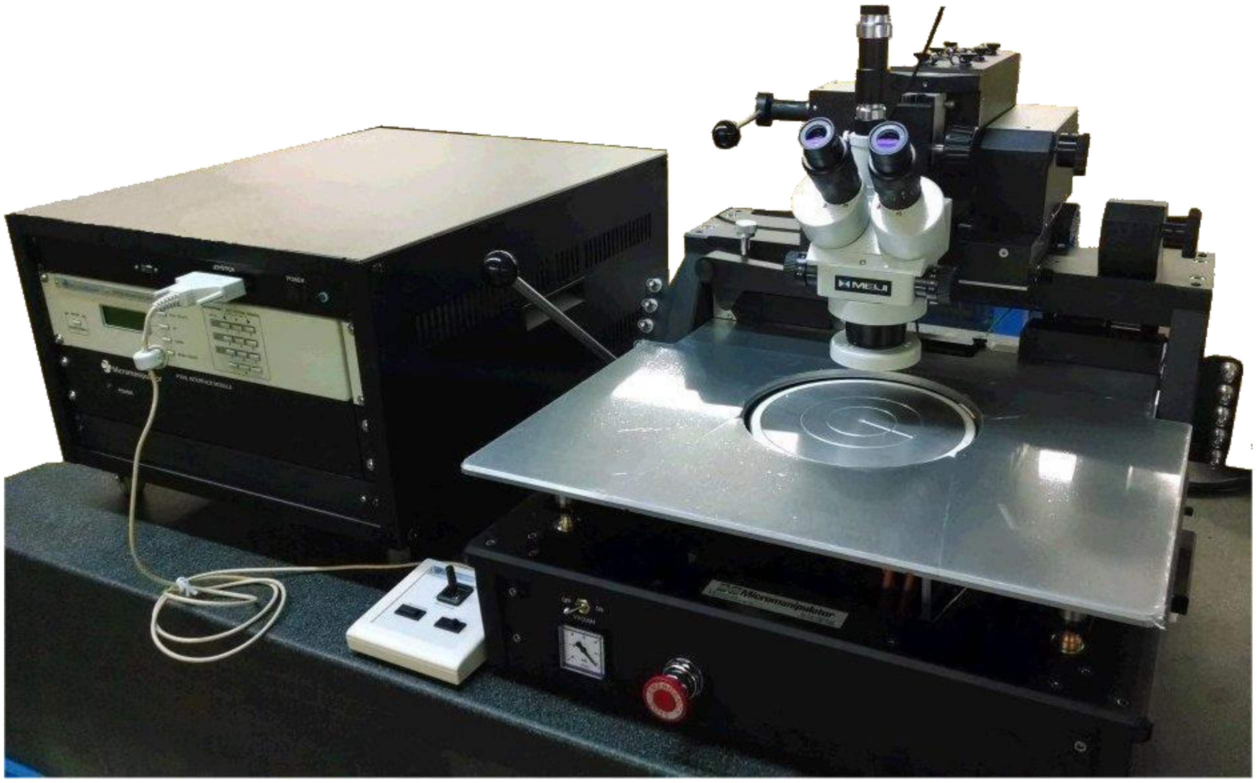


P200L Prober

OPERATOR MANUAL

Revision B
1 December 2014

PN: A1017659

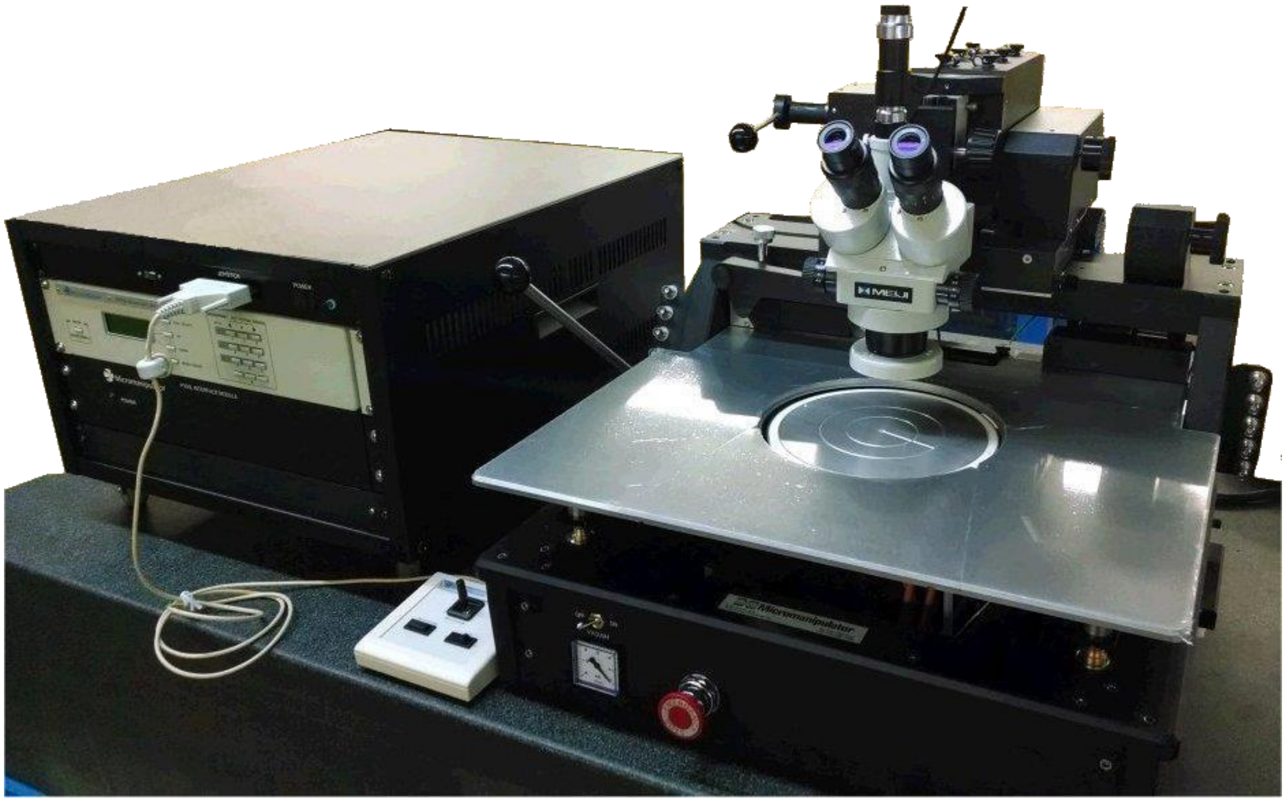
Micromanipulator

1555 Forrest Way
Carson City, Nevada 89706

sales@Micromanipulator.com

Table of Contents

1 System Overview

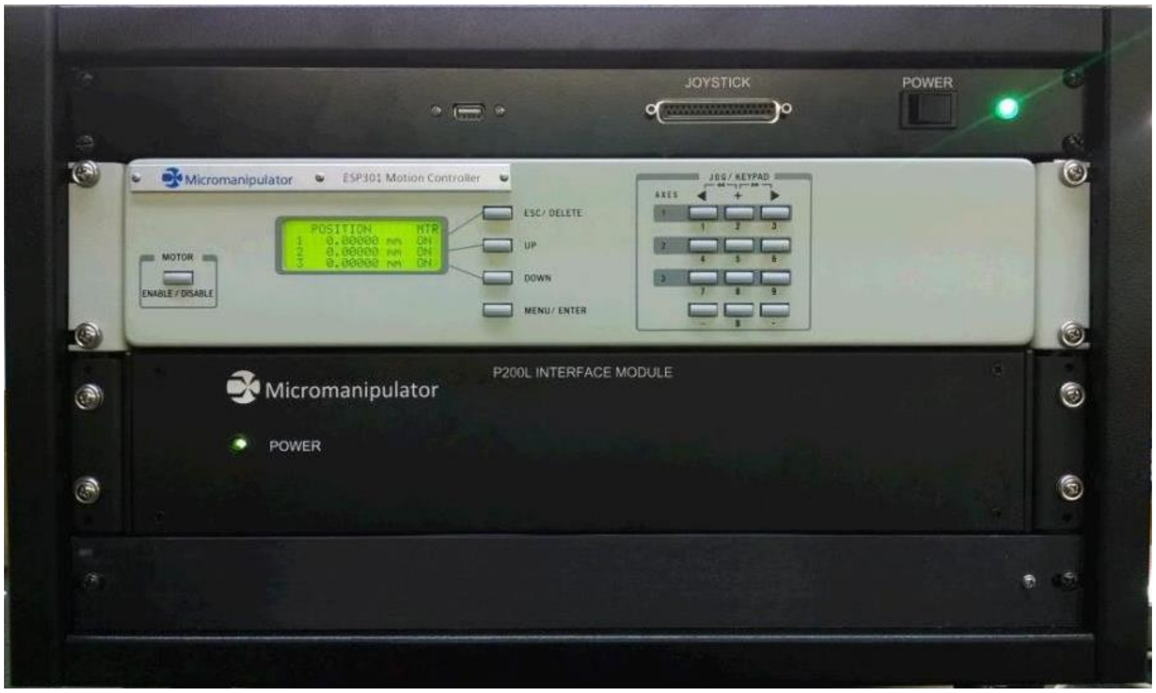


The P200L is a motorized 200mm prober with programmable capability. It is ready-to-run out of the box. No host software is required or provided. The basic P200L system consists of the P200L Controller, P200L Prober, and customer specified microscope.

The prober has 200mm of motorized XY stage travel and 25 mm of motorized Z Platen travel (Z travel may be reduced to accommodate different height wafer chucks). The microscope has 50mm of XY and Z travel. The type of microscope varies based on the customer requirements.

The P200L controller drives the motorized axes and manages the limit logic. The joystick provides operator control of the motorized stage and platen.

1.1 P200L Controller

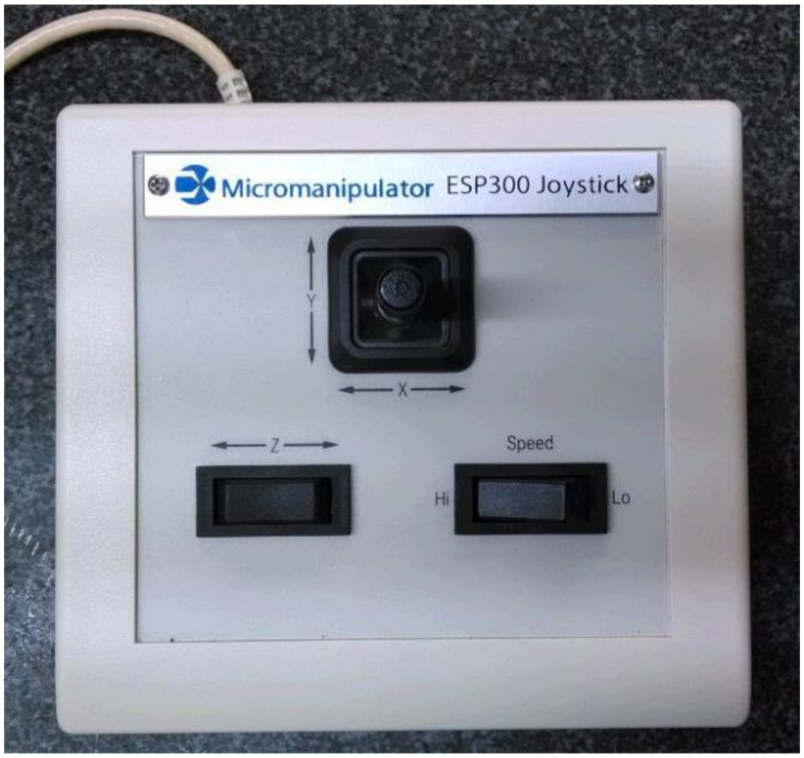


The P200L Motion Controller manages the motor driven axes of the prober. The standard configuration has a motorized XY stage with Z Platen. The motion controller has the following components:

- ESP301 Motion controller
- P200L Interface Module
- ESP300 Joystick
- Prober power management

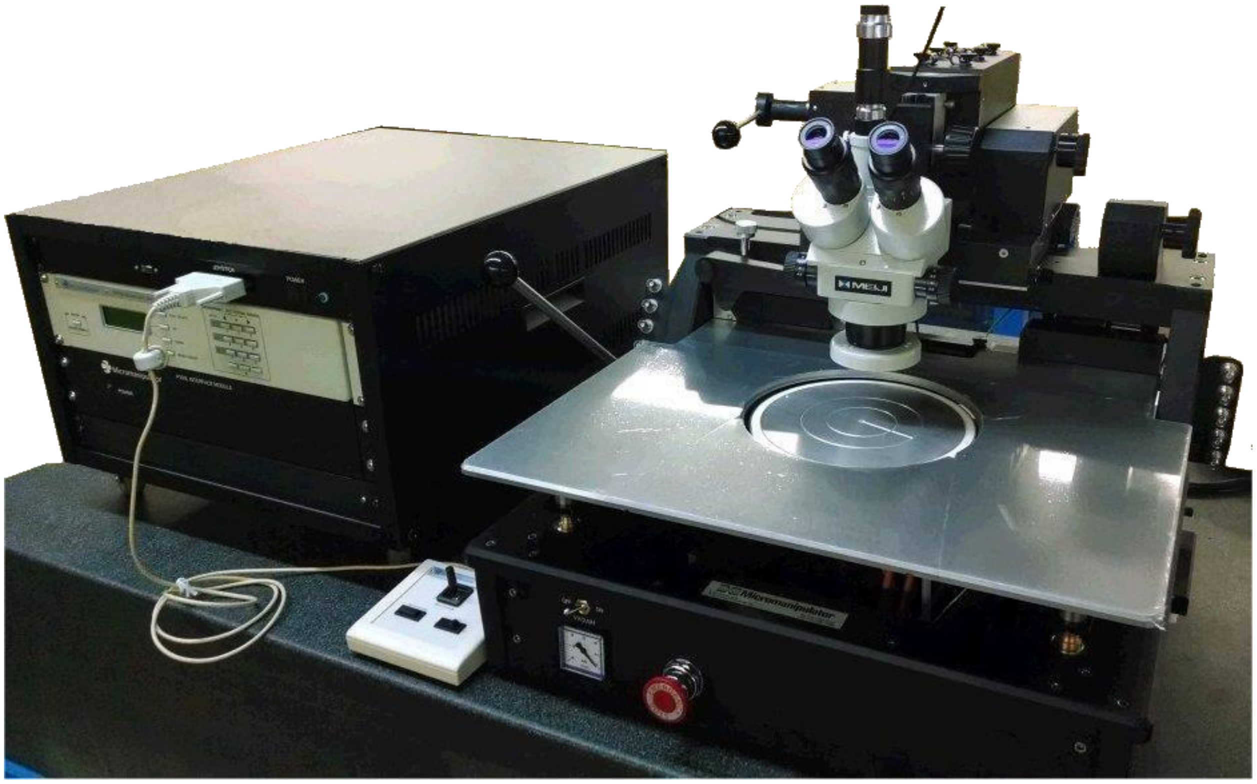
The ESP301 controller provides position readout and more advanced positioning controls. An extensive command set is available through the standard RS-232 and USB interfaces. An IEEE-488 interface is optional. The Emergency Stop (EMS) switch on the prober is connected to the ESP301. Pressing the EMS switch immediately stops all motors.

The P200L Interface Module provides digital and analog compatibility between the P200L prober and the ESP301 controller. It is backward compatible with the Micromanipulator 8860 prober stages.



The ESP300 joystick has three axes of control and a High/Low speed switch.

1.2 P200L Prober



The test station proper consists of a motorized wafer stage assembly, microscope bridge and platen assembly. The motorized stage assembly has a chuck mounted to it which holds a wafer to be either viewed or probed. Several chuck options are available. The microscope bridge provides for a microscope to be conveniently mounted over the probing surface. Manipulators are mounted on the platen, which accommodates both vacuum and magnetic base manipulators. The platen is motorized as well.

2 SPECIFICATIONS

2.1 Platen

- Accepts up to 10 or more vacuum or magnetic base manipulators
- 10 BNC strain reliefs, (5 on the left side and 5 on the right)
- 10 vacuum connections for vacuum base manipulators, (5 on the left side and 5 on the right)
- Platen surface grounded to chassis and accessible via connector located on the right side of the Test Station
- Four point support and drive system for the platen
- Platen positioning resolution is 0.1 um
- Total motorized platen travel: 1 inch (25.4 mm) but may be less depending on the chuck option installed.
- Platen Quick Lift Control raises platen and microscope with adjustable microscope lift delay with a range of 1.5 inch (38.1 mm) maximum

NOTE: When platen QUICK LIFT is locked in the UP position, motorized positioning of XY Stage and Z PLATEN are inhibited

2.2 X - Y Stage Movement

- Cross roller translation way construction
- Stage translation and vacuum stage permanently planarized at factory
- Range of motion: 200 mm
- Resolution (minimum move): 0.1 micron
- Repeatability of positioning better than 15.0 microns
- Travel speed: 12 mm/sec. maximum
- Manual ultrafine theta rotation control full 15 degrees

2.3 Vacuum Chuck

- 200 mm with wafer tweezers slot
- Gold plated brass (ambient version) for low contact resistance (stainless steel optional)
- Flatness: +/-0.0005 in. (+/-12 microns) or better
- Electrical isolation 5 gigaohms @500VDC minimum
- Chuck surface: Is electrically connected to the BNC terminal at rear right side of baseplate

2.4 Microscope Bridge

- X/Y translation: 50 mm by 50 mm permanently planarized at the factory
- Cross roller translation way construction
- Lift delay; (raise probes while maintaining focus, offset adjustable)
- Independent microscope lift
- 100 mm vertical range of motion by microscope lift delay arm
- Bridge support structure independently mounted to baseplate
- Microscope swing-up function to provide clear access to chuck via platen quick lift and microscope lift delay

2.5 Test Station Dimensions and Finish

- 27 in. wide by 27 in. deep by 22 in. high (68.6 cm by 68.6 cm by 55.8 cm)
- Leg footprint 24 in. wide by 24 in. deep (61 cm by 61 cm)
- Weight: 280 lbs. (123 kg.)
- Grained black anodized aluminum for long life

2.6 Facility Requirements

- Power: 100-240V AC; 60/50 Hz; 8 Amps
- Internally overcurrent protected to 10 Amps at 125 VAC and 6.3 Amps at 250 VAC with 100 and 65 AIC ratings respectively and must be on the load side of over current protection device with 10,000 AIC rating.
- Vacuum: 20 in. mercury.
- The P200L Series probe stations are equipped with an emergency stop of automated motion. The emergency stop is not intended to operate as an Emergency Manual Override (EMO). The P200L series probe station, and its accessories, are components of an overall wafer test system. In applications where an EMO is required, it is a facility's responsibility to supply an overall EMO.
- Consult the factory for installation recommendations on vibration isolation table integrated systems for Seismic Zone 4 locations.

3 INSTALLATION AND SETUP PROCEDURES

3.1 PRE-INSTALLATION PROCEDURES

3.1.1 Shipping Carton Inspection

Carefully check the packing materials for signs of visible damage when they are received. If any damage is noted, photograph the damaged cartons immediately, and contact the shipping company to report the damage and to obtain a damage claim form. Notify **The Micromanipulator Co., Inc.** Service Department.

3.1.2 Test Station Removal

When removing the test station from the cartons, check all components for shipping damage. If possible, photograph any damage for use when claims are submitted to the shipping company. If the test station is damaged, notify the shipping company and The MICROMANIPULATOR Co. promptly. All containers should be checked against the packing list as soon as received. If a shortage is noted, notify **The MICROMANIPULATOR Co.** immediately. We will not be responsible for shortages unless they are reported at the time of receipt. Include the following on all claim reports:

- INSTRUMENT MODEL NUMBER
- INSTRUMENT SERIAL NUMBER
- YOUR PURCHASE ORDER NUMBER
- YOUR LIST OF THE SHORTED MATERIALS

3.1.3 Test Station Inspection

All instruments should be thoroughly inspected immediately upon receipt. If the contents are damaged, or the instrument fails to operate properly, the carrier and The Micromanipulator Co., must be notified at once. The following documents are required to support claims:

- ORIGINAL FREIGHT BILL AND BILL OF LADING ORIGINAL INVOICE OR PHOTOCOPY OF ORIGINAL INVOICE
- COPY OF PACKING LIST
- PHOTOGRAPHS OF THE DAMAGED EQUIPMENT AND CONTAINER(S)

Should any damage be noted, contact **The MICROMANIPULATOR Co.** Customer Service Department.

3.2 INSTALLATION AND SETUP

NOTE: **The MICROMANIPULATOR Co.** recommends the use of a vibration/isolation table to provide extreme stability and freedom from vibration. Recommended tables are available from **The MICROMANIPULATOR Co.**

When all major components have been removed from their shipping cartons, place the base of the test station in its final position on the aforementioned table, ensuring that it is level in both X and Y directions.

NOTE: Prior to any type of installation or operation the microscope locks and stage locks (both X and Y) **MUST** be removed. Each of these locks is tagged with a card with RED printing. For systems with integrated thermal chuck plumbing option, remove lower stage lock screws on Y stage (Ref. Fig. 3-1)

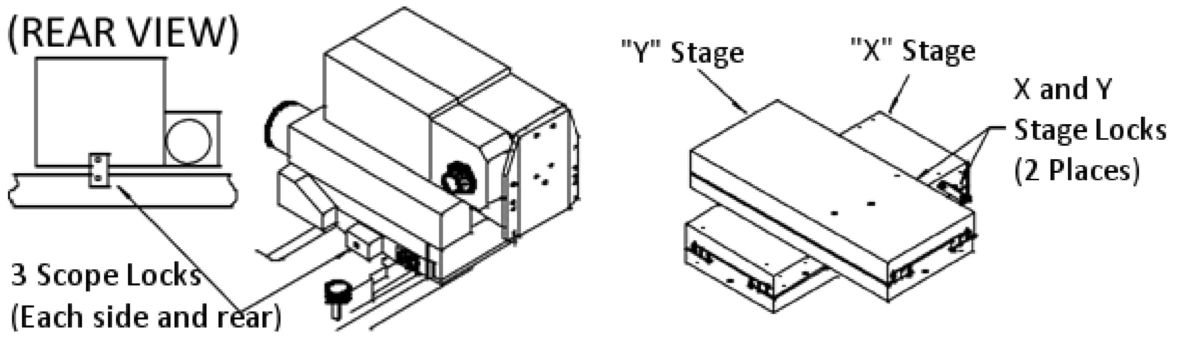
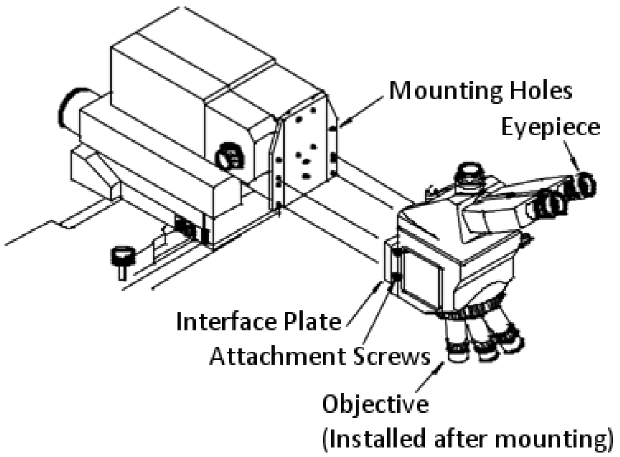


FIGURE 3-1. LOCKING DEVICES

3.2.1 Microscope Installation

Install the microscope as shown in Figure 3-2. Attach the scope to the lower set of holes on the scope mounting plate using the four large knurled screws attached to the scope. The other set of holes are used with packaged parts and socket card adapters.

Attach the threaded objective lenses to the turret, making certain that the color code markings are followed. That is, insert the objective having a particular color band into the



threaded hole having the same color code. This aids in parcentering and parfocality. Finally, install the eyepieces by carefully inserting them into the microscope head receptacles.

FIGURE 3-2. MICROSCOPE MOUNTING

3.2.2 Interface Cable Connection

CAUTION
DO NOT APPLY POWER TO THE **P200L CONTROLLER** AT THIS TIME.

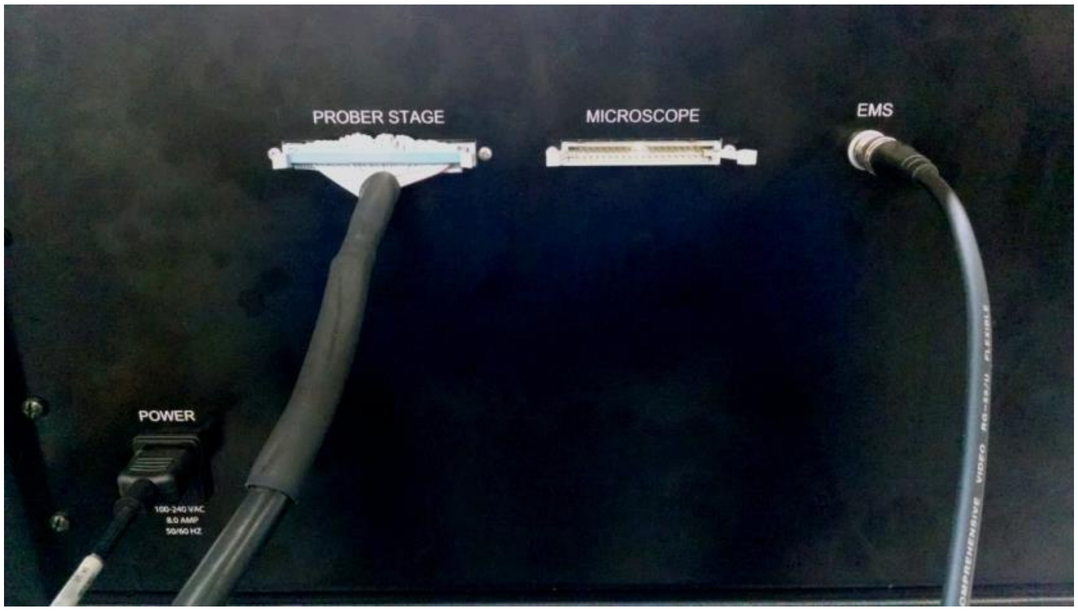
Stage Interconnect Cable (Programmable/Motorized): The STAGE interconnect cable (50 pin to 50 pin) supplies power to the STAGE motors, the PLATEN motor and LIMIT circuitry.

STAGE/PLATEN: The DC power delivered to the stage is used to drive the **X , Y** and **Z** drive motors and powers the HALL EFFECT limits at the end of each direction of travel.

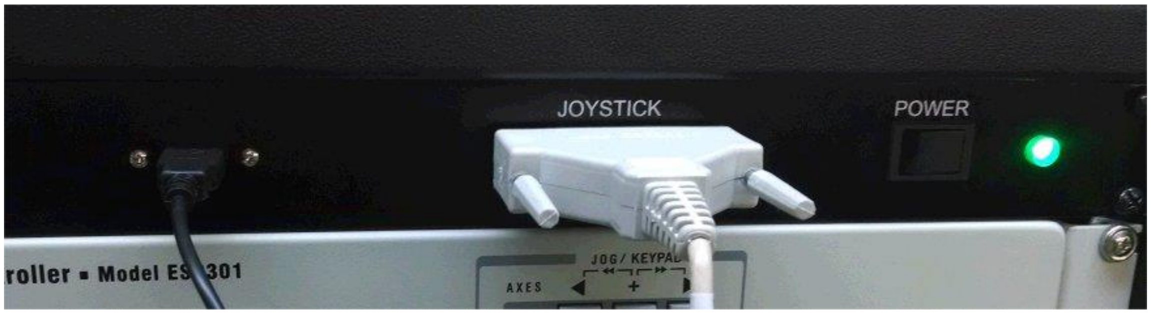
EMERGENCY STOP (EMS): When this button is depressed, the ESP301 controller stops all motor motion. The BNC connection labeled “EMS” is connected to the P200L controller “EMS” input.

Activating the Emergency Stop switch causes the ESP301 controller to turn off power to the motors. The “Enable Motors” button on the front of the controller must be pressed to enable motor motion. The ESP301 may need to be re-homed in order to maintain absolute positional accuracy after activating the Emergency Stop. Errors displayed on the front panel can be cleared by viewing the “ERRORS” menu contents. The EMERGENCY STOP button also powers off the HC1000, if one is present.

3.2.3 P200L Controller Connections

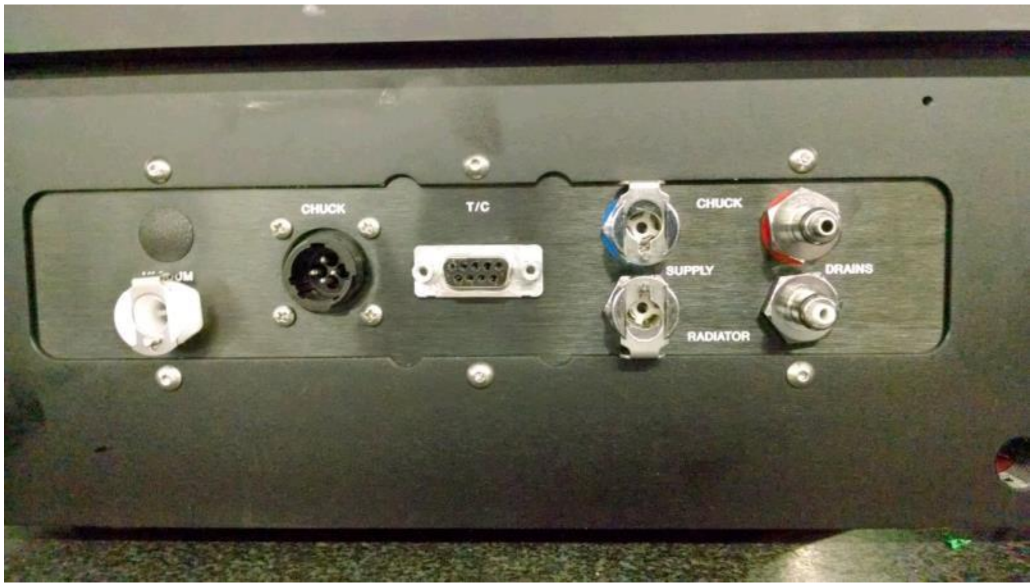


Connect the STAGE [50 conductor], AC Line cord, and EMS BNC cable to the P200L Controller rear panel.

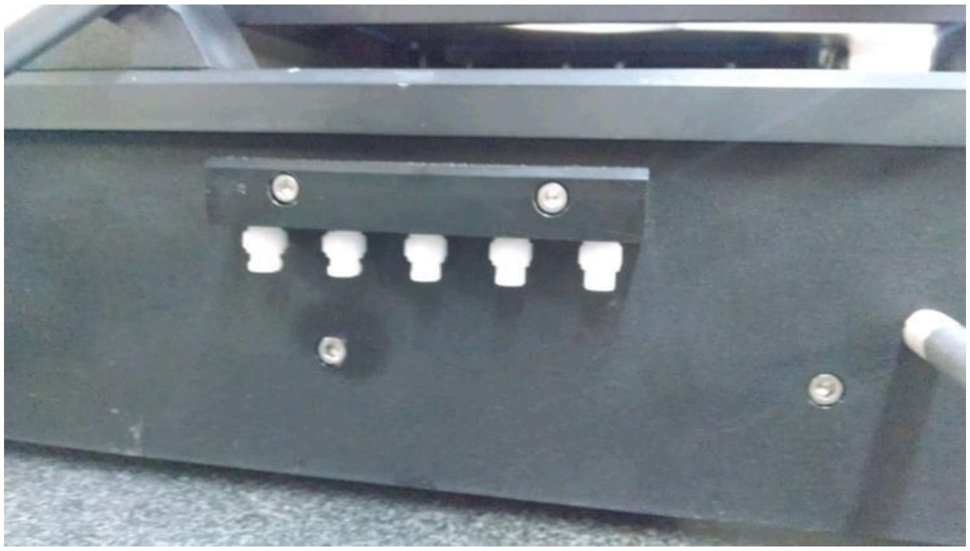


Connect the JOYSTICK to the front panel of the P200L. If applicable, attach the host USB cable to the front panel at this time.

3.2.4 Vacuum Connection

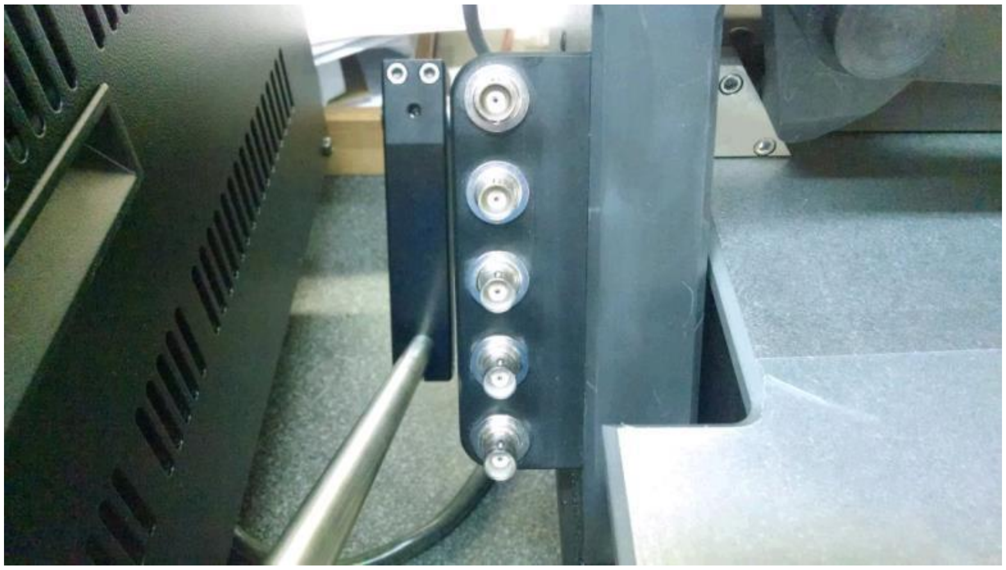


Connect a vacuum source by applying 20mm Hg to the system vacuum input located on the rear of the system.



If vacuum manipulators are to be used, the vacuum lines used for the manipulators can now be connected to the vacuum manifolds located on the left and right sides of the test station. Vacuum adapters are included.

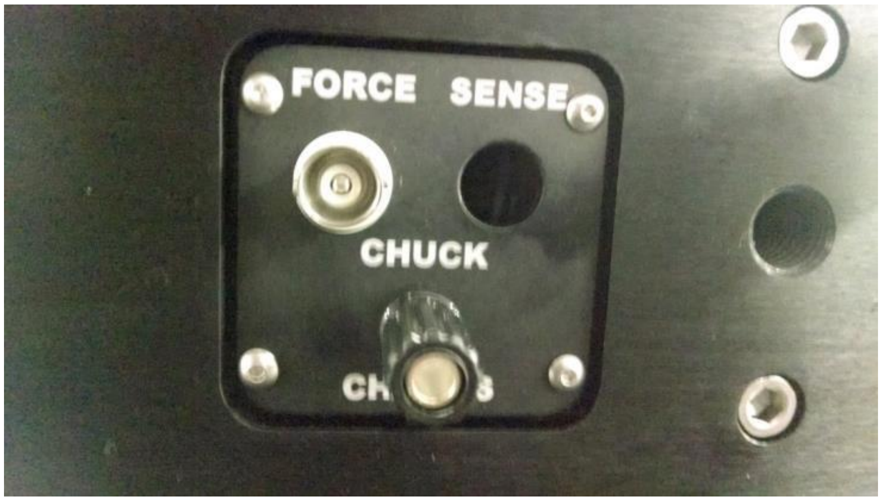
3.2.5 BNC Connections



Ten isolated BNC feed throughs are provided to interface from the DUT's to the test equipment, five on the left side of the platform and five on the right. They provide isolation and strain relief between any probe cables and external test equipment. These connectors are mounted in Delrin in order to isolate them from the chassis. Each BNC is tested to ensure better than 5 teraohms isolation between each conductor and its outer shell. Also each BNC has greater than 5 teraohms isolation between itself and any other BNC connector as well as the test station.

There are two optional electrical interface connection brackets readily available as standard accessories. These kits are easily installed in place of the standard BNC bracket. The two types of connector kits are: **TRI BRACKET KIT** (two brackets with 5 each isolated 3 lug triax [TRB] connectors) or **TRI-BNC BRACKET KIT** (two brackets with 2 each isolated 3 lug triax [TRB] connectors and 3 each isolated BNC connectors).

3.2.6 Grounding Connections



Located on the right side of the test station is a mounting plate with at least two electrical connectors. The plate connectors are labeled **FORCE, SENSE** and **CHASSIS**. Typically the sense connector is not present unless KELVIN design vacuum chucks are ordered with the prober. TRIAXIAL KELVIN as well as COAXIAL KELVIN vacuum chucks are available upon request.

This plate facilitates field and factory upgrades for vacuum chuck electrical wiring. For example, when the prober is purchased with a triaxial vacuum chuck design, the chuck BNC may be converted at the factory to a 3 lug triaxial jack, or an adapter that ties the BNC shield to the triaxial guard (leaving the triaxial shield to be connected to chassis ground) may be used .

Probers with this connection plate can be field upgraded to accept special KELVIN vacuum chucks.

⚠ Chassis ground connection provides the earth ground connection for thermal chuck accessories. It is provided for the user to connect to test and measurement equipment grounds to minimize noise. This connection must be connected to test equipment ground when used with thermal chucks to prevent electrical shock hazard in the event of a failure with the thermal chuck.

The CHUCK FORCE BNC is used to connect the surface of the VACUUM CHUCK to an external ground plane, or applicable potential. The coax shield is isolated from the prober chassis.

The CHASSIS BANANA JACK is used to connect the test station chassis to the user external equipment so that both have a common ground.

3.2.7 Single Point Grounding

The presence of ground loops in the fixturing and test station directly affect the electrical noise levels present in a measurement. Current flowing through the ground loops may produce electrical noise sufficient to make it impossible to resolve the characteristics of the actual device measurement.

The MICROMANIPULATOR Co.'s prober design has incorporated a special grounding technique. Ground straps connect all major chassis elements to a singlel grounding block inside the station. A CHASSIS GROUND connector on the side provides the user access to this ground point. The wafer surface of our standard vacuum chucks has better than 5 x 10^9 ohms (at 500 VDC) electrical isolation from the chassis.

3.3 SYSTEM FAMILIARIZATION

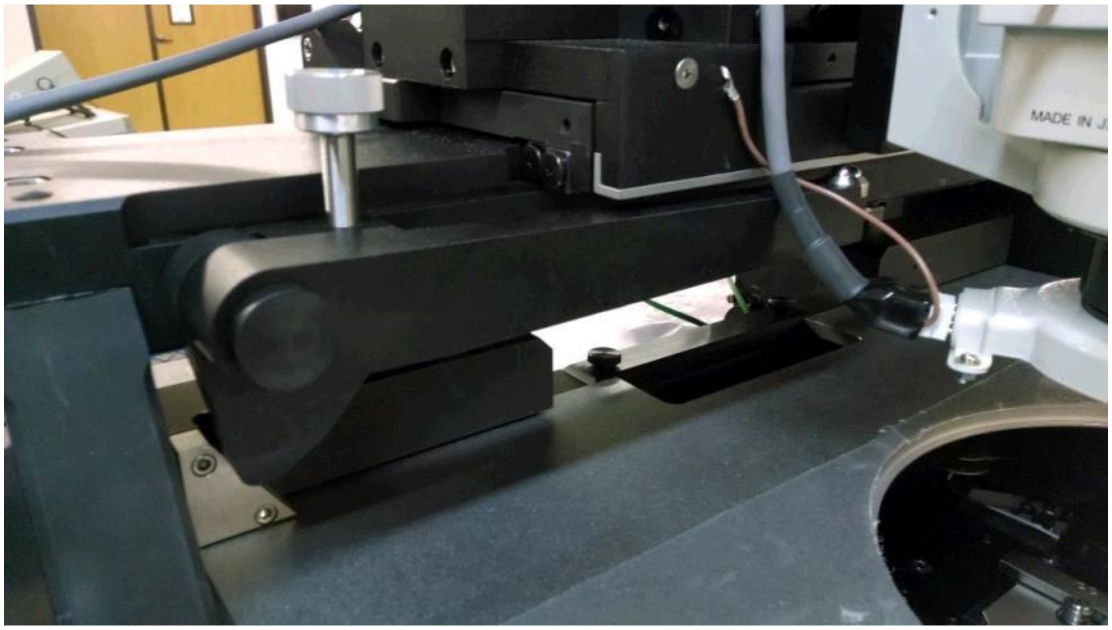
This section is designed to get you familiar with the individual components of the test station prior to beginning operation.

3.3.1 Manual Platen Quick Lift Lever

The platen QUICK LIFT lever allows all probes, probe card holder, and platen to be raised vertically for sample changes or adjustments. This lever can also lift the microscope through the microscope lift delay arm. The actual amount of microscope lift depends upon the microscope position along the **Y** axis and the setting of the LIFT DELAY.

NOTE: Lifting the platen disables platen motion on the ESP301 controller. It is necessary to turn the Z axis motor back ON after using Quick Lift. Z only works when the quick lift handle is in the down position.

3.3.2 Microscope Lift Delay



The LIFT DELAY adjustment determines at what point the microscope begins to lift as the platen is raised by the QUICK LIFT handle. This is useful when stepping from site to site with a platen lift between each X/Y move. The microscope will remain in focus as long as the microscope lift delay is set to a dimension greater than the platen lift distance.

⚠ Using the platen quick lift may cause damage to the microscope if the lever slips from the operator's hand while raising or lowering the platen with microscope lift delay engaged. Be sure to have firm control of the lever when raising or lowering the platen quick lift.

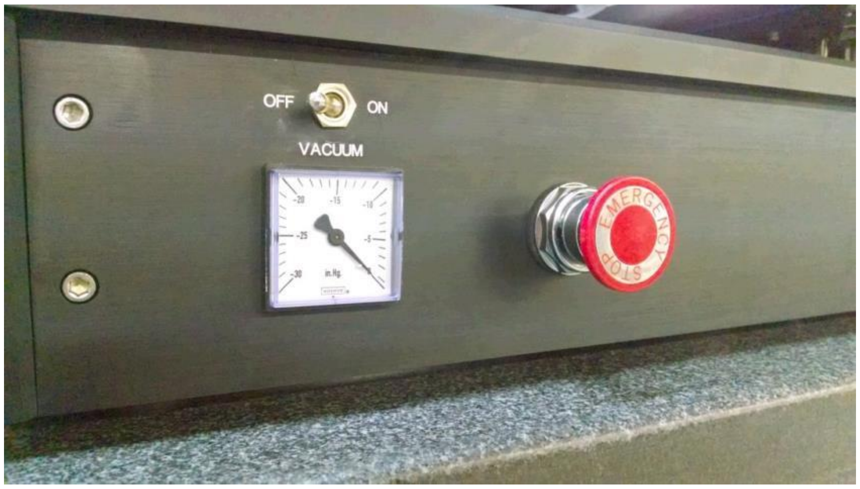
3.3.3 Platen Support System

The PLATEN SUPPORT SYSTEM is a heavy-duty four point platen support system with mechanical isolation for enhanced stability.

3.3.4 Theta Adjust

The THETA ADJUST knob allows you to adjust the chuck, in a thirty degree arc ($\pm 15^o$) for minor substrate rotational adjustments.

3.3.5 Vacuum Gauge



The VACUUM GAUGE indicates the amount of vacuum, measured in inches-mercury (in. Hg), that is being delivered to the test station vacuum system which includes the CHUCK, and MANIPULATOR VACUUM MANIFOLDS. The vacuum inlet is located on the rear of test station.

Station Vacuum: The STATION VACUUM supplies two manifolds, located on both sides of the station.. The valves supply vacuum for up to 10 manipulators (5 each side). There should be a continuous vacuum source of minimum **.25 CFM, 20 IN-HG.**

Wafer Vacuum: WAFER VACUUM is controlled via the VACUUM ON/OFF switch located on the front of the test station.

3.3.6 Emergency Stop Button

The EMERGENCY STOP BUTTON is utilized to immediately stop any and all motion of the PLATEN and STAGE in emergency situations. The ESP301 Motors must be re-enabled from the ESP301 front panel after disengaging the EMS button. The EMS button stops any thermal chuck control when and HC1000 is attached to the prober.

3.3.7 Stage

The STAGE is a motor driven (X and Y motion) mechanical assembly which carries the VACUUM CHUCK. Its primary function is to position the DEVICE UNDER TEST (DUT) which can either be wafers or packaged parts using a socket adapter.

3.3.8 Vacuum Chuck

The VACUUM CHUCK provides a precise, planer, rigid base for the wafer or device. The top disk of the chuck is manufactured to have excellent planarity and standard chucks are gold plated for good electrical contact and corrosion resistance. One shallow slot, in the front of the chuck, is provided for tweezers access to the wafer (N/A on thermal chucks). When the chuck is properly seated on the chuck post, the tweezers slot opens to the front of the station.

3.3.9 Chuck Release Knob

The CHUCK RELEASE KNOB locks the vacuum chuck firmly onto its mounting post which is attached to the stage assembly. Ensure that the chuck hub pin is aligned with the notched keyway on the chuck post before tightening chuck release.

3.3.10 Platen

The PLATEN is a highly stable flat platform, made of heavy ground stainless steel, that is parallel to the vacuum chuck surface. It is designed to accommodate both vacuum and magnetic base manipulators as well as probe card holder adapters. The platen moves only in a true planer vertical direction.

4 ACCESSORY INSTALLATION

4.1 INTRODUCTION

This section will instruct you on how to install the following types of accessories:

- AMBIENT CHUCK
- THERMAL CHUCK
- PROBE CARD HOLDER
- MANUAL MANIPULATOR(S)

4.2 AMBIENT AND THERMAL CHUCK REMOVAL AND INSTALLATION

There are two basic types of CHUCKS that may be mounted in the Test Station. These chucks are AMBIENT and THERMAL. The AMBIENT chuck is equipped with only vacuum to hold the wafer to its surface while under test; however, the THERMAL chuck may be equipped with a plumbing option for coolant and vacuum so that the wafer can be tested at temperature.

All Test Stations are equipped with a PLATEN with a removal wedge (Ref. Fig. 4-1) which is usually removed unless your operation employs multiple manipulators in such a number that the center section is needed for manipulator mounting. If the wedge is inserted, it must be removed prior to removal or installation of a chuck.

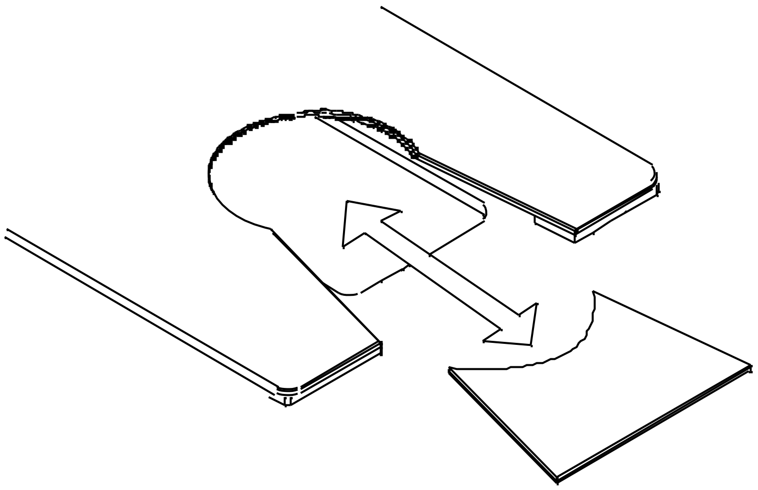


FIGURE 4-1. PLATEN WEDGE REMOVAL

The **P200L Test Station** can be purchased with an assortment of three chuck configurations which are:

- AMBIENT chuck (gold and stainless steel) only with no plumbing.
- LOW NOISE AMBIENT chuck with appropriate internal wiring.
- THERMAL chuck with plumbing and an accompanying AMBIENT chuck that can replace the THERMAL chuck.
- SOCKET STAGE ADAPTER for packaged devices.
- Plumbing for other manufacturer's THERMAL CHUCKS are available upon request.

This section will deal with chuck removal and installation in the above conditions only. If you ordered only the AMBIENT chuck without plumbing you can still have a THERMAL chuck installed. To install a THERMAL chuck, contact **The MICROMANIPULATOR, Co.** to determine if the installation can be performed at your site or if the test station must be sent back to the factory.

Caution, the thermal chuck is capable of reaching temperature extremes, do not touch the chuck surface at any time, risk of severe burn may result and the surface can become contaminated when at ambient temperatures.

4.2.1 Ambient Chuck and Socket Stage Adapter Removal and Installation

To remove the AMBIENT chuck from its post (Ref. Fig. 4-2), first clear the platen of all manipulators. Center the chuck relative to the platen so that it will clear the access hole. Then, move the microscope to the far right and tilt the microscope up with the lift lever. Once these preliminary steps have been accomplished, perform the following steps:

- Disconnect the vacuum, from the chuck, by unplugging it from the under side.
- Disconnect all electrical connections (CHUCK potential) also located on the under side.
- Rotate the CHUCK LOCKING KNOB counterclockwise and very carefully lift the chuck straight up using both hands to remove it from the chuck mounting post.

CAUTION - Do not allow chuck to come in contact with platen, equipment damage may result. This can best be accomplished when the stage is front and center, microscope lift delay is engaged, and the quick lift is up.

To install the CHUCK, first clear the platen of all manipulators. Center the chuck post relative to the platen so that the CHUCK will clear the access hole. Then move the microscope to the far right and tilt the microscope up with the lift lever. Once these preliminary steps have been accomplished, perform the following steps:

- Lower the CHUCK to the platen access hole and place it on the post.
- Attach the vacuum hose and electrical connections.
- Ensure that the chuck hub pin is aligned with the notched keyway on the mounting post.
- Rotate the CHUCK LOCKING KNOB clockwise so that the CHUCK is firmly locked in place. Press on the opposite side of the chuck when setting the lock.

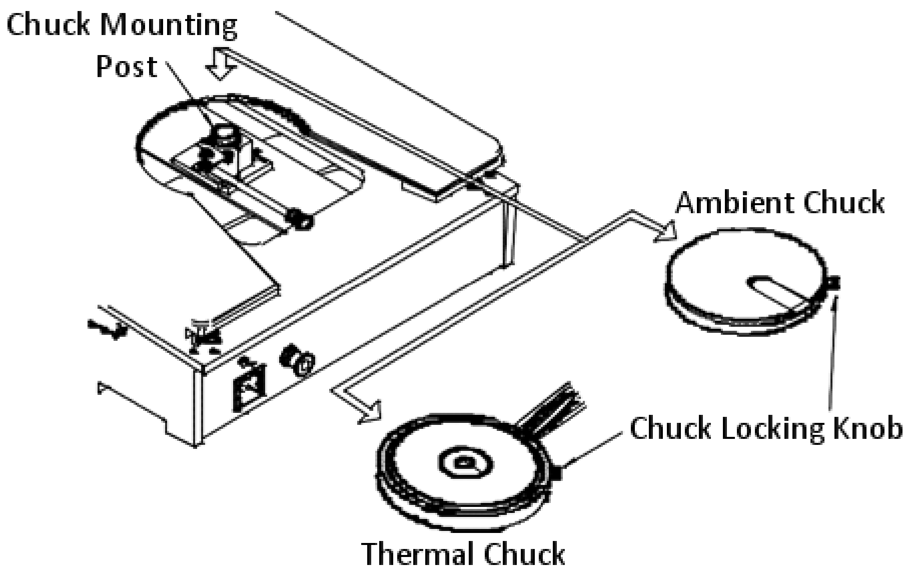


FIGURE 4-2. CHUCK REMOVAL AND INSTALLATION

4.2.2 Thermal Chuck Installation

NOTE: If you are replacing an AMBIENT chuck with a THERMAL chuck, ensure that you remove the AMBIENT chuck in accordance with section 4.2.1. The installation of a THERMAL chuck is more complicated than installing an AMBIENT CHUCK. This is due to the plumbing, coolant, vacuum, and electrical connections that are necessary.

To install a THERMAL CHUCK, perform the following steps:

1. Ensure that the CHUCK LOCKING KNOB is backed out past its mounting ring on the inside wall.
2. Ensure that the CHUCK MOUNTING POST is centered in the PLATEN ring with hose and cabling exiting to the right.
3. Place the chuck loosely down on the chuck mounting post and align the chuck hub (theta) pin to its slot by gently rotating the chuck from side to side until the pin slips into place. Then tighten the CHUCK LOCKING KNOB.
4. Starting from the rear, attach the associated hoses in the following sequence:
   - VACUUM (SMALL HOSE)
   - COOLING RING (SMALL HOSES)
   - THERMAL CHUCK COOLANT (LARGE HOSES)
5. After all of the hoses have been connected, make the following electrical connections:
   - CHUCK POTENTIAL
   - CONNECTOR THERMAL
   - CHUCK HEATER/THERMOCOUPLE CONNECTOR

4.3 PROBE CARD HOLDER (8800-FPC-FRX)

The optional PROBE CARD HOLDER (Ref. Fig. 4-4) is designed to mount to the platen and extend over the chuck and under the microscope objectives.

The card holder is equipped with two small mounting blocks and one larger block with a theta adjustment knob. All three of the blocks have two screws - one for mounting the holder to the platen and the other to adjust coarse planarity.

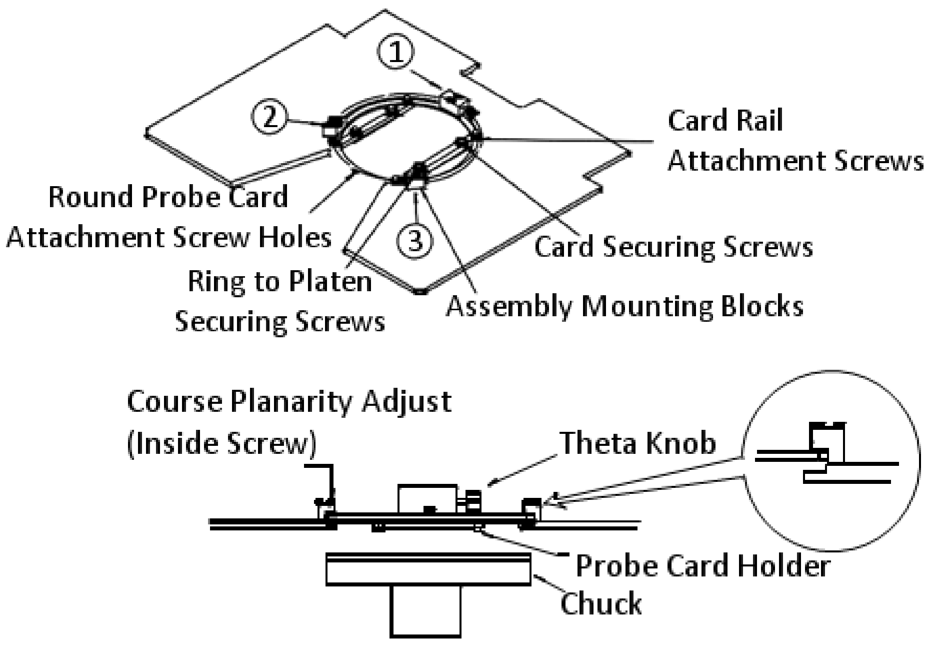

1
2
3
Card Rail Attachment Screws
Round Probe Card Attachment Screw Holes
Card Securing Screws
Ring to Platen Securing Screws
Assembly Mounting Blocks
Course Planarity Adjust (Inside Screw)
Theta Knob
Probe Card Holder
Chuck

FIGURE 4-4. PROBE CARD HOLDER INSTALLATION

CAUTION - Do not allow probe card holder to come in contact with the wafer chuck, equipment damage may result.

To install the PROBE CARD HOLDER, perform the following steps:

1. Place the PROBE CARD HOLDER in the platen receiving ring with the mounting blocks resting on the edge of the platen ring. Align mounting block pins and the ring securing screws with the holes in the platen and tighten.

2. Install mounting block number 1 first (Ref. Fig. 4-4) loosely. Install numbers 2 and 3 and apply light pressure on angles on corner of bearing block ensuring contact of bearing to outer diameter of ring as shown in figure 4-4 detailed circle.

3. Adjust the course planarity screws so that visually the PROBE CARD HOLDER is in the same plane or level with the platen.

4. If you have to rotate the card holder, adjust the THETA ADJUST KNOB on the larger mounting block to obtain the proper alignment.

4.4 PROBE CARD INSTALLATION

The WEDGE should be removed prior to removing or installing the probe card. The probe card is inserted into the PROBE CARD HOLDER from the under side (Ref. Fig. 4-5) of the HOLDER. The probe card contacts may face the front or rear of the station, depending upon which direction the test cable leaves the test station. Once the card is inserted, tighten the four CARD SECURING SCREWS to hold the card in place. A cable exit port is provided at the rear of the test station platen.

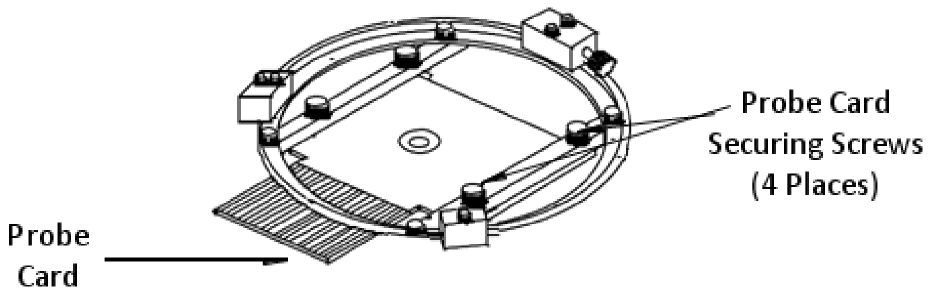


FIGURE 4-5. PROBE CARD INSERTION

4.5 MANIPULATORS

Prior to placing the manipulators, the platen quick lift should be in the down position and the manipulators should roughly centered in their respective travels. The manipulators are positioned on the platen around the periphery of the wafer. Suitable probe holders are attached via collet shaft and closure. This assembly is then positioned so the probe tip approximates its final position. For final positioning, the X, Y, and Z axis adjustments on the manipulator are maneuvered to position the probe tip precisely on the correct pad or feature of the wafer. This operation applies to both manual and programmable manipulators.

To save time when gross positioning, position the microscope over the area to be probed. Switch the illuminator on and position the probe point under the cone of the light. Be sure the Z setting is high enough to clear the specimen with the manipulator base in full contact with the platen surface and vacuum engaged. Fine positioning can now be accomplished with increasing magnification and fine manipulator control.

5 MAINTENANCE

5.1 INTRODUCTION

Maintenance of the **P200L Test Station** consists primarily of proper lubrication of the moving parts and a continuing effort to keep the equipment free from dust and other contaminants, such as wafer fragments or other particles.

Disconnect all power sources prior to maintenance or servicing. Disconnect controller power cord from line and rear panel to prevent inadvertent power application while performing maintenance or service. Reference the safety section on electrical hazards for details. Service should only be performed by a Micromanipulator authorized technician. Contact the Micromanipulator Service department with any problems.

The prober lubrication interval is dependent upon the location and usage levels of the prober. If it is located in an open area subject to dust or other contaminants, the station should be cleaned and lubricated more frequently than directed in this section. However, if the station is located in a clean area, such as a clean room, where the environment is more closely monitored, then less maintenance is required. In the event the test station is located in a corrosive atmosphere, it must be cleaned more frequently to remove corrosive materials.

At frequent intervals, all work surfaces should be wiped down with clean, lint free cloths or wipers, moistened with denatured alcohol to aid in this process. Also, periodically clean the lenses on the microscope with a lens brush, followed by a nonabrasive wiper moistened with denatured alcohol. View the eyepieces while rotating them to reveal particles that should be removed.

The Micromanipulator Co. programmable positioning products are designed to give the user many years of specified performance when proper care, handling and maintenance practices are observed.

Calibration of semiautomatic probers may be required if the instrument in question has experienced any degradation due to excessive wear caused by improper maintenance, impact, tampering or unauthorized adjustment to any of the pertinent positioning system components such as leadscrews, nuts, bearing ways or support hardware. Further, if the operator notices any degradation in performance or inability to hit targets dimensionally within the specified positioning performance of the product, calibration may be required. It should be noted that lack of lubrication, excess dirt or grime on the positioning system components as well as improper setup may result in inaccurate positioning and potential damage. The user should consult the operations manual or contact the factory with any questions or concerns arising with the use of these instruments.

5.2 LOWER LINEAR STAGE

The X and Y axis linear bearings (Fig. 5-2) should have lubricant (**The MICROMANIPULATOR Co.** P/N A1011545) applied with a clean, dry cloth . This cleans and protects the bearings from rust and dirt buildup. Do this every 50 hours of use, as the minimum.

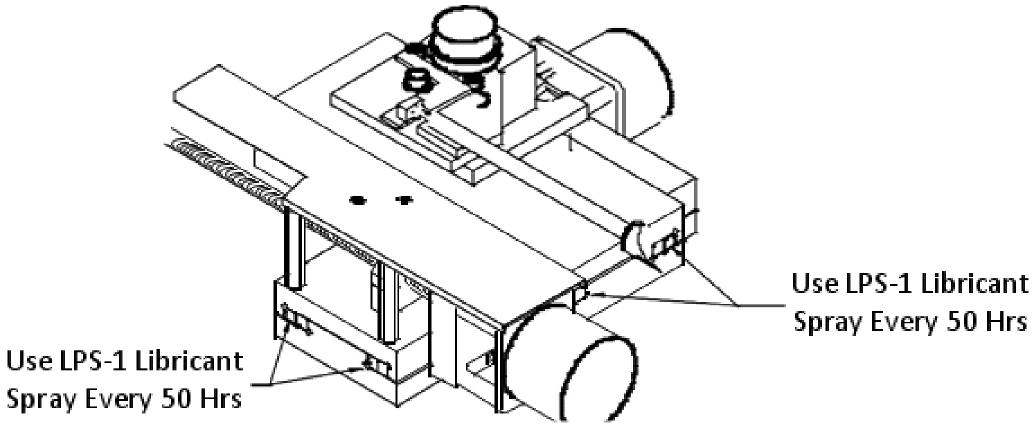


FIGURE 5-2. X AND Y AXIS STAGE BEARING LUBRICATION

5.3 LEADSCREWS

The leadscrews (Ref. Fig. 5-3) should be lubricated every 100 hours minimum with the lubricant provided (**The MICROMANIPULATOR CO.** P/N A1011545). Prop the front of the station up. Locate both the **X** and **Y** lead screws and lightly apply an even coat of lubricant along the full length of the screws. Then move the stage through its full travel in both directions. This ensures that the lubricant spreads equally along the full length of the screws. Wipe off any excess lubricant that builds up on either side of the lead nut.

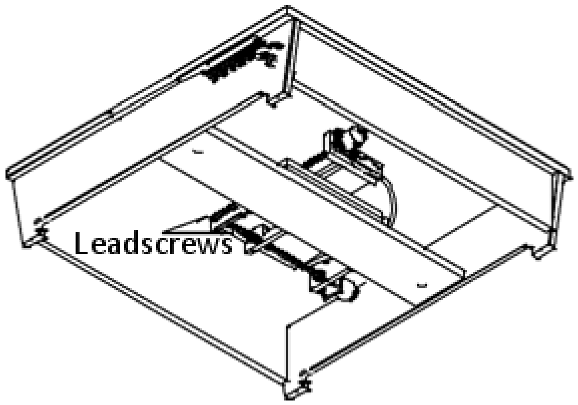


FIGURE 5-3. LUBRICATING THE X AND Y LEADSCREWS

5.4 Z DRIVE (PLATEN) JACKSCREWS

The raising and lowering of the PLATEN is controlled by four (4) jackscrews located under each corner of the PLATEN (Ref. Fig 5-4). Each of these screws must be lubricated to ensure smooth operation of the **Z** motion. Apply a light coating of lubricant (**The MICROMANIPULATOR Co.** P/N A1011545) on each of the screws. As with the stage, move the PLATEN to its extremes of travel and wipe off any excess lubricant collected on the bottom of the riser screws.

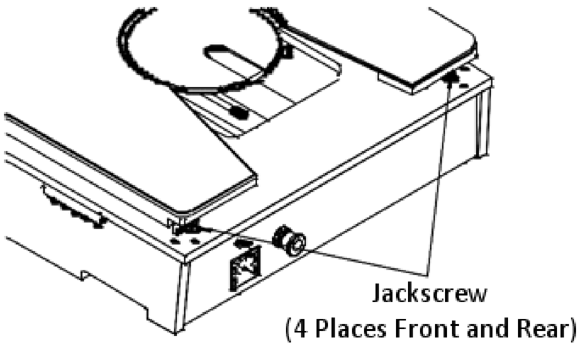


FIGURE 5-4. PLATEN JACKSCREW(S) LOCATION

5.4.1 CHUCK PLANARIZATION

When a chuck is replaced with another, it should be checked for PLANARIZATION. To check and/or adjust for planarization (Ref. Fig. 5-5), perform the following procedure:

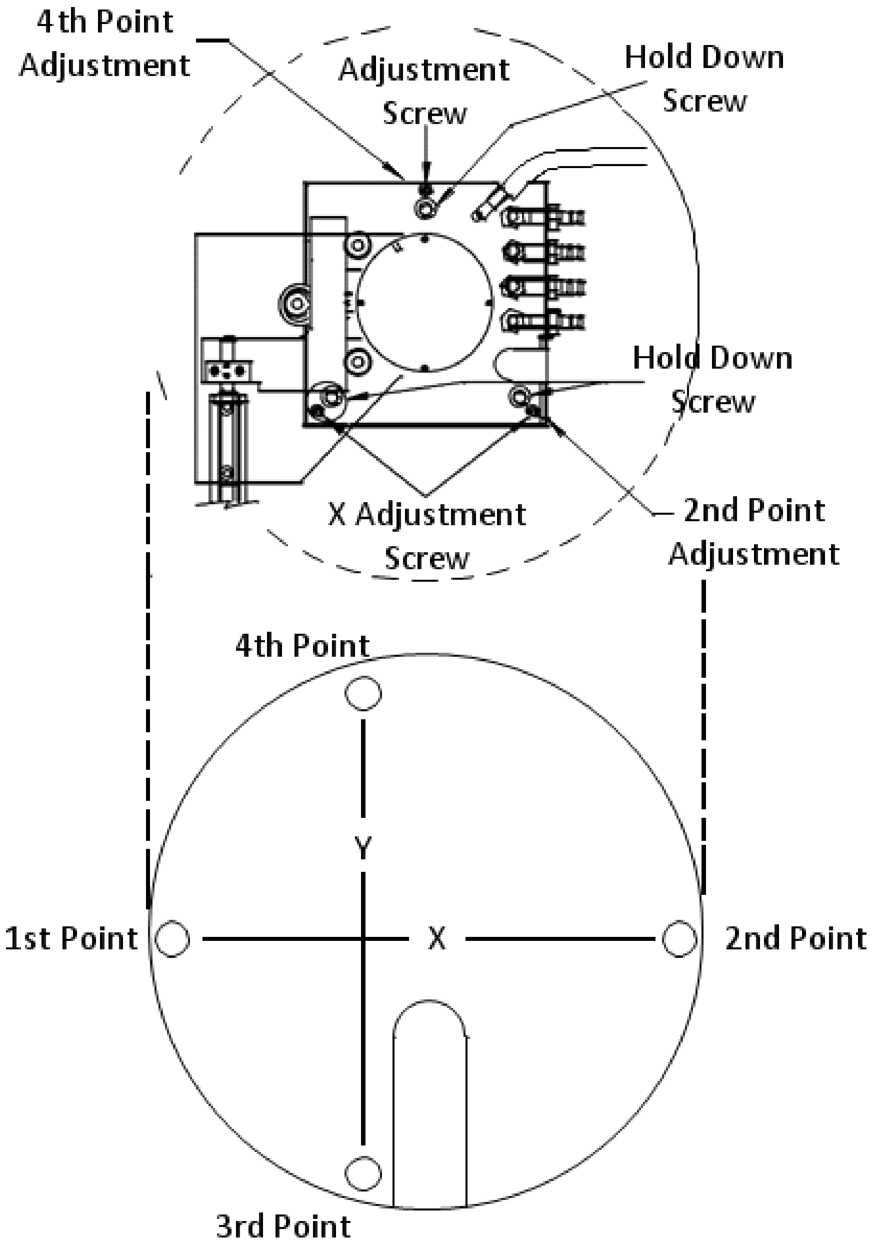
1. Select four fixed points at the chuck corners approximately as shown in figure 5-5.
2. Leave the microscope at its approximate center of travel. Select the 20X objective, if available.
3. Moving **ONLY THE CHUCK,** position point 1 under the microscope and focus on the chuck surface using the 20X objective.
4. Move the chuck to the second point. If the second point is **NOT** in focus, loosen the hold down screw located next to the second focus point **X** adjustment screw and adjust until second point is in focus.

NOTE: Adjust the second focus point only -- do not readjust the first point.

5. Move back to the first focus point and re-focus scope.
6. Move to second focus point and check focus. Readjust per step 3 above until in focus. Repeat steps 4 and 5 until both the first and second focus points both remain in focus.

7. Move chuck to third focus point and focus on the chucks' surface.
8. Move chuck to fourth focus point (**DO NOT** change focus). If fourth point is not in focus, loosen **Y** adjustment hold down screw and adjust **Y** adjustment screw until fourth point is in focus then lightly re-tighten hold down screw.
9. Move back to third focus point and check focus.
10. If necessary, repeat steps 6 through 8 until third and fourth focus points are in focus.
11. After all focus points are in focus, check to ensure that all three adjustment hold down screws are tightened.
12. When all wafer chuck assemblies are equally adjusted to this standard (or each other), replanarizing the prober for each chuck, when replaced, is no longer required.

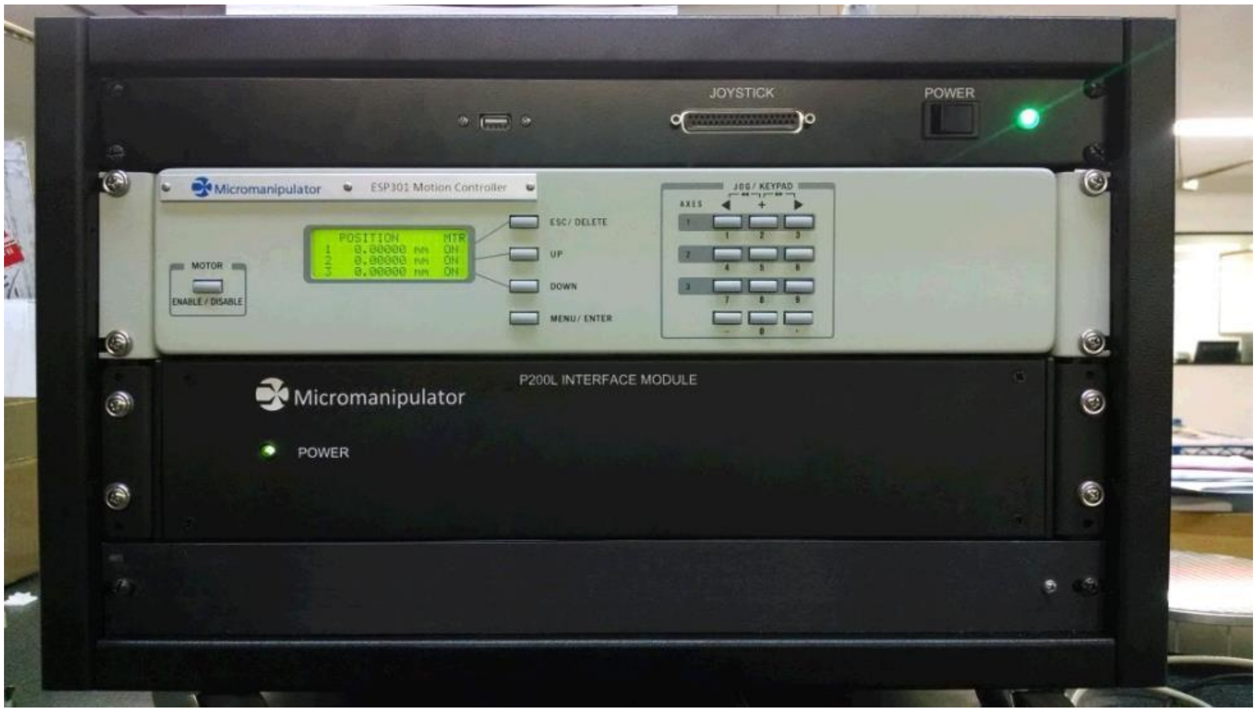
NOTE: Individual TRIAXIAL CHUCKS and 8800 series wafer chuck assembly planarization is different than planarizing a chuck to a prober. All chucks manufactured after AUGUST 1994 can be separately PLANARIZED to a standard and consequently to each other. Contact **The MICROMANIPULATOR Co.** for more details on this type of global planarizing.



FIGURE

5-5. CHUCK PLANARIZATION

6 P200L Controller



6.1 ESP301 Motion Controller

The ESP301 Motion Controller provides 3 axes of stepper motor control. It has basic navigation functions including:

- Move to an XYZ position or by an XYZ distance
- Home XYZ
- Jog XYZ
- Reset positions
- Execute stored programs
- Configuration

The ESP301 also has a USB interface available for host control.

NOTE: The ESP301 front panel and host commands refer to each axis by a number, not a letter. The axis number to letter mapping is as follows: 1 = X 2=Y 3=Platen (Z)

The complete ESP301 manual can be found at:
http://assets.newport.com/webDocuments-EN/images/14294.pdf

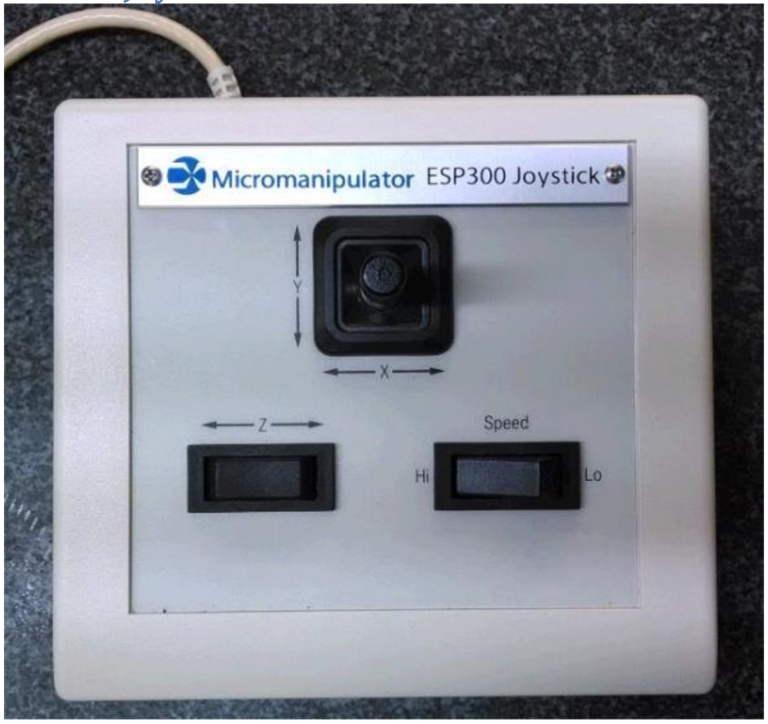
VISA drivers required to communicate via USB can be found at:
http://lumen.ni.com/nicif/US/GB_NIDU/content.xhtml?du=http://www.ni.com/download/ni-visa-5.4/4230/en/

Labview Drivers for the ESP301 can be found at:
http://assets.newport.com/webDocuments-EN/images/ESP301_LABVIEW_Drivers.zip

6.2 P200L Interface Module

The P200L Interface Module converts the ESP301 motor control and limit inputs to those compatible with the P200L prober. The interface module also provides backward compatibility with the 8860 prober.

6.3 ESP300 Joystick



The joystick controls stage and platen motion as follows:

Joystick Axis Actuation	Stage/Platen Motion
X Right	Left (+X)
X Left	Right (-X)
Y Away	Towards (+Y)
Y Towards	Away (-Y)
Z Right	Down (+Z)
Z Left	Up (-Z)

The Speed button toggles between high speed and low speed jog modes.

NOTE: The move speed used by the MOVE ABSOLUTE and MOVE RELATIVE commands will be the same as the most recent joystick or jog move. Therefore be sure to jog an axis at high speed prior to performing an absolute or relative move.

6.4 ESP301 Front Panel Operation

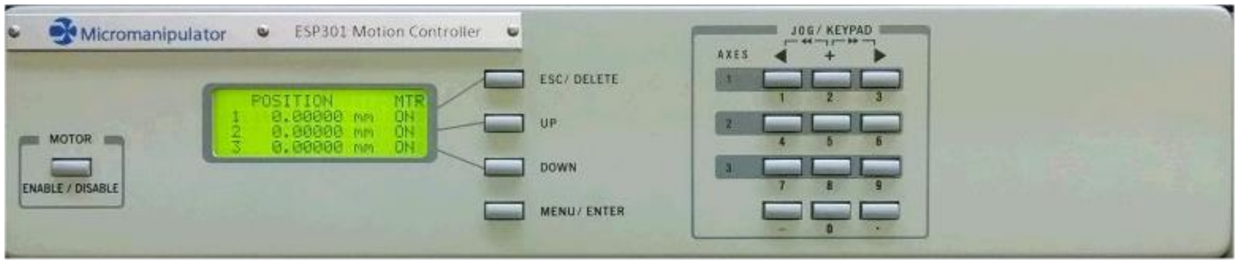


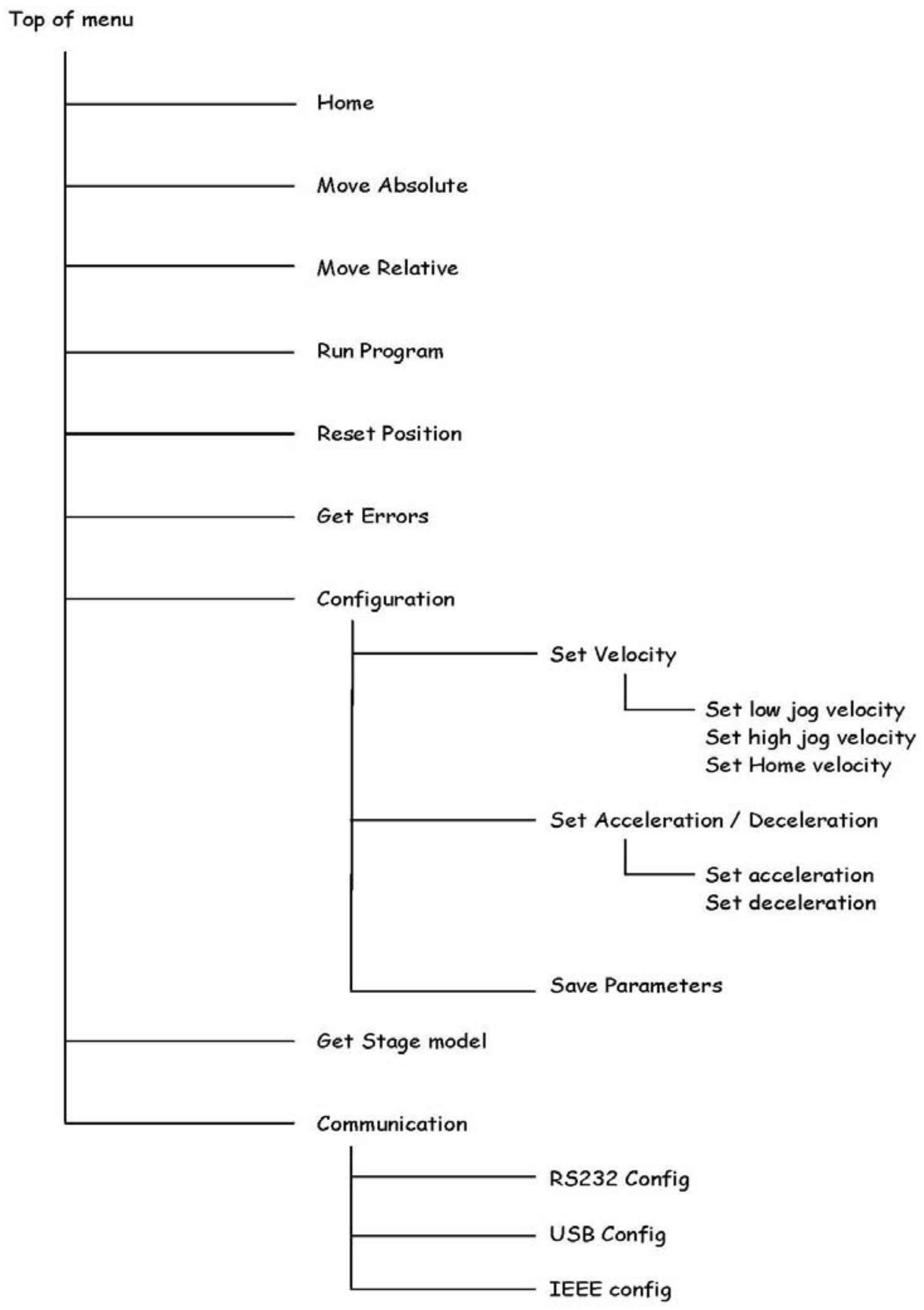
Figure 6.1 ESP301 Front Panel

The ESP301 provides basic navigation and configuration through the front panel interface. The POSITION display shows the position and motor on state for each axis. Axis 1 controls the X stage, axis 2 controls the Y stage, and axis 3 controls the Z Platen. The Cartesian coordinate system matches that of the DUT.

The buttons and their associated functions are outlined in the following table.

Button	Function
MOTOR ENABLE/DISABLE	Enable or disable power to the motors
ESC/DELETE	Exit the current menu or delete a numeric entry. If the POSITION display is active, this key turns on or off the axis 1 (X) motor.
UP	Scroll up through the active menu. If the POSITION display is active, this key turns on or off the axis 2 (Y) motor.
DOWN	Scroll down through the active menu. If the POSITION display is active, this key turns on or off axis 3 (Z Platen) motor.
MENU/ENTER	Open the MENU display when in the POSITION display. Press this button to access the selected menu entry and to enter numeric data.
JOG/KEYPAD	If the controller is at the POSITION screen, these buttons jog the associated axis in a + or – direction at low or high speed. JOG mode requires running program "1" first. Otherwise, these keys are used to enter numbers.

The menu structure is as follows:



6.4.1 Clearing Errors

Occasionally a blinking “E” appears in the upper left corner of the display. This indicates that one of the enabled errors has occurred. A description of the error(s) is accessed and cleared as follows:

1. Press MENU/ENTER and scroll down to the ERRORS menu item.
2. Press MENU/ENTER. A list of the current errors is displayed.
3. After reading the messages, make note of any that need resolution.
4. Press MENU/ENTER to clear the list or ESC/DELETE to exit without clearing the list.

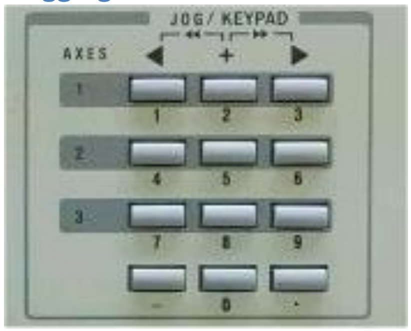
6.4.2 Run Program

The ESP301 has the capability to store program scripts and run them. It comes with two programs, labeled “1” and “2”. Program “1” enables joystick and jog button operation. Program “2” enables the Move Absolute and Move Relative menu functions. By default, the joystick and jog buttons are active. Program “1” is run as follows:

1. Select “MENU/ENTER”
2. Scroll down with the “DOWN” button until the “RUN PROGRAM” menu is selected. Press “MENU/ENTER”.
3. Press the “1” button on the JOG/KEYPAD and then run the program by pressing “DOWN”.

Any numbered program can be executed in this manner.

6.4.3 Jogging



The JOG/KEYPAD can be used to move the X,Y and Z axis at either low or high speed. The ESP301 must be in jog mode (Run program “1”) to use these buttons. The “<” and “>” buttons associated with each axis jog it in the indicated direction in slow speed. Pressing the “+” button associated with the axis in conjunction with the jog button causes a high speed jog. These buttons function the same as the joystick.

6.4.4 Homing

Homing is not necessary for basic joystick and jog operations. If repeatable programmed positioning is required from one session to the next, home should be performed at the beginning of each session. Be sure to always home the Z Platen first. Home is accessed as follows:

1. Select “MENU/ENTER”. The “HOME” menu item should be selected.
2. Select “MENU/ENTER”. A list of the axes is presented.
3. Scroll down to axis 3 (Z) and press “MENU/ENTER” to home the platen to the top limit. The platen should always be homed first to prevent damage to probes and the DUT when homing the X and Y axes.
4. Scroll up using the “UP” button to select axis 2 (Y) and press “MENU/ENTER”. The Y axis should home to the back of the travel.
5. Scroll up using the “UP” button to select axis 1 (X) and press “MENU/ENTER”. The X axis should home to the left of travel.
6. Press “ESC/DELETE” twice to return to the position display.

6.4.5 Move Absolute

Absolute moves are used to move an axis to a specific location. The move speed used is the same as the most recent joystick or jog button move. Therefore, be sure to move at high speed prior to performing an absolute move. Absolute moves are enabled by running program “2”.

1. Select “MENU/ENTER”. Use the “DOWN” button to select the “MOVE ABSOLUTE” menu item.
2. Select “MENU/ENTER”. A list of the axes is presented.
3. Scroll to the axis to move using the “UP” and “DOWN” keys.
4. Enter the destination number using the “JOG/KEYPAD”.
5. Press “MENU/ENTER” to move the axis to the destination.
6. Scroll to the next axis to move and repeat steps 4 and 5.
7. Press “ESC/DELETE” twice to return to the position display.

6.4.6 Move Relative

Relative moves are used to move an axis by a distance. The move speed used is the same as the most recent joystick or jog button move. Therefore, be sure to move at high speed prior to performing a relative move. Relative moves are enabled by running program “2”.

1. Select “MENU/ENTER”. Use the “DOWN” button to select the “MOVE ABSOLUTE” menu item.
2. Select “MENU/ENTER”. A list of the axes is presented.
3. Scroll to the axis to move using the “UP” and “DOWN” keys.
4. Enter the distance number using the “JOG/KEYPAD”.
5. Press “MENU/ENTER” to move the axis by the distance.
6. Scroll to the next axis to move and repeat steps 4 and 5.
7. Press “ESC/DELETE” twice to return to the position display.

6.4.7 Reset Position

1. Select “MENU/ENTER”. Use the “DOWN” button to select the “MOVE ABSOLUTE” menu item.
2. Select “MENU/ENTER”. A list of the axes is presented.
3. Scroll to the axis to move using the “UP” and “DOWN” keys.
4. Enter the distance number using the “JOG/KEYPAD”.
5. Press “MENU/ENTER” to move the axis by the distance.
6. Scroll to the next axis to move and repeat steps 4 and 5.
7. Press “ESC/DELETE” twice to return to the position display.

6.5 Host Control

The ESP301 has an extensive command set available. The ESP301 Manual provides complete descriptions of the commands, with a summary found in section 3.5.

There are some caveats to keep in mind when operating the system remotely and from the front panel menus. These are as follows:

1. Avoid changing the axis motor, limit, and stepping configurations. Changing the system configuration values can render the ESP301 incapable of moving the motors and even cause physical damage. The following commands should be specifically avoided: BA, FR, QM, QV, QI, QS, ZA, ZB, ZE, ZF, ZH, ZS, ZZ. If the ESP301 ceases operating due to configuration changes, it will be necessary to reconfigure it using the commands in section 6.6.1 SETTINGS.CFG
2. The ESP301 ships with program "1" and program "2" loaded. Program "1" enables the joystick, disables programmable motion, and sets the trajectory mode to Jog. Program "2" enables programmable motion, disables the joystick, and sets the trajectory mode to Trapezoidal. DO NOT DELETE EITHER OF THESE PROGRAMS! If they are deleted the joystick will cease to function, even after restart. They can be restored by executing the commands in section 6.6.2 PROGRAMS.CFG
3. The "Move Absolute" and "Move Relative" menu options in the ESP301 are operational only when the trajectory mode is set to Trapezoidal. This is done by running program "2".
4. All changes to settings are temporary and are lost when the ESP301 is powered down. Settings can be stored permanently by either selecting the menu "Configuration/Save Parameters" or by sending the "SM" command before powering down.
5. Some errors, and activating the EMS switch cause the motors to turn off, as indicated on the main display under the "MTR" column. Motors can be turned back on by pressing the "Motor Enable/Disable" button or by sending "1MO;2MO;3MO".

THE ESP301 DOES NOT RAISE THE PLATEN BEFORE AN XY MOVE! It is up to the operator and developer to ensure the platen has lifted probes off the DUT before moving in XY.

6.5.1 Command Usage

There are several commands and command sequences which are specific to the manner in which the ESP301 is configured. They are outlined below.

NOTE: Axis 1 corresponds to the X stage, axis 2 to the Y stage, and axis 3 to the Z Platen.

Command Type	Command Sequence	Description

Home	`3OR3;3WS100;`
	`1OR3;1WS100;`
	`2OR4;2WS100`
		Home Z First, wait for it to complete, then home X and Y

Enable Joystick (Disables Programmed moves)	Call ESP301 Program "1":
	`1EX`
	OR execute this sequence:
	`BO0;`
	`1BP11,10,12;`
	`2BP9,8,12;`
	`3BP14,13,12;`
	`1BQ1;2BQ1;3BQ1;`
	`1TJ3;2TJ3;3TJ3;`
	`1MO;2MO;3MO`
		Enables navigation using the joystick. This DISABLES programmable moves from the front panel of the ESP301 as well as from the host.

Enable Programmed Moves(Disables Joystick)	Call ESP301 Program "2"
	`2EX`
	OR execute this sequence:
	`1BQ0;2BQ0;3BQ0;`
	`1TJ1;2TJ1;3TJ1`
		Enables navigation using the front panel "Move to Absolute" and "Move Relative" as well as host controlled moves. The joystick is DISABLED.

Set Speeds	`1VA12.0;2VA12.0;3VA0.4`	Sets the programmed move speed to the maximum for each axis. Other speeds less than the maximums can be used if required

Move to Absolute Position	XY Move:
	`1VA12.0;2VA12.0;`
	`1PA<xpos>;2PA<ypos>;`
	`1WS100;2WS100;`
	Z Move:
	`3VA0.4;3PA<zpos>;3WS100`
		Set the move speed, move to the destination, and wait for the move to complete. <xpos> and <ypos> are the XY destination positions. <zpos> is the Z destination.

Move by Relative Distance	XY Move:
	`1VA12.0;2VA12.0;`
	`1PR<xrel>;2PR<yrel>;`
	`1WS100;2WS100;`
	Z Move:
	`3VA0.4;3PR<zrel>;3WS100`
		Set the move speed, move by the distance, and wait for the move to complete. <xrel> and <yrel> are the XY distances. <zrel> is the Z distance.

Get Position	`1TP?;2TP?;3TP?`	Get the XYZ position
Stop Motion	`1ST;2ST;3ST`	Stops motion on all axes
Set XY Position	`1DH<xpos.;2DH<ypos>`	Sets the XY position to <xpos> and <ypos>.
Get Moving Status	`TS`	Returns controller status as a two

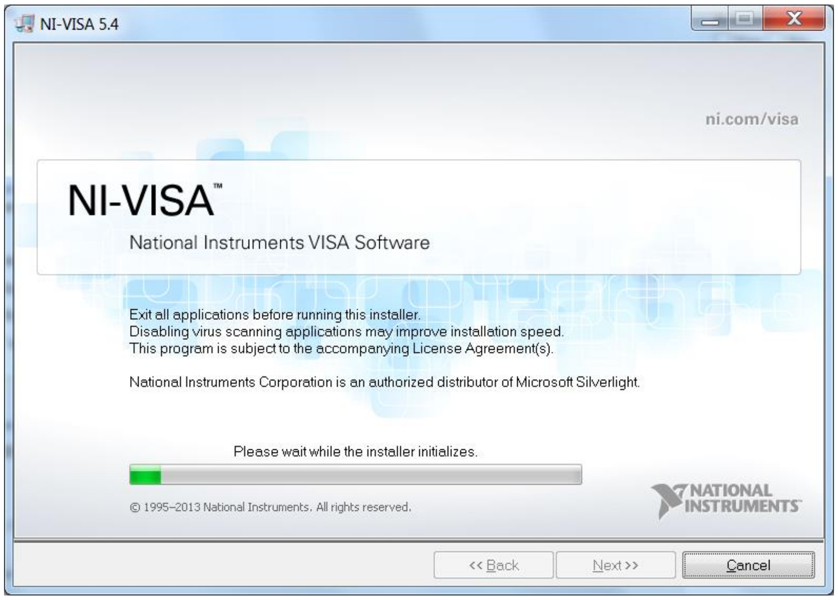
		character string. The leftmost character is the XYZ status. Binary AND the leftmost character with 0x07. If the result is non-zero, an axis is moving.
Save Settings	`SM`	Saves all parameters, including position, for use when the ESP301 boots. This command should be used sparingly since the non-volatile RAM of the ESP301 is limited to around 10000 storage cycles.

6.5.2 Installing and configuring the VISA driver

The VISA installer is found at:
http://lumen.ni.com/nicif/US/GB_NIDU/content.xhtml?du=http://www.ni.com/download/ni-visa-5.4/4230/en/

Installation is as follows:

1. Download the installer and unzip it per the instructions.
2. Start the Installer. The window below is shown. Click on the “Next >>” button when it is enabled.



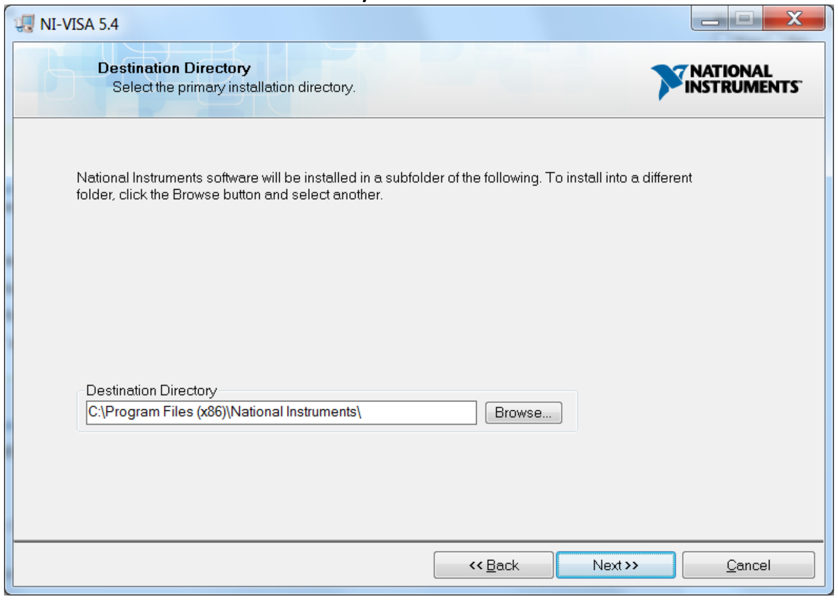
Exit all applications before running this installer.
Disabling virus scanning applications may improve installation speed.
This program is subject to the accompanying License Agreement(s).

National Instruments Corporation is an authorized distributor of Microsoft Silverlight.

Please wait while the installer initializes.

© 1995–2013 National Instruments. All rights reserved.

3. Once the installer is initialized, the following screen is shown. Do not change the destination directory. Select “Next>>”.

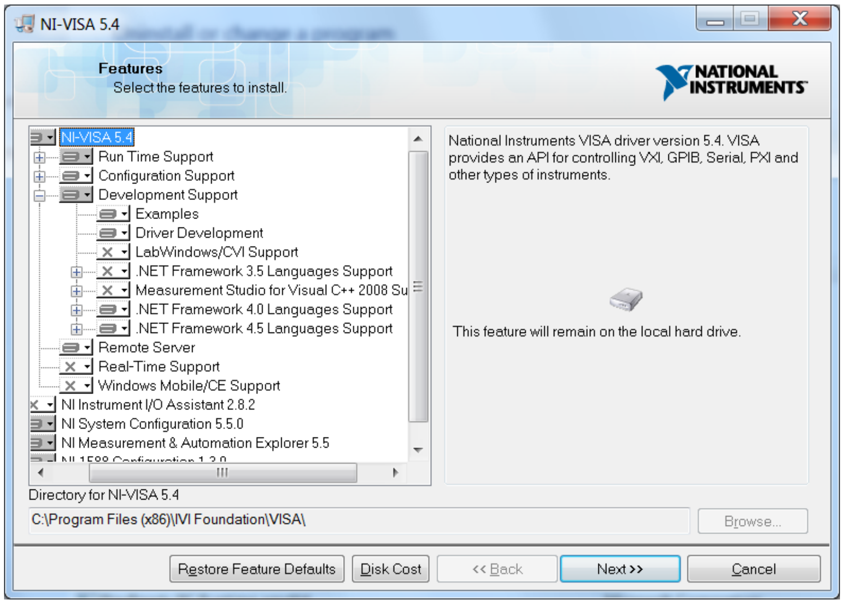


Destination Directory
Select the primary installation directory.

National Instruments software will be installed in a subfolder of the following. To install into a different folder, click the Browse button and select another.

Destination Directory
C:\Program Files (x86)\National Instruments\

4. The default feature set requires the addition of .NET Framework 4.0 Language Development Support. Change the settings as shown below. Select "Next>>" to continue the install.


NI-VISA 5.4

Features
Select the features to install.

NI-VISA 5.4
- Run Time Support
- Configuration Support
- Development Support
  - Examples
  - Driver Development
  - ✗ LabWindows/CVI Support
  - ✗ .NET Framework 3.5 Languages Support
  - ✗ Measurement Studio for Visual C++ 2008 Su
  - .NET Framework 4.0 Languages Support
  - .NET Framework 4.5 Languages Support
- Remote Server
- ✗ Real-Time Support
- ✗ Windows Mobile/CE Support

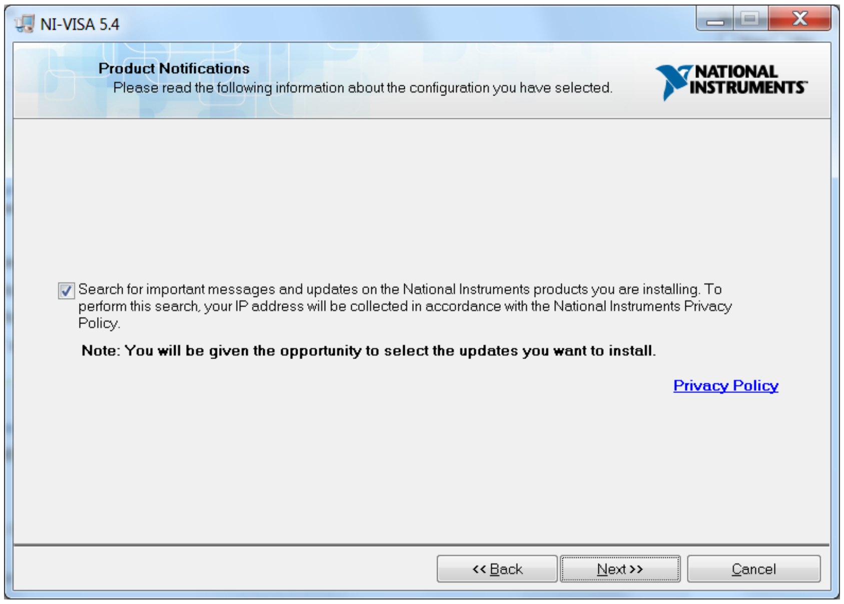
✗ NI Instrument I/O Assistant 2.8.2
NI System Configuration 5.5.0
NI Measurement & Automation Explorer 5.5
NI 1588 Configuration 1.3.0

National Instruments VISA driver version 5.4. VISA provides an API for controlling VXI, GPIB, Serial, PXI and other types of instruments.

This feature will remain on the local hard drive.

Directory for NI-VISA 5.4
C:\Program Files (x86)\IVI Foundation\VISA\

5. If the host is connected to the Internet, then check the "Search…" check box. Select "Next>>" to continue.

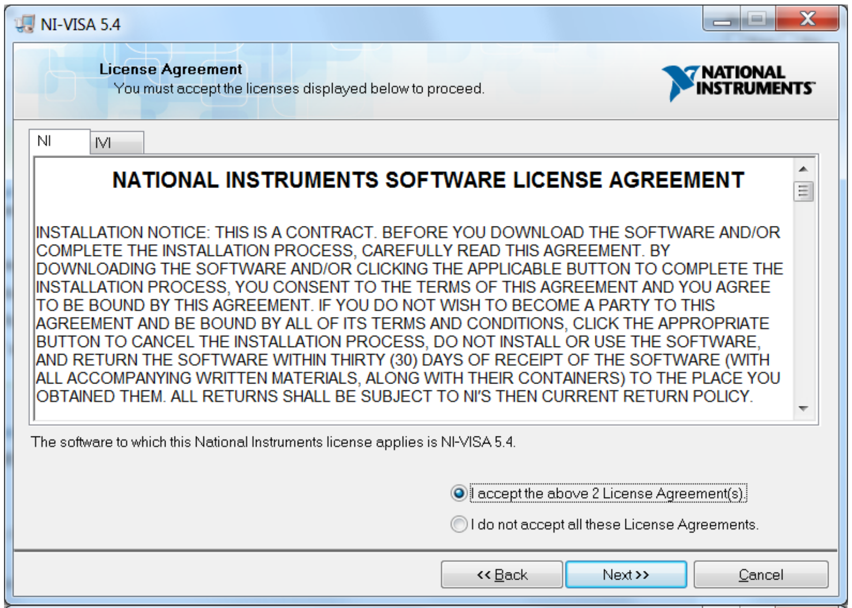

NI-VISA 5.4

Product Notifications
Please read the following information about the configuration you have selected.

[✓] Search for important messages and updates on the National Instruments products you are installing. To perform this search, your IP address will be collected in accordance with the National Instruments Privacy Policy.

Note: You will be given the opportunity to select the updates you want to install.

6. Accept both National Instruments and Microsoft license agreements. Select "Next>>" to continue.



NI-VISA 5.4

License Agreement

You must accept the licenses displayed below to proceed.

NI | IVI

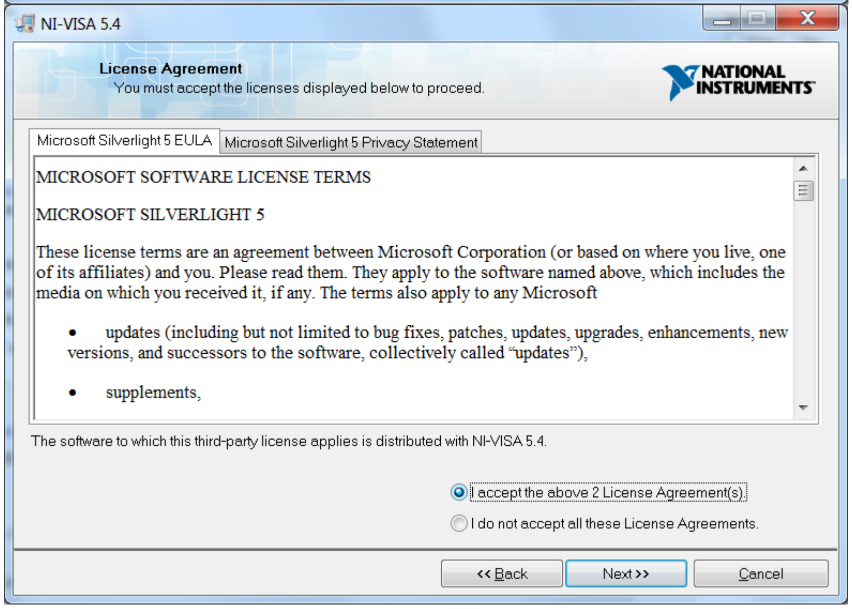
NATIONAL INSTRUMENTS SOFTWARE LICENSE AGREEMENT

INSTALLATION NOTICE: THIS IS A CONTRACT. BEFORE YOU DOWNLOAD THE SOFTWARE AND/OR COMPLETE THE INSTALLATION PROCESS, CAREFULLY READ THIS AGREEMENT. BY DOWNLOADING THE SOFTWARE AND/OR CLICKING THE APPLICABLE BUTTON TO COMPLETE THE INSTALLATION PROCESS, YOU CONSENT TO THE TERMS OF THIS AGREEMENT AND YOU AGREE TO BE BOUND BY THIS AGREEMENT. IF YOU DO NOT WISH TO BECOME A PARTY TO THIS AGREEMENT AND BE BOUND BY ALL OF ITS TERMS AND CONDITIONS, CLICK THE APPROPRIATE BUTTON TO CANCEL THE INSTALLATION PROCESS, DO NOT INSTALL OR USE THE SOFTWARE, AND RETURN THE SOFTWARE WITHIN THIRTY (30) DAYS OF RECEIPT OF THE SOFTWARE (WITH ALL ACCOMPANYING WRITTEN MATERIALS, ALONG WITH THEIR CONTAINERS) TO THE PLACE YOU OBTAINED THEM. ALL RETURNS SHALL BE SUBJECT TO NI'S THEN CURRENT RETURN POLICY.

The software to which this National Instruments license applies is NI-VISA 5.4.

[✓ I accept the above 2 License Agreement(s).]

I do not accept all these License Agreements.



NI-VISA 5.4

License Agreement

You must accept the licenses displayed below to proceed.

Microsoft Silverlight 5 EULA | Microsoft Silverlight 5 Privacy Statement

MICROSOFT SOFTWARE LICENSE TERMS

MICROSOFT SILVERLIGHT 5

These license terms are an agreement between Microsoft Corporation (or based on where you live, one of its affiliates) and you. Please read them. They apply to the software named above, which includes the media on which you received it, if any. The terms also apply to any Microsoft

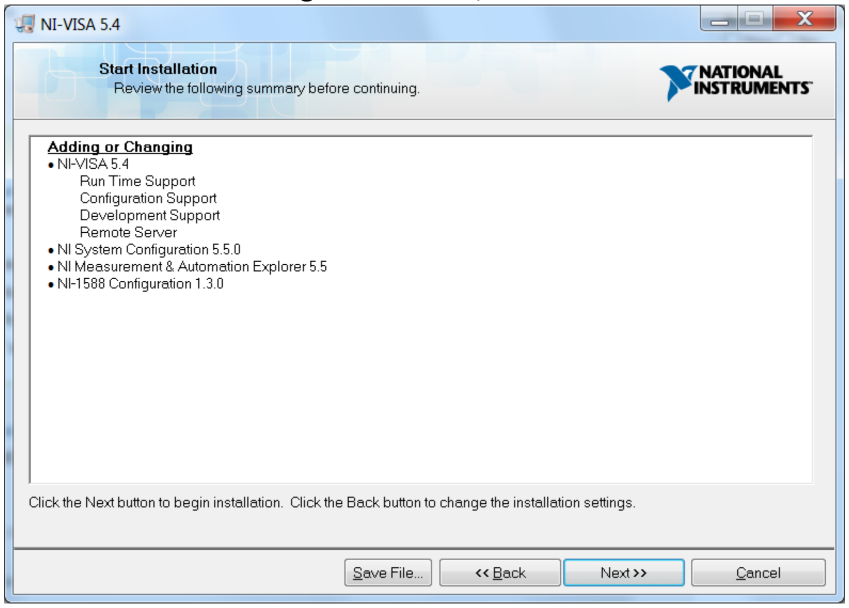
- updates (including but not limited to bug fixes, patches, updates, upgrades, enhancements, new versions, and successors to the software, collectively called "updates"),
- supplements,

The software to which this third-party license applies is distributed with NI-VISA 5.4.

[✓ I accept the above 2 License Agreement(s).]

I do not accept all these License Agreements.

7. All install settings are correct, so select “Next>>”.



Start Installation

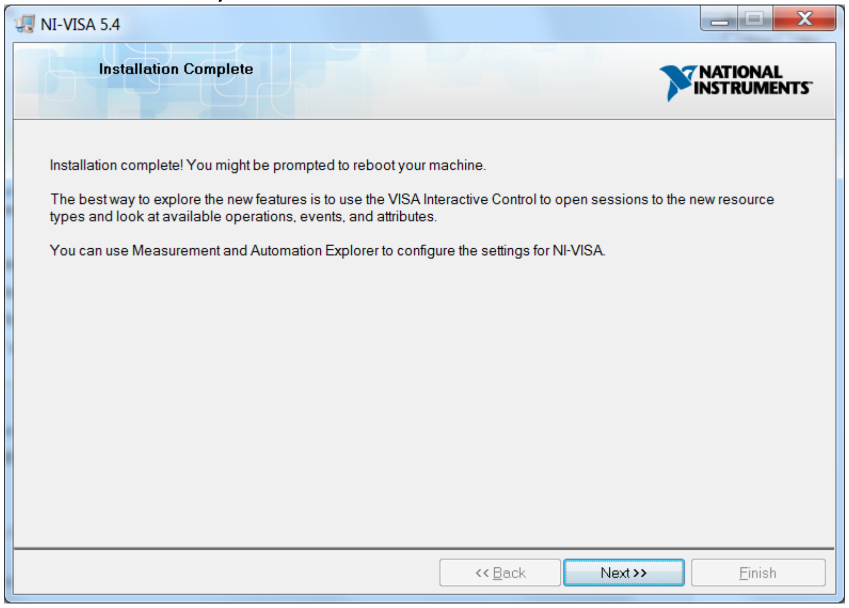
Review the following summary before continuing.

Adding or Changing

- NI-VISA 5.4
  - Run Time Support
  - Configuration Support
  - Development Support
  - Remote Server
- NI System Configuration 5.5.0
- NI Measurement & Automation Explorer 5.5
- NI-1588 Configuration 1.3.0

Click the Next button to begin installation. Click the Back button to change the installation settings.

8. Once installation is complete, select “Next>>” to finish. A reboot is likely necessary.



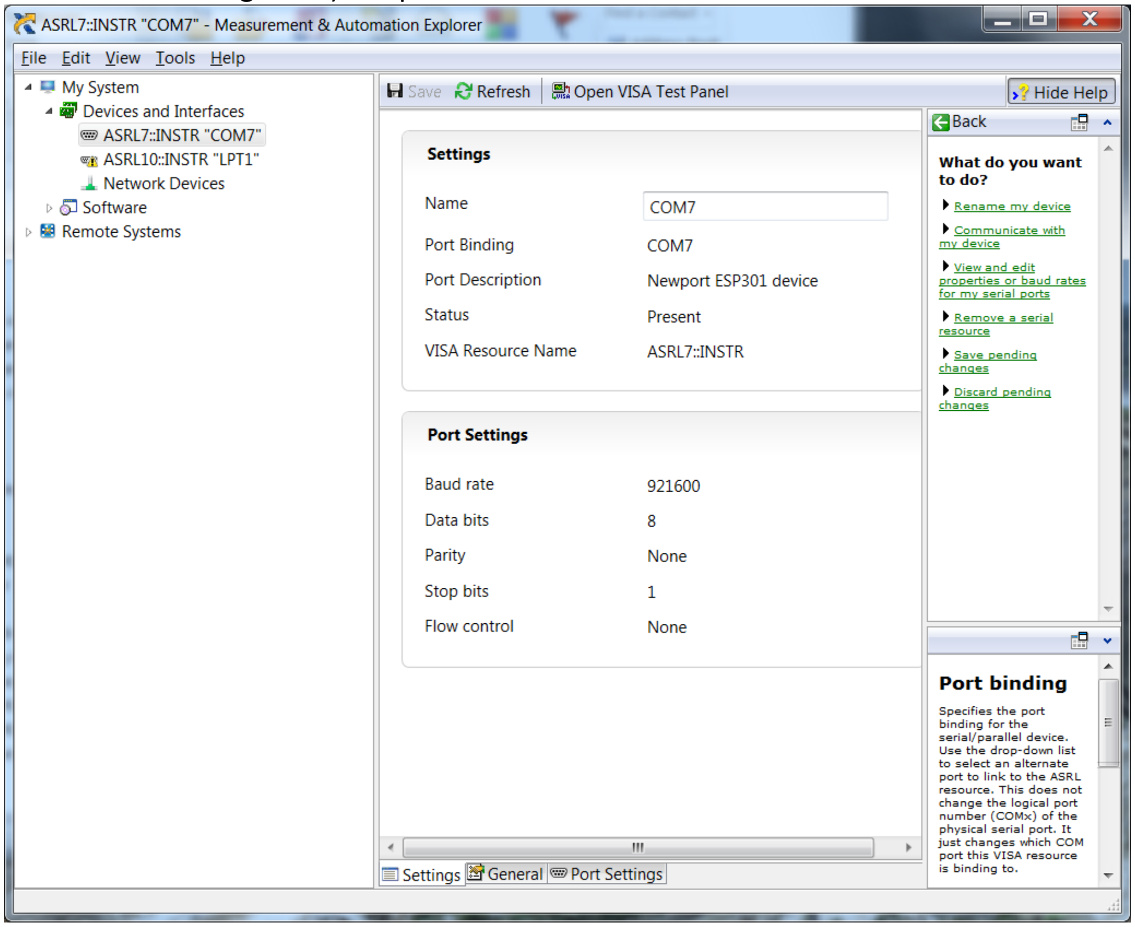
Installation Complete

Installation complete! You might be prompted to reboot your machine.

The best way to explore the new features is to use the VISA Interactive Control to open sessions to the new resource types and look at available operations, events, and attributes.

You can use Measurement and Automation Explorer to configure the settings for NI-VISA.

9. Once the computer restarts, the ESP301 must be connected to the host via USB before configuring the VISA driver. There should now be an application on the desktop named “NI MAX”. It is used to configure VISA interfaces. The USB interface is added to the VISA driver automatically. Run NI MAX to ensure the USB port is available as shown below. The Baud rate MUST be set to 921600 (use Port Settings tab”) to operate with the ESP301.



Settings

Name	COM7
Port Binding	COM7
Port Description	Newport ESP301 device
Status	Present
VISA Resource Name	ASRL7::INSTR

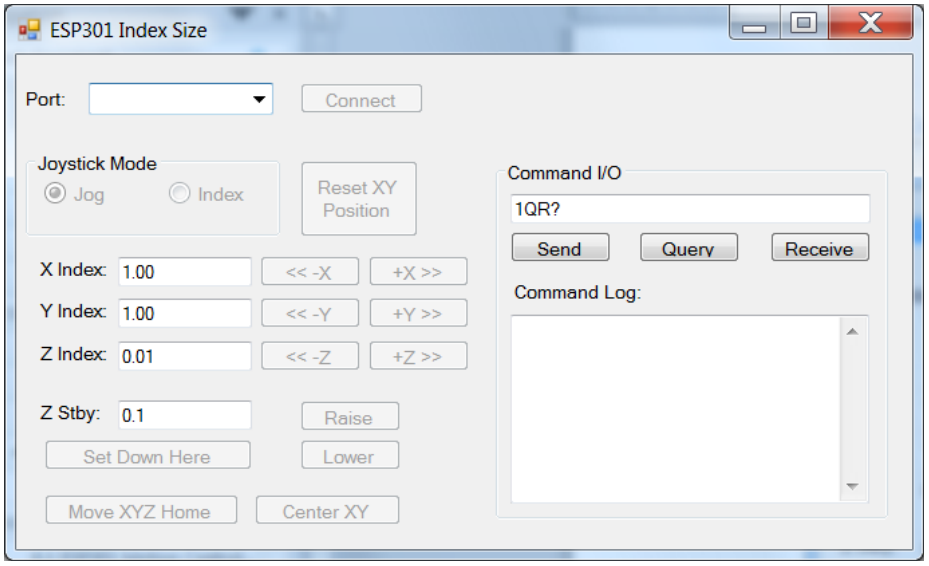
Port Settings

Baud rate	921600
Data bits	8
Parity	None
Stop bits	1
Flow control	None

Port binding

Specifies the port binding for the serial/parallel device. Use the drop-down list to select an alternate port to link to the ASRL resource. This does not change the logical port number (COMx) of the physical serial port. It just changes which COM port this VISA resource is binding to.

6.5.3 Sample Code (C#)



Port:

Joystick Mode: [✓ Jog] Index

X Index: 1.00

Y Index: 1.00

Z Index: 0.01

Z Stby: 0.1

Command I/O: 1QR?

The code sample provided in this section demonstrates how to connect to the ESP301 via USB. The code has some scripts embedded in it which show how to enable/disable joystick control and perform basic directed positioning functions. "Index" mode causes the ESP301 to index by the indicated distance when the axis index buttons are pressed. The Command I/O section provides a way to send and receive information from the controller. Operate the program as follows:

1. Select the serial port attached to the ESP301.
2. Select the "Connect" button and wait for the button label to change to "Disconnect".
3. Move the stage using the joystick noting that motion is continuous when the joystick is actuated.
4. Set the joystick mode to "Index" and wait for the additional buttons to be enabled.
5. Move the stage using the index buttons, noting that it moves by the indicated index distance for each axis.
6. Change any of the index values then select a jog button adjacent to the value to use it for indexing.
7. Set the joystick mode back to "Jog" before exiting the program.

The controls on the GUI are as follows:

- **Port** -- List of VISA devices available
- **Connect/Disconnect** – Connect to the selected USB/Serial port
- **Joystick Mode** – Select either joystick continuous jog mode or indexed mode
- **Reset XY Position** – Sets the XY position to 0.0000,0.0000
- **X/Y/Z Index Size** – Index distances associated with index buttons
- **Index by X/Y/Z** – Index by the index distances when index mode

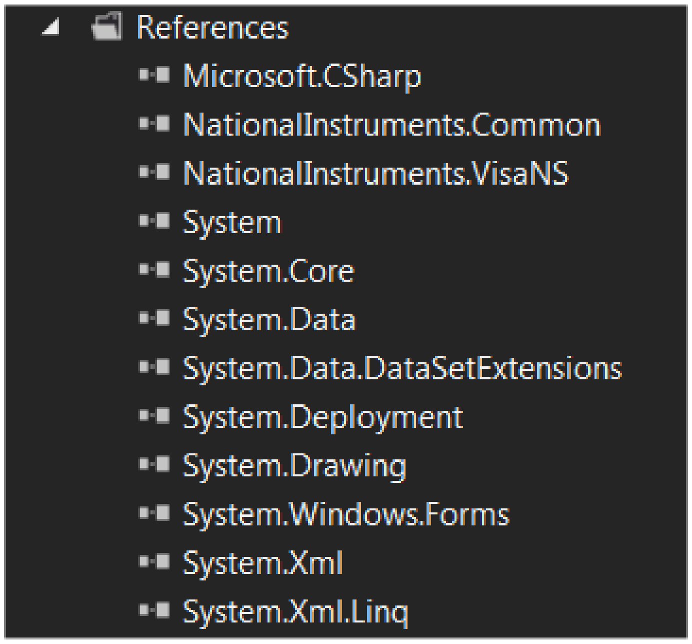
- **Z Stby** – Platen standby height above down set point
- **Set Down Here** – Set the current Z platen position as the down position
- **Raise** – Move the platen above the Z down set position to the standby height
- **Lower** – Move the platen to the down set point
- **Move XYZ Home** – Moves all axes to their home limits
- **Center XY** – Move the XY axes to the center of their respective travels.
- **Command I/O** – Enter a command to the ESP301 controller
- **Send** – Send the command to the controller
- **Query** – Send the command to the controller and read the response
- **Receive** – Read the response from the controller
- **Command Log** – Displays sent commands and received data

The code modules are as follows:

ESP301_Event.cs -- Launches the program and opens the GUI
ESP301_Form.cs – Handles all the operator input and generates the associated ESP301 action
ESP301_Form.Designer.cs – Creates and displays the GUI Interface
ESP301_Serial.cs – Serial I/O to the ESP301

6.5.3.1 Visual Studio 2012 Project References

The references provided may not match installations on all computers. They do show which resources are necessary to compile the program.

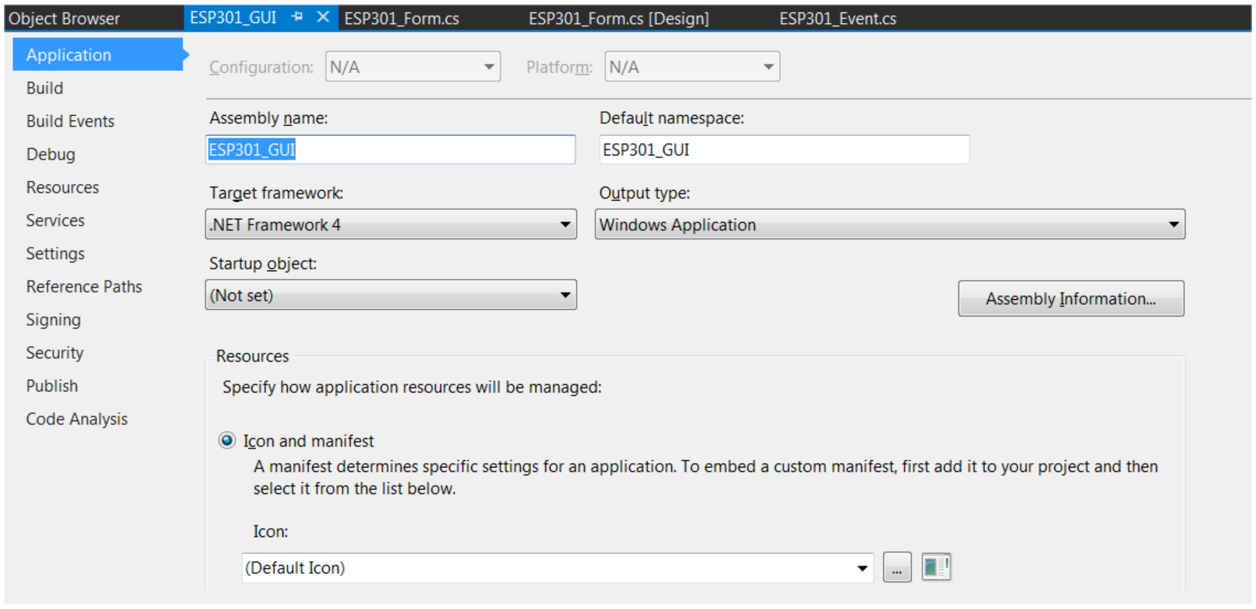


```
References
    Microsoft.CSharp
    NationalInstruments.Common
    NationalInstruments.VisaNS
    System
    System.Core
    System.Data
    System.Data.DataSetExtensions
    System.Deployment
    System.Drawing
    System.Windows.Forms
    System.Xml
    System.Xml.Linq
```

Reference Paths:

Reference	Path
NationalInstruments.Common	C:\Windows\Microsoft.NET\assembly\GAC_MSIL\NationalInstruments.Common\v4.0_13.0.40.190__dc6ad606294fc298\NationalInstruments.Common.dll
NationalInstruments.VisaNS	C:\Windows\Microsoft.NET\assembly\GAC_MSIL\NationalInstruments.VisaNS\v4.0_13.0.40.167__dc6ad606294fc298\NationalInstruments.VisaNS.dll

The project properties should be as follows:



Configuration: N/A Platform: N/A

Assembly name: ESP301_GUI

Default namespace: ESP301_GUI

Target framework: .NET Framework 4

Output type: Windows Application

Startup object: (Not set)

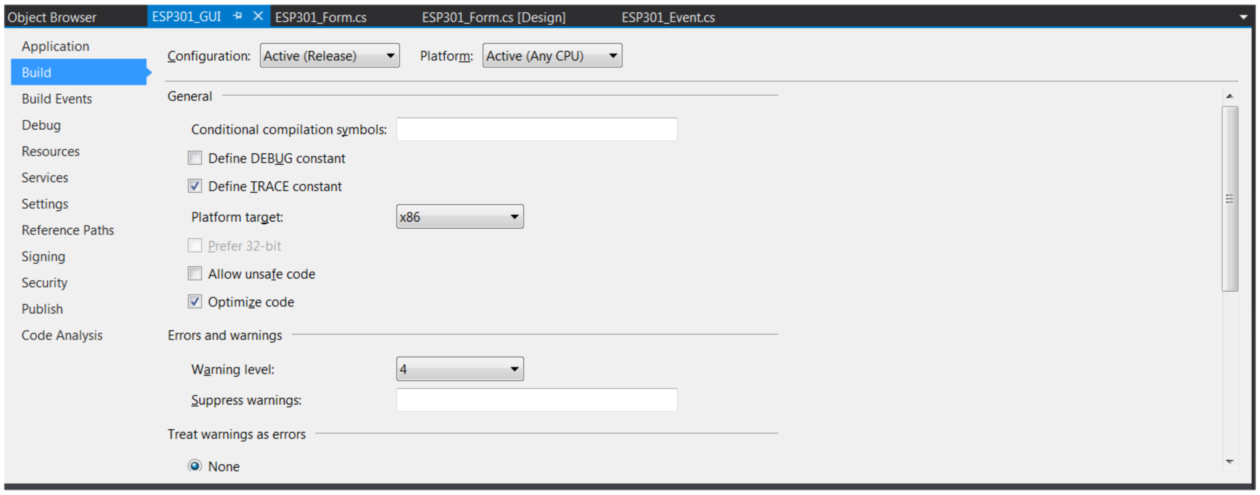
Resources

Specify how application resources will be managed:

(●) Icon and manifest

A manifest determines specific settings for an application. To embed a custom manifest, first add it to your project and then select it from the list below.

Icon: (Default Icon)



Configuration: Active (Release) Platform: Active (Any CPU)

General

Conditional compilation symbols:

☐ Define DEBUG constant

☑ Define TRACE constant

Platform target: x86

☐ Prefer 32-bit

☐ Allow unsafe code

☑ Optimize code

Errors and warnings

Warning level: 4

Suppress warnings:

Treat warnings as errors

(●) None

6.5.3.2 Startup (ESP301_Event.cs)

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace ESP301_GUI
{
    static class ESP301_Event
    {
        /// <summary>
        /// The main entry point for the application.
        /// </summary>
        [STAThread]
        static void Main()
        {
            Application.EnableVisualStyles();
            Application.SetCompatibleTextRenderingDefault(false);
            Application.Run(new ESP301_Form());
        }
    }
}
```

6.5.3.3 GUI Handler (ESP301_Form.cs)

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace ESP301_GUI
{
    /// <summary>
    /// Main form class for handling operator input
    /// </summary>
    public partial class ESP301_Form : Form
    {
        ESP301_Serial sp;
        bool connected = false;
        bool zdownset = false;
        Double downpos = 0.0;

        public ESP301_Form()
        {
            InitializeComponent();
            enabledButtons(false);
            sp = new ESP301_Serial();
            string[] ports = ESP301_Serial.getPorts();
            if (ports != null)
            {
                cbPorts.Items.AddRange(ports);
                cbPorts.SelectedIndex = 0;
            }
            else
            {
                MessageBox.Show("The ESP301 is not found.  Close this application, connect 
the ESP301, and restart.", "ESP301 Error", MessageBoxButtons.OK, MessageBoxIcon.Error );
                pbConnect.Enabled = false;
            }

        }

        /******************************************
         * NOTE!!
         * All positioning values are in millimeters
         *
         * */


        private string[] script_reset_position = { "1DH0.0;2DH0.0"};

        private void enabledButtons(bool en)
        {
           pbRaise.Enabled = en;
            pbLower.Enabled = en;
            pbResetPosition.Enabled = en;
            pbHome.Enabled = en;
            pbCenter.Enabled = en;
           pbZDown.Enabled = en;
            rbIndex.Enabled = en;
            rbJog.Enabled = en;
            pbXMinus.Enabled = en;
            pbXPlus.Enabled = en;
            pbYMinus.Enabled = en;
            pbYPlus.Enabled = en;
            pbZMinus.Enabled = en;
            pbZPlus.Enabled = en;
        }

        /// <summary>
        /// Connect to the ESP301 and upload the indexing programs
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void pbConnect_Click(object sender, EventArgs e)
        {
            if (pbConnect.Text == "Connect")
            {
                //  Validate text entries
                //  Open COM port
                string port = cbPorts.Text;
                sp.openSerial(port);
                pbConnect.Text = "Disconnect";
                connected = true;
                rbJog.Checked = true;
            }
            else
            {
                //  Place in jog mode again
                rbJog.Checked = true;
                sp.writeSerial("1EX");
                //  Close COM port
//                sp.closeSerial();
                pbConnect.Text = "Connect";
                connected = false;
            }
            enabledButtons(false);
            rbIndex.Enabled = true;
            rbJog.Enabled = true;
        }

        /// <summary>
        /// Enable Jog Mode
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void rbJog_CheckedChanged(object sender, EventArgs e)
        {

            if (rbJog.Checked)
            {
                if (connected)
                    sp.writeSerial("1EX");
                enabledButtons(false);
                rbJog.Enabled = true;
                rbIndex.Enabled = true;
            }
        }

        /// <summary>
        /// Enable Index Mode
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void rbIndex_CheckedChanged(object sender, EventArgs e)
        {
            if (rbIndex.Checked)
            {
                if (connected)
                    sp.writeSerial("2EX");
                enabledButtons(true);
                rbJog.Enabled = true;
                rbIndex.Enabled = true;
            }
        }


        /// <summary>
        /// Reset the positions to 0.0
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void pbResetPosition_Click(object sender, EventArgs e)
        {
            if (!connected)
                return;
            sp.sendScript(script_reset_position);
        }

        /// <summary>
        /// Lift the platen to the up position
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void pbRaise_Click(object sender, EventArgs e)
        {
            if (!connected)
                return;
            string dpos;
            Double zdest = downpos-Convert.ToDouble(etStby.Text);
            if (zdownset)
            {
                dpos = "3PA" + Convert.ToString(zdest);
            }
            else
            {
                dpos = "3PR-" + etStby.Text;
            }
            sp.writeSerial("3VA0.4");  //  Set speed to 0.4 mm/s

            sp.writeSerial(dpos);
            sp.writeSerial("3WS100");
        }

        /// <summary>
        /// Lower the platen to the down position
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void pbLower_Click(object sender, EventArgs e)
        {
            if (!connected)
                return;
            string dpos;
            Double zdest = downpos;
            if (zdownset)
            {
                dpos = "3PA" + Convert.ToString(zdest);
            }
            else
            {
                dpos = "3PR" + etStby.Text;
            }
            sp.writeSerial("3VA0.4");  //  Set speed to 0.4 mm/s
            sp.writeSerial(dpos);
            sp.writeSerial("3WS100");

        }

        /// <summary>
        /// Set the Z Down position
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void pbZDown_Click(object sender, EventArgs e)
        {
            if (!connected)
                return;
            if (zdownset)
            {
                zdownset = false;
                pbZDown.Text = "Set Down Here";
                return;
            }
            string pos = "0.0";
            sp.writeSerial("3TP?");
            sp.readSerial(ref pos);
            if (pos.Length > 0)
            {
                downpos = Convert.ToDouble(pos);
                zdownset = true;
                pbZDown.Text = "Clear Down Set";
            }
            else
            {
                zdownset = false;
            }

        }

        /// <summary>
        /// Home the prober axes
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void pbHome_Click(object sender, EventArgs e)
        {
            if (!connected)
                return;
            sp.writeSerial("3OR3;3WS100;1OR3;1WS100;2OR4;2WS100");

        }

        /// <summary>
        /// Center the prober XY axes
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void pbCenter_Click(object sender, EventArgs e)
        {
            if (!connected)
                return;
            sp.writeSerial("1VA12.0;2VA12.0;3VA0.4");  //  Set speed to 0.4 mm/s
            sp.writeSerial("1PA-100;2PA100;1WS100;2WS100");
        }

        /// <summary>
        /// Send ASCII command to ESP301
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void pbSend_Click(object sender, EventArgs e)
        {
            if (!connected)
                return;
            tbLog.AppendText("-->");
            sp.writeSerial(etCommand.Text);
            tbLog.AppendText(etCommand.Text + "\n");
        }

        /// <summary>
        /// Send ASCII command to ESP301 and read back data
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void pbQuery_Click(object sender, EventArgs e)
        {
            string s = "";

            if (!connected)
                return;
            tbLog.AppendText("-->");
            sp.writeSerial(etCommand.Text);
            tbLog.AppendText(etCommand.Text + "\n");
            tbLog.AppendText("<--");
            sp.readSerial(ref s);
            if (s.Length > 0)
            {
                tbLog.AppendText(s + "\n");
            }
            else
                tbLog.AppendText("READ FAILED\n");
        }

        /// <summary>
        /// Read data from the ESP301
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void pbRecv_Click(object sender, EventArgs e)
        {
            string s = "";

            if (!connected)
                return;
            tbLog.AppendText("<--");
            sp.readSerial(ref s);
            if (s.Length > 0)
            {
                tbLog.AppendText(s + "\n");
            }
            else
                tbLog.AppendText("READ FAILED\n");
        }

        /// <summary>
        /// Index -X using entered value
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void pbXMinus_Click(object sender, EventArgs e)
        {
            string s = "";
            double idx = double.Parse(etX.Text);
            etX.Text = string.Format("{0,4}", Math.Abs(idx));

            if (!connected)
                return;
            s = string.Format( "1PR{0,4}",-idx);
            sp.writeSerial(s);
        }

        /// <summary>
        /// Index +X using entered value
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void pbXPlus_Click(object sender, EventArgs e)
        {
            string s = "";
            double idx = double.Parse(etX.Text);
            etX.Text = string.Format("{0,4}", Math.Abs(idx));

            if (!connected)
                return;

            s = string.Format("1PR{0,4}", idx);
            sp.writeSerial(s);

        }

        /// <summary>
        /// Index -Y using entered value
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void pbYMinus_Click(object sender, EventArgs e)
        {
            string s = "";
            double idx = double.Parse(etY.Text);
            etY.Text = string.Format("{0,4}", Math.Abs(idx));

            if (!connected)
                return;

            s = string.Format("2PR{0,4}", -idx);
            sp.writeSerial(s);

        }

        /// <summary>
        /// Index +Y using entered value
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void pbYPlus_Click(object sender, EventArgs e)
        {
            string s = "";
            double idx = double.Parse(etY.Text);
            etY.Text = string.Format("{0,4}", Math.Abs(idx));

            if (!connected)
                return;

            s = string.Format("2PR{0,4}", idx);
            sp.writeSerial(s);

        }

        /// <summary>
        /// Index -Z using entered value
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void pbZMinus_Click(object sender, EventArgs e)
        {
            string s = "";
            double idx = double.Parse(etZ.Text);
```

```
            etZ.Text = string.Format("{0,4}", Math.Abs(idx));

            if (!connected)
                return;

            s = string.Format("3PR{0,4}", -idx);
            sp.writeSerial(s);

        }

        /// <summary>
        /// Index +Z using entered value
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void pbZPlus_Click(object sender, EventArgs e)
        {
            string s = "";
            double idx = double.Parse(etZ.Text);
            etZ.Text = string.Format("{0,4}", Math.Abs(idx));

            if (!connected)
                return;

            s = string.Format("3PR{0,4}", idx);
            sp.writeSerial(s);
        }


    }
```

6.5.3.4 Form Definition (ESP301_Form.Designer.cs)

```
namespace ESP301_GUI
{
    partial class ESP301_Form
    {
        /// <summary>
        /// Required designer variable.
        /// </summary>
        private System.ComponentModel.IContainer components = null;

        /// <summary>
        /// Clean up any resources being used.
        /// </summary>
        /// <param name="disposing">true if managed resources should be disposed;
otherwise, false.</param>
        protected override void Dispose(bool disposing)
        {
            if (disposing && (components != null))
            {
                components.Dispose();
            }
            base.Dispose(disposing);
        }

        #region Windows Form Designer generated code


        /// <summary>
        /// Required method for Designer support - do not modify
        /// the contents of this method with the code editor.
        /// </summary>
        private void InitializeComponent()
        {
            this.label1 = new System.Windows.Forms.Label();
            this.label2 = new System.Windows.Forms.Label();
            this.etX = new System.Windows.Forms.TextBox();
            this.etY = new System.Windows.Forms.TextBox();
            this.label3 = new System.Windows.Forms.Label();
            this.cbPorts = new System.Windows.Forms.ComboBox();
            this.rbJog = new System.Windows.Forms.RadioButton();
            this.rbIndex = new System.Windows.Forms.RadioButton();
            this.groupBox1 = new System.Windows.Forms.GroupBox();
            this.pbConnect = new System.Windows.Forms.Button();
            this.etZ = new System.Windows.Forms.TextBox();
            this.label4 = new System.Windows.Forms.Label();
            this.pbResetPosition = new System.Windows.Forms.Button();
            this.pbRaise = new System.Windows.Forms.Button();
            this.etStby = new System.Windows.Forms.TextBox();
            this.label5 = new System.Windows.Forms.Label();
            this.pbLower = new System.Windows.Forms.Button();
            this.pbZDown = new System.Windows.Forms.Button();
            this.pbHome = new System.Windows.Forms.Button();
            this.pbCenter = new System.Windows.Forms.Button();
            this.groupBox2 = new System.Windows.Forms.GroupBox();
            this.tbLog = new System.Windows.Forms.TextBox();
            this.lbRecv = new System.Windows.Forms.Label();
            this.pbRecv = new System.Windows.Forms.Button();
            this.pbQuery = new System.Windows.Forms.Button();
            this.pbSend = new System.Windows.Forms.Button();
            this.etCommand = new System.Windows.Forms.TextBox();
            this.pbXMinus = new System.Windows.Forms.Button();
            this.pbXPlus = new System.Windows.Forms.Button();
            this.pbYMinus = new System.Windows.Forms.Button();
            this.pbYPlus = new System.Windows.Forms.Button();
            this.pbZMinus = new System.Windows.Forms.Button();
            this.pbZPlus = new System.Windows.Forms.Button();
            this.groupBox1.SuspendLayout();
            this.groupBox2.SuspendLayout();
            this.SuspendLayout();
            //
            // label1
            //
            this.label1.AutoSize = true;
            this.label1.Location = new System.Drawing.Point(15, 151);
            this.label1.Name = "label1";
            this.label1.Size = new System.Drawing.Size(58, 17);
            this.label1.TabIndex = 0;
            this.label1.Text = "X Index:";
            //
            // label2
            //
            this.label2.AutoSize = true;
            this.label2.Location = new System.Drawing.Point(15, 182);
            this.label2.Name = "label2";
            this.label2.Size = new System.Drawing.Size(58, 17);
            this.label2.TabIndex = 1;
            this.label2.Text = "Y Index:";
            //
            // etX
            //
            this.etX.Location = new System.Drawing.Point(76, 151);
            this.etX.Name = "etX";
            this.etX.Size = new System.Drawing.Size(100, 22);
            this.etX.TabIndex = 2;
            this.etX.Text = "1.00";
            //
            // etY
            //
            this.etY.Location = new System.Drawing.Point(76, 182);
            this.etY.Name = "etY";
            this.etY.Size = new System.Drawing.Size(100, 22);
            this.etY.TabIndex = 3;
            this.etY.Text = "1.00";
            //
            // label3
            //
            this.label3.AutoSize = true;
            this.label3.Location = new System.Drawing.Point(4, 24);
            this.label3.Name = "label3";
```

```
            this.label3.Size = new System.Drawing.Size(38, 17);
            this.label3.TabIndex = 5;
            this.label3.Text = "Port:";
            //
            // cbPorts
            //
            this.cbPorts.FormattingEnabled = true;
            this.cbPorts.Location = new System.Drawing.Point(54, 21);
            this.cbPorts.Name = "cbPorts";
            this.cbPorts.Size = new System.Drawing.Size(138, 24);
            this.cbPorts.TabIndex = 6;
            //
            // rbJog
            //
            this.rbJog.AutoSize = true;
            this.rbJog.Checked = true;
            this.rbJog.Enabled = false;
            this.rbJog.Location = new System.Drawing.Point(14, 21);
            this.rbJog.Name = "rbJog";
            this.rbJog.Size = new System.Drawing.Size(52, 21);
            this.rbJog.TabIndex = 7;
            this.rbJog.TabStop = true;
            this.rbJog.Text = "Jog";
            this.rbJog.UseVisualStyleBackColor = true;
            this.rbJog.CheckedChanged += new
System.EventHandler(this.rbJog_CheckedChanged);
            //
            // rbIndex
            //
            this.rbIndex.AutoSize = true;
            this.rbIndex.Enabled = false;
            this.rbIndex.Location = new System.Drawing.Point(107, 21);
            this.rbIndex.Name = "rbIndex";
            this.rbIndex.Size = new System.Drawing.Size(62, 21);
            this.rbIndex.TabIndex = 8;
            this.rbIndex.Text = "Index";
            this.rbIndex.UseVisualStyleBackColor = true;
            this.rbIndex.CheckedChanged += new
System.EventHandler(this.rbIndex_CheckedChanged);
            //
            // groupBox1
            //
            this.groupBox1.Controls.Add(this.rbJog);
            this.groupBox1.Controls.Add(this.rbIndex);
            this.groupBox1.Location = new System.Drawing.Point(7, 72);
            this.groupBox1.Name = "groupBox1";
            this.groupBox1.Size = new System.Drawing.Size(190, 64);
            this.groupBox1.TabIndex = 9;
            this.groupBox1.TabStop = false;
            this.groupBox1.Text = "Joystick Mode";
            //
            // pbConnect
            //
            this.pbConnect.Location = new System.Drawing.Point(212, 21);
            this.pbConnect.Name = "pbConnect";
            this.pbConnect.Size = new System.Drawing.Size(92, 23);
            this.pbConnect.TabIndex = 10;
            this.pbConnect.Text = "Connect";
            this.pbConnect.UseVisualStyleBackColor = true;
            this.pbConnect.Click += new System.EventHandler(this.pbConnect_Click);
            //
            // etZ
            //
            this.etZ.Location = new System.Drawing.Point(76, 214);
            this.etZ.Name = "etZ";
            this.etZ.Size = new System.Drawing.Size(100, 22);
            this.etZ.TabIndex = 12;
            this.etZ.Text = "0.01";
            //
            // label4
            //
            this.label4.AutoSize = true;
            this.label4.Location = new System.Drawing.Point(15, 214);
            this.label4.Name = "label4";
            this.label4.Size = new System.Drawing.Size(58, 17);
            this.label4.TabIndex = 11;
            this.label4.Text = "Z Index:";
            //
            // pbResetPosition
            //
            this.pbResetPosition.Enabled = false;
            this.pbResetPosition.Location = new System.Drawing.Point(212, 79);
            this.pbResetPosition.Name = "pbResetPosition";
            this.pbResetPosition.Size = new System.Drawing.Size(88, 57);
            this.pbResetPosition.TabIndex = 15;
            this.pbResetPosition.Text = "Reset XY Position";
            this.pbResetPosition.UseVisualStyleBackColor = true;
            this.pbResetPosition.Click += new
System.EventHandler(this.pbResetPosition_Click);
            //
            // pbRaise
            //
            this.pbRaise.Enabled = false;
            this.pbRaise.Location = new System.Drawing.Point(212, 258);
            this.pbRaise.Name = "pbRaise";
            this.pbRaise.Size = new System.Drawing.Size(75, 23);
            this.pbRaise.TabIndex = 18;
            this.pbRaise.Text = "Raise";
            this.pbRaise.UseVisualStyleBackColor = true;
            this.pbRaise.Click += new System.EventHandler(this.pbRaise_Click);
            //
            // etStby
            //
            this.etStby.Location = new System.Drawing.Point(76, 258);
            this.etStby.Name = "etStby";
            this.etStby.Size = new System.Drawing.Size(100, 22);
            this.etStby.TabIndex = 17;
            this.etStby.Text = "0.1";
            //
            // label5
            //
            this.label5.AutoSize = true;
            this.label5.Location = new System.Drawing.Point(15, 258);
            this.label5.Name = "label5";
            this.label5.Size = new System.Drawing.Size(53, 17);
            this.label5.TabIndex = 16;
            this.label5.Text = "Z Stby:";
            //
            // pbLower
            //
            this.pbLower.Enabled = false;
            this.pbLower.Location = new System.Drawing.Point(212, 287);
            this.pbLower.Name = "pbLower";
            this.pbLower.Size = new System.Drawing.Size(75, 23);
            this.pbLower.TabIndex = 19;
            this.pbLower.Text = "Lower";
            this.pbLower.UseVisualStyleBackColor = true;
            this.pbLower.Click += new System.EventHandler(this.pbLower_Click);
            //
            // pbZDown
            //
            this.pbZDown.Enabled = false;
            this.pbZDown.Location = new System.Drawing.Point(21, 287);
            this.pbZDown.Name = "pbZDown";
            this.pbZDown.Size = new System.Drawing.Size(155, 23);
            this.pbZDown.TabIndex = 20;
            this.pbZDown.Text = "Set Down Here";
            this.pbZDown.UseVisualStyleBackColor = true;
            this.pbZDown.Click += new System.EventHandler(this.pbZDown_Click);
            //
            // pbHome
            //
```

```
this.pbHome.Enabled = false;
this.pbHome.Location = new System.Drawing.Point(21, 328);
this.pbHome.Name = "pbHome";
this.pbHome.Size = new System.Drawing.Size(145, 23);
this.pbHome.TabIndex = 21;
this.pbHome.Text = "Move XYZ Home";
this.pbHome.UseVisualStyleBackColor = true;
this.pbHome.Click += new System.EventHandler(this.pbHome_Click);
//
// pbCenter
//
this.pbCenter.Enabled = false;
this.pbCenter.Location = new System.Drawing.Point(178, 328);
this.pbCenter.Name = "pbCenter";
this.pbCenter.Size = new System.Drawing.Size(109, 23);
this.pbCenter.TabIndex = 22;
this.pbCenter.Text = "Center XY";
this.pbCenter.UseVisualStyleBackColor = true;
this.pbCenter.Click += new System.EventHandler(this.pbCenter_Click);
//
// groupBox2
//
this.groupBox2.Controls.Add(this.tbLog);
this.groupBox2.Controls.Add(this.lbRecv);
this.groupBox2.Controls.Add(this.pbRecv);
this.groupBox2.Controls.Add(this.pbQuery);
this.groupBox2.Controls.Add(this.pbSend);
this.groupBox2.Controls.Add(this.etCommand);
this.groupBox2.Location = new System.Drawing.Point(358, 79);
this.groupBox2.Name = "groupBox2";
this.groupBox2.Size = new System.Drawing.Size(293, 268);
this.groupBox2.TabIndex = 23;
this.groupBox2.TabStop = false;
this.groupBox2.Text = "Command I/O";
//
// tbLog
//
this.tbLog.Location = new System.Drawing.Point(11, 115);
this.tbLog.Multiline = true;
this.tbLog.Name = "tbLog";
this.tbLog.ScrollBars = System.Windows.Forms.ScrollBars.Vertical;
this.tbLog.Size = new System.Drawing.Size(268, 141);
this.tbLog.TabIndex = 5;
//
// lbRecv
//
this.lbRecv.AutoSize = true;
this.lbRecv.Location = new System.Drawing.Point(11, 89);
this.lbRecv.Name = "lbRecv";
this.lbRecv.Size = new System.Drawing.Size(103, 17);
this.lbRecv.TabIndex = 4;
this.lbRecv.Text = "Command Log:";
//
// pbRecv
//
this.pbRecv.Location = new System.Drawing.Point(205, 53);
this.pbRecv.Name = "pbRecv";
this.pbRecv.Size = new System.Drawing.Size(75, 23);
this.pbRecv.TabIndex = 3;
this.pbRecv.Text = "Receive";
this.pbRecv.UseVisualStyleBackColor = true;
this.pbRecv.Click += new System.EventHandler(this.pbRecv_Click);
//
// pbQuery
//
this.pbQuery.Location = new System.Drawing.Point(107, 53);
this.pbQuery.Name = "pbQuery";
this.pbQuery.Size = new System.Drawing.Size(75, 23);
this.pbQuery.TabIndex = 2;
this.pbQuery.Text = "Query";
this.pbQuery.UseVisualStyleBackColor = true;
this.pbQuery.Click += new System.EventHandler(this.pbQuery_Click);
//
// pbSend
//
this.pbSend.Location = new System.Drawing.Point(11, 53);
this.pbSend.Name = "pbSend";
this.pbSend.Size = new System.Drawing.Size(75, 23);
this.pbSend.TabIndex = 1;
this.pbSend.Text = "Send";
this.pbSend.UseVisualStyleBackColor = true;
this.pbSend.Click += new System.EventHandler(this.pbSend_Click);
//
// etCommand
//
this.etCommand.Location = new System.Drawing.Point(11, 25);
this.etCommand.Name = "etCommand";
this.etCommand.Size = new System.Drawing.Size(269, 22);
this.etCommand.TabIndex = 0;
this.etCommand.Text = "1QR?";
//
// pbXMinus
//
this.pbXMinus.Location = new System.Drawing.Point(182, 150);
this.pbXMinus.Name = "pbXMinus";
this.pbXMinus.Size = new System.Drawing.Size(75, 23);
this.pbXMinus.TabIndex = 24;
this.pbXMinus.Text = "<< -X";
this.pbXMinus.UseVisualStyleBackColor = true;
this.pbXMinus.Click += new System.EventHandler(this.pbXMinus_Click);
//
// pbXPlus
//
this.pbXPlus.Location = new System.Drawing.Point(263, 150);
this.pbXPlus.Name = "pbXPlus";
this.pbXPlus.Size = new System.Drawing.Size(75, 23);
this.pbXPlus.TabIndex = 25;
this.pbXPlus.Text = "+X >>";
this.pbXPlus.UseVisualStyleBackColor = true;
this.pbXPlus.Click += new System.EventHandler(this.pbXPlus_Click);
//
// pbYMinus
//
this.pbYMinus.Location = new System.Drawing.Point(182, 181);
this.pbYMinus.Name = "pbYMinus";
this.pbYMinus.Size = new System.Drawing.Size(75, 23);
this.pbYMinus.TabIndex = 26;
this.pbYMinus.Text = "<< -Y";
this.pbYMinus.UseVisualStyleBackColor = true;
this.pbYMinus.Click += new System.EventHandler(this.pbYMinus_Click);
//
// pbYPlus
//
this.pbYPlus.Location = new System.Drawing.Point(263, 181);
this.pbYPlus.Name = "pbYPlus";
this.pbYPlus.Size = new System.Drawing.Size(75, 23);
this.pbYPlus.TabIndex = 27;
this.pbYPlus.Text = "+Y >>";
this.pbYPlus.UseVisualStyleBackColor = true;
this.pbYPlus.Click += new System.EventHandler(this.pbYPlus_Click);
//
// pbZMinus
//
this.pbZMinus.Location = new System.Drawing.Point(182, 213);
this.pbZMinus.Name = "pbZMinus";
this.pbZMinus.Size = new System.Drawing.Size(75, 23);
this.pbZMinus.TabIndex = 28;
this.pbZMinus.Text = "<< -Z";
this.pbZMinus.UseVisualStyleBackColor = true;
this.pbZMinus.Click += new System.EventHandler(this.pbZMinus_Click);
//
// pbZPlus
```

```
            //
            this.pbZPlus.Location = new System.Drawing.Point(263, 213);
            this.pbZPlus.Name = "pbZPlus";
            this.pbZPlus.Size = new System.Drawing.Size(75, 23);
            this.pbZPlus.TabIndex = 29;
            this.pbZPlus.Text = "+Z >>";
            this.pbZPlus.UseVisualStyleBackColor = true;
            this.pbZPlus.Click += new System.EventHandler(this.pbZPlus_Click);
            //
            // ESP301_Form
            //
            this.AutoScaleDimensions = new System.Drawing.SizeF(8F, 16F);
            this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;
            this.ClientSize = new System.Drawing.Size(669, 369);
            this.Controls.Add(this.pbZPlus);
            this.Controls.Add(this.pbZMinus);
            this.Controls.Add(this.pbYPlus);
            this.Controls.Add(this.pbYMinus);
            this.Controls.Add(this.pbXPlus);
            this.Controls.Add(this.pbXMinus);
            this.Controls.Add(this.groupBox2);
            this.Controls.Add(this.pbCenter);
            this.Controls.Add(this.pbHome);
            this.Controls.Add(this.pbZDown);
            this.Controls.Add(this.pbLower);
            this.Controls.Add(this.pbRaise);
            this.Controls.Add(this.etStby);
            this.Controls.Add(this.label5);
            this.Controls.Add(this.pbResetPosition);
            this.Controls.Add(this.etZ);
            this.Controls.Add(this.label4);
            this.Controls.Add(this.pbConnect);
            this.Controls.Add(this.groupBox1);
            this.Controls.Add(this.cbPorts);
            this.Controls.Add(this.label3);
            this.Controls.Add(this.etY);
            this.Controls.Add(this.etX);
            this.Controls.Add(this.label2);
            this.Controls.Add(this.label1);
            this.Name = "ESP301_Form";
            this.Text = "ESP301 Index Size";
            this.groupBox1.ResumeLayout(false);
            this.groupBox1.PerformLayout();
            this.groupBox2.ResumeLayout(false);
            this.groupBox2.PerformLayout();
            this.ResumeLayout(false);
            this.PerformLayout();

        }

        #endregion

        private System.Windows.Forms.Label label1;
        private System.Windows.Forms.Label label2;
        private System.Windows.Forms.TextBox etX;
        private System.Windows.Forms.TextBox etY;
        private System.Windows.Forms.Label label3;
        private System.Windows.Forms.ComboBox cbPorts;
        private System.Windows.Forms.RadioButton rbJog;
        private System.Windows.Forms.RadioButton rbIndex;
        private System.Windows.Forms.GroupBox groupBox1;
        private System.Windows.Forms.Button pbConnect;
        private System.Windows.Forms.TextBox etZ;
        private System.Windows.Forms.Label label4;
        private System.Windows.Forms.Button pbResetPosition;
        private System.Windows.Forms.Button pbRaise;
        private System.Windows.Forms.TextBox etStby;
        private System.Windows.Forms.Label label5;
        private System.Windows.Forms.Button pbLower;
        private System.Windows.Forms.Button pbZDown;
        private System.Windows.Forms.Button pbHome;
        private System.Windows.Forms.Button pbCenter;
        private System.Windows.Forms.GroupBox groupBox2;
        private System.Windows.Forms.TextBox tbLog;
        private System.Windows.Forms.Label lbRecv;
        private System.Windows.Forms.Button pbRecv;
        private System.Windows.Forms.Button pbQuery;
        private System.Windows.Forms.Button pbSend;
        private System.Windows.Forms.TextBox etCommand;
        private System.Windows.Forms.Button pbXMinus;
        private System.Windows.Forms.Button pbXPlus;
        private System.Windows.Forms.Button pbYMinus;
        private System.Windows.Forms.Button pbYPlus;
        private System.Windows.Forms.Button pbZMinus;
        private System.Windows.Forms.Button pbZPlus;
    }
}
```

6.5.3.5 Serial Interface Class (ESP301_Serial.cs)

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using NationalInstruments.VisaNS;

namespace ESP301_GUI
{
    /// <summary>
    /// ESP301 USB Interface Class
    /// </summary>
    class ESP301_Serial
    {
        /// <summary>
        /// Error returns for functions
        /// </summary>
        public const int SERR_BASE = 0x800000;
        public const int SERR_NONE = SERR_BASE;
        public const int SERR_INVALID_PORT = SERR_BASE+1;
        public const int SERR_UNABLE_TO_OPEN = SERR_BASE + 2;
        public const int SERR_WRITE_FAILED = SERR_BASE + 3;
        public const int SERR_READ_FAILED = SERR_BASE + 4;
        public const int SERR_UNABLE_TO_CLOSE = SERR_BASE + 5;

        /// <summary>
        /// Default USB Baud rate required
        /// </summary>
        public const int BAUD_USB = 921600;

        string terminator = "\r";
        static string port_id = "ASRL7:INSTR";
        bool initialized = false;
        int baud_rate = BAUD_USB;

        /// <summary>
        /// Serial Port selection
        /// </summary>
        private SerialSession serial_port = null;

        public ESP301_Serial()
        {
        }

        /// <summary>
        /// Get a list of available VISA resources
        /// </summary>
        /// <returns>String array of available VISA resources</returns>
        static public string[] getPorts()
        {
            try
            {
                string[] resource_list =
ResourceManager.GetLocalManager().FindResources("?*");
                return resource_list;
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message);
            }
            return null;
        }

        /// <summary>
        ///
        /// </summary>
        /// <param name="commport">Name of VISA resource to open of the form "ASRLn:INSTR"
where n is the port ID</param>
        /// <returns>Error Code</returns>
        public int openSerial(string commport)
        {
            int rtn = SERR_NONE;
            if (serial_port != null)
            {
                serial_port.Dispose();
            }
            else
            {
                serial_port = new SerialSession(commport);
            }
            port_id = commport;
            serial_port.BaudRate = BAUD_USB;
            serial_port.Parity = Parity.None;
            serial_port.StopBits = StopBitType.One;
            serial_port.DataBits = 8;
            return rtn;
        }

        /// <summary>
        /// Closes the active serial port
        /// </summary>
        /// <returns>Error Code</returns>
        public int closeSerial()
        {
            int rtn = SERR_NONE;
            if (serial_port != null)
                serial_port.Dispose();
            return rtn;
        }

        /// <summary>
        /// Send a string to the ESP301
        /// </summary>
        /// <param name="output">Non terminated string to send</param>
        /// <returns>Error Code</returns>
        public int writeSerial(string output)
        {
            int rtn = SERR_NONE;
            serial_port.Write(output+this.terminator);
            System.Threading.Thread.Sleep(100);
            return rtn;
        }

        /// <summary>
        /// Receive data from the ESP301
        /// </summary>
        /// <param name="output">Destination string for data</param>
        /// <returns>Error Code</returns>
        public int readSerial(ref string output)
        {
            int rtn = SERR_NONE;
            output = serial_port.ReadString();
            return rtn;
        }

        /// <summary>
        /// Send a list of commands to the ESP301
        /// </summary>
        /// <param name="script">List of commands to send</param>
        /// <returns></returns>
        public int sendScript(string [] script)
        {
            int rtn = SERR_NONE;
            foreach (string s in script)
            {
                rtn = writeSerial(s);
                if (rtn != SERR_NONE)
                    break;
            }

            return rtn;
        }
    }
}
```

6.6 ESP301 Default Configuration Files

Two ESP301 command files are used to set up the ESP301 through the USB port:

- SETTINGS.CFG – Configures the XYZ Axes and I/O. This ensures all limits and axis motion matches the P200L hardware
- PROGRAMS.CFG – Programs to Enable/Disable attached Joystick. Program “1” enables the joystick but disables all the Move Absolute and Move Relative functions. Program “2” disables the attached joystick and enables use of the Move Absolute and Move Relative functions.

The comments shown are not present in the file stored on disk.

6.6.1 SETTINGS.CFG

```
// Motor Types (Commutated Stepper)
1QM3;2QM3;3QM3
// Motor drive voltages
1QV5;2QV5;3QV5
// Motor drive currents in Amps
1QI1.0;2QI1.0;3QI1.3
// Torque reduction delay and power reduction after move
1QR5000,50;2QR5000,50;3QR5000,50
// Microsteps per full step factor
1QS125;2QS125;3QS125
// Amplifier configuration as follows:
//      Bit     Setting  Description
//      ---     -------  -------------------
//      0       1        Enable amplifier fault input checking
//      1       1        Disable motor on amplifier fault event
//      2       1        Abort motion on amplifier fault event
//      3       0        n/a
//      4       0        n/a
//      5       1        Amplifier fault input active high
//      6       0        Stepper outputs are Step/direction
//      7       0        Step output is active low
//      8       1        Direction output is active high
//      9       0        Do not invert servo DAC output
//      10      0        Amplifier output enable active low
//      11      0        Stepper motor winding is FULL
1ZA127H;2ZA127H;3ZA127H
// Feedback disabled
1ZB0H;2ZB0H;3ZB0H
// E-Stop configuration as follows:
//      Bit     Setting  Description
//      ---     -------  -------------------
//      0       1        Enable E-Stop
//      1       1        Disable motors on E-Stop
//      2       1        Abort motion on E_Stop
1ZE7H;2ZE7H;3ZE7H
// Following error disabled
1ZF0H;2ZF0H;3ZF0H
```

```
// Hardware limit configuration as follows:
//        Bit      Setting   Description
//        ---      -------   -------------------
//        0        1         Enable Limit Checking
//        1        0         Do not disable motor on limit
//        2        1         Abort motion on travel limit event
//        3-4      0         n/a
//        5        0         Hardware limit active low
1ZH05H;2ZH05H;3ZH05H
// Software limits are disabled
1ZS0H;2ZS0H;3ZS0H
// Set system configuration as follows:
//        Bit      Setting   Description
//        ---      -------   -------------------
//        0        1         Enable 100-pin interlock error checking
//        1        1         Disable all axes on 100-pin interlock error
//        2        0         n/a
//        3        0         n/a
//        4        1         Interlock fault is active high
//        5        0         n/a
//        6        0         n/a
//        7        0         Route auxiliary I/O encoder counter channels
//        8        1         Protect ESP system-critical settings
//        9        0         Enable queue purge on time expiration
//        10       0         Do not display units on certain responses
//        11       0         Enable timeout during homing
1ZZ113H;2ZZ113H;3ZZ113H
// Units are millimeters
1SN2;2SN2;3SN2
// Display 4 significant digits
1FP4;2FP4;3FP4
// Full step resolution in mm
1FR0.00635;2FR0.00635;3FR0.000369443
// No Backlash
1BA0.0;2BA0.0;3BA0
// Maximum velocity in mm/s
1VU13.0;2VU13.0;3VU1.0
// Base velocity in mm/s
1VB0.15;2VB0.15;3VB0.01
// Move velocity in mm/s
1VA12.0;2VA12.0;3VA0.4
// Maximum jog speed
1JH12.0;2JH12.0;3JH0.4
// Base jog speed
1JW0.03;2JW0.03;3JW0.001
// Maximum acceleration in mm/s^2
1AU80.0;2AU80.0;3AU20.0
// Acceleration in mm/s^2
1AC50.0;2AC50.0;3AC10.0
// Deceleration in mm/s^2
1AG50.0;2AG50.0;3AG10.0
```

```
// Home search mode to +limit,-limit,-limit
1OM3;2OM4;3OM4
// Set axis positions to 0.0 after home
1SH0;2SH0;3SH0
// Home search high speed in mm/s
1OH10.0;2OH10.0;3OH-0.3
// Home search low speed in mm/s
1OL1.0;2OL1.0;3OL0.02
// Trajectory mode set to Jog mode (3)
// Set trajectory mode to Trapezoidal (1) in order to use programmable motion functions (see
PROGRAMS.CFG)
1TJ3;2TJ3;3TJ3
```

6.6.2 PROGRAMS.CFG

```
// Program “1” enables joystick navigation
1XX
1EP JOYSTICK ON
// Set DIO port bits 0..15 as inputs
BO0
// Assign X axis -/+ jog to port bits 11 and 10
1BP11,10,12
// Assign Y axis -/+ jog to port bits 9 and 8
2BP9,8,12
// Assign Z axis -/+ jog to port bits 14 and 13
3BP14,13,12
// Enable DIO bits for jog mode
1BQ1;2BQ1;3BQ1
// Trajectory mode set to Jog mode (3)
1TJ3;2TJ3;3TJ3
// Power on the motors
1MO;2MO;3MO
// End of Program “1”
QP
// Program “2” enables programmable navigation (e.g. “Move Absolute”) and disables the joystick
2XX
2EP JOYSTICK OFF
// Disable DIO bits for jog mode
1BQ0;2BQ0;3BQ0
// Set trajectory mode to Trapezoidal (1) for programmable motion functions
1TJ1;2TJ1;3TJ1
// End of Program “2”
QP
// Execute Program “1” at startup
1EO
```

6.7 Calibration

The P200L is an open loop system, and is subject to the effects of thermal expansion. Therefore, there may be a mismatch between physical position on a wafer and the reported position of the ESP301. The ESP301 provides access to a linear compensation factor to correct this error. It is accessed through the "CO" command. See the ESP301 user's manual page 3-41 for calculation of error and command usage. The best method for calculating the linear compensation factor is to take 10 or so points and perform linear regression on the data set.

If very precise positioning is required, an XY grid of positioning errors over the entire XY stage travel should be made to provide correction for repeatability, straightness, and accuracy. The recommended method to use for grid mapping correction is Bilinear Interpolation.

The following section provides guidelines for creation and use of a positional grid map.

6.7.1 Grid Mapping

Algorithms have been developed over the years which compensated for lead screw run out error, single axis straightness, and orthogonality error between paired axes. It has been discovered that although these methods greatly improved performance, they were not able to compensate adequately for variations in mechanical conditions caused by the stage moving in two axes. The availability of pattern recognition and calibration standard wafers enables a new type of positional compensation algorithm – grid mapping.

Grid mapping is a process whereby the X/Y stage position is correlated with displacement information obtained using pattern recognition. Data points are calculated based on expected locations of a feature with very precise spacing across the entire wafer. The stage is moved to all the features of that type on the wafer and the positional error is recorded. The resulting data set is then arranged in a grid which is used to calculate the true position of the stage. The algorithm is based on bilinear interpolation.

The key advantage of the grid mapping method is that the entire X/Y range of travel is characterized, rather than just one axis. There is one data set which describes X/Y run out, straightness, orthogonality, and other repeatable biaxial behaviors.

Grid Mapping Logic

The grid map locations are acquired as follows:

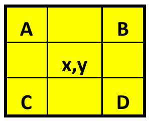
						159	160	161						
				152	153	154	155	156	157	158				
		141	142	143	144	145	146	147	148	149	150	151		
		130	131	132	133	134	135	136	137	138	139	140		
	117	118	119	120	121	122	123	124	125	126	127	128	129	
	104	105	106	107	108	109	110	111	112	113	114	115	116	
89	90	91	92	93	94	95	96	97	98	99	100	101	102	103
74	75	76	77	78	79	80	81	82	83	84	85	86	87	88
59	60	61	62	63	64	65	66	67	68	69	70	71	72	73
	46	47	48	49	50	51	52	53	54	55	56	57	58	
	33	34	35	36	37	38	39	40	41	42	43	44	45	
		22	23	24	25	26	27	28	29	30	31	32		
		11	12	13	14	15	16	17	18	19	20	21		
				4	5	6	7	8	9	10				
						1	2	3						

1. Create a calibration file with an xy grid of calibration points
2. Each calibration point has x and y correction values.
3. The file starts with a grid point having the smallest y value first, then the smallest x value.
4. The next point has the next larger x value and so on across the first row.
5. The second row starts with the next larger y value and the smallest x value.
6. Repeat until all locations are stored.

The resulting file contains a list of all locations and their associated error.

Interpolation Strategy

The value of the compensation for a given location is calculated from the X,Y locations of the four corners around the position as shown below.



1. Before a move to a target location (xm,ym) is executed, correction values are read from the map.

2. Search map for 4 grid calibration points (ABCD) which encompass the target (xy)

Where:

A=	1ST VALUE >y, LAST VALUE <x
B=	1ST VALUE >y, 1ST VALUE >x
C=	LAST VALUE<y, LAST VALUE <x
D=	LAST VALUE <y, 1ST VALUE >x

The inverse of the above process is used to convert from a “corrected” value to an uncorrected value.

Search Logic:

1. Calculate corrected x value (xcor):
2. Interpolate between A+B: linab = axr+(xm-ax)*((bxr-axr)/(bx-ax))
3. Interpolate between C+D: lincd = cxr+(xm-cx)*((dxr-cxr)/(dx-cx))
4. Interpolate between linab+lincd: linabcd = lincd+(ym-cy)*((linab-lincd)/(ay-cy))
5. xcor = xm+round(linabcd,6)
6. Calculate corrected y value (ycor)
7. Interpolate between C+A: linca = cyr+(ym-cy)*((ayr-cyr)/(ay-cy))
8. Interpolate between D+B: lindb = dyr+(ym-dy)*((byr-dyr)/(by-dy))
9. Interpolate between lindb+linca: lincadb = lindb+(xm-bx)*((lindb-linca)/(bx-ax))
10. ycor = ym+round(lincadb,6)
11. Use the results to move.