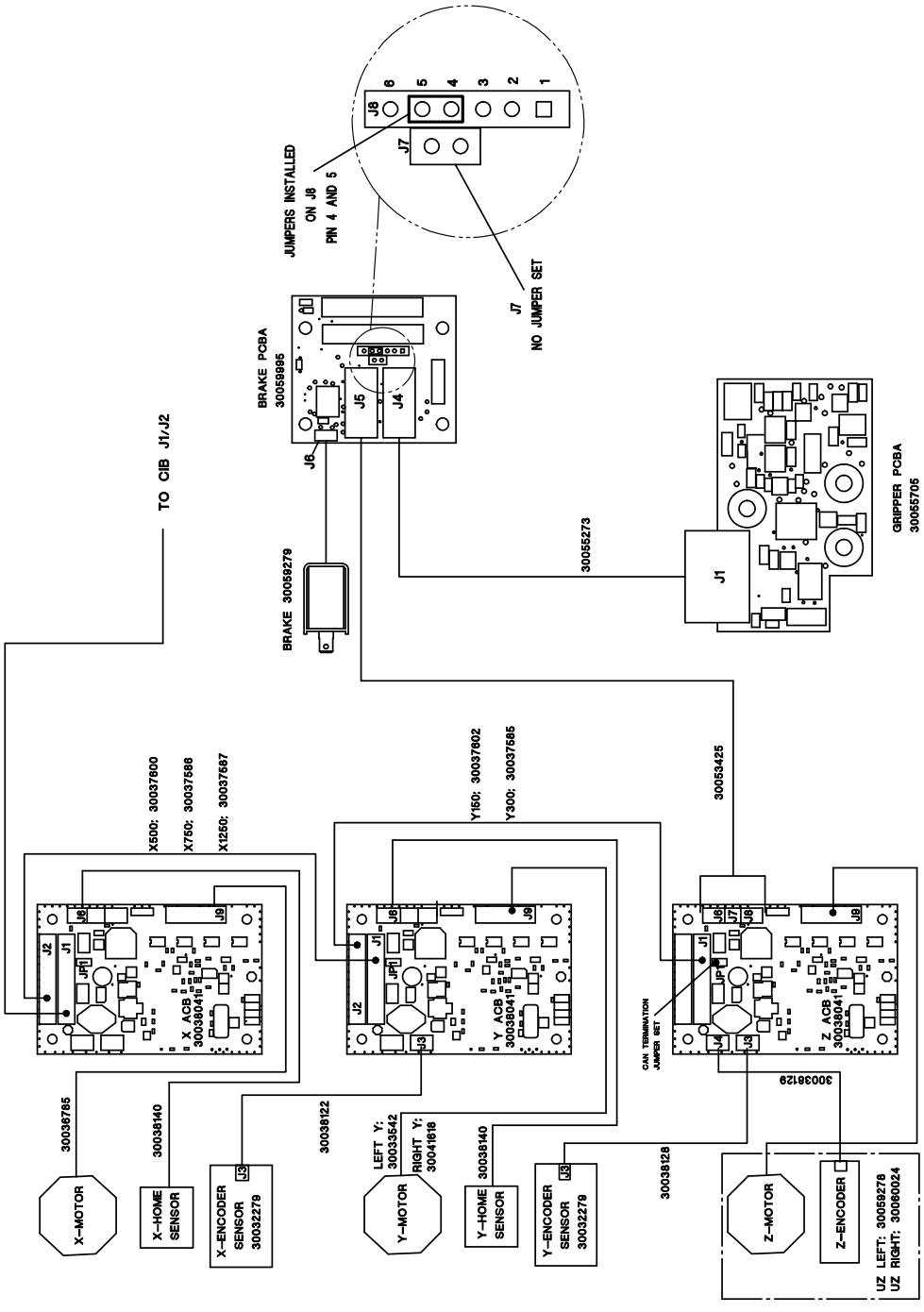


Figure C-19 *XYZ axis group with Universal Z axis and Gripper*

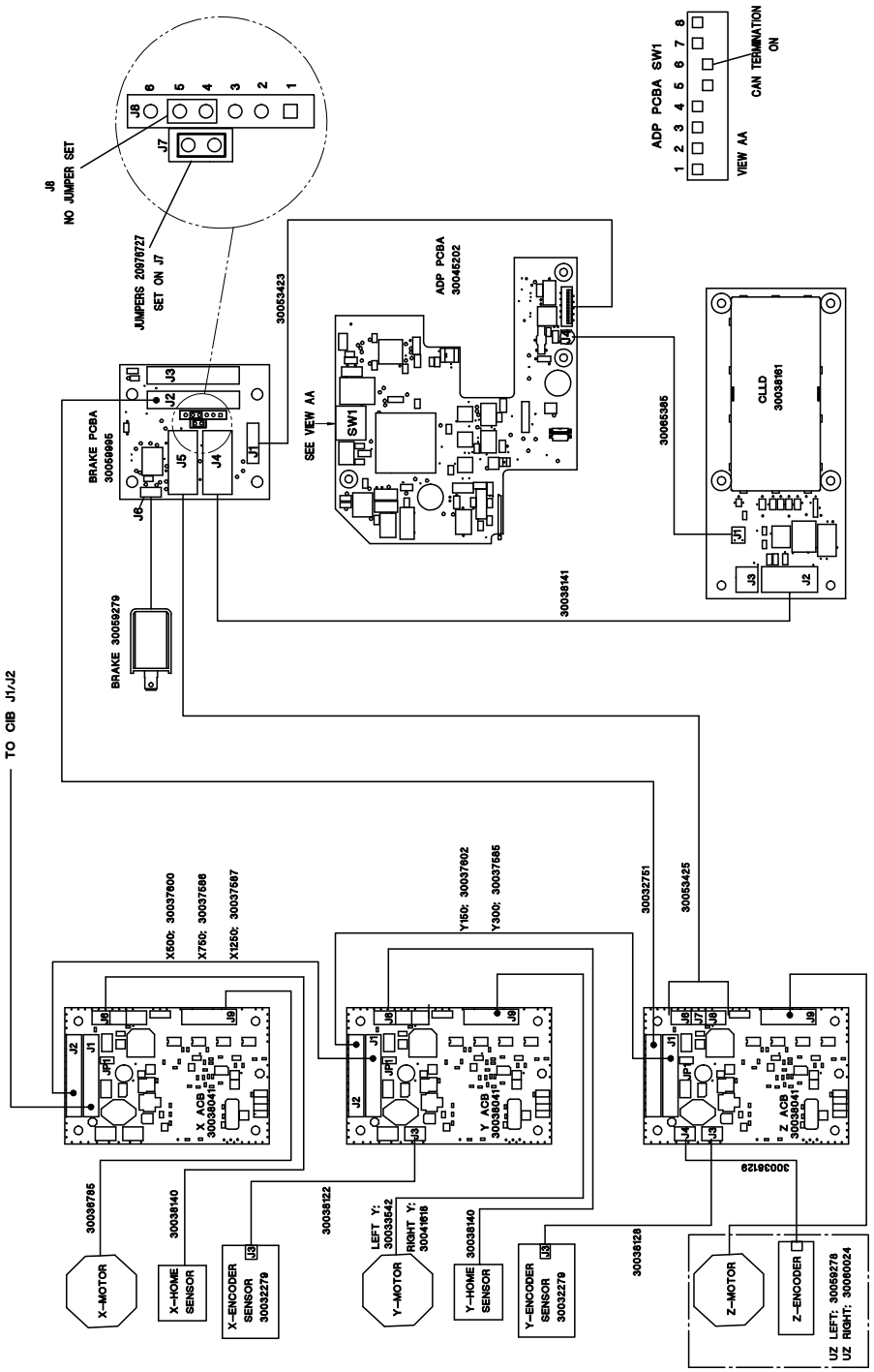


Figure C-20 *XYZ axis group with Universal Z axis with ADP*

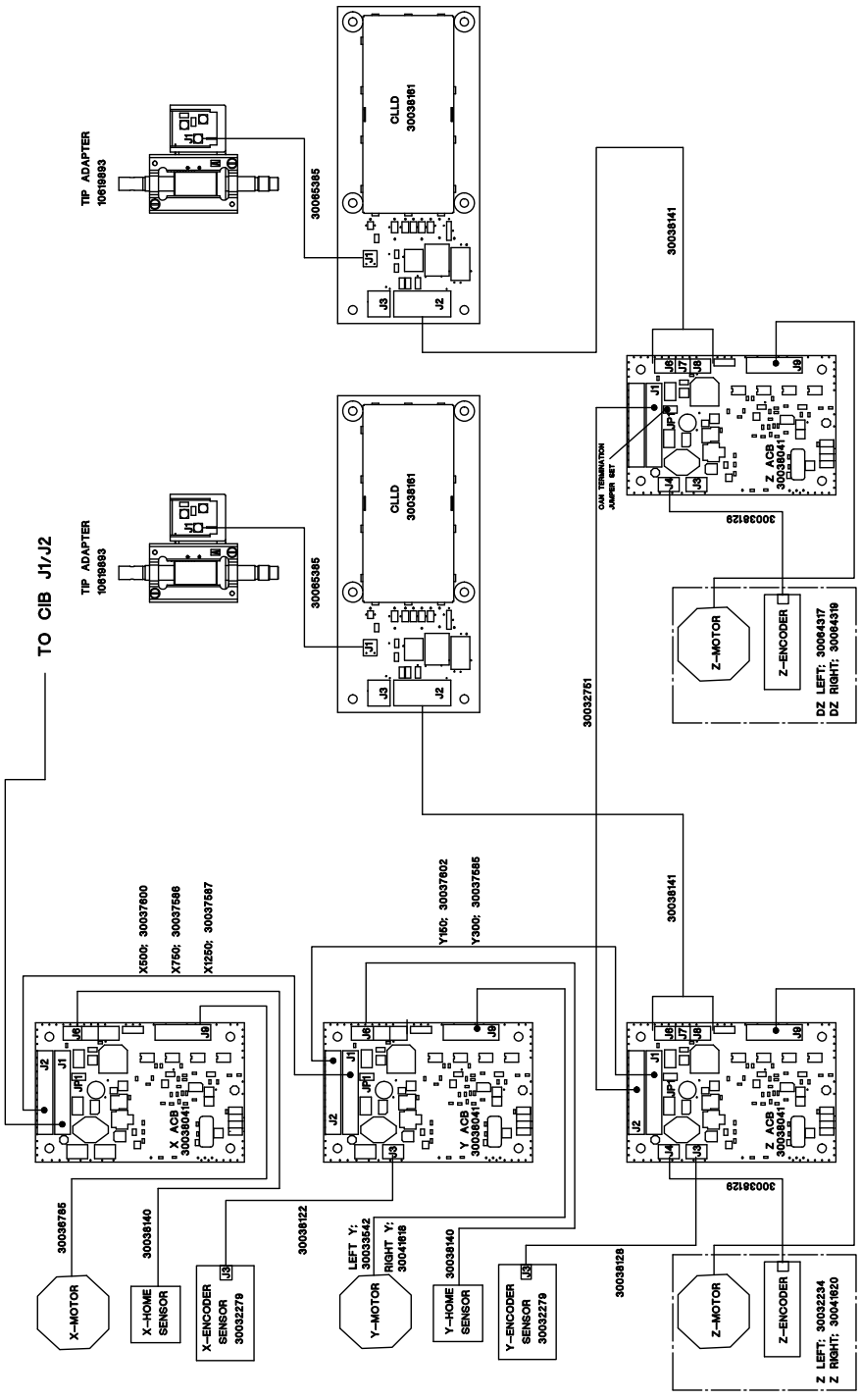


Figure C-21 *XYZ axis group with Dual Z axis*

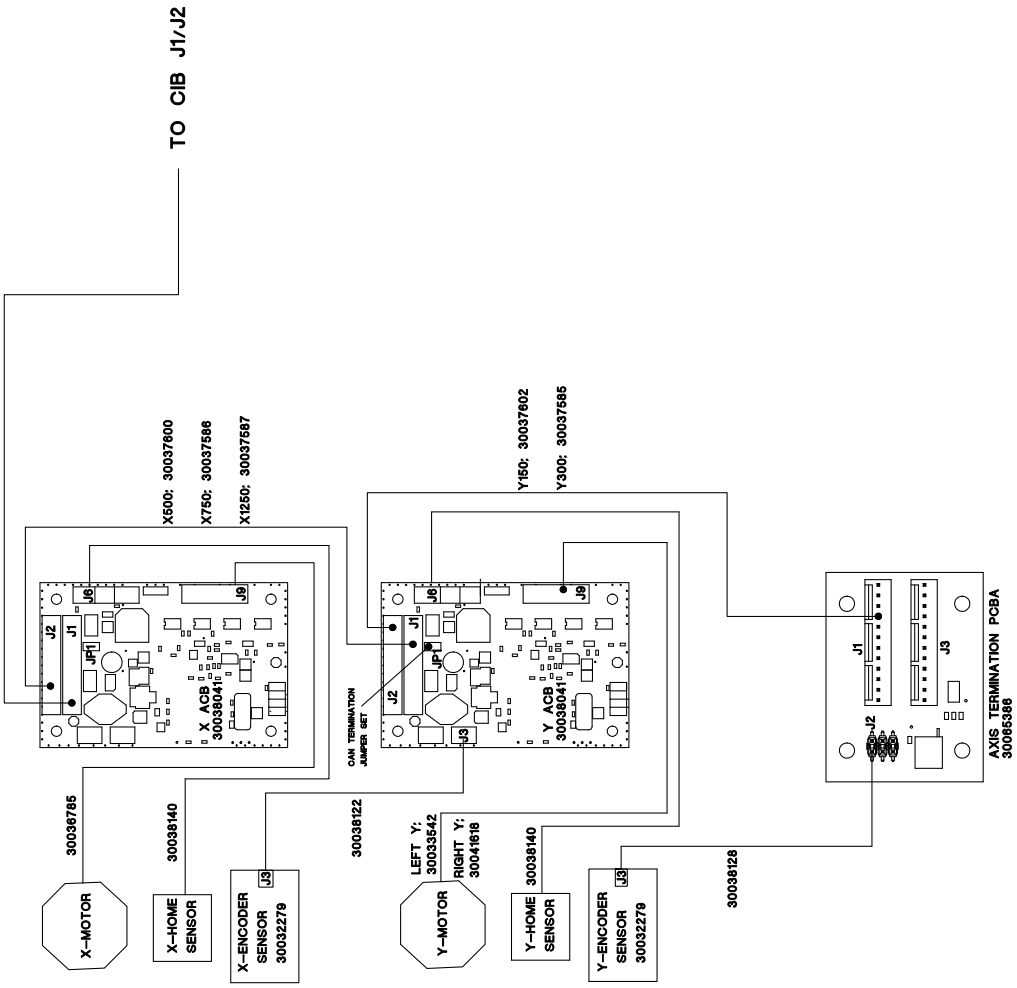


Figure C-22 *Axis group without a Z axis*

Table C-14 lists the electrical specifications of the Omni Robot.

Table C-14 Electrical specifications

Operating voltage at CIB power input	24 (+10/-2)% VDC
Ripple voltage	Less than or equal to 0.5V peak to peak @ 100 Hz typical
Fuse rating	T4A 5X20mm
Peak current	16A
Note that 16A is the maximum rated current of the CIB. The actual current drawn by your Omni Robot will vary based on the configuration. A robot configured with a single X, single Y, and single Z axis will draw approximately 3A.	

Table C-15 lists the manufacturers and part numbers for the power supply components for assembling mating receptacles recommended by Tecan.

Table C-15 Recommended power supply connector components

Connector	ITT CANNON DAM3W3SA197 or DAM3W3SK126
Contact Socket	ITT CANNON DM53744-6 or DM53744-25
Housing	TYCO ELECTRONICS AMP 5748676-2

Table C-16 lists the manufacturers and part numbers for the DA15 connector recommended by Tecan.

Table C-16 Recommended DA15 connector components

Connector	NORCOMP 171-015-103L031
Housing	TYCO ELECTRONICS AMP 5748676-2

Table C-17 lists the electrical specifications for the 8-channel pipetting head probe with the cLLD coax cable disconnected.

Table C-17 8-Channel head probe electrical specifications

Electrical resistance between probes (measured probe to probe between all 8 probes)	< 10 Ohm
Insulation resistance between probes and chassis, measured between all 8 probes and Z block (which represents the chassis) with cLLD coax cable disconnected (connecting the coax creates a circuit to ground via the cLLD board).	> 10 MOhm

Note: *Probes are electrically connected. Electrical resistance measured at 0.3 to 1.0 VDC, 20 +/- 2°C ambient and < 70% RH.*

C.9 Connector Pin Out Diagrams

Figure C-23 contains the pin out diagram of the power supply port.

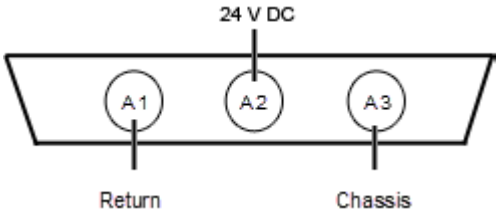


Figure C-23 *Power supply port pin out*

Figure C-24 contains the pin out diagram of the DA15 port of the CIB and two additional Cavro® modules.

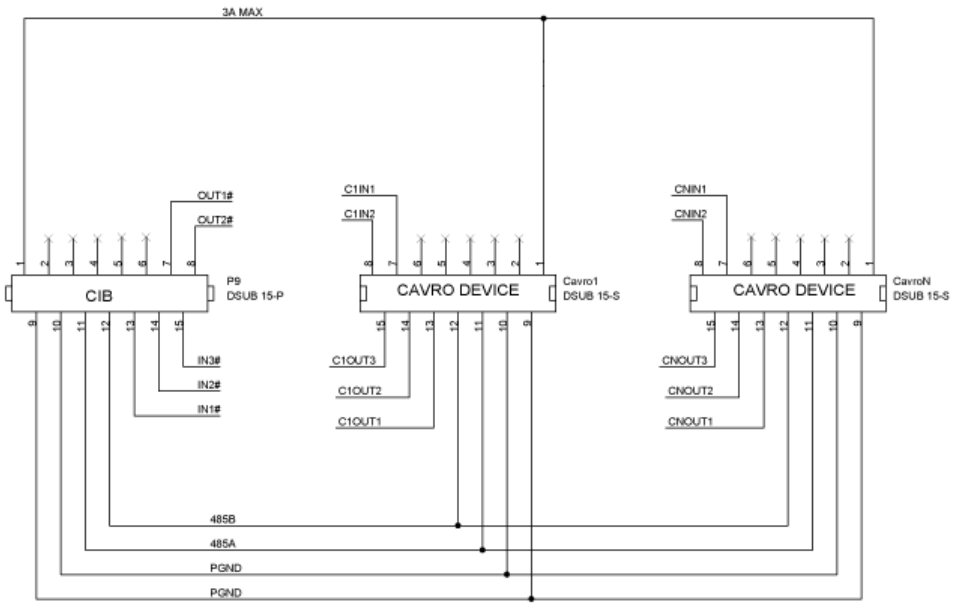


Figure C-24 *DA15 port pin out*

Caution! *Any cable that is connected to the DA15 port must not use pins 5 (CANH) and 6 (CANL); these pins are reserved for the Cavro® CAN bus. Use of these pins for another purpose could disrupt communication with the robot.*

Table C-18 contains the pin out information for the Ethernet port.

Table C-18 Ethernet port pin out information

Pin Number	Data
1	Transmit +
2	Transmit -
3	Receive +
4	N/A
5	N/A
6	Receive -
7	N/A
8	N/A

Table C-19 CAN connections (Reserved for future use)

Pin Number	Data
1	N/A
2	CANL
3	N/A
4	N/A
5	N/A
6	GND
7	CANH
8	N/A
9	N/A

Table C-20 RS-232 connections

Pin Number	Data
1	N/A
2	Transmit
3	Receive
4	N/A
5	GND
6	N/A
7	N/A
8	N/A
9	N/A

C.9.1 Internal CIB Connector Pin Out Diagrams

The internal connectors can be accessed only when the CIB is removed from the system.

WARNING! *When the CIB is removed, there is a safety risk due to high amperage exposure. Always remove power before removing the CIB.*

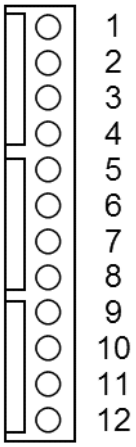
WARNING! *Static electricity can damage the CIB if proper precautions are not taken.*

Contact Tecan Systems for more information on how to use the internal connectors.

Note: *See the diagram in Figure C-16, "Omni Robot electrical diagram" on page C-23 for the locations of these connectors.*

Note: *All inputs and outputs are maximum 5V logic level unless noted otherwise. See Table C-21, CIB board input and output specifications.*

Figure C-25 contains the pin out diagram of the CIB J1 and J2 connectors.



Pin #	Function	Pin #	Function
1	Chassis (connects to the ACB and CIB enclosure)	7	CAN High
2	24V	8	CAN Low
3	24V return	9	24V return
4	24V	10	No connection
5	24V return	11	No connection
6	24V	12	No connection

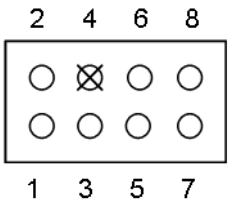
Figure C-25 *CIB J1 and J2 pin outs*

- 24V RETURN is connected to CIB J3-A1 and connected to J3-A3 through solder bridge SB2
- 24V is connected to CIB J3-A2 through fuse on CIB
- Earth ground should be connected to CIB J3-A3 and is connected to the chassis through solder bridge SB1 and to the ACB mounting holes.

Note: *The chassis wire is connected to the CIB and ACB mounting holes.*

For more information about grounding the Omni Robot, see Section 4.4, "Grounding".

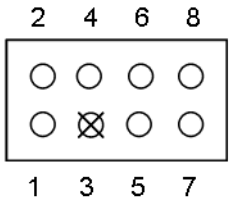
Figure C-26 contains the pin out diagram of the J4 connector.



Pin #	Function	Pin #	Function
1	VPP / 24VDC	5	I/O 1
2	VPP / 24VDC	6	Input 4
3	VCC / 5VDC	7	GND
4	key	8	GND

Figure C-26 *J4 pin out*

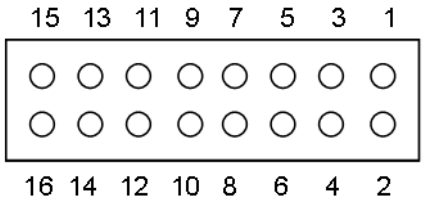
Figure C-27 contains the pin out diagram of the J5 connector.



Pin #	Function	Pin #	Function
1	VPP / 24VDC	5	I/O 2
2	VPP / 24VDC	6	Input 5
3	key	7	GND
4	VCC/5VDC	8	GND

Figure C-27 *J5 pin out*

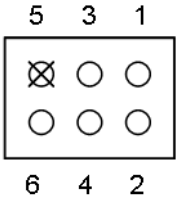
Figure C-28 contains the pin out diagram of the J8 connector.



Pin #	Function	Pin #	Function
1	VPP / 24VDC	9	CAN High
2	GND	10	Input 1
3	nc	11	CAN Low
4	GND	12	Input 2
5	nc	13	Output 1
6	RS-485A	14	Input 3
7	nc	15	Output 2
8	RS-485B	16	nc

Figure C-28 *J8 pin out*

Figure C-29 contains the pin out diagram of the J17 connector.

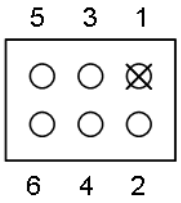


Pin #	Function	Pin #	Function
1	VCC / 5VDC	4	Input 7 / INT5*
2	VCC / 5VDC	5	key
3	Output 3	6	GND

* *Low to high transition will trigger event ID INT5. See Table 7-11, "EventID values," on page 7-89.*

Figure C-29 *J17 pin out*

Figure C-30 contains the pin out diagram of the J18 connector.



Pin #	Function	Pin #	Function
1	key	4	VPP / 24VDC
2	VPP / 24VDC	5	Input 6
3	Output 4*	6	GND

* *Open drain power output, 0.8A maximum at 24VDC.*

Figure C-30 *J18 pin out*

Table C-21 lists the input and output specifications for the CIB board.

Table C-21 CIB board input and output specifications

Minimum input high level voltage	3.7V
Maximum input low level voltage	1.0V
Digital output min. high level voltage (exclude I/O1&2)	3.7V @ 4mA
Digital output max. low level voltage (exclude I/O1&2)	0.4V @ 4mA
Digital output min. high level voltage (I/O1&2)	>2.8V @ 1.5mA (Note)
Digital output max. low level voltage (I/O1&2)	<0.48V@1.5mA

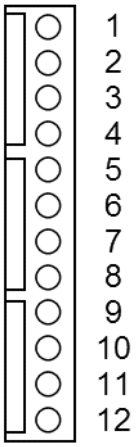
Note: *All inputs and I/Os have pull-up resistors. The I/Os can be configured by firmware either as input or output. The pull-up resistors of the I/Os will only be effective when configured as input.*

C.9.2 Internal ATB Connector Pin Out Diagram

WARNING! *Static electricity can damage the ATB if proper precautions are not taken.*

- J1 - axis in - standard axis cable connection to Axis Control Board (ACB)
- J2 - encoder in - standard encoder cable for remote ACB
- J3 - spare - future expansion (24V = 24VDC)

Figure C-31 contains the pin out diagram of the J3 ATB connector.



Pin #	Function	Pin #	Function
1	Chassis (connects to the ACB and CIB enclosure)	7	CAN High
2	24V	8	CAN Low
3	24V return	9	24V return
4	24V	10	No connection
5	24V return	11	No connection
6	24V	12	No connection

Figure C-31 *J3 ATB pin out*

- 24V RETURN is connected to CIB J3-A1 and connected to J3-A3 through solder bridge SB2,
- 24V is connected to CIB J3-A2 through fuse on CIB
- Earth ground should be connected to CIB J3-A3 and is connected to the chassis through solder bridge SB1 and to the ACB mounting holes.

Note: *The chassis wire is connected to the ATB and ACB mounting holes.*

For more information about grounding the Omni Robot, see Section 4.4, ”Grounding”.

C.10 Preventing Interference Between Two Robots

If an Omni Robot is operating in close proximity to another Omni Robot, the two cLLD modules could interfere with one another because they are operating at the same frequency. This could lead to false liquid detection.

You can prevent interference between two cLLD modules in the following ways:

- Maintain a minimum tip-to-tip distance between the two robots.

  Table C-22 lists the minimum distance to be maintained tip-to-tip between two Omni Robot modules operating within close proximity to one another.

  The minimum tip-to-tip distance may vary depending on system liquid properties.

Table C-22 Minimum distance to maintain between two Omni Robot modules

cLLD Sensitivity	Minimum Distance Without Shield		
	Standard Z	Dual Z	Universal Z with 8-channel head
≥ 15	600 mm	-	-
≥ 32	-	-	650 mm
≥ 63	300 mm	300 mm	450 mm

- Place a grounded piece of sheet metal between the two adjacent robots, to act as a shield and prevent interference.

  The shield must cover the entire travel length of the Y axis and Z axis, as well as the tip length (300 mm x 300 mm minimum).

For more information, see Section 4.6, ”Operating Multiple Omni Robots” on page 4-5.

C.11 Preventing Z Axis cLLD Interference in Robots with Multiple Z Axes

To avoid interference between multiple cLLD modules on an Omni Robot, the frequency has to be set to a different value for each cLLD module. This is accomplished automatically by assigning a frequency setting (*PWMFrequency*) at start up to each Z axis, based on its hardware address ID.

In addition, there is a minimum setting for the *LLDSensitivity* depending on the Omni configuration. Please refer to Table C-23, Minimum settings for LLDSensitivity for information about the minimum setting for your configuration.

Table C-23 Minimum settings for LLDSensitivity

	Single arm	Dual arm	
		NOT facing toward each other	Facing toward each other
Combinations containing Standard Z and ADP only	15	15	24
At least one Universal Z (8-channel)	32	63	63
At least one Dual Z	63	63	63

Caution! *The appropriate sensitivity needs to be determined for each application. Conductive system liquids can increase EMI susceptibility and interference between OMNI arms significantly. For reference, EMI testing was conducted using a sensitivity setting of 63 on the Standard Z, Dual Z, and Universal Z.*

C.12 Environmental Specifications

Table C-24 lists the environment conditions in which to operate and store the Omni Robot.

Table C-24 Environmental specifications

Operating temperature	10-40°C **Note:** If an ADP is mounted on the Omni, the operating temperature range is only15-35°C, per ADP operating temperature specifications.
Storage temperature	-20-70°C
Operating humidity	30-80% RH, non condensing

C.13 Chemical Resistance Information

The following table shows the chemical resistance of some commonly used materials. The table is not intended to be comprehensive; the OEM user is responsible for compiling the complete chemical resistance information for his or her particular OEM system.

Legend:

o	Resistant
x	Partly resistant; use is possible with frequent replacements
/	Not resistant; unsuitable for use
nd	Not determined

Table C-25 Chemical resistance table

Material	FEP	PVC	Silicone	POM	PP	PTFE	PCTFE (Kel-F)
Acetone	o	/	o	x	o	o	o
Acetonitrile (C_2H_3N)	o	/	/	/	o	nd	nd
Formic acid 100%	o	x	x	/	o	o	o
Ammonium hydroxide 25%	o	x	o	/	o	o	o
Chloroform	o	/	/	x	x	o	x
Dimethylformamide	o	/	/	/	o	o	o
DMSO	o	/	x	o	o	nd	nd
Acetic acid 96%	o	/	x	/	x	o	o
Acetic acid ethylester	o	/	/	x	x	nd	nd
Ethanol 96%	o	x	x	x	o	o	o
Formaldehyde 40%	o	x	/	/	o	o	o
Sulfuric acid 96%	o	/	/	/	x	o	o
Isopropanol	o	/	x	o	o	o	o
Diluted bleach, NaOCl	o	x	x	/	x	o	o
Methanol	o	x	o	x	o	o	o
Methylene chloride	o	/	/	x	/	o	o
Sodium hydroxide 10M	o	x	o	/	o	nd	nd

Table C-25 Chemical resistance table (continued)

Material	FEP	PVC	Silicone	POM	PP	PTFE	PCTFE (Kel-F)
Perchloric acid 60%	o	/	/	x	x	o	x
Petroleum ether 30/50	o	x	/	x	/	nd	nd
Hydrochloric acid 32%	o	x	/	/	o	o	o
Trichloroacetic acid 40%	o	/	/	o	/	o	o

C.14 Default IP Address and Port Number

The following table shows default settings that the OEM user will likely change.

Table C-26 Default IP address and port number, Command Processor mode

Default IP address	192.168.0.198
Default port number	9898

Table C-27 Default IP address, port number, and baud rate, Embedded mode

Default IP address	192.168.0.198
Default port number	9897
Default RS-232 baud rate	38400,n,8,1

D Abbreviations and Symbols

Table D-1 provides a list of abbreviations used in this manual. Table D-2 provides a list of symbols used in this manual.

Table D-1 *List of abbreviations*

Abbreviation	Meaning	Abbreviation	Meaning
A	Ampere (Amp)	kg	Kilogram
AC	Alternating Current	lbs	Pounds
ACB	Axis Control Board	m	Meter
acc.	Acceleration	mm	Millimeter
ATB	Axis Termination Board	N	Newton
CAN	Controller Area Network	NVMEM	Non-volatile memory
CIB	Communication Interface Board	OEM	Original Equipment Manufacturer
cLLD	Capacitive liquid level detection	PCBA	Printed Circuit Board Assembly
cm	Centimeter	PLC	Programmable Logic Controller
DC	Direct Current	RS-232	Single-ended full duplex interface
DIP	Dual inline package	RS-485	Differential-pair networked interface
DiTi	Disposable tips	s	Second
ef	End Frequency	sf	Start frequency
ft	Foot	TCP/IP	Transmission Control Protocol / Internet Protocol
Hz	Hertz	V	Volt
in	Inch	W	Watt

Table D-2 *List of symbols*

Symbol	Meaning	Symbol	Meaning
≈	Approximately equal to	≠	Not equal to
°C	Degrees Celsius	Ω	Ohm(s)
°F	Degrees Fahrenheit	±	Plus or minus

E Command Processor Command and Error Quick Reference Guide

E.1 Get and Set Commands

GET (“read”) commands report the specified values stored in the Command Processor (“

SET (“write”) commands store specified values in the Command Processor.

Commands are organized by device type:

- C - CIB (System commands)
- A - Arm commands
- M - Axis (motor) commands
- D - Dilutor commands

Table E-1 *System Get/Set commands (device type C)*

Command	Brief Description	Value Range	Default Value	Get	Set	See Page
BootVersion	Version of boot code	--	--	**Yes**	No	7-25
CANBaudRate	CAN baud rate	500K/s (fixed)	500K/s	**Yes**	No	7-25
ClearInt5	Re-enable Omni axes to start accepting commands after a stop triggered by Int5.	0	--	**Yes**	No	7-27
EnableCavroCAN	Enable Cavro® CAN communication support for devices.	0	--	**Yes**	No	7-31
FWRevision	FW Part number & revision number	Pre-set	Pre-set	**Yes**	No	7-32
FWVersion	Version of firmware	Pre-set	Pre-set	**Yes**	No	7-33
Input1	Status of input 1	0 or 1	--	**Yes**	No	7-36
Input2	Status of input 2	0 or 1	--	**Yes**	No	7-36
Input3	Status of input 3	0 or 1	--	**Yes**	No	7-36
Input4	Status of input 4	0 or 1	--	**Yes**	No	7-37
Input5	Status of input 5	0 or 1	--	**Yes**	No	7-37

Table E-1 *System Get/Set commands (device type C) (continued)*

Command	Brief Description	Value Range	Default Value	Get	Set	See Page
Input6	Status of input 6	0 or 1	--	**Yes**	No	7-37
Input7	Status of input 7	0 or 1	--	**Yes**	No	7-37
Int5Mode	Enable or disable use of Int5 to trigger axis stop.	0	--	**Yes**	**No**	7-35
IO1	Status of I/O 1 when IO1Mode = 1 (output)	0 (low) or 1 (high)	--	**Yes**	**Yes**	7-38
IO2	Status of I/O 2 when IO2Mode = 1 (output)	0 (low) or 1 (high)	--	**Yes**	**Yes**	7-39
IO1Mode	I/O direction for I/O 1	0 (input) or 1 (output)	0	**Yes**	**Yes**	7-38
IO2Mode	I/O direction for I/O 2	0 (input) or 1 (output)	0	**Yes**	**Yes**	7-39
IPAddress	IP address of CIB	0.0.0.0 to 255.255.255.255	192.168.0.198	**Yes**	**Yes**	7-40
MACAddress	MAC Address of CIB	--	--	**Yes**	No	7-44
Output1	Status of output 1	0 (low) or 1 (high)	0	**Yes**	**Yes**	7-44
Output2	Status of output 2	0 (low) or 1 (high)	0	**Yes**	**Yes**	7-44
Output3	Status of output 1	0 (low) or 1 (high)	0	**Yes**	**Yes**	7-45
Output4	Status of output 1	0 (low impedance to ground) or 1 (high impedance to ground)	0	**Yes**	**Yes**	7-45
PowerOnMinutes	Cumulative number of minutes powered on		Set resets counter to 0	**Yes**	**Yes**	7-47
PowerUps	Cumulative number of power ups		Set resets counter to 0	**Yes**	**Yes**	7-48
RS485Config	Select port settings for RS-485 communications	Mode 0 or 1	0 = 9600 baud, no parity, 8 data bits, 1 stop bit 1 = 38400 baud, no parity, 8 data bits, 1 stop bit	**Yes**	**Yes**	7-52

Table E-1 *System Get/Set commands (device type C) (continued)*

Command	Brief Description	Value Range	Default Value	Get	Set	See Page
SerialNumber	Serial number of the CIB	Pre-set	Pre-set	**Yes**	No	7-52
SWRevision	Part number and revision of Command Processor software	Pre-set	Pre-set	**Yes**	No	7-53
SWVersion	Version number of Command Processor	Pre-set	Pre-set	**Yes**	No	7-53
UserData	User data stored in CIB non-volatile memory	0 to 256 bytes of ASCII data except "/ "or ", "characters	--	**Yes**	**Yes**	7-56
VoltageFU1	Power supply voltage after FU1	Pre-set	Pre-set	**Yes**	No	7-56
VoltageFU2	Power supply voltage after FU2	Pre-set	Pre-set	**Yes**	No	7-57
VoltageFU3	Power supply voltage after FU1	Pre-set	Pre-set	**Yes**	No	7-57
VoltageFU4	Power supply voltage after FU1	Pre-set	Pre-set	**Yes**	No	7-57

Table E-2 *Arm Get/Set commands (device type A)*

Command	Brief Description	Value Range	Default Value	Get	Set	See Page
Axes	ID number of ACBs that comprise the logical arm	0 to 6	--	**Yes**	No	7-24

Table E-3 *Axis Get/Set commands (device type M)*

Command	Brief Description	Value Range	Default Value	Get	Set	See Page
AxisDesc	Axis type, including TML script version number	format aabbcccc	aa = major version bb = minor version cccc = axis type	**Yes**	No	7-24
AxisOffset	Axis home position coordinates	Depends on reference position	Pre-set	**Yes**	No	7-25
CANegativeAxisID	ID of neighboring axis in negative direction	0 to 15	0 if no axis in the negative direction.	**Yes**	**Yes**	7-26

Table E-3 *Axis Get/Set commands (device type M) (continued)*

Command	Brief Description	Value Range	Default Value	Get	Set	See Page
CANegativeDist	Minimum distance required to avoid collisions, in negative direction. Calibrate before setting.	1 to 10,000 mm	0 if no axis in the negative direction.	**Yes**	**Yes**	7-26
CAPositiveAxisID	ID of neighboring axis in positive direction	0 to 15	0 if no axis in the positive direction.	**Yes**	**Yes**	7-26
CAPositiveDist	Minimum distance required to avoid collisions, in positive direction. Calibrate before setting.	1 to 10,000 mm	0 if no axis in the positive direction.	**Yes**	**Yes**	7-27
DefAcceleration	Axis acceleration during any move command (except CavroInit)	1 to 100,000 mm/s^2	X axis: 3500 mm/s^2 Y axis: 3900 mm/s^2 Standard Z axis: 6000 mm/s^2 Universal Z axis: 1000 mm/s^2 Dual Z axis: 6000 mm/s^2	**Yes**	**Yes**	7-28
DefInitAcceleration	Axis acceleration during CavroInit command	1 to 100,000 mm/s^2	X axis: 3500 mm/s^2 Y axis: 3900 mm/s^2 Standard Z axis: 6000 mm/s^2 Universal Z axis: 1000 mm/s^2 Dual Z axis: 6000 mm/s^2	**Yes**	**Yes**	7-28
DefInitSpeed	Axis initialization speed during CavroInit command	0.1 to 200 mm/s	X axis: 80 mm/s Y axis: 80 mm/s Standard Z axis: 40 mm/s Universal Z axis: 40 mm/s Dual Z axis: 40 mm/s	**Yes**	**Yes**	7-29

Table E-3 *Axis Get/Set commands (device type M) (continued)*

Command	Brief Description	Value Range	Default Value	Get	Set	See Page
DefSpeed	Default axis movement speed	0.1 to 10000 mm/s	X axis: 800 mm/s Y axis: 600 mm/s Standard Z axis: 600 mm/s Universal Z axis: 250 mm/s Dual Z axis: 600 mm/s	**Yes**	**Yes**	7-29
DropForceFactor	Force used to drop disposable tip (Standard Z and Dual Z only)	8000 to 22000	17000	**Yes**	**Yes**	7-30
DropSpeed	Travel speed used by Z axis to drop disposable tip (Standard Z and Dual Z only)	1 to 150 mm/s	60 mm/s	**Yes**	**Yes**	7-30
ExitLimit	Acceptance limit for successful exit signal check	0 to 50 mm	5 mm	**Yes**	**Yes**	7-31
ExitRetractDist	Z axis travel distance during CheckForExit command	1 to 100 mm	10 mm	**Yes**	**Yes**	7-31
ExitRetractSpeed	Z axis travel speed when retracting and searching for exit signal	1 to 150 mm/s	60 mm/s	**Yes**	**Yes**	7-32
FirmwareID	Axis firmware part number and revision	Pre-set	Pre-set	**Yes**	No	7-32
HighestTemp	Axis' initial temperature value, in Celsius	0 (This is initial or reset value)	0	**Yes**	**Yes**	7-33
HighestTempAbs	Axis' highest recorded temperature ever, in Celsius	As recorded	Cannot be reset	**Yes**	No	7-33
InitAcceptanceWindow	Allowable position difference between two initializations in a double initialization using CavroInit command	0 to 10 mm	0.5 mm	**Yes**	**Yes**	7-34

Table E-3 *Axis Get/Set commands (device type M) (continued)*

Command	Brief Description	Value Range	Default Value	Get	Set	See Page
InitForceFactor	Force used by Z axis to search for a hard stop during CavroInit command	6000 to 22000	Standard Z axis: 8000 Universal Z axis: 22000 Dual Z axis: 8000	**Yes**	**Yes**	7-34
InitMode	Z axis initialization mode	0 (double init) or 1 (single init)	0	**Yes**	**Yes**	7-35
LLDAcceptanceWindow	Using double detection, the maximum acceptance difference between the 2 detection positions (positive or negative)	0.5 to 50 mm	5 mm	**Yes**	**Yes**	7-40
LLDDetectionMethod	Z axis liquid detection method used	0 (double detection) or 1 (single detection)	0	**Yes**	**Yes**	7-41
LLDRetractDist	Z axis retraction distance after first detection of a double liquid level detection	*LLDAcceptance Window* to 100 mm	10 mm/s	**Yes**	**Yes**	7-41
LLDRetractOnError	Indicator of whether or not Z axis will retract to start position if no liquid detected	0 (retract) and 1 (do not retract)	0	**Yes**	**Yes**	7-42
LLDSearchSpeed	Z axis speed used when searching for liquid	1 to 150 mm/s	60 mm/s	**Yes**	**Yes**	7-42
LLDSensitivity	Sensitivity, in increments, used when searching for liquid. Settings vary based on arm configurations.	3 (highest sensitivity) to 120 (lowest sensitivity)	63	**Yes**	**Yes**	7-42
LLDSubmergeDist	DIstance the Z axis will extend after successfully detecting liquid	0 to 100 mm	0	**Yes**	**Yes**	7-43
PickForceFactor	Force used to engage disposable tip during pickup (Standard Z axis, Universal Z Axis with ADP, and Dual Z axis only)	1000 to 20,000	Standard Z axis: 11800 Universal Z axis: 7000 Dual Z axis: 11800	**Yes**	**Yes**	7-45

Table E-3 *Axis Get/Set commands (device type M) (continued)*

Command	Brief Description	Value Range	Default Value	Get	Set	See Page
PickSpeed	Z axis travel speed when picking up disposable tip (Standard Z axis, Dual Z axis, and Universal Z axis with ADP only)	1 to 150 mm/s	60 mm/s	**Yes**	**Yes**	7-46
Pos	Axis position, per device encoder	"Get" returns actual physical position	"Set" calibrates axis coordinates to worktable coordinates	**Yes**	**Yes**	7-47
PWMDutyCycle	PWM "on" (high) state time, in 25 ns increments.		See complete *PWMDuty Cycle* command instructions.	**Yes**	**Yes**	7-49
PWMFrequency	PWM Output signal period, in 25 ns increments. PWMFrequency allows separation of frequencies on configurations with multiple Z axes.	See complete PWMFrequency command instructions.	See complete command instructions. Inappropriate PWMFrequency settings may cause false LLD.	**Yes**	**Yes**	7-37
RangeHigh	Highest reachable axis position, in mm	RangeHigh is calculated; it cannot be set.	X axis left: *AxisOffset* + *WorkingLen* X axis right: *AxisOffset* Y axis: *AxisOffset* + *WorkingLen* Z axis: *AxisOffset*	**Yes**	No	7-51
RangeLow	Lowest reachable axis position, in mm	RangeLow is calculated; it cannot be set.	X axis left: *AxisOffset* X axis right: *AxisOffset* - *WorkingLen* Y axis: *AxisOffset* Z axis: *AxisOffset* - *WorkingLen*	**Yes**	No	7-51

Table E-3 *Axis Get/Set commands (device type M) (continued)*

Command	Brief Description	Value Range	Default Value	Get	Set	See Page
Status	Current status of the axis	0 (initialized) or 1 (not initialized) If initialized, may see second code: 0 (device error) 1 (disabled) or 2 (busy)	--	**Yes**	No	7-52
TipPresence	Tip attached to Z axis or not	0 (not present) or 1 (present)	--	**Yes**	No	7-53
TotalMoveCount	Cumulative number of movements performed by axis since last reset	Calculated	Set resets counter to 0	**Yes**	**Yes**	7-54
TotalMoveCountAbs	Absolute cumulative number of movements performed by axis	Calculated	Cannot be reset	**Yes**	No	7-54
TotalTravelDist	Cumulative travel distance performed by axis since last reset, in mm	Calculated	Set resets counter to 0	**Yes**	**Yes**	7-54
TotalTravelDistAbs	Absolute cumulative travel distance performed by axis	Calculated	Cannot be reset	**Yes**	No	7-55
TravelPos	Position to which the axis moves first, before executing MoveAbs command	See *MoveAbs* and *CavroInit* commands	0 indicates no movement	**Yes**	**Yes**	7-55
WorkingLen	Distance between axis home position and maximum reachable axis position, used to check for out of range	Calculated	--	**Yes**	**Yes**	7-57

E.2 Action Commands

Action commands are organized by device type:

- A - Arm commands
- M - Axis (motor) commands

Table E-4 *Arm action commands (device type A)*

Command	Brief Description	Parameters and Ranges	Default Values	Notes	See Page
CavroInit	Initialize robotic arm	--	--	Axes with defined *TravelPos* values are moved first	7-59
CheckForExit	Moves tip out of liquid and checks for exit signal using cLLD system	[RetractSpeed] *1 to 150 mm/s* [RetractDist] *1 to 100 mm* [Limit] *0 to 50 mm* [Sensitivity] *3 to 120 increments*	*ExitRetractSpeed* *ExitRetractDist* *ExitLimit* *LLDSensitivity*	Z axis only	7-69
DetectLiquid	Detects liquid using cLLD	[StartPos] *RangeLow to RangeHigh* [SearchLim] *StartPos to RangeLow* [SubmergeDist] *0 to 100 mm* [Sensitivity] *3 to 120 increments* [DetectionMethod] *0 (double)* *1 (single)* [RetractOnError] *0 (retract)* *1 (don't retract)*	-- -- *LLDSubmergeDist* *LLDSensitivity* 0, unless overridden by *Set,LLDDetectionMethod* command 0, unless overridden by *Set,LLDRetractOnError* command	Z axis only	7-72

Table E-4 *Arm action commands (device type A) (continued)*

Command	Brief Description	Parameters and Ranges	Default Values	Notes	See Page
DropTip	Drops a disposable tip from current position	[StartPos] *RangeLow to RangeHigh - 10 mm* [Limit] *StartPos to RangeHigh + 50 mm* [AcceptanceLimit] *StartPos to Limit* [Speed] *1 to 150 mm/s* [ForceFactor] *8000 to 22000*	*AxisOffset - 20 mm* *AxisOffset + 20 mm* *AxisOffset* *DropSpeed* *DropForceFactor*	Standard Z axis or Dual Z axis only	7-76
MoveAbs	Moves robotic arm to an absolute position	[pos1]-[pos15] *RangeLow to RangeHigh*	--	Axes with defined *TravelPos* values are moved first	7-60
PickTip	Picks up a disposable tip	[StartPos] *RangeLow to RangeHigh - 10 mm* [Limit] *StartPos to RangeLow* [AcceptanceLimit] *StartPos to Limit* [Speed] *1 to 150 mm/s* [ForceFactor] *8000 to 22000*	*AxisOffset - 20 mm* *AxisOffset + 20 mm* *AxisOffset* *PickSpeed* *PickForceFactor*	Standard Z axis, Dual Z axis, or Universal Z axis with ADP only	7-79
TerminateMotion	Stops motion of all axes on the arm immediately			Accepted only during *MoveAbs* or *CavroInit*	7-61

Table E-5 *Axis action commands (device type M)*

Command	Brief Description	Parameters and Ranges	Default Values	Notes	See Page
CavroInit	Initialize a single axis of motion	X axis - search for home sensor Y axis - search for home sensor Z axis - search for hard stop	Uses *DefInitSpeed, DefInitAcceleration* Uses *DefInitSpeed, DefInitAcceleration* Uses *DefInitSpeed, DefInitAcceleration, InitForceFactor and InitMode.*	Double init of Z axis is successful if position difference between the two initializations is within *Init Acceptance-Window*	7-63
CheckForExit	Moves tip out of liquid and checks for exit signal using cLLD system	[RetractSpeed] *1 to 150 mm/s* [RetractDist] *1 to 100 mm* [Limit] *0 to 50 mm* [Sensitivity] *3 to 120 increments*	*ExitRetractSpeed* *ExitRetractDist* *ExitLimit* *LLDSensitivity*	Z axis only	7-69
ClearError	Clears all errors preventing action commands from executing on the axis, and resets error status				7-64

Table E-5 *Axis action commands (device type M) (continued)*

Command	Brief Description	Parameters and Ranges	Default Values	Notes	See Page
DetectLiquid	Detects liquid using cLLD	[StartPos] *RangeLow to RangeHigh* [SearchLim] *StartPos to RangeLow* [SubmergeDist] *0 to 100 mm* [Sensitivity] *3 to 120 increments* [DetectionMethod] *0 (double)* *1 (single)* [RetractOnError] *0 (retract)* *1 (don't retract)*	-- -- *LLDSubmergeDist* *LLDSensitivity* 0, unless overridden by *Set, LLDDetectionMethod* command 0, unless overridden by *Set, LLDRetractOnError* command	Z axis only	7-72
Disable	Disables the motor current/drive for an axis				7-64
DropTip	Drops a disposable tip from current position	[StartPos] *RangeLow to RangeHigh - 10 mm* [Limit] *StartPos to RangeHigh + 50 mm* [AcceptanceLimit] *StartPos to Limit* [Speed] *1 to 150 mm/s* [ForceFactor] *8000 to 22000*	*AxisOffset - 20 mm* *AxisOffset + 20 mm* *AxisOffset* *DropSpeed* *DropForceFactor*	Standard Z axis or Dual Z axis only	7-76

Table E-5 *Axis action commands (device type M) (continued)*

Command	Brief Description	Parameters and Ranges	Default Values	Notes	See Page
Enable	Enables the motor current/drive for an axis.			Should not be used while axis is being moved by hand	7-65
MoveAbs	Moves a single axis to an absolute position	[pos] *RangeLow to RangeHigh*	Uses *DefSpeed*	Axes with defined *TravelPos* values are moved first	7-65
PickTip	Picks up a disposable tip	[StartPos] *RangeLow to RangeHigh - 10 mm* [Limit] *StartPos to RangeLow* [AcceptanceLimit] *StartPos to Limit* [Speed] *1 to 150 mm/s* [ForceFactor] *8000 to 22000*	*AxisOffset - 20 mm* *AxisOffset + 20 mm* *AxisOffset* *PickSpeed* *PickForceFactor*	Standard Z axis, Dual Z axis, or Universal Z axis with ADP only	7-79
TerminateMotion	Stops motion of a single axis immediately			Accepted only during *MoveAbs* or *CavroInit*	7-68

E.3 Error Codes

Table E-6 *System or API error codes*

Error No.	Code	Description
301	SYS_Config_error	System configuration error
302	SYS_Unable_to_connect_to_CIB	System cannot find CIB
303	SYS_Failed_to_start_logger	Logger failed to start

Table E-6 *System or API error codes (continued)*

Error No.	Code	Description
304	SYS_Unexpected_ACB_initialization_error	Unexpected error occurred during ACB initialization
305	SYS_Unexpected_error	Unexpected error occurred
306	SYS_Low_free_disk_space	Free disk space is down to XX%
307	SYS_ACB_checksum_error	Checksum error in ACB
308	SYS_TCP_connection_not_established	No TCP/IP connection established
309	SYS_Command_syntax_error	Command syntax error
310	SYS_Unexpected_error_during_cmd_execution	Unexpected error occurred during execution of system command
311	SYS_Unexpected_pump_execution_error	Unexpected error occurred during execution of a pump command
312	SYS_CIB_checksum_error	Checksum error in CIB
313	SYS_Data_not_initialized	System data not initialized
314	SYS_Unexpected_CIB_IO_error	Unexpected error during CIB I/O
315	SYS_System_halted	System operation halted
316	SYS_InitOmniApi_failed	API InitiOmniApi failed
317	SYS_OmniApi_not_initialized	API OmniApi was not initialized

Table E-7 *Command error codes*

Error No.	Code	Description
0	No_Error	No error - system is operating properly and ready for commands
1	Initialization_Error	Error during initialization
2	Invalid_Command	Invalid command sent
3	Invalid_Operand	Operand provided is invalid (out of range)
4	Timeout_Error	Command response not received before timeout threshold reached
5	Device_Not_Implemented	Specified ACB not found or incompatible TML
6	Arm_Collision_Avoided	Collision avoided
7	Device_Not_Initialized	Intended device has not been initialized

Table E-7 *Command error codes (continued)*

Error No.	Code	Description
8	Device_Busy	Intended device is executing another command
9	Device_Error	Unexpected error in the hardware
10	Command_Aborted	Command not executed
11	Double_init_acceptance_window_exceeded	Initialization value exceeds the window for double initialization acceptance
12	Invalid_Number_of_Operands	Too many or too few operands
13	Device_Disabled	Device disabled by user or by hardware error
14	Invalid_travel_pos	Value of TravelPos is invalid
15	cLLD_No_Liquid_Detected	No liquid detected by tip using cLLD
16	cLLD_Not_Enough_Liquid_Detected	cLLD detected liquid, but insufficient amount for operation
17	cLLD_Liquid_detection_failure	No liquid detected by cLLD
18	cLLD_Exit_Limit_Exceeded_at_Exit_Detection	Exit detection value exceeded limit
19	cLLD_No_Exit_Signal_at_Exit_Detection	Exit detection value not received
20	cLLD_not_ready	cLLD liquid level detection not ready
21	(Not currently used)	(Not currently used)
22	DiTi_Tip_not_dropped	Tip was not dropped
23	DiTi_Tip_drop_error	Error during tip drop
24	DiTi_Hard_stop_not_found	Hard stop not found when moving tip
25	DiTi_Tip_not_fetched	Axis reaches limit and tip is found, however the axis fails to reach the force set by ForceFactor
26	DiTi_Tip_crash	Probe tip has crashed into something
27	DiTi_No_Tip_mounted	No tip detected on probe
28	DiTi_Tip_already_mounted	PickTip command attempted on a probe that already has a tip
29	DiTi_Tip_not_found	Probe tip not found
30	DiTi_Tip_not_mounted_properly	Probe tip not properly mounted
41	Gripper_not_attached	Gripper not attached
42	Gripper_not_initialized	Gripper not initialized
43	Gripper_move_error	Gripper move error

Table E-7 *Command error codes (continued)*

Error No.	Code	Description
44	Gripper_CAM_at_intermediate_starting_position	Gripper CAM at intermediate starting position (Normal Initmode only)
45	Gripper_object_detected	Gripper object detected
46	Gripper_no_object_detected	Gripper no object detected
48	Gripper_sensor_error	Gripper sensor error
60	Brake_test_failed	Brake test failed
101	PreInitMoveRel_move_blocked	Axis was physical blocked before the move was completed
102	PreInitMoveRel_not_allowed_at_current_state	This error returned if command PreInitMoveRel was issued in initialized state
103	PreInitMoveRel_encoder_error	Encoder doesn't generate the expected pulse sequence

F Embedded Command and Error Quick Reference Guide

F.1 Embedded Mode Quick Start

Before programming the Omni in Embedded mode, follow the steps in this section to get acquainted with basic operations. This section assumes that setup of the system is complete as described in Chapter 3, “Mechanical Integration”, Chapter 4, “Electrical Integration”, and Chapter 5, “Integration with Omni Configurator and Cavro Fusion”.

WARNING! *Personnel who are not operating the Omni Robot module should take extra care when standing near the module, as an unanticipated movement by one or more axes could cause injury to a bystander. This is especially important if the Omni Robot is being operated remotely through the Ethernet connection.*

F.1.1 Set up a static IP address on your computer.

These instructions are for Windows 7.

1. Open the Control Panel and select the Network and Sharing Center.
2. Click on the Local Area Connection link.
3. Click on the Properties button.
4. Select Internet Protocol Version 4 from the list of options and click the Properties button.
5. Select Use the Following IP Address and enter 192.168.0.200. The subnet mask will auto-fill.
6. Click OK.
7. Close the Local Area Connection Properties window.

F.1.2 Connect the PC to the Omni.

1. Connect your PC to the Omni with an Ethernet cable.
2. Switch power on for the Omni.

F.1.3 Start the OE UserApp

1. Start the application by clicking on the **`OEUserApp`** executable file in the **`\OEUserApp`** subdirectory of the root directory of the installation CD.
2. In the Settings window, select the Ethernet option and click OK.

F.1.4 Execute basic Omni commands.

1 In the OE UserApp, enter a command string and click Send.

- The following command string initializes axes X, Y, and Z, and then moves each axis to a fixed location.

```
1/1Z/2Z/3Z/1A200/2A100/3A150
```

- The following command string also initializes axes X, Y, and Z, and then moves each axis to the same fixed location. By using parentheses, it is easier to see what is being accomplished.

```
1((/3Z)/1Z/2Z)((/1A100/2A100)/3A150)
```

F.2 Parameter Get and Set Commands

GET (?) commands report the specified values related to single axis commands. SET (u) commands store specified values related to single-axis commands.

Table F-1 Parameter get and set commands

Var ID	Name	Brief Description	Value Range	Default Value	Get?	Set?	See Page
0	Pos	Axis position, per device encoder	Actual physical position	--	Yes	Yes	G: 6-55 S: 6-71
1	AxisOffset	Axis home position coordinates	Depends on reference position	Pre-set	Yes	No	G: 6-34
2	AxisOrientation	Orientation of axis	0 (left oriented) or 1 (right oriented)	0	Yes	No	G: 6-34
3	InitMode	Z axis initialization mode	0 (double init) or 1 (single init)	0	Yes	Yes	G: 6-48 S: 6-66
4	InitAcceptanceWindow	Allowable position difference between two initializations in a double initialization using Initialize Axis command	0 to 10 mm	0.5 mm	Yes	Yes	G: 6-46 S: 6-64
5	InitForceFactor	Force used by Z axis to search for a hard stop during Initialize Axis command	6000 to 22000	Standard Z axis: 8000 Universal Z axis: 22000 Dual Z axis: 8000	Yes	Yes	G: 6-46 S: 6-65

Table F-1 Parameter get and set commands (continued)

Var ID	Name	Brief Description	Value Range	Default Value	Get?	Set?	See Page
6	InitForceRatio	Percentage of *InitForceFactor* used for first initialization in a double hardstop initialization.	1-100	Standard Z axis: 50 Universal Z axis: 55 Dual Z axis: 50	Yes	Yes	G: 6-46 S: 6-65
7	DefInitAcceleration	Axis acceleration during Initialize Axis command	1-100,000 mm/s^2	X axis: 3500 mm/s^2 Y axis: 3900 mm/s^2 Standard Z axis: 6000 mm/s^2 Universal Z axis: 1000 mm/s^2 Dual Z axis: 6000 mm/s^2	Yes	Yes	G: 6-37 S: 6-60
8	DefInitSpeed	Axis initialization speed during Initialize Axis command	0.1 to 200 mm/s	X axis: 80 mm/s Y axis: 80 mm/s Standard Z axis: 40 mm/s Universal Z axis: 250 mm/s Dual Z axis: 40 mm/s	Yes	Yes	G: 6-37 S: 6-61

Table F-1 Parameter get and set commands (continued)

Var ID	Name	Brief Description	Value Range	Default Value	Get?	Set?	See Page
9	DefAcceleration	Axis acceleration during any move command (except Initialize Axis)	1-100,000 mm/s^2	X axis: 3500 mm/s^2 Y axis: 3900 mm/s^2 Standard Z axis: 6000 mm/s^2 Universal Z axis: 1000 mm/s^2 Dual Z axis: 6000 mm/s^2	Yes	Yes	G: 6-36 S: 6-60
10	DefSpeed	Default axis movement speed	0.1 to 10000 mm/s	X axis: 800 mm/s Y axis: 600 mm/s Standard Z axis: 600 mm/s Universal Z axis: 250 mm/s Dual Z axis: 600 mm/s	Yes	Yes	G: 6-37 S: 6-61
11	Status (DeviceState)	Current status of the axis	0x00: Initialized 0x20: Not Initialized 0x01: Disabled 0x02: Device Error 0x04: Device Busy	--	Yes	No	G: 6-57
12	DropSpeed	Travel speed used by Z axis to drop disposable tip (Standard Z or Dual Z axis only)	1 to 150 mm/s	60 mm/s	Yes	Yes	G: 6-40 S: 6-62

Table F-1 Parameter get and set commands (continued)

Var ID	Name	Brief Description	Value Range	Default Value	Get?	Set?	See Page
13	DropForceFactor	Force used to drop disposable tip (Standard Z or Dual Z axis only)	8000 to 22000	17000	Yes	Yes	G: 6-40 S: 6-62
14	EncoderDirection	Direction of the encoder	0: Normal 1: Reverse	0	Yes	Yes	G: 6-40 S: 6-62
15	ExitRetractSpeed	Z axis travel speed when retracting and searching for exit signal	1 to 150 mm/s	60 mm/s	Yes	Yes	G: 6-42 S: 6-63
16	ExitRetractDist	Z axis travel distance during Check For Exit command	1 to 100 mm	10 mm	Yes	Yes	G: 6-42 S: 6-63
17	ExitLimit	Acceptance limit for successful exit signal check	0 to 50 mm	5 mm	Yes	Yes	G: 6-42 S: 6-63
18	FirmwareID	Axis firmware part number and revision	Pre-set	Pre-set	Yes	No	G: 6-43
19	GripperInitMode	Initialization mode of the gripper	0: Normal init, closed 1: Alternate init, closed 2: Normal init, open 3: Alternate init, open	0	Yes	Yes	G: 6-45 S: 6-64
20	GripperState	State of the gripper	0 - Gripper Open 1 - Gripper Not Attached 2 - Gripper Move Error 3 - Gripper Not Initialized 4 - Gripper Object Detected 5 - Gripper Closed	--	Yes	No	G: 6-45
21	LLDSubmergeDist	Distance the Z axis will extend after successfully detecting liquid	0 to 100 mm	0	Yes	Yes	G: 6-52 S: 6-69

Table F-1 Parameter get and set commands (continued)

Var ID	Name	Brief Description	Value Range	Default Value	Get?	Set?	See Page
22	LLDSensitivity	Sensitivity, in increments, used when searching for liquid. Settings vary based on arm configurations.	3 (highest sensitivity) to 120 (lowest sensitivity)	63	Yes	Yes	G: 6-51 S: 6-68
23	AxisDesc	Axis type, including TML script version number	format aabbcccc	aa = major version bb = minor version cccc = axis type	Yes	No	G: 6-34
24	LastDeviceErrorInfo	Information about the most recent device error	See Table 6-4	--	Yes	No	G: 6-48
25	LLDDetectionMethod	Z axis liquid detection method used	0 (double detection) or 1 (single detection)	0	Yes	Yes	G: 6-49 S: 6-66
26	LLDRetractOnError	Indicator of whether or not Z axis will retract to start position if no liquid detected	0 (retract) and 1 (do not retract)	0	Yes	Yes	G: 6-49 S: 6-67
27	LLDRetractDist	Z axis retraction distance after first detection of a double liquid level detection	*LLDAcceptance Window* to 100 mm	10 mm/s	Yes	Yes	G: 6-49 S: 6-67
28	LLDAcceptanceWindow	Using double detection, the maximum acceptance difference between the 2 detection positions (positive or negative)	0.5 to 50 mm	5 mm	Yes	Yes	G: 6-48 S: 6-66
29	LLDSearchSpeed	Z axis speed used when searching for liquid	1 to 150 mm/s	60 mm/s	Yes	Yes	G: 6-50 S: 6-67
30	LLDSelfTestSensitivity	Sensitivity test level	3 (most sensitive) to 120 (least sensitive)	63	Yes	Yes	G: 6-51 S: 6-68

Table F-1 Parameter get and set commands (continued)

Var ID	Name	Brief Description	Value Range	Default Value	Get?	Set?	See Page
31	LLDSelfTestLevel	Capacitance test level	0: high capacitance test level, ΔC ~ 0.75pF 1: low capacitance test level, ΔC ~ 0.15pF	0	Yes	Yes	G: 6-50 S: 6-68
32	LLDSelfTestLastResult	Result of the most recent cLLD Self Test command	See complete command instructions	See complete command instructions	Yes	No	G: 6-50
33	MoveAbsHiForceSpeed	Speed used during the Move Axis to Absolute Position with High Force command.	0.1 to 10000 mm/s	40 mm/s	Yes	Yes	G: 6-52 S: 6-69
34	PWMFrequency	PWM Output signal period, in 25 ns increments. PWMFrequency allows separation of frequencies on configurations with multiple Z axes.	PWMDutyCycle+1 to 65535	See complete command instructions.	Yes	Yes	G: 6-56 S: 6-72
35	PWMDutyCycle	PWM “on” (high) state time, in 25 ns increments.	0 to PWMFrequency-1	See complete command instructions.	Yes	Yes	G: 6-55 S: 6-71
36	TipPresence	Tip attached to Z axis or not	0 (not present) or 1 (present)	--	Yes	No	G: 6-57
37	PickSpeed	Z axis travel speed when picking up disposable tip (Standard Z axis, Dual Z axis, or Universal Z axis with ADP only)	1 to 150 mm/s	60 mm/s	Yes	Yes	G: 6-54 S: 6-70
38	PickForceFactor	Force used to engage disposable tip during pickup (Standard Z axis, Dual Z axis, or Universal Z axis with ADP only)	1000 to 20,000	Standard Z axis: 11800 Universal Z axis: 7000 Dual Z axis: 11800	Yes	Yes	G: 6-53 S: 6-70

Table F-1 Parameter get and set commands (continued)

Var ID	Name	Brief Description	Value Range	Default Value	Get?	Set?	See Page
39	RangeHigh	Highest reachable axis position, in mm	RangeHigh is calculated; it cannot be set.	X axis left: *AxisOffset* + *WorkingLen* X axis right: *AxisOffset* Y axis: *AxisOffset* + *WorkingLen* Standard Z axis: *AxisOffset* Universal Z axis: *AxisOffset* Dual Z axis: *AxisOffset*	Yes	No	G: 6-56
40	RangeLow	Lowest reachable axis position, in mm	RangeLow is calculated; it cannot be set.	X axis left: *AxisOffset* X axis right: *AxisOffset* - *WorkingLen* Y axis: *AxisOffset* Standard Z axis: *AxisOffset* - *WorkingLen* Universal Z axis: *AxisOffset* - *WorkingLen* Dual Z axis: *AxisOffset* - *WorkingLen*	Yes	No	G: 6-57

Table F-1 Parameter get and set commands (continued)

Var ID	Name	Brief Description	Value Range	Default Value	Get?	Set?	See Page
41	WorkingLength	Distance between axis home position and maximum reachable axis position, used to check for out of range	0 to 10000 mm	--	Yes	Yes	G: 6-58 S: 6-73
71	ACB move count	Number of ACB move counts since last reset	0 to 4294967295 (0xFFFFFFFF)	--	Yes	Yes	G: 6-33 S: 6-74
72	ACB move count total	Total number of recorded ACB move counts	0 to 4294967295 (0xFFFFFFFF)	--	Yes	No	G: 6-33
73	ACB highest temperature recorded	Highest recorded ACB temperature since last reset	Any valid temperature	--	Yes	Yes	G: 6-33 S: 6-75
74	ACB highest temperature ever recorded	Highest ever recorded ACB temperature in Celsius	Any valid temperature	--	Yes	No	G: 6-32
76	ACB configuration	Parameters that affect the general ACB configuration	ASCII text string	--	Yes	Yes	G: 6-43 S: 6-73 S: 6-62 S: 6-60
77	Initialization Configuration	Initialization configuration	ASCII text string	--	Yes	Yes	G: 6-47 S: 6-61 S: 6-60 S: 6-65 S: 6-64 S: 6-66 S: 6-65
78	Get Move Configuration	Parameters that affect the move commands	ASCII text string	--	Yes	Yes	G: 6-52 S: 6-61 S: 6-60 S: 6-69

Table F-1 Parameter get and set commands (continued)

Var ID	Name	Brief Description	Value Range	Default Value	Get?	Set?	See Page
79	Detect liquid configuration	Parameters that affect the Detect Liquid command	ASCII text string	--	Yes	Yes	G: 6-38 S: 6-67 S: 6-60 S: 6-68 S: 6-69 S: 6-67 S: 6-66 S: 6-72 S: 6-66 S: 6-67
80	Check for exit configuration	Parameters that affect the Check for Exit command	ASCII text string	--	Yes	Yes	G: 6-35 S: 6-63 S: 6-60 S: 6-68 S: 6-63 S: 6-63
81	cLLD self test configuration	Parameters that affect the cLLD Self Test Configuration command	ASCII text string	--	Yes	Yes	G: 6-35 S: 6-68 S: 6-68
82	Get Pick Tip Configuration	Parameters that affect the Get Pick Tip Configuration command	ASCII text string	--	Yes	Yes	G: 6-54 S: 6-70 S: 6-60 S: 6-70
83	Drop Configuration	Parameters that affect the Drop Configuration command	ASCII text string	--	Yes	Yes	G: 6-39 S: 6-62 S: 6-60 S: 6-62
84	Gripper Configuration	Gripper Configuration	ASCII text string	--	Yes	No	G: 6-44 S: 6-64

F.3 CIB Control Get Command: F?

Table F-2 *List of Var IDs for the F? command*

Var ID	Name	Brief Description	Value Range	Default Value	Set?	See Page
0	IO1	Status of I/O 1 when IO1Mode = 1 (output)	0 (low) or 1 (high)	--	**Yes**	6-115

Table F-2 *List of Var IDs for the F? command (continued)*

Var ID	Name	Brief Description	Value Range	Default Value	Set?	See Page
1	IO2	Status of I/O 2 when IO2Mode = 1 (output)	0 (low) or 1 (high)	--	**Yes**	6-116
2	Input1	Status of input 1	0 or 1	--	No	6-113
3	Input2	Status of input 2	0 or 1	--	No	6-113
4	Input3	Status of input 3	0 or 1	--	No	6-114
5	Input4	Status of input 4	0 or 1	--	No	6-114
6	Input5	Status of input 5	0 or 1	--	No	6-114
7	Input7	Status of input 7	0 or 1	--	No	6-115
8	Output3	Status of output 3: stop state	0 (low) or 1 (high)	0	**Yes**	6-118
9	Output4	Status of output 4: lock state	0 (low impedance to ground) or 1 (high impedance to ground)	0	**Yes**	6-118
10	Output1	Status of output 1	0 (low) or 1 (high)	0	**Yes**	6-117
11	Output2	Status of output 2	0 (low) or 1 (high)	0	**Yes**	6-117
12	Input6	Status of input 6	0 or 1	--	No	6-114
13	IO1Mode	I/O direction for I/O 1	0 (input) or 1 (output)	0	**Yes**	6-115
14	IO2Mode	I/O direction for I/O 2	0 (input) or 1 (output)	0	**Yes**	6-116
23	FWRevision	FW Part number & revision number	Pre-set	Pre-set	No	6-112
24	FWVersion	Version of firmware	Pre-set	Pre-set	No	6-113
25	BootVersion	Version of boot code	Pre-set	Pre-set	No	6-111
30	VoltageFU1	Power supply voltage after FU1	--	--	No	6-120
31	VoltageFU2	Power supply voltage after FU2	--	--	No	6-121
32	VoltageFU3	Power supply voltage after FU1	--	--	No	6-121

Table F-2 *List of Var IDs for the F? command (continued)*

Var ID	Name	Brief Description	Value Range	Default Value	Set?	See Page
33	VoltageFU4	Power supply voltage after FU1	--	--	No	6-121
34	VoltageV3.3D	3.3V power supply rail	--	--	No	6-121
35	VoltageV1.8D	1.8V power supply rail	--	--	No	6-122
36	VoltageTPPO	On/Off state of J18 pin 3	--	--	No	6-122
40	MACAddress	MAC Address of CIB	--	--	No	6-117
41	IPAddress	IP address of CIB	0.0.0.0 to 255.255.255.255	192.168.0.198	**Yes**	6-116
42	PortNumber: Command Processor	Port number for Command Processor	1-65535	9898	**Yes**	6-118
44	PortNumber: Embedded	Port number for Omni Embedded communication	1-65535	9897	**Yes**	6-119
50	PowerOnMinutes	Cumulative number of minutes powered on since last reset	0 to 4294967295 (0xFFFFFFFF)	Set resets counter to 0	**Yes**	6-119
51	Absolute PowerOnMinutes	Highest number of power on minutes ever recorded	0 to 4294967295 (0xFFFFFFFF)	--	No	6-119
52	PowerUps	Cumulative number of power ups since last reset	0 to 65535	--	**Yes**	6-120
53	Absolute PowerUps	Highest number of power ups ever recorded	0 to 65535	--	**No**	6-120
76	CIB configuration	Gets the following configuration info, separated by | characters: 1: RS-232 Baud Rate 2: RS-485 Baud Rate 3: Can0 Baud Rate 4: CAN1 Baud Rate 5: Device Protocol 6: Device Error termination type 7: Device Event termination type 8: INT5 trigger type	See Get CIB Configuration for details	See Get CIB Configuration for details	No	6-111

F.4 CIB Control Set Command (u and U)

Table F-3 *List of Var IDs for the u command*

Var ID1	Var ID2	Name	Brief Description	Value Range	Default Value	See Page
0	0 1	Output1	Set value of output 1	0 (low) or 1 (high)	0	6-126
1	0 1	Output2	Status of output 2	0 (low) or 1 (high)	0	6-126
2	0 1	IO1	Status of I/O 1 when IO1Mode = 1 (output)	0 (low) or 1 (high)	0	6-124
3	0 1	IO2	Status of I/O 2 when IO2Mode = 1 (output)	0 (low) or 1 (high)	0	6-125
4	0 1	Output3	Status of output 3: stop state	0 (low) or 1 (high)	0	6-126
5	0 1	Output4	Status of output 4: lock state	0 (low impedance to ground) or 1 (high impedance to ground)	0	6-127

Table F-4 *List of Var IDs for the U command*

Var ID1	Var ID2	Name	Brief Description	Value Range	Default Value	See Page
0	0	CAN0 Baud Rate	Set the CAN0 baud rate	500K/s	500K/s	6-129
1	0 1 2 3	CAN1 Baud Rate	Set the CAN1 baud rate	500K/s 250K/s 125K/s 100K/s	500K/s	6-129
10	0 1	RS-232 Baud Rate	Set the RS-232 baud rate	0: 9600 1: 38400	1	6-131
15	0 1	RS-485 Baud Rate	Set the RS-485 baud rate	0: 9600 1: 38400	0	6-132
20	n	IP Address:1	First octet of IP address of CIB	0-255	192	6-130
21	n	IP Address:2	Second octet of IP address of CIB	0-255	168	6-130
20	n	IP Address:3	Third octet of IP address of CIB	0-255	0	6-130

Table F-4 *List of Var IDs for the U command (continued)*

Var ID1	Var ID2	Name	Brief Description	Value Range	Default Value	See Page
20	n	IP Address:4	Fourth octet of IP address of CIB	0-255	198	6-130
24	n	Port Number: Command Processor	Port number for the command processor connection	1-65535	9898	6-130
34	n	Port Number: Embedded	Port number for the embedded connection	1-65535	9897	6-130

F.5 Action Commands

Table F-5 *Axis action commands*

Name	Brief Description	Parameters and Ranges	Default Values	Notes	See Page
Z	Initialize Axis	--	--	Double init of Z axis is successful if position difference between the two initializations is within *Init Acceptance-Window*	6-78
O	Check for Exit: Moves tip out of liquid and checks for exit signal using cLLD system	--	--	Z axis only	6-81
C	Clear Error: Clears all errors preventing action commands from executing on the axis, and resets error status	--	--		6-83

Table F-5 *Axis action commands (continued)*

Name	Brief Description	Parameters and Ranges	Default Values	Notes	See Page
B <*LLDStartPos*>, <*SearchLimit*>	Detect Liquid: Detects liquid using cLLD	<*LLDStartPos*> RangeLow to RangeHigh <*SearchLimit*> RangeLow to <*LLDStartPos*>	--	Z axis only	6-84
N <*Var_ID*>	Disable/Enable: Disables or enables the motor current/ drive for an axis	0: Disable 1: Enable	--	Enable should not be used while axis is being moved by hand	6-94
E <*DropStartPos*>, <*DropLimit*>, <*DropAcceptanceLimit*>	Drop Tip: Drops a disposable tip from current position	<*DropStartPos*> *RangeLow to RangeHigh - 10 mm* <*DropLimit*> <*DropStartPos*> to *RangeHigh + 50 mm* <*DropAcceptanceLimit*> *StartPos to RangeHigh + 50 mm*	AxisOffset - 20 AxisOffset + 20 AxisOffset	Standard Z axis or Dual Z axis	6-91
W	Gripper operations	0: Initialize 1: Close 2: Open	--	Gripper Z axis only	6-103
A <*TargetPos*>	Move Axis to Absolute Position: Moves a single axis to an absolute position	<*TargetPos*>: *RangeLow to RangeHigh*	--		6-94
F <*TargetPos*>	Move Axis to Absolute Position with High Force: Moves a single axis to an absolute position	<*TargetPos*>: 0.1 to 10000 mm/sec	--	Standard Z axis and Dual Z axis only	6-95
I <*Offset*>	Move Axis to Relative Position Before Initialization	<*Offset*>: Units in mm	--	X and Y axes only	6-96

Table F-5 *Axis action commands (continued)*

Name	Brief Description	Parameters and Ranges	Default Values	Notes	See Page
P <*StartPos*>, <*Limit*>, <*AcceptanceLimit*>	Pick Tip: Picks up a disposable tip	<*StartPos*>: *RangeLow to RangeHigh - 10 mm* <*Limit*>: <*StartPos*> *to RangeLow* <*AcceptanceLimit*>: <*StartPos*> *to* <*Limit*>	--	Standard Z axis, Dual Z axis, or Universal Z axis with ADP only	6-97
T	Terminate Motion: Stops motion of a single axis immediately	--	--		6-102

F.6 Status Codes, Error Codes, and Events

See section Section 6.6, ”Status Codes, Error Codes, and Events”for lists of:

- ACB and Gripper Error Codes
- ACB and Gripper Event Codes
- CIB (Device F) Error Codes
- CIB (Device F) Event Codes
- Command Response String Status Codes

G Conversion Guide: Command Processor to Embedded Commands

G.1 Arms and Axes

In Command Processor Mode, there are commands that operate on a logical grouping of an X, Y, and Z axis as a unified arm. In Embedded Mode, all commands operate only on single axes. Parentheses can be used in Embedded Mode to group an X, Y, and Z axis so that the embedded commands approximate the behavior of arm commands in the Command Processor.

G.1.1 Initializing an Arm

In Command Processor Mode, when initializing an arm, all the axes with TravelPos not equal 0 (Z axis) will always initialize first. Then the other axes in the group with TravelPos equal 0 (X and Y axes) initialize simultaneously. To create the same behavior in Embedded Mode, use parentheses as follows:

```
1/3Z(/1Z/2Z)
or
1((/3Z)/1Z/2Z)
```

G.1.2 Moving an Arm to an Absolute Position

In Command Processor Mode, the absolute move command causes all Z axes to first move to TravelPos, then the X and Y axes move to target position at the same time, and finally Z moves to the target position. To create the same behavior in Embedded Mode assuming TravelPos is 190 and X,Y, and Z target positions are 450, 75, and 50, use parentheses as follows:

```
1/3A190(/1A450/2A75)/3A50
or
1(((/3A190)/1A450/2A75)/3A50)
```

G.2 Command Equivalence Map

Table G-1 *Single axis control commands*

Command Processor	Omni Embedded
M<AxisID>BrakeTest	/<AxisID>b
M<AxisID>CavroInit/	/<AxisID>Z[R]
M1CavroInit/	/1ZR , /3Z
M<AxisID>CheckForExit,[RetractSpeed],[RetractDist],[Limit],[Sensitivity]/	/<AxisID>O[R]
M3CheckForExit / M3CheckForExit,,20,,63/	/3O /3u27,20u22,63O
M<AxisID>ClearError/	/<AxisID>C[R]
M1ClearError/	/1C
M<AxisID>cLLDSelfTest,[testlevel],[LLDSensitivity]/	/<AxisID>c[R]
M3cLLDSelfTest/ M3cLLDSelfTest,0,63/ M3cLLDSelfTest,,63/	/3c /3u31,0u30,63c /3u30,63c
Get most recent value of cLLDSelfTest	/1?32
Get most recent value of <TestLevel> (or default) Set value of <TestLevel> (to 1 in this example)	/1?31 /1u31,1
Get most recent value of <LLDSensitivity> Set value of <LLDSensitivity> (to 70 in this example)	/1?30 /1u30,70
M<AxisID>DetectLiquid,<StartPos>,<SearchLim>,[SubmergeDist],[Sensitivity],[DetectionMethod],[RetractOnError]/	/<AxisID>B<StartPos>,<SearchLim> [R]
M3DetectLiquid,50,20 M3DetectLiquid,50,20,,,,1/	/3B50,20 /3u26,1B50,20
M<AxisID>Disable/	/<AxisID>N<0> [R]
M1Disable/	/1N0
M<AxisID>DropTip,[StartPos],[Limit],[AcceptanceLimit],[Speed],[ForceFactor],[NumberOfAxis],[AxisID]/	/<AxisID>E<StartPos>,<Limit>,<AcceptanceLimit> [R]
M3DropTip,200,220,210/ M3DropTip,200,220,210,,1500/	/3E200,220,210R /3u13,1500E200,220,210
M<AxisID>Enable/	/<AxisID>N<1> [R]
M1Enable/	/1N1

Table G-1 *Single axis control commands*

Command Processor	Omni Embedded
M<AxisID>GripperClose/	/<AxisID>W<1> [R]
M3GripperClose/	/3W1
M<AxisID>GripperInit,[InitMode]/	/<AxisID>W<0> [R]
M3GripperInit/ M3GripperInit,1/	/3W0 /3u19,1W0
M<AxisID>GripperOpen/	/<AxisID>W<2> [R]
M3GripperOpen/	/3W2
M<AxisID>MoveAbs,<pos>/	/<AxisID>A<pos> [R]
M1MoveAbs,100.54/ M3MoveAbs,-50.5/	/1A100.54 /3A-50.5
M<AxisID>MoveAbsHiForce,<pos>,[speed]/	/<AxisID>F<pos> [R]
M3MoveAbsHiForce,50.5/ M3MoveAbsHiForce,50.5,100/	/3F50.5 /3u33,100F50.5
Get value of most recent <speed> (or default) Set value of <speed> (to 800 in this example)	/1?33 /1u33,800
M<AxisID>PickTip,<StartPos>,<Limit>,<AcceptanceLimit>,[PickSpeed], [PickForceFactor]/	/<AxisID>P<StartPos>,<Limit>,<AcceptanceLimit> [R]
M3PickTip,50,20,20.5/ M3PickTip,50,20,20.5,,1200/	/3P50,20,25.5R /3u38,1200P50,20,25.5
M<AxisID>PreInitMoveRel,<offset>/	/<AxisID>I<offset> [R]
M1PreInitMoveRel,-2000 M2PreInitMoveRel,3000	/1I-2000 /2I3000
M<AxisID>TerminateMotion/	/<AxisID>T[R]
M1TerminateMotion/	/1T

Table G-2 *Single axis Set and Get parameter commands*

Command Processor	Omni Embedded
M<AxisID>Set,<Property>,<Value>/	/<AxisID>u<Property>,<Value> [R]
M<AxisID>Get,<Property>/	/<AxisID>?<Property> [R]
M1Get,AxisDesc/	/1?23
M1Get,AxisOffset/	/1?1

Table G-2 *Single axis Set and Get parameter commands*

Command Processor	Omni Embedded
M1Get,CANegativeAxisID	N/A
M1Set,CANegativeAxisID,2/	N/A
M1Get,CANegativeDist	N/A
M1Set,CANegativeDist,10/	N/A
M1Get,CAPositiveAxisID	N/A
M1Set,CAPositiveAxisID,1/	N/A
M1Get,CAPositiveDist	N/A
M1Set,CAPositiveDist,10/	N/A
M1Get,DefAcceleration/	/1?9
M1Set,DefAcceleration,1200/	/1u9,1200
M1Get,DefInitAcceleration/	/1?7
M1Set,DefInitAcceleration,2000/	/1u7,2000
M1Get,DefInitSpeed/	/1?8
M1Set,DefInitSpeed,80/	/1u8,80
M1Get,DefSpeed/	/1?10
M1Set,DefSpeed,800/	/1u10,800
M3Get,DropSpeed/	/3?12
M3Set,DropSpeed,80/	/3u12,80
M3Get,DropForceFactor/	/3?13
M3Set,DropForceFactor,2000/	/3u13,2000
M3Get,ExitLimit/	/3?17
M3Set,ExitLimit,50/	/3u17,50
M3Get,ExitRetractDist/	/3?16
M3Set,ExitRetractDist,40/	/3u16,40
M3Get,ExitRetractSpeed/	/3?15
M3Set,ExitRetractSpeed,35/	/3u15,35
M1Get,FirmwareID/	/1?18
GripperInit,1	/1u19,1

Table G-2 *Single axis Set and Get parameter commands*

Command Processor	Omni Embedded
M3Get,GripperState/	/3?20
M1Get,HighestTemp	/1?73
M1Set,HighestTemp,0/	/1U73,0
M1Get,HighestTempAbs	/1?74
M3Get,InitAcceptanceWindow/	/3?4
M3Set,InitAcceptanceWindow,5.5/	/3u4,5.5
M3Get,InitForceFactor/	/3?5
M3Set,InitForceFactor,2000	/3u5,2000
M3Get,InitForceRatio/	/3?6
N/A	/1u6,55 (Set InitForceRatio)
M3Get,InitMode/	/3?3
M3Set,InitMode,1/	/3u3,1
M3Get,LLDAcceptanceWindow/	/3?28
M3Set,LLDAcceptanceWindow,10/	/3u28,10
M3Get,LLDDetectionMethod/	/3?25
M3Set,LLDDetectionMethod,1/	/3u25,1
M3Get,LLDRetractDist/	/3?27
M3Set,LLDRetractDist,5/	/3u27,5
M3Get,LLDRetractOnError/	/3?26
M3Set,LLDRetractOnError,0/	/3u26,0
M3Get,LLDSearchSpeed/	/3?29
M3Set,LLDSearchSpeed,50/	/3u29,50
M3Get,LLDSensitivity/	/3?22
M3Set,LLDSensitivity,63/	/3u22,63
M3Get,LLDSubmergeDist/	/3?21
M3Set,LLDSubmergeDist,5/	/3u21,5
M3Get,PickForceFactor/	/3?38
M3Set,PickForceFactor,3000/	/3u38,3000

Table G-2 *Single axis Set and Get parameter commands*

Command Processor	Omni Embedded
M3Get,PickSpeed/	/3?37
M3Set,PickSpeed,80/	/3u37,80
M1Get,Pos/	/1?0
M1Set,Pos,100.5/	/1u0,100
M3Get,PWMDutyCycle/	/3?35
M3Set,PWMDutyCycle,63/	/3u35,63
M3Get,PWMFrequency/	/3?34
M3Set,PWMFrequency,600/	/3u34,600
M1Get,RangeHigh/	/1?39
M1Get,RangeLow/	/1?40
M1Get,Status/	/1?11
M3Get,TipPresence/	/3?36
M1Get,TotalMoveCount	/1?71
M1Set,TotalMoveCount,20/	/1U71,20
M1Get,TotalMoveCountAbs	/1?72
M1Get,TotalTravelDist	N/A
M1Set,TotalTravelDist,20/	N/A
M1Get,TotalTravelDistAbs	N/A
M1Get,TravelPos	N/A
M3Set,TravelPos,210/	N/A
M1Get,WorkingLen/	/1?41
M1Set,WorkingLen/	/1u41,1250
Most recent error, regardless of the command that resulted in the error	/1?24 (Get LastDeviceErrorInfo)

Table G-3 *CIB Get and Set parameter commands*

Command Processor	Omni Embedded
C0Get,<Property>	/F?<Property>[R]
C0Get,BootVersion/	/F?25
C0Get,CANBaudRate/	N/A (Part of ?76)
N/A	/FU5,0 (Set CAN0 baud rate)
N/A	/FU6,0 (Set CAN1 baud rate)
C0Set,EnableCavroCAN,0/	N/A Command Processor Function only
C0Get,FWRevision/	/F?23
C0Get,FWVersion/	/F?24
C0Get,Input1/	/F?2
C0Get,Input2/	/F?3
C0Get,Input3/	/F?4
C0Get,Input4/	/F?5
C0Get,Input5/	/F?6
C0Get,Input6/	/F?12
C0Get,Input7/	/F?7
C0Get,IO1/	/F?0
C0Set,IO1,1/	/Fu2,1
C0Get,IO1Mode/	/F?13
C0Set,IO1Mode,1/	/Fu2,3 Note that the arguments to the embedded Set commands do not match the arguments for the command processor Set commands.
C0Get,IO2/	/F?1
C0Set,IO2,1/	/Fu3,1
C0Get,IO2Mode/	/F?14

Table G-3 *CIB Get and Set parameter commands*

Command Processor	Omni Embedded
C0Set,IO2Mode,1/	/Fu3,3 Note that the arguments to the embedded Set commands do not match the arguments for the command processor Set commands.
C0Set,Int5Mode,0/	/FU2,0
C0Get,IPAddress/	/F?41
C0Set,IPAddress,192.168.0.198/	/FU20,192/FU21,168/FU22,0/ FU23,198
C0Get,MACAddress/	/F?40
C0Get,Output1/	/F?10
C0Set,Output1,1/	/Fu0,1
C0Get,Output2/	/F?11
C0Set,Output2,1/	/Fu1,1
C0Get,Output3/	/F?8
C0Set,Output3,1/	/Fu4,1
C0Get,Output4/	/F?9
C0Set,Output4,1/	/Fu5,1
C0Get,PortNumber/	/F?42 (Get Port Num. for Cmd. Processor)
N/A	/F?44 (Get Port Num. for Omni Embedded)
C0Set,PortNumber,9898/	/FU24,9898 (Set Port Num. for Cmd. Processor)
N/A	/FU34,9897 (Set Port Num. for Omni Embedded)
C0Get,PowerOnMinutes/	/F?50
C0Set,PowerOnMinutes,50/	/FU50,50
C0Get,PowerOnMinutesAbs	/F?51 (Get Absolute Power On Minutes)
C0Get,PowerUps/	/F?52
C0Set,PowerUps,100/	/FU52,100

Table G-3 *CIB Get and Set parameter commands*

Command Processor	Omni Embedded
C0Get,PowerUpsAbs	/F?53 (Get Absolute Num. of Power Ups)
C0Get,RS485Config/	N/A (Part of ?76)
C0Set,RS485Config,0/	/FU15,0 also available in OE: /FU10,0 (Set RS232 baud rate)
C0Get,SerialNumber/	N/A
C0Get,SWRevision/	N/A Command Processor Function only
C0Get,UserData/	N/A Command Processor Function only
C0Set,UserData,1234/	N/A Command Processor Function only
C0Get,VoltageFU1/	/F?30
C0Get,VoltageFU2/	/F?31
C0Get,VoltageFU3/	/F?32
C0Get,VoltageFU4/	/F?33
N/A	/F?34 (Get V3.3 voltage)
N/A	/F?35 (Get V1.8 voltage)
N/A	/F?36 (Get TPPO voltage)
N/A	/F?76 (Reports CIB Configuration)

Table G-4 *Cavro CAN device command formats*

Command Processor	Omni Embedded
E<DeviceID>-<FrameType>,<CommandString>/	/<DeviceID><CommandString> or / <DeviceID><FrameType><CommandString>
E10-1,ZgA10A100G5R/	/aZgA0A100G5R , /a1ZgA0A100G5R

G.3 Solutions

G.3.1 Aspirate and Dispense with ADP

The following sequence of command strings demonstrates how to program the Omni left arm to pick up and drop off a disposable tip, detect liquid using the ADP's pLLD mechanism, then aspirate and dispense a reagent with the ADP.

The hardware configuration for this example is as follows:

- Left arm Omni with a Universal Z with ADP, addresses are:
  - Axis X: 1
  - Axis Y: 2
  - Axis Z: 3
  - ADP: 0

  Z is assumed to have a configured travel pos of 200 in Command Processor Mode.

Table G-5 *Command Processor and Embedded example with ADP*

Command Processor Mode	Embedded Mode
// Initialize left arm and ADP. Make sure no tip is mounted on ADP. // Note that this W command is an ADP command that is passed to the ADP for execution.	
A0CavroInit **E0-1,WR**	**1((/3Z)/1Z/2Z)/0WR**
// Move arm to DiTi tip rack positioned at X:335, Y:150	
A0MoveAbs,335,150,210	**1((((/3A200)/1A335/2A150)/3A210)**
// Pick up tip, starting from position 100, descending up to 75 mm from 100, and stopping at or // before 85 mm below 100	
A0PickTip,100,75,85	**1/3P100,75,85**
// Raise arm to position Z:200 and move arm to a reagent rack positioned at X:450, Y:75	
A0MoveAbs,450,75,200	**1((/3A200)/1A450/2A75)**
// Set ADP to pLLD mode. Liquid detection must use single detection mode. // Note that this U command is an ADP command // and is passed to the ADP for execution.	
E0-1,U70R	**1/0U70R**

// Detect liquid with submerge distance set to 2 mm using single detection mode.
// Start from position 180 and descend as far as 0.
// Note that for pLLD mode, both the Omni and the ADP need to execute the B command to
// detect liquid with the ADP.	

Table G-5 *Command Processor and Embedded example with ADP*

Command Processor Mode	Embedded Mode
E0-1,B1R **A0DetectLiquid,180,0,2,,1** ***Note:*** These commands must be executed in parallel. Please see Section 7.11, "Using the Omni Command Processor APIs"and Section 8.4.1, "Integration Sequences"for information about the BeginSendCmd method.	**1(/0B1R/3u21,2u25,1B180,0)**
// Aspirate 500 ml. Note that this P command is an ADP command that is passed to the ADP // for execution	
E0-1,P500,1R	**1/0P500,1R**
// Raise arm to position Z:200, then move arm to a sample rack positioned at X:250, Y:125, then // lower arm to position Z:100	
A0MoveAbs,250,125,100	**1(((/3A200)/1A250/2A125)/1A100)**
// Dispense 500 ml. Note that the D command is an ADP command that is passed to the ADP // for execution.	
E0-1,D500,1R	**1/0D500,1R**
// Raise arm to position Z:200, then move arm to a waste container positioned at X:50, Y:50 and // eject tip	
A0MoveAbs,50,50,200 **E0-1,ER**	**1((/3A200)/1A50/2A50)/0ER**

G.3.2 Aspirate and Dispense with XLP Pumps

The following sequence of command strings demonstrates how to program the Omni right arm to prime the tubing system, pick up and drop off disposable tips, detect liquid using the cLLD mechanism, and aspirate and dispense a reagent using XLP pumps.

The hardware configuration for this example is as follows:

- Right arm has a Dual Z with two XLP pumps, each with a 3-port valve. Axis addresses are:
  - X: 4
  - Y: 5
  - Z1: 6
  - Z2: 8
  - XLP1: 9
  - XLP2: C

Z1 and Z2 are assumed to have a configured travel pos of 180 in Command Processor Mode.

Table G-6 *Command Processor and Embedded Example with XLP pumps*

Command Processor Mode	Embedded Mode
// Initialize right arm and pumps	
A1CavroInit **E9-1,ZR** **E12-1,ZR**	**1((/6Z/8Z)/4Z/5Z)/9ZR/CZR**
// Move arm to a waste container at position X:700,Y:50 and lower Z1 and Z2 to a height of 100	
A1MoveAbs,700,50,100,100	**1(((/6A180/8A180)/4A700/5A50)/ 6A100/8A100)**
// Prime system. Note that these commands are a mixture of Omni embedded commands and // XLP pump commands that are passed on to the XLP pumps 9 and C for processing	
E9-1, gOM10A3000IM10A0G5R **E12-1, gOM10A3000IM10A0G5R** ***Note:*** These commands must be executed in parallel. Please see Section 7.11, "Using the Omni Command Processor APIs"for information about the BeginSendCmd method.	**1(/9gOM10A3000IM10A0G5R/ CgOM10A3000IM10A0G5R)**
// Move arm to a tip rack at position X:375, Y:150, with Z1 at position 180 and Z2 at position 180	
A1MoveAbs,375,150,180,180	**1((/6A180/8A180)/4A375/5A150)**
// Pick up tips: both Z1 and Z2 starting at position 55 with a limit of 20 and an acceptance limit // of 50. Note that under most circumstances both Z axes will use the same starting position, // limit, and acceptance limit. If both Z axes are picking tips from separate racks, these parameters // might differ between the two axes. **// Note:** Axis 6 pick tip start position is 65 in OE mode to prevent picking tip simultaneously with // axis 8	
A1PickTip,55,20,50 **M6MoveAbs,180** **M8MoveAbs,180**	**1(/6P65,20,50/8P55,20,50)/6A180/ 8A180**
// Move arm to reagent rack at position X:450, Y:75	
A1MoveAbs,450,75,180,180	**1(/4A450/5A75)**
// Detect liquid with submerge distance set to 2 mm, a start position of 100, and a search limit of 0 // for both Z axes	
A1DetectLiquid,100,0,2	**1(/6u21,2B100,0/8u21,2B100,0)**
//One XLP pump aspirates 500 ml, another aspirates 250 ml	

Table G-6 *Command Processor and Embedded Example with XLP pumps*

Command Processor Mode	Embedded Mode
E9-1,IM10P1500R **E12-1,IM10P750R** ***Note:*** These commands must be executed in parallel. Please see Section 7.11, "Using the Omni Command Processor APIs"for information about the BeginSendCmd method.	**1(/9IM10P1500R/CIM10P750R)**
// Raise both Z axes to position 180, move arm to a sample rack at position X:250, Y:125, and // lower both Z axes to position 150	
A1MoveAbs,250,125,150,150	**1(((/6A180/8A180)/4A250/5A125)/ 6A150/8A150)**
// Pumps dispense reagent. Note that this D command is an XLP command and is passed // directly to the XLP pumps for execution.	
E9-1,D1500R **E12-1,D750R** ***Note:*** These commands must be executed in parallel. Please see Section 7.11, "Using the Omni Command Processor APIs"for information about the BeginSendCmd method.	**1(/9D1500R/CD750R)**
// Raise arm to position 180, move arm to a waste container located at position X:700, Y:50, and // drops off tips with a drop limit of 220 and a drop acceptance limit of 200	
A1MoveAbs,700,50,180,180 **A1DropTip,180,220,200**	**1(/6A180/8A180)/4A700/5A50)(/ 6E180,220,200/8E180,220,200)**

This page has been left intentionally blank.

Index

D

E

F

G

M

T

U

V

W

X

Y

Z

This page has been left intentionally blank.