www.keithley.com

S530 Parametric Test System

Prober and Prober Driver Manual

S530-911-01 Rev. B / January 2014

S530-911-01

A Greater Measure of Confidence

KEITHLEY
A Tektronix Company

S530 Parametric Test Systems

Prober and Prober Driver

Manual

Document Number: S530-911-01 Rev. B / January 2014

KEITHLEY
A Tektronix Company

Safety precautions

The following safety precautions should be observed before using this product and any associated instrumentation. Although some instruments and accessories would normally be used with nonhazardous voltages, there are situations where hazardous conditions may be present.

This product is intended for use by qualified personnel who recognize shock hazards and are familiar with the safety precautions required to avoid possible injury. Read and follow all installation, operation, and maintenance information carefully before using the product. Refer to the user documentation for complete product specifications.

If the product is used in a manner not specified, the protection provided by the product warranty may be impaired.

The types of product users are:

Responsible body is the individual or group responsible for the use and maintenance of equipment, for ensuring that the equipment is operated within its specifications and operating limits, and for ensuring that operators are adequately trained.

Operators use the product for its intended function. They must be trained in electrical safety procedures and proper use of the instrument. They must be protected from electric shock and contact with hazardous live circuits.

Maintenance personnel perform routine procedures on the product to keep it operating properly, for example, setting the line voltage or replacing consumable materials. Maintenance procedures are described in the user documentation. The procedures explicitly state if the operator may perform them. Otherwise, they should be performed only by service personnel.

Service personnel are trained to work on live circuits, perform safe installations, and repair products. Only properly trained service personnel may perform installation and service procedures.

Keithley Instruments products are designed for use with electrical signals that are measurement, control, and data I/O connections, with low transient overvoltages, and must not be directly connected to mains voltage or to voltage sources with high transient overvoltages. Measurement Category II (as referenced in IEC 60664) connections require protection for high transient overvoltages often associated with local AC mains connections. Certain Keithley measuring instruments may be connected to mains. These instruments will be marked as category II or higher.

Unless explicitly allowed in the specifications, operating manual, and instrument labels, do not connect any instrument to mains.

Exercise extreme caution when a shock hazard is present. Lethal voltage may be present on cable connector jacks or test fixtures. The American National Standards Institute (ANSI) states that a shock hazard exists when voltage levels greater than 30 V RMS, 42.4 V peak, or 60 VDC are present. A good safety practice is to expect that hazardous voltage is present in any unknown circuit before measuring.

Operators of this product must be protected from electric shock at all times. The responsible body must ensure that operators are prevented access and/or insulated from every connection point. In some cases, connections must be exposed to potential human contact. Product operators in these circumstances must be trained to protect themselves from the risk of electric shock. If the circuit is capable of operating at or above 1000 V, no conductive part of the circuit may be exposed.

Do not connect switching cards directly to unlimited power circuits. They are intended to be used with impedance-limited sources. NEVER connect switching cards directly to AC mains. When connecting sources to switching cards, install protective devices to limit fault current and voltage to the card.

Before operating an instrument, ensure that the line cord is connected to a properly-grounded power receptacle. Inspect the connecting cables, test leads, and jumpers for possible wear, cracks, or breaks before each use.

When installing equipment where access to the main power cord is restricted, such as rack mounting, a separate main input power disconnect device must be provided in close proximity to the equipment and within easy reach of the operator.

For maximum safety, do not touch the product, test cables, or any other instruments while power is applied to the circuit under test. ALWAYS remove power from the entire test system and discharge any capacitors before: connecting or disconnecting cables or jumpers, installing or removing switching cards, or making internal changes, such as installing or removing jumpers.

Do not touch any object that could provide a current path to the common side of the circuit under test or power line (earth) ground. Always make measurements with dry hands while standing on a dry, insulated surface capable of withstanding the voltage being measured.

For safety, instruments and accessories must be used in accordance with the operating instructions. If the instruments or accessories are used in a manner not specified in the operating instructions, the protection provided by the equipment may be impaired.

Do not exceed the maximum signal levels of the instruments and accessories, as defined in the specifications and operating information, and as shown on the instrument or test fixture panels, or switching card.

When fuses are used in a product, replace with the same type and rating for continued protection against fire hazard.

Chassis connections must only be used as shield connections for measuring circuits, NOT as protective earth (safety ground) connections.

If you are using a test fixture, keep the lid closed while power is applied to the device under test. Safe operation requires the use of a lid interlock.

If a  screw is present, connect it to protective earth (safety ground) using the wire recommended in the user documentation.

The  symbol on an instrument means caution, risk of danger. The user must refer to the operating instructions located in the user documentation in all cases where the symbol is marked on the instrument.

The  symbol on an instrument means caution, risk of electric shock. Use standard safety precautions to avoid personal contact with these voltages.

The  symbol on an instrument shows that the surface may be hot. Avoid personal contact to prevent burns.

The  symbol indicates a connection terminal to the equipment frame.

If this  symbol is on a product, it indicates that mercury is present in the display lamp. Please note that the lamp must be properly disposed of according to federal, state, and local laws.

The **WARNING** heading in the user documentation explains dangers that might result in personal injury or death. Always read the associated information very carefully before performing the indicated procedure.

The **CAUTION** heading in the user documentation explains hazards that could damage the instrument. Such damage may invalidate the warranty.

Instrumentation and accessories shall not be connected to humans.

Before performing any maintenance, disconnect the line cord and all test cables.

To maintain protection from electric shock and fire, replacement components in mains circuits — including the power transformer, test leads, and input jacks — must be purchased from Keithley Instruments. Standard fuses with applicable national safety approvals may be used if the rating and type are the same. Other components that are not safety-related may be purchased from other suppliers as long as they are equivalent to the original component (note that selected parts should be purchased only through Keithley Instruments to maintain accuracy and functionality of the product). If you are unsure about the applicability of a replacement component, call a Keithley Instruments office for information.

To clean an instrument, use a damp cloth or mild, water-based cleaner. Clean the exterior of the instrument only. Do not apply cleaner directly to the instrument or allow liquids to enter or spill on the instrument. Products that consist of a circuit board with no case or chassis (e.g., a data acquisition board for installation into a computer) should never require cleaning if handled according to instructions. If the board becomes contaminated and operation is affected, the board should be returned to the factory for proper cleaning/servicing.

Safety precaution revision as of January 2013.

Table of Contents

Section	Topic	Page
1	**Scope**	1-1
	Trademarks and registered trademarks	1-2
	Manual contents	1-2
	Safety symbols and terms	1-3
2	**Installation**	2-1
	Prober requirements	2-2
	General (all prober models)	2-2
	Electroglas probers	2-2
	TEL probers	2-3
	TSK probers	2-4
	Mechanical interfacing	2-4
	Connection	2-4
	IEEE-488 connection	2-4
	Configuration files (prbcnfg_xxxx.dat)	2-6
	S530 KTE	2-6
3	**Prober and Execution Engine Operation**	3-1
	General information	3-2
	Target die	3-2
	Probe die	3-3
	Chuck movement	3-4
	Prober mode of operation	3-5
	External mode	3-5
	Execution engine	3-6
	Initialize prober	3-4
	Wafer loop	3-8
	Exit execution engine	3-15
	Typical cycle of execution	3-16
4	**Troubleshooting**	4-1
	Environment variables used to debug probers	4-2
	Error level -- KI_PRB_ERROR_LEVEL 0-3	4-2
	Enable prober debug message while prober I/O is running	4-3
	Error messages and recovery procedures	4-3
	Return values	4-3
	Examples	4-7
	Handling alignment errors	4-8
	Error detection when writing code	4-9
	Using a UAP for error recovery	4-9
	Example	4-9
5	**Reference**	5-1
	Introduction to command summary and syntax	5-2
	Command format	5-2
	Prober command return status	5-2
	Quick reference table	5-3
	Prober commands	5-7
	Calling prober commands as functions	5-27

Acquiring prober data 5-28

6 Modifying Prober Software 6-1
Introduction 6-2
Linux 6-2
Adding a function to an existing prober driver 6-3
Alternative to adding a new function to a prober driver 6-14
Debugging probers 6-14
Enabling the prober transaction log 6-15
FAKE prober debug 6-16
Using the transaction log to isolate problems 6-19

7 Sample Files 7-1
General prober files (all supported models) 7-2
Detailed Explanation of prbcnfg.dat fields 7-2
Electroglas 2001/2010/3001/4085 (EG2X driver) 7-4
Electroglas 4060/4080/4090 (EG40 driver) 7-4
Tel P8 (TELP8 driver) 7-8
TSK/Accretech probers (TSK9 driver) 7-11
T19S GPIB (T19S driver) 7-15

8 Prober AddIn Library 8-1
Overview 8-2
SofTouch 8-2

9 Prober I/O 9-1
Summary of usage 9-2
Prober I/O description 9-2
Example 1: Idle TSK prober 9-3
Example 2: Active TSK prober 9-3
Prober I/O configuration 9-4
New environment variables used 9-4
Prober I/O Access Point (PAP) routines 9-5
Prober I/O commands 9-5
“mbx_do_xact” 9-6
prbsrq_xxxx.dat 9-6
Example (TSK prober driver) 9-6
FAKE probers, no probers, and real probers 9-8
PrbGen Client/Server/IO 9-8
PRBGEN_SERVER/PRBGEN_CLIENT 9-8
PRBGEN_IO Modification 9-8
PRBGEN_SERVER Modification 9-9
PRBGEN_CLIENT Modification 9-9
GPIB adaptor installation 9-9
Adaptor installation 9-9
NI-488.2 software installation procedure 9-9

A EG 2001/2010/3001/4085 Prober Information A-1
Prober start-up A-2
Prober setup and initialization A-3
Installation verification program A-7
EG2X driver configuration A-7
Command list A-8
Error symbols and returned values table A-8

B EG4060/4080/4090/5|300 Prober Information B-1
Prober GPIB connection B-2
Prober start-up B-2
Prober setup and initialization B-2
Quadrant setup B-6
Installation verification program B-8
Updating prbsqr_EG40.dat B-9

Command list B-10
Error symbols and returned values table B-10

C TEL Prober Information C-1
Prober start-up C-2
Prober setup and initialization C-2
Installation verification program C-4
P8 driver configuration C-4
Load first wafer remotely C-4
Updating prbsqr_P8.dat C-5
Environment variable KI_PRB_CONFIG C-5
Command list C-6
Error symbols and returned values table C-6

D TSK/Accretech Prober Information D-1
Prober start-up D-2
Prober configuration D-4
Prober setup and initialization D-5
Installation verification program D-6
Command list D-8
Updating prbsqr_TSK9.dat D-8
Error symbols and returned values table D-9

E T19S (GPIB Driver) Prober Information E-1
Prober start-up E-2
Additional Documentation E-2
Prober setup E-2
Prober DIP switch settings E-2
Menu setups E-2
GPIB setup E-9
Installation verification program E-9
Command list E-9
Error symbols and returned values table E-10

F GPIB Adapter Installation F-1
GPIB adapter installation F-2
NI-488.2 software installation procedure F-2
GPIB Configuration F-2

1
Scope

Trademarks and registered trademarks

Manual contents

NOTE *This manual covers software versions KTE 5.3 or greater on the S530 platform.*

Information about the following probers is contained in this manual:

Electroglas models: EG2001, EG2010, EG3001, EG4060, EG4080, EG4085, EG4090, EG5|300 (GPIB or RS-232 interface—system dependent).

TSK/Accretech models: A-PM-90A, UF190, UF200, UF300, and UF3000 (GPIB only)

TEL models: P8-series, P12-series, T19S, Precio (GPIB only)

This manual is organized as follows:

Section 2 — Installation information including: prober requirements (hardware and software), mechanical interfacing, and connection (GPIB) information.

Section 3 — Prober and execution engine operation information including: general information (target die, probe die, and chuck movement), prober modes of operation (external mode), and the execution engine including a typical cycle of execution.

Section 4 — Troubleshooting information including: error messages and recovery procedures, handling alignment errors, and error detection when writing code.

Section 5 — Command reference information including: command format, prober command return status, prober commands, and calling prober commands in functions.

Section 6 — Modifying prober software information including: adding a function to an existing prober driver and an alternative, and debugging probers.

Section 7 — Sample configuration files.

Section 8 — Prober AddIn library information which contains additional functions performed by the prober.

Section 9 — Prober I/O provides a means to quickly launch prober specific processes for unsolicited SRQs.

Appendices contain prober specific information including:

Appendix A — Electroglas EG2001/2010/3001/4085 Prober information

Appendix B — Electroglas 4060/4080/4090/5|300 Prober information

Appendix C — TEL: P8-series, P12-series, T19S, Precio Prober information

Appendix D — TSK/Accretech: A-PM-90A, UF-190/200/300/3000 Prober information

Appendix E — T19S (GPIB Driver) Prober information

Appendix F — GPIG adapter installation

Safety symbols and terms

The following symbols and terms may be found on an instrument or used in this manual.

The  symbol on an instrument indicates that the user should refer to the operating instructions located in the instruction manual.

The  symbol indicates that voltage levels greater than 30V RMS, 42.4V peak, or 60VDC are present. Use standard safety precautions to avoid personal contact with these voltages.

The **WARNING** heading used in this manual explains dangers that might result in personal injury or death. Always read the associated information very carefully before performing the indicated procedure.

The **CAUTION** heading in a manual explains hazards that could result in minor injury or damage the instrument. Such damage may invalidate the warranty.

The **ATTENTION** heading in a manual explains hazards that could damage the instrument. Such damage may invalidate the warranty.

2
Installation

Prober requirements

General (all prober models)

Probers have specific installation requirements (system dependent). Refer to Table 2-1 for general requirements and to Table 2-2 through Table 2-6 for specific requirements along with part numbers and recommendations for options.

Table 2-1
General prober requirements

Requirement	Description
Chuck connection requirements	A specific connection cable is required. The cable provides the connection point for the S530. The chuck connection is treated as a pin called Chuck. For more information, see the S530 KTE Linear Parametric Test Library manual.
Other options	Recommendations for triax chucks, thermal chucks, software options, etc. — contact Keithley for recommended options.

Electroglas probers

In addition to the general requirements contained in Table 2-1, Electroglas probers have specific installation requirements (system dependent). Table 2-2 and Table 2-3 contain these requirements along with part numbers.

Table 2-2
EG2001/2010/3001/4085 Prober requirements

	S530
Keithley Prober Support Package Model Number	Use of an adapter is required (e.g., Model 9139A-PCA).
Ring Carrier/Top plate	Dependent on adapter/method used to connect to the DUT.
Software version*	Handler Software 247845-001.HB Prober Software 249779-121.DA

*Software version—version of prober software used to verify prober driver and configuration.

Table 2-3
EG4060/4080/4090 Prober requirements

	S530
Keithley Prober Support Package Model Number	Use of an adapter is required (e.g., Model 9139A-PCA).
Ring Carrier/Top plate	Dependent on adapter/method used to connect to the DUT.
Software version*	Prober Control Software 4.702 EG Comm Firmware EGCOMM C Z Motion Control Firmware E XY Motion Control Firmware A

*Software version—version of prober software used to verify prober driver and configuration.

Table 2-4
EG4|200E and EG5|300E Prober requirements

	S530
Keithley Prober Support Package Model Number	Use of an adapter is required (e.g., Model 9139A-PCA).
Ring Carrier/Top plate	Dependent on adapter/method used to connect to the DUT.

TEL probers

In addition to the general requirements contained in Table 2-1, TEL P8 probers have specific installation requirements (system dependent). Table 2-5 contains these requirements along with part numbers.

Table 2-5
TEL P8 Prober requirements

	S530
Keithley Prober Support Package Model Number	Use of an adapter is required (e.g., Model 9139A-PCA).
Ring Carrier/Top plate	Dependent on adapter/method used to connect to the DUT.
Software version(s)*	Pap00-R011.02-2 and PZZ00-R009.02.2

*Software version(s)—version(s) of prober software used to verify prober driver and configuration.

TSK probers

In addition to the general requirements contained in Table 2-1, TSK probers have specific installation requirements (system dependent). Table 2-6 contains these requirements along with part numbers.

Table 2-6
TSK/Accretech A-PM-90A, TSK UF190, TSK UF200, TSK UF300, and TSK UF3000 Prober requirements

	S530
Keithley Prober Support Package Model Number	Use of an adapter is required (e.g., Model 9139A-PCA).
Ring Carrier/ Top plate	Dependent on adapter/method used to connect to the DUT.

Mechanical interfacing

To properly connect a prober to a system, an adapter is needed. For the S530, a separate adapter (such as the Model 9139A-PCA) is required. Referring to the instructions provided with the adapter, install the adapter on the prober. Reference the specific prober manufacturer's manual prior to installing the adapter for any applicable precautions or adapter installation instructions. Make sure adapter grounds have been correctly installed.

CAUTION **Noise problems may result if the adapter grounds are not correctly installed.**

Connection

Use the information contained in this section to connect the prober to the tester (see "IEEE-488 connection," page 2-4).

Refer to Table 2-7 for information on IEEE-488 connections.

Table 2-7
Communication configuration

System	Communication type	Cable	See paragraph titled
S530	**IEEE-488**	IEEE-488 (for pinouts, refer to Table 2-8)	"IEEE-488 connection," page 2-4.

IEEE-488 connection

Make the IEEE-488 connection using a shielded IEEE-488 cable (Table 2-8). The S530 uses a USB-GPIB adaptor for control of the prober.

NOTE *When attaching instruments to the IEEE-488 bus:*

- *allow a maximum separation of 4 meters between any two instruments on the bus.*

- *limit maximum total cable length to 20 meters.*
- *allow no more than 15 devices on the bus.*
- *do not allow two instruments to have the same address.*

If you cannot meet these requirements, use bus extenders (recommended).

CAUTION **IEEE-488 common is connected to digital common. Maximum voltage between digital common and earth ground is 0V.**

Connectors may be stacked to allow a number of parallel connections to one instrument. Two screws located on a standard connector maintain secure connections between connectors.

NOTE *To minimize interference caused by electromagnetic radiation, use shielded IEEE-488 cables.*

NOTE *On Electroglas probers, use the GPIB connection labeled IEEE-488 (do not connect the prober to the connection labeled Tester).*

Table 2-8
IEEE-488 contact designation

Contact number	Contact designation	Type
1	DIO1	Data
2	DIO2	Data
3	DIO3	Data
4	DIO4	Data
5	EOI (24)*	Management
6	DAV	Handshake
7	NRFD	Handshake
8	NDAC	Handshake
9	IFC	Management
10	SRQ	Management
11	ATN	Management
12	SHIELD	Ground
13	DIO5	Data
14	DIO6	Data
15	DIO7	Data
16	DIO8	Data
17	REN (24)*	Management
18	Gnd, (6)*	Ground
19	Gnd, (7)*	Ground
20	Gnd, (8)*	Ground
21	Gnd, (9)*	Ground
22	Gnd, (10)*	Ground
23	Gnd, (11)*	Ground
24	Gnd, LOGIC	Ground

* Number in parentheses refers to signal ground return of reference contact number. EOI and REN signal lines return on contact 24.

Configuration files (prbcnfg_xxxx.dat)

S530 KTE

The unit numbers defined in the `prbcnfg_XXXX.dat` files must be configured properly. The USB-GPIB adaptor that is used for prober communications should be configured as Unit 1. Make sure to configure the UNIT numbers correctly.

3
Prober and Execution Engine Operation

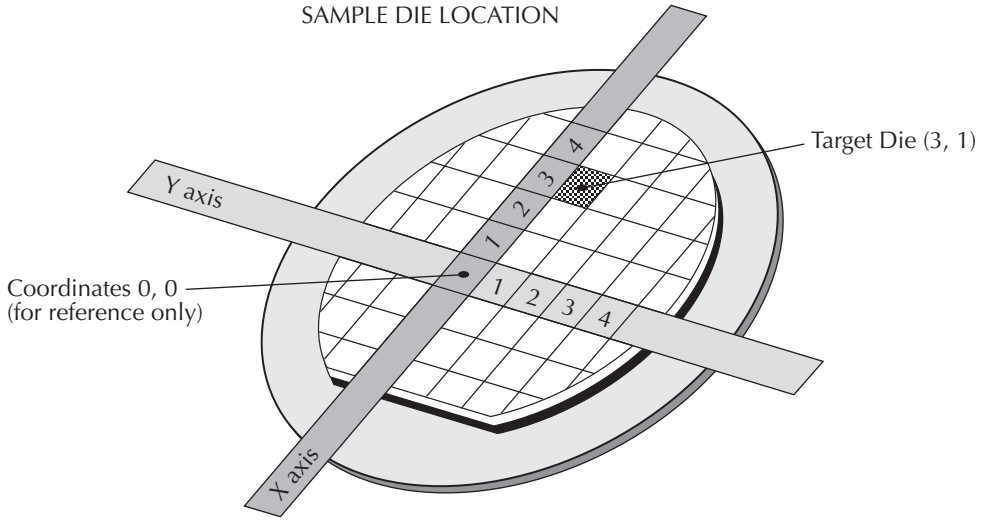
General information

Target die

The die designated as the target die defines the prober's first stopping point after alignment. The target die's physical location may be any coordinate selected on the wafer. The target die may be selected for probing or marked as the target through WDU. In WDU, the wafer's coordinate system is defined by the target die's coordinates. For example, the target die's coordinates shown in Figure 3-1 are (3, 1).

NOTE *The Target Die defined in WDU must agree with the physical location of the wafer. This is the location on the wafer directly below the probe pins after the prober loads the wafer and the chuck movement stops.*

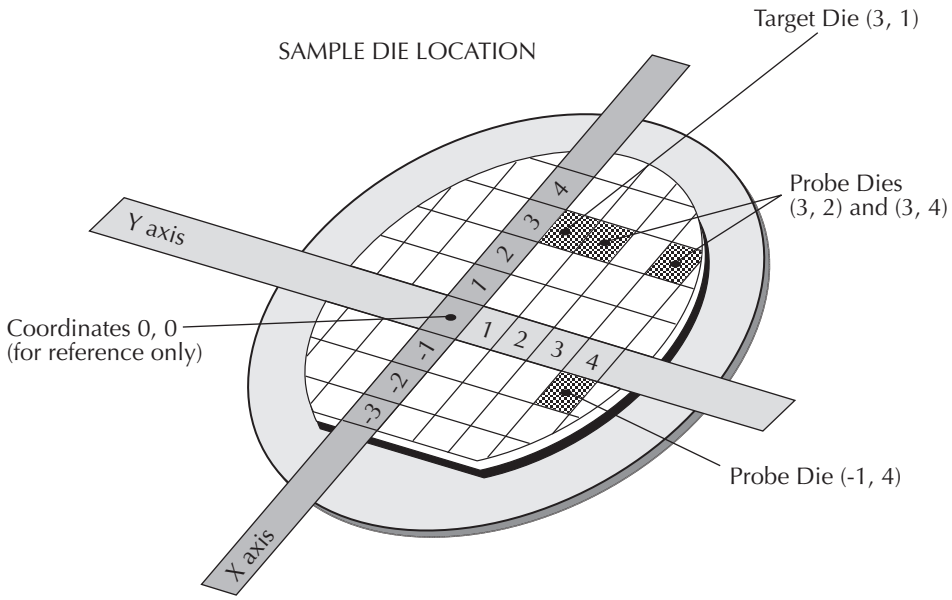
Figure 3-1
Sample target die location



Probe die

Dies marked as probe dies define the areas to be tested. The probe die's physical location can be any coordinates selected on the wafer. Marking a die as a probe die also selects the die for probing. In WDU, each probe die's coordinates are referenced with respect to the Target Die's coordinates. For example, with the target die of (3, 1), the coordinates of the three probe dies shown in Figure 3-2 are (3, 2), (3, 4), and (-1, 4).

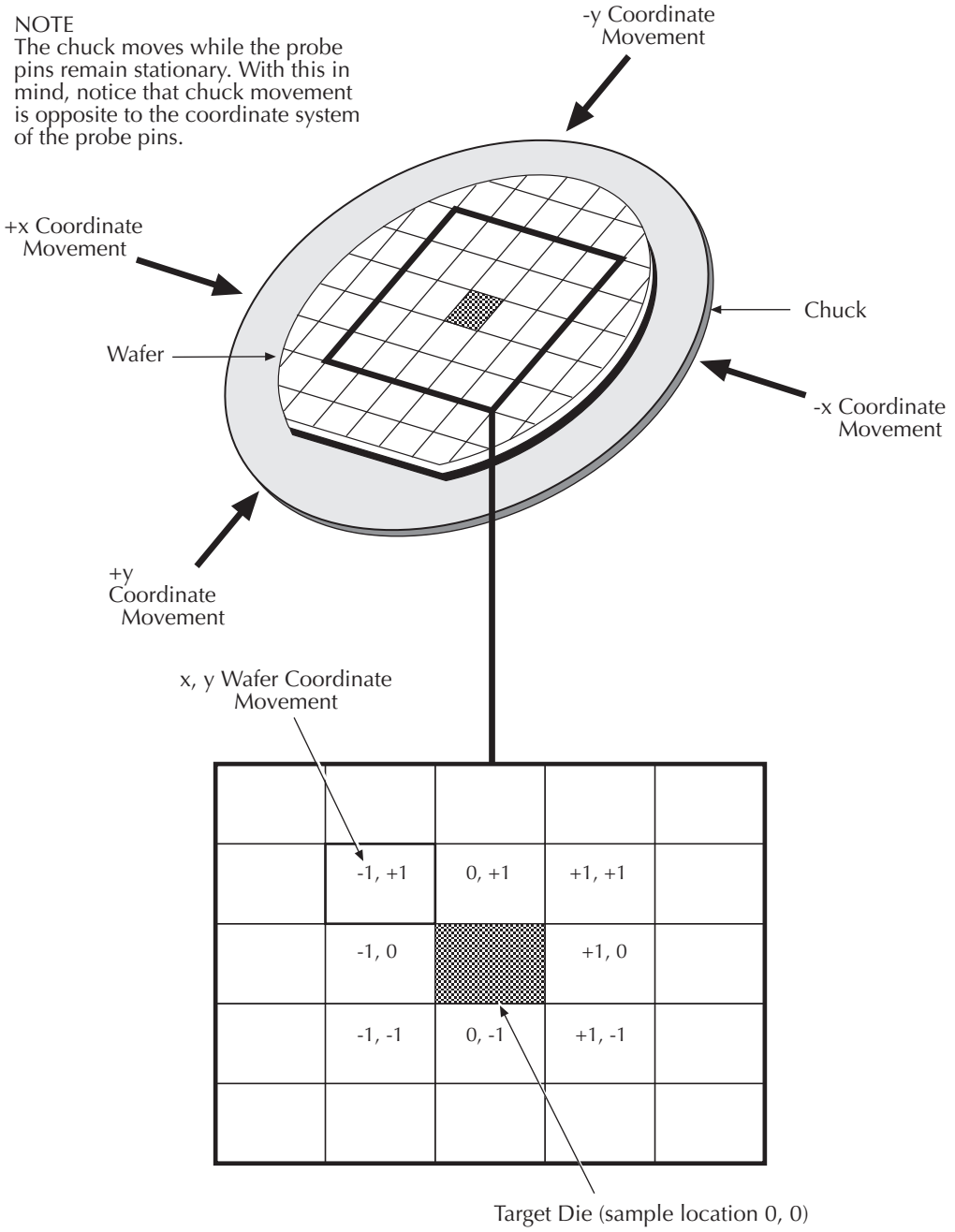
Figure 3-2
Sample probe die locations



Chuck movement

Coordinate movements are described using a First Quadrant coordinate system and x, y coordinates (+x values move East and +y values move North). To accommodate this system, the correct quadrant must be configured (prober dependent). Applicable quadrant setup instructions are contained in the applicable appendix (prober specific). When specifying chuck movements, use the coordinates of the desired site. The chuck will automatically move in the proper direction to position the probe pins over the correct die. For example, to move from the Target Die to the die up one and over one, command the chuck to move (1, 1). Refer to Figure 3-3 for a representation of the relationship between chuck movement and (x, y) coordinates.

Figure 3-3
Chuck movement



Prober mode of operation

External mode

The external mode is used to control the prober on an individual die basis. In this mode, the tester will tell the prober to go to a location (1,2), perform a test, and then go to another location (3,4), perform a test, etc. In this mode, each movement of the prober is completely controlled by the tester. On some probers, the chuck position must be controlled. In order to perform a move, the chuck must first be set down, the stage moved to the next probe site, and then the chuck must be raised. For example, if the chuck is already up at the beginning of a move, the prober will set it down, move to the new location, and automatically raise the chuck again. If, however, the chuck is down at the beginning of a move, it will remain down when the move is completed.
The main concern when using this mode is that external mode moves must be made using the PrMove command. The PrMove command has parameters for specifying the x and y locations of the next die to be probed.
For probers operating in external mode, the PrRelMove and PrRelReturn commands are used to perform subsite probing. With this method:

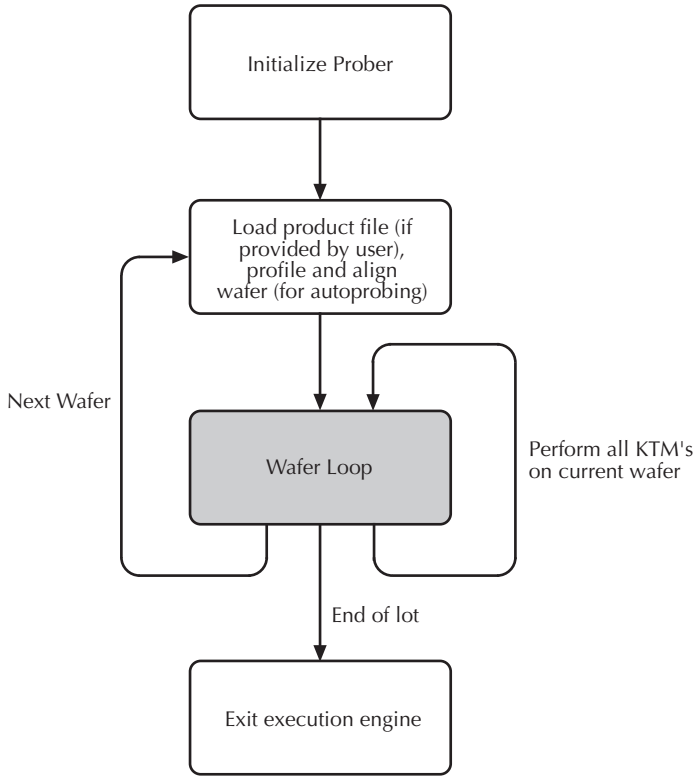
NOTE *KTXE will not issue any type of XY chuck movement command(s) if it is determined that the physical location will not change.*

1. An alignment at the first subsite to be probed is performed.
2. After probing the first probe site, the probe is moved a specified number of millimeters (or mils) to the next subsite to be probed using the PrRelMove command.
3. After using this method to probe all subsites within a given major site, the PrRelReturn command is used to move the probe tips back to the first subsite probed so that a major site move can be initiated.
4. After movement by an integral number of die-size steps with the PrMove command, the prober is at a location where it can begin subsite probing again.

Execution engine

The generic KTE 5.3 execution engine for probers is customized by UAPs (user access points). UAPs provide information to the system such as program arguments, cassette loading information, lot information, and the prober based initialization filename. Figure 3-4 illustrates the typical order of execution engine events with respect to the prober.

Figure 3-4
Overview KTXE prober interaction



Initialize prober

The PrCheckOptions() command (internal to the execution engine) determines which options/commands are available on the prober. With this information, the execution engine configures itself to use only available options.

NOTE *The PrCheckOptions() command will interpret a disabled prober option as unavailable. The system will not be capable of auto-probing until all required options are enabled at the prober.*

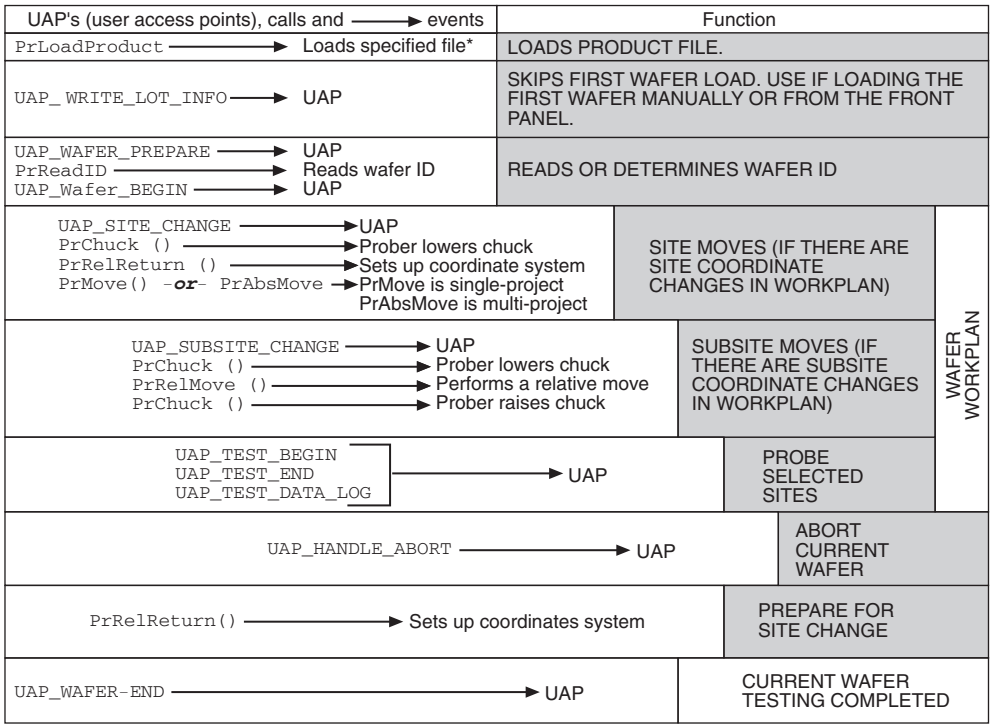
Through dialog with the operator, the execution engine loads the wafer cassette on the prober. The execution engine finds wafers to probe using the PrCassetteMap() and PrSetSlotStatus() commands if applicable. These commands, along with the probing mode used, determine which wafers to probe. The results of this search are based on the cassette plan (testing plan) and the prober cassette contents. From the cassette plan, the slot status will be set to be probed or skipped. Through the test plan, it is determined whether to test in absolute (test all slots) or relative mode (test only slots containing wafers). The PrInit() command is used to set the prober units and prober mode. After the PrInit command has been completed, the product file is loaded (if the product filename was provided by the user through a UAP). Enter a product file name (typically case sensitive) in the Device Name field (lot struct) after you start the engine. Before entering a product file name, you need to add the following UAP: KTXEAddIn/KTXEGetProductFile at UAP_PROBER_INIT. In addition, a network drive

name can be specified where the product file resides — specify the network drive letter when specifying a product file to be loaded. The product file loading sequence is the same except the engine will now allow for a single letter drive name followed by a colon (:) then the product file. The format is as follows: `d:filename`, where d is a single letter drive name and filename is the name of the product file to load. If no product file name is provided, the execution engine uses the present prober setup. Dependent on the options available, the PrProfileWafer() and PrAutoAlign() commands will then be executed which ends initialization and starts the wafer loop.

Wafer loop

At the beginning of the wafer loop, the wafer ID is read by an OCR unit (if available). If an OCR unit is not available, the engine assigns an ID to the wafer in all modes. In RELative and ABSolute mode, the user can (if desired) assign a specific ID to the wafer in the cassette plan. If in Operator mode, user specifies the wafer ID for the wafers to be probed. For more information, refer to the specific platforms engineering documentation. For every KTM (Keithley Test Macro) tested, there may be a site coordinate change, a subsite coordinate change, or no site changes. If there are no site changes, all the KTMs are performed at the same coordinates. The execution engine also controls the chuck position (up or down). This is accomplished by the PrChuck() command. The execution engine lowers the chuck before each chuck movement and raises the chuck making contact with the wafer before probing. The site and subsite moves, along with the wafer testing, are contained in the Wafer Workplan. Figure 3-5 illustrates the basic process of the wafer loop.

Figure 3-5
Wafer loop process



*PrLoadProduct - Refer to the specific prober's addendix for details.

Exit execution engine

After the wafer loop and testing is completed for a given wafer, the cycle continues loading another wafer. Based on the status returned from the load operation, wafers are loaded and tested until the lot ends. The end of the lot is determined one of two ways: either the cassette is empty or there are no entries left in the wafer testing list.

Typical cycle of execution

Figure 3-6 details a typical execution engine's interaction with KTXE.

Figure 3-6
Prober interactions in KTXE

```
UAP-PROG-ARGS                                                    INITIALIZE PROBER
    UAP-CASSETTE_LOAD
    UAP_LOT_INFO
    UAP_PROBER_INIT
    EnableTransactionLogging()
    KTXESetProductFile()
    PrCheckOptions()
     Rules for PrCheckOptions parameters
        prober_has_id_reader:
        Means prober driver supports PrReadId call.
        prober_has_autoalign:
        Means prober driver supports PrAutoAlign call.
        prober_has_profiler:
        Means prober driver supports PrProfile call.
        prober_has_hot_chuck:
        Means prober has a hot chuck. No calls at time of publication.
        prober_has_handler:
        Means prober driver supports PrCassetteMap call.
        prober_has_probe2pad:
        Means prober supports pin-to-pad alignment. No calls at time of publication.
Dialog with operator to load wafer cassette on prober
PrCassetteMap () and PrSetSlotStatus()
PrInit()
```

```
   PrLoad()                                                  LOAD PRODUCT FILE AND WAFER
 PrProfileWafer()
  PrAutoAlign()
```

```
    UAP_WAFER-PREPARE                                                      WAFER LOOP
           PrReadID
           UAP_WAFER_BEGIN
                UAP_SITE_CHANGE
                 PrChuck()
                 PrRelReturn()
                 PrMove() or PrAbsMove()
                     UAP_SUBSITE_CHANGE
                     PrChuck()
                     PrRelMove()
                     PrChuck()
                          UAP_TEST_BEGIN
                          UAP_TEST_END
                          UAP_TEST_DATA_LOG
                             UAP_HANDLE_ABORT
                     PrRelReturn()
            UAP_WAFER_END
            PrLoad()
            PrProfile()
            PrAutoAlign()
```

```
 UAP_LOT_END                                              COMPLETE LOOPS FOR ALL WAFERS
 UAP_ENGINE_EXIT
```

Definitions
UAP_PROBER_INIT — Used to load prober based initialization file (i.e., production file).
PrCassetteMap() and PrSetSlotStatus() — Find wafers to prober based on cassette plan and prober cassette contents.
PrCheckOptions() — Checks which options are supported and should be used for the specific prober. Some options may be disabled at the prober. These options may need to be enabled before full automatic operation can occur.

4
Troubleshooting

Environment variables used to debug probers

Error level -- KI_PRB_ERROR_LEVEL 0-3

0 - Verbose System Defined (stdout)

Default error message file prbmsg.msg located in KIDAT.

User Configurable Input file: N/A

1 - DisplayErrorMsg User Defined (stdout)

Default error message input file prb_errm.dat located in KIDAT.

User Configurable Input file prb_errm.dat located in KI_PRB_ERRM KIDAT.

2 - Disabled

3 - Output to file

Default error message input file: prb_errm.dat located in KIDAT (default error message input file).

User Configurable:

prb_errm.dat located in KI_PRB_ERRM KIDAT

KI_PRB_ERROR_LOG

filename/stdout (default == KILOG/prb_errs.log)

written to if err level == 3

Auditing Tester/Prober communications

KI_PRB_AUDIT_LOG

Default output filename prober.log located in KILOG.

Useful commands

EnableTransactionLogging — opens file KI_PRB_AUDIT_LOG and writes header information.

EnableErrorLogging(x) — sets error level to x (0 - 3 see above).

DisableTransactionLogging — stops transaction logging to KI_PRB_AUDIT_LOG

KI_PRB_DEBUG — write to KI_PRB_AUDIT_LOG

PRINT — prb_debug_print_flag — general debug information.

TRANS — prb_debug_trans_flag — displays the communication between the prober and tester.

CMNDS — prb_debug_cmnds_flag — identified the prober function called.

Examples

This section provides Linux® command line syntax for setting and unsetting environment variables.

Linux

1. setenv KI_PRB_AUDIT_LOG $KILOG/prober.log
2. setenv KI_PRB_DEBUG PRINTTRANSCMNDS

Enable prober debug message while prober I/O is running

Use the ‘proberio_util’ script to enable/disable prober debug messages while the prober I/O is running.

script usage: `proberio_util -t N -q NNNN -dest fn [ -on -off ]`

where -t N--> test station number

where -q NNNN--> QMO number

where -dest fn--> destination of prober log file

where -on--> enable logging

where -off--> disable logging

NOTE *It is still possible to enable/disable debug messages as noted at the start of this chapter, but the prober I/O process(es) must first be stopped. Once this is accomplished, prober I/O may be started using the command line. The environment where prober I/O is started must have the correct environment variables established for prober debug messages to appear.*

Error messages and recovery procedures

After calling a prober command as a function, a negative number for a response indicates an error. Error codes are either issued by the prober (refer to the prober manufacturer’s documentation), or generated by the tester-prober interface (documented in this section).

Return values

The value of the number returned by the function will designate the type of response or error (shown in Table 4-1).

Table 4-1
Return values

Returned number between	Response description
0 and +14	Positive feedback (command completed successfully). The return values are defined in the file pointed to by KI_PRB_CONFIG. See Section 7 for sample prb.h file. See Table 4-2 for explanation.
-1001 and -1097	Tester-prober interface error (errors documented in this section). See Table 4-3 for explanation. The return values are defined in prb_msg.h. See Section 7 for a sample.
-1500 and -1999	Prober error. Add +1500 to the returned error number, and then refer to the prober manufacturer for error handling. For example, if a returned error number is -1573, adding +1500 to -1573 = -73. Use this number when referring to the prober manufacturer’s documentation.

Table 4-2
Positive return value description

Constant definitions	Explanation
PR_CASSETTECOMPLETE	End of lot (no more wafers).
PR_LOTEND	End of lot (no more wafers).
PR_MOVECOMPLETE	Prober moved to next die (confirmed).
PR_OK	Command executed properly.
PR_WAFERCOMPLETE	Next wafer loaded (confirmed).
PR_WAFER_REJECT_LOAD	After a failed wafer, the operator must load the next wafer.
PR_WAFER_REJECT_NO LOAD	After a failed wafer, the prober loads the next wafer automatically.

Table 4-3
Error return values

Error number	Error symbol	Description
-1001	BAD_TST_NUM	Bad test station number
-1002	BAD_CONFIG_DAT	Bad prober configuration data
-1003	FEAT_NOT_SUPPORT	Feature not supported
-1004	PORT_ASS_FAIL	Failure in assigning prober port
-1005	SET_UNITS_FAIL	Failure setting units
-1006	INVAL_MODE	Invalid mode
-1007	CLR_WAF_MAP	Error in clearing wafer mapping
-1008	SET_MODE_FAIL	Failure setting mode
-1009	SET_DIE_FAIL	Failure setting die size
-1010	SET_PRESET_FAIL	Failure setting preset
-1011	BAD_MODE	Operation invalid in mode
-1012	TST_COMPL_FAIL	Test complete failure
-1013	UNINTEL_RESP	Unintelligible response
-1014	MOVE_FAIL	Movement failure
-1015	UNEXPE_ERROR	Unexpected error number
-1016	LOAD_ERROR	Error on load
-1017	BAD_CHUCK	Bad chuck position
-1018	CHUCK_NOT_UP	Chuck not up for inking
-1019	INK_FAIL	Error in inking
-1020	TIME_GONE	Timeout value expired
-1021	BAD_LEARN_FUNCT	Bad learn list function
-1022	SET_DIAM_FAIL	Failure setting diameter
-1023	SET_MATRIX_FAIL	Failure setting matrix size
-1024	BAD_Z_PARAM	Bad Z parameter function
-1025	NO_RESPONSE	No response from the prober
-1026	TOO_MANY_NAKS	Too many NoAcKnowledge packets from prober
-1027	INVAL_PARAM	Invalid parameter
-1028	ERR_LOAD_FILE	Error loading file
-1029	ALIGN_FAIL	An alignment failure has occurred
-1030	GPIB_ERROR	GPIB error
-1031	CHK_PORT_ASS	Check assigned logical name
-1032	CHK_DEV_PROT	Check device protection on prober port
-1033	PR_MEM_ALLOC_ERR	Could not allocate dynamic memory
-1034	PR_NO_CASSETTE	No cassette
-1035	SET_FLAT_FAIL	Failure setting wafer flat
-1036	SET_REF_DIE_FAIL	Failure setting reference die
-1037	Z_PARAM_FAIL	Failure setting Z param
-1038	ERR_OPEN_PRBCNFG	Error opening prober configuration file
-1039	NO_CNFG_DATA	No data in prober configuration file

Table 4-3 (continued)
Error return values

Error number	Error symbol	Description
-1040	NO_PRB_TYPE	No prober type in prober configuration file
-1041	INVAL_PRB_MODE	Invalid prober mode in prober configuration file
-1042	INVAL_PRB_OPTIONS	Invalid prober options in prober configuration file
-1043	INVAL_DEVICE_NAME	Invalid device name in prober configuration file
-1044	INVAL_DEVICE_IRQ	Invalid device IRQ in prober configuration file
-1045	INVAL_BAUD_RATE	Invalid baud rate in prober configuration file
-1046	INVAL_TIMEOUT	Invalid timeout in prober configuration file
-1047	INVAL_GPIB_UNIT	Invalid GPIB unit in configuration file
-1048	INVAL_GPIB_SLOT	Invalid GPIB slot in configuration file
-1049	INVAL_GPIB_ADDRESS	Invalid GPIB addressed in configuration file
-1050	INVAL_GPIB_WRITE_MODE	Invalid GPIB write mode in configuration file
-1051	INVAL_GPIB_READ_MODE	Invalid GPIB read mode in configuration file
-1052	INVAL_GPIB_TERMINATOR	Invalid GPIB terminator in configuration file
-1053	INVAL_KULT_PATH	Invalid KULT path in configuration file (KI_KULT_PATH)
-1054	INVAL_PRB_LIB	Invalid prober library (EG40)
-1055	INVAL_PRB_CNFG_FUNC	Invalid prober configuration function (cannot find configuration function)
-1056	INVAL_PRB_MAX_SLOTS	Invalid maximum prober slots in configuration file
-1057	INVAL_PRB_MAX_CASSETTES	Invalid maximum prober cassettes in configuration file
-1058	SET_QUAD_FAIL	Failure setting quadrants
-1059	NO_UNPROBED_WAFERS	No unprobed wafers
-1060	SET_TEMPERATURE_FAIL	Failure setting chuck temperature
-1061	QUERY_TEMPERATURE_FAIL	Failure querying chuck temperature
-1062	SMIF_LOCK_FAIL	Failure setting SMIF lock
-1063	INVALID_ARRAY_SIZE	Invalid array size
-1064	NO_SRQ_FOUND	SRQ not found
-1065	UNKNOWN_SRQ_FOUND	Unknown SRQ found
-1066	INVALID_SRQ_FILE	Invalid SRQ file
-1067	INVALID_NEEDLE_CLEANING_STATION	Invalid parameter to PrNeedleClean
-1068	ERROR_CLEANING_NEEDLES	An error occurred during the attempt to clean the needles
-1069	ERROR_MOVING_ZFINE	An error occurred while performing a fine adjustment on the z axis
-1070	INVALID_Z_HT_RETURNED	An invalid z height was returned from the prober
-1071	INVALID_Z_SET_HT	New z height was not set
-1072	PR_MAX_Z_CT_EXCEEDED	User defined maximum z height exceeded
-1073	INVALID_USER_FUNCTION	AutoZ user function was not found
-1074	INVALID_RETURN_USER_FUNC	AutoZ user function returned an invalid number
-1075	INVALID_AUTOZ_LIB	User-defined AutoZ library is invalid
-1076	INVALID_AUTOZ_FUNC	User-defined AutoZ function is invalid
-1077	AUTOZ_NOT_ON_PROBER	AutoZ functionality is either not available or not enabled on the prober

Table 4-3 (continued)
Error return values

Error number	Error symbol	Description
-1078	NO_LICENSE	License is not available on the system
-1079	NO_NEEDLE_CLEANING_LICENSE	Needle cleaning license is not available on the system
-1080	KI_ABORT	User-defined function for abort signal
-1081	INVAL_PRB_P8_OPTION	Invalid P8 option
-1082	NI_UNIX_ERROR	NI board Unix error
-1083	NI_BOARD_NOT_CIC	NI board not controller in charge
-1084	NI_NO_LISTENER	NI board no listeners
-1085	NI_BOARD_NOT_ADDRESSES	NI board not addressed
-1086	NI_INVALID_FUNCTION_ARG	NI board invalid function argument
-1087	NI_BOARD_NOT_SYS_CONTROLLER	NI board not system controller
-1088	NI_IO_ABORTED_TIMO	NI board timeout
-1089	NI_BOARD_NOT_PRESENT	NI board not present
-1090	NI_DMA_HW_PROBLEM	NI board DMA hardware problem
-1091	NI_DMA_HW_TIMO	NI board DMA hardware timeout
-1092	NI_NO_CAPABILITY	NI board no capability
-1093	NI_FILE_SYS_ERROR	NI board file system error
-1094	NI_GPIB_BUS_ERROR	NI board bus error
-1095	NI_SRQ_BYTE_Q_OVERFLOW	NI board SRQ overflow
-1096	NI_SRQ_STUCK_ON	NI board SRQ stuck on
-1097	NI_TABLE_PROBLEM	NI board table problem

Examples

1. After a PrMove, a value of PR_MOVECOMPLETE is returned. From Table 4-2, this return value indicates that the “Prober moved to next die (confirmed).”
2. After a prober command is issued, a negative number of -1020 is returned. From Table 4-3, a -1020 return value indicates that the “Timeout value expired.”

Handling alignment errors

If the prober cannot auto-align a particular wafer, the prober generates an alignment error. Keithley prober control software can detect this alignment error. For example, the following code shows one way of handling an alignment error:

```
int CommandStatus;
int count;
       •
       •
       •
CommandStatus = PrAutoAlign();
count = 1;
while ((CommandStatus == ALIGN_FAIL) && (count <= 3))
{
       /* align fail, retry up to 3 times */
       CommandStatus = PrAutoAlign();
       count++;
}

if (CommandStatus  == ALIGN_FAIL)
{
       /* ask for operator intervention */
       •
       •
       •
}
```

In the sample code, when an alignment error is detected (CommandStatus == `ALIGN_FAIL`), the program instructs the prober to retry the alignment (up to 3 additional times). If the wafer fails to align, a code to request operator intervention could be added (in the example, this code would be added after the `/* ask for operator intervention */`). KTXE, the Keithley Execution Engine, handles alignment and pre-alignment errors in precisely this manner.

Error detection when writing code

The prober control software returns a negative number when an error is detected. While the Keithley Execution Engine (KTXE) handles these types of errors, make sure to trap for errors in any user written code (refer to the Execution engine paragraph of Section 3 for information on the engine). To write code with this ability, always call prober library commands as C functions. Also, always check the return value for any negative numbers. For example:

```
int CommandStatus;
        •
        •
        •
CommandStatus = PrChuck(PR_CHUCK_UP); /* raise the chuck */
if (CommandStatus < 0)
{
        /* Error */
        •
        •
        •
}
```

Using a UAP for error recovery

NOTE *The UAP for error recovery is part of KTE 4.0 and greater.*

An included UAP (User Access Point: UAP_PRB_ERR_HDLR) can be used to execute specific error recovery code. An example module (KTXEPrbErrHdlr) is provided in KTXEADDIN user library.

Use the provided module as a template enabling creation of prober specific error recovery code.

This module contains a ‘switch’ statement with all available prober functions listed. Even though all prober functions are listed, the Keithley Test eXecution Engine does not necessarily call each function. To use this UAP, create a prober function specific error recovery module using KULT. This function call should be added to the corresponding position in the switch statement.

Example

To perform a special prober error recovery sequence in the event of a PrLoad error, create a function named ‘CustLoadRecover’ in the KULT library. The function, CustLoadRecover, then needs to be placed in the PRLOAD case.

```
Case PRLOAD:
     CustLoadRecover();
break;
```

NOTES *The user IS RESPONSIBLE for correctly adding the ‘break;’ statement(s). The ‘break;’ statements have NOT been included in the example code. This allows the default case to be used in the event of an error if the user did not supply their own recovery function in the correct case.*

*The **KTXEPrbErrHdlr** must be re-compiled and re-built after it has been edited.*

*The **KTXEPrbErrHdlr** function is a template, the user SHOULD save the **KTXEPrbErrHdlr** file to a unique name and then edit that newly created file. This will keep specific user's prober error recovery function from being over-written when **KTE** is upgraded.*

*Add **KTXEPrbErrHdlr** at the following UAP (User Access Point) **UAP_PRB_ERR_HDLR**. This UAP is located in the prober error trapping function within the engine. The function will not work correctly if **KTXEPrbErrHdlr** is not added at the **UAP_PRB_ERR_HDLR** UAP.*

5
Reference

Introduction to command summary and syntax

This section details Keithley Prober Driver commands for supported probers. Not all commands are available for each supported prober. Refer to the appropriate prober specific appendix for a list of supported commands.

Command format

Praaaa

Purpose

This paragraph describes the format of all prober commands in the following command summary. Each command supports the C language. If the command has a character string parameter, the format of the form is Praaaa (C language).

Syntax

```
status = Praaaa(parameters 0-n)
```

Parameters

Refer to Table 5-1 for data type.

NOTE *The returned status of all commands is always an integer.*

Table 5-1
Parameter data type

Language	Integer	Real	Character string
C	int	double	char*

Prober command return status

When commands are called as functions, their values should be placed in integer variables. The values returned can be error codes if negative, or status codes. Refer to Section 4 for Error codes. Refer to the command definitions in this section for status codes. Also refer to the sample prb.h file contained in Section 7.

NOTE *Call commands as functions to return status codes (recommended).*

All automatic mode commands and most external prober commands return a function value of PR_OK when they have successfully completed. The following list contains the exceptions:

In external mode, exceptions include:

- PrMove: When a successful move to a new site has been completed, a functional value of PR_MOVECOMPLETE is returned from the PrMove.
- PrLoad:
  1. When a PrLoad command is executed and a new wafer is successfully loaded, a functional value of PR_WAFERCOMPLETE is returned from PrLoad.
  2. PrLoad: When a PrLoad command is executed and no more wafers are available for loading, a functional value of PR_LOTEND is returned from PrLoad.

The only other command that does not return a PR_OK when properly executed would be the PrError command. This command returns the prober error condition number.

Quick reference table

Table 5-2 contains all available commands for supported probers. Refer to the Prober Commands paragraph for a detailed description of each command.

NOTE *Not all commands are supported by all probers. Refer to the appropriate prober specific appendix for a list of supported commands.*

Keithley's execution engine and KTXE are used interchangeably in this text.

Table 5-2
Quick reference table

Command	Description	Syntax	Page
PrAbsMove	Tells prober to move to an absolute number of millimeters or mils relative to the wafer origin. Available command, but not used by Keithley's execution engine.	status = PrAbsMove(double XDistance, double YDistance);	page 5-7
PrAdjustZHeight	Tells the prober to move the Z-stage (chuck surface) a relative number of units (as defined on the prober) up/ contact (+) or down/separate (-).	status = PrAdjustZHeight(double z_height);	page 5-7
PrAutoAlign	Causes an auto-align to be performed.	status = PrAutoAlign();	page 5-8
PrCassetteMap	Returns a map of the specified cassette to an integer array. Used by Keithley's execution engine only if PrCheckOptions command returns that the required options are present on the prober.	status = PrCassetteMap(int CassetteNumber, int *Map[The array size is from 1 to a defined constant. The constant is defined in the file pointed to by KI_PRB_CONFIG]);	page 5-8
PrCassetteMask	Causes the indexer to be lowered and the desired cassette to be masked. The mask is based on calls to the PrSetSlotStatus function.	status = PrCassetteMask(int cassette)	page 5-9
PrCheckOptions	Returns optional features present on the prober. Used by Keithley's execution engine.	status = PrCheckOptions(int *OcrPresent, int *AutoAlnPresent, int *ProfilerPresent, int *HotchuckPresent, int *HandlerPresent, int *Probe2PadPresent);	page 5-9
PrChuck	Raises or lowers the chuck. Used by Keithley's execution engine.	status = PrChuck (int chuck_position);	page 5-9
PrClearAll	Causes all the wafers to be returned to the cassette and the termination of the lot.	status = PrClearAll()	page 5-10
PrClearPipeline	Clears the pipeline.	status = PrClearPipeline();	page 5-10
PrError	Returns the error condition from the prober and attempts to clear the error. Used by Keithley's execution engine.	status = PrError();	page 5-10
PrGetNxtWafer	Loads the next "unprobed" wafer from the specified cassette.	status = PrGetNxtWafer(int cassette_number);	page 5-10
PrGetProduct	Returns the currently loaded product file.	status = PrGetProduct(char *file_name);	page 5-11

Table 5-2 (continued)
Quick reference table

Command	Description	Syntax	Page
PrGetWafer	Loads a wafer from the specified cassette and slot to the chuck.	status = PrGetWafer(int cassette_number, int slot_number);	page 5-11
PrInit	Initializes the prober. Used by Keithley's execution engine.	status = PrInit (int mode, double x_die_size, double y_die_size, int x_start_coordinate, int y_start_coordinate, int units, int sub_type);	page 5-11
PrLoad	Unloads presently loaded wafer and loads the next wafer on the chuck. Used by Keithley's execution engine.	status = PrLoad();	page 5-12
PrLoadProduct	Instructs the prober to load a product file from its internal disk drive. Used by Keithley's execution engine only if PrCheckOptions command returns that the required options are present on the prober.	status = PrLoadProduct(char *ProductFilename, char *DriveName);	page 5-12
PrLowerBoat	Causes the indexer to be lowered and the desired cassette to be mapped.	status = PrLowerBoat(int i_pod_number)	page 5-12
PrMove	Moves the chuck in die steps (X-Y). Use PrRelMov to move the chuck in machine steps. Used by Keithley's execution engine.	status = PrMove(int x_location, int y_location, int ink_number);	page 5-13
PrNeedleClean	Directs the prober to clean the needles on the probe card by using the internal prober cleaning method(s).	status = PrNeedleClean(int clean_function);	page 5-13
PrProfile	Forces wafer profiling.	status = PrProfile();	page 5-13
PrPutNxtSlot	Unloads the wafer presently on the chuck to the next empty slot in the specified cassette.	status = PrPutNxtSlot(int cassette_number, int reason_code);	page 5-14
PrPutWafer	Unloads the wafer presently on the chuck to the specified cassette and slot.	status = PrPutWafer(int cassette_number, int slot_number, int reason_code);	page 5-14
PrQueryChuckTemp	Queries chuck temperature.	status = PrQueryChuckTemp (double *chuck_temp);	page 5-15
PrQueryZHeight	Returns the current Z-stage (chuck) height in units defined on the prober.	status = PrQueryZHeight(double *z_height);	page 5-15
PrReadId	Reads wafer ID information from the prober. Used by Keithley's execution engine only if PrCheckOptions command returns that the required options are present on the prober.	status = PrReadId(char *user_buffer);	page 5-16
PrRelMove	Causes a move in a first quadrant coordinate system (in relation to the probe pins) of the indicated x and y distances. Used by Keithley's execution engine.	status = PrRelMove(double x_value, double y_value);	page 5-16
PrRelReturn	Terminates a sequence of PrRelMove commands. Used by Keithley's execution engine.	status = PrRelReturn();	page 5-16

Table 5-2 (continued)
Quick reference table

Command	Description	Syntax	Page
PrStart	Causes the prober to re-start probing after a PrStop.	status = PrStart();	page 5-16
PrStop	Causes the prober to stop probing.	status = PrStop();	page 5-17
PrSerialPoll	Performs a serial poll of the prober GPIB interface. Used by Keithley's execution engine.	status = PrSerialPoll();	page 5-17
PrSetChuckTemp	Sets chuck temperature.	status = PrSetChuckTemp(double chuck_temp);	page 5-17
PrSetDiam	Makes the wafer diameter used in the test known to the prober.	status = PrSetDiam(int diameter);	page 5-17
PrSetDieSize	Sets the (x,y) die size of a wafer.	status = PrSetDieSize(double XDieSize, double YDieSize);	page 5-17
PrSetFlat	Tells the prober in which direction to place the flat of the wafer.	status = PrSetFlat (int flat_number);	page 5-18
PrSetMode	Allows for changes to the probing mode after the PrInit command is sent.	status = PrSetMode (int mode);	page 5-19
PrSetPipeline	Enables/disables pipelining on the prober.	status = PrSetPipeline (int pipe_line);	page 5-19
PrSetQuadrant	Sets the directions in which x and y coordinates will increase.	status = PrSetQuadrant (int quadrant_number);	page 5-19
PrSetRefDie	Assigns an (x,y) coordinate to the prober's target or reference die.	status = PrSetRefDie(int XDieCoordinate, int YDieCoordinate);	page 5-20
PrSetSlotStatus	Modifies the prober's internal cassette map.	status = PrSetSlotStatus(int Cassette, int Slot, int StatusCode);	page 5-20
PrSetTime	Changes the system default timeout value of 120 seconds.	status = PrSetTime (int time_value);	page 5-21
PrSetUnits	Sets the prober to metric (PR_METRIC) or English (PR_ENGLISH) units.	status = PrSetUnits(int Units);	page 5-21
PrSetZHeight	Tells the prober to set the current Z-stage (chuck) height as the new contact position.	status = PrSetZHeight();	page 5-21
PrSmifClamp	Causes the SMIF (FOUP) pod to be engaged (clamped) or disengaged (unclamped). If engaged (clamped) the cassette may NOT be removed from the indexer, conversely, if disengaged (unclamped) the cassette may be removed from the indexer.	status = PrSmifClamp(int i_pod_number, int i_clamp_state)	page 5-22
PrSmifLock	Causes the SMIF (FOUP) pod to be engaged (locked) or disengaged (unlocked). If engaged (locked) the cassette may NOT be lowered, conversely, if disengaged (unlocked) the cassette may be lowered.	status = PrSmifLock(int i_pod_number, int i_lock_state)	page 5-22
PrSmifStatus	Returns the status of various items on the prober, they are: operator mode, latch/lock status, cassette home status, pod present status, and clamp status.	status = int PrSmifStatus(int I_pod_number, int *I_status_array, int i_status_array_size)	page 5-22

Table 5-2 (continued)
Quick reference table

Command	Description	Syntax	Page
PrStatus	Returns the present x,y location, chuck position, and prober ready status. Available command, but not used by Keithley's execution engine.	status = PrStatus (int *ready_for_probing, int *x_location, int *y_location, int *chuck_position, int *prober_mode);	page 5-23
PrUnLoad	Forces wafer presently on chuck to unload to its original cassette and slot.	status = PrUnLoad();	page 5-24
PrWriteRead	Sends a string to the prober and reads it's response (prober communication I/O primitive). Available command, but not used by Keithley's execution engine.	status = PrWriteRead (char *input_buffer, int input_buffer_length, char *output_buffer, int output_buffer_maximum_length, int terminator, int number_of_terminators);	page 5-24
PrWriteReadSRQ	Sends a selected string to a GPIB prober.	status = PrWriteReadSRQ (char *input_buffer, int input_buffer_length, char *output buffer, int output_buffer_maximum_length, int timeout, int *i_srq);	page 5-25
PrZParams	Sets certain height parameters from the tester.	status = PrZParams (int function, int value);	page 5-26
PrZTravel	Sets the prober z travel mode from the tester.	status = PrZTravel (int number);	page 5-27

Prober commands

This section contains a detailed description of each command for all available commands.

NOTE *Not all commands are supported by all probers. Refer to the appropriate prober specific appendix for a list of supported commands. Appendix A and Appendix B contain Electroglas specifics, Appendix contains TEL P8 specifics, and Appendix D contains TSK A-PM-90A specifics.*

PrAbsMove

Purpose

The PrAbsMove command tells the prober to move to an absolute number of millimeters or mils relative to the wafer origin. (PrRelMove by contrast tells the prober to move a relative number of millimeters or mils relative to the present position.)

Syntax

```
status = PrAbsMove(double x...value. double Y...value);
```

Parameters

x_value—This number indicates the number of mm or mils to move along the x-axis with respect to the wafer origin.

y_value—This number indicates the number of mm or mils to move along the y-axis with respect to the wafer origin.

PrAdjustZHeight

Purpose

The PrAdjustZHeight command tells the prober to move the Z-stage (chuck surface) a relative number of units (as defined on the prober) up/contact (+) or down/separate (-).

Syntax

```
status = PrAdjustZHeight(double z_height);
```

Parameters

z_height—The number of units to move the chuck from its current position. A valid Z height is defined for each prober. The supported prober drivers are EG40, TSK9, P8.

Constants associated with PrAdjustZHeight
(See specific prober appendix for supported probers):
EG40
EG40_MIN_Z_ADJUST
EG40_MAX_Z_ADJUST
PR_ZFINE_RESOLUTION_EG40

P8
P8_MIN_Z_ADJUST
P8_MAX_Z_ADJUST
PR_ZFINE_RESOLUTION_P8

TSK9
TSK9_MIN_Z_ADJUST
TSK9_MAX_Z_ADJUST
PR_ZFINE_RESOLUTION_TSK9

Refer to sample prb.h file contained in Section 7.

PrAutoAlign

Purpose

Before using this command, an auto alignment must be stored on the prober. This optical pattern is determined during the wafer setup on the prober.

Syntax

```
status = PrAutoAlign();
```

Parameters

None

Remarks

Keithley's execution engine will only run this command if the PrCheckOptions command's return value indicates that the prober possesses all the features (or options) required for
PrAutoAlign to execute.

PrCassetteMap

Purpose

The PrCassetteMap command returns a map of the slot status for the specified cassette to an integer array.

Syntax

```
status = PrCassetteMap (int cassette_number, int *cassette_map);
```

Parameters

cassette_number—This parameter identifies the cassette (in a single or multi-cassette prober) for which a map will be returned. The value of cassette_number ranges from 1 to PROBER_n_MAX_CASSETTE. PROBER_n_MAX_CASSETTE is a constant found in the file, KIDAT/prbcnfg_XXXX.dat, which represents the total number of cassettes contained in prober model XXXX. This file may also be referenced by its environmental variable, KI_PRB_CONFIG. The n in PROBER_n_MAX_CASSETTE is substituted by the test station number in the actual file.

cassette_map—This array is a map of the slot statuses of the cassette. The size of the array, which corresponds to the number of slots in the cassette, is given by the constant, PROBER_n_MAX_SLOT; this constant can be found in the KIDAT/prbcnfg_XXXX.dat file. The array is indexed from 0 to (PROBER_n_MAX_SLOT - 1). Each element of the array represents a single slot and is assigned one of the statuses listed below. Each status has a corresponding constant integral value; these values may be found in the prb.h file.

PR_SLOT_STATUS_UNMAPPED
PR_SLOT_STATUS_IN PROCESS
PR_SLOT_STATUS_EMPTY
PR_SLOT_STATUS_UNPROBED
PR_SLOT_STATUS_PROBED
PR_SLOT_STATUS_PROBLEM
PR_SLOT_STATUS_UNSCHEDULED
DEFAULT_SLOT_STATUS

Remarks

Keithley's execution engine will run this command only if the PrCheckOptions command's return value indicates that the prober possesses all the features (or options) required for
PrCassetteMap to execute.

PrCassetteMask

Purpose The PrCassetteMask will allow the user to select which slots in the specified cassette will be available for probing. The PrCassetteMask function must be used in conjunction with the PrSetSlotStatue command.

Syntax
```
status = PrCassetteMask (int cassette);
```

Parameters cassette—a variable that contains the desired cassette number.

Remarks Currently this command is only valid on P8XL model probers. For example, a series of calls to PrSetSlotStatus will be followed by a call to PrCassetteMask. This will configure the prober as to which wafers will be probed.

PrCheckOptions

Purpose This command returns which optional features are present on the prober.

Syntax
```
status = PrCheckOptions( int *OcrPresent, int *AutoAlnPresent, int *Pro-
filerPresent, int *HotchuckPresent, int *HandlerPresent, int
*Probe2PadPresent );
```

Parameters OcrPresent—the status of the OCR subsystem is returned to this variable. 1 indicates OCR is present and 0 indicates no OCR. (INTEGER)

AutoAlnPresent—the Auto Align subsystem status is returned to this variable. 1 indicates autoalign is enabled and present, while 0 indicates that the autoalign system is disabled. (INTEGER)

ProfilerPresent—the Profiler subsystem status is returned to this variable. 1 indicates profiler is enabled and present, while 0 indicates that the profiler is disabled. (INTEGER)

HotchuckPresent—the prober's hot chuck status is returned to this variable. 1 indicates the presence of a hot chuck, while 0 indicates that the option is either not installed or is disabled. (INTEGER)

HandlerPresent—the prober's random access handler status is returned to this variable. 1 indicates the presence of the random access system, while 0 indicates the system is disabled. (INTEGER)

Probe2PadPresent—the prober's automatic probe-to-pad system status is returned to this variable. 1 indicates that the probe-to-pad system is enabled, while 0 indicates that it is disabled. (INTEGER)

Remarks This command is used by Keithley’s execution engine.

PrChuck

Purpose This command raises or lowers the chuck.

Syntax
```
status = PrChuck(int chuck_position);
```

Parameters The parameter chuck_position is an integer. (INTEGER: PR_CHUCK_DOWN or PR_CHUCK_UP)

Lower chuck: PR_CHUCK_DOWN

Raise chuck: PR_CHUCK_UP

Remarks To lower chuck, let chuck_position = PR_CHUCK_DOWN. To raise chuck, let chuck_position = PR_CHUCK_UP. This command is used by Keithley's execution engine. Refer to sample prb.h file contained in Section 7.

PrClearAll

Purpose — The PrClearAll command causes all wafers to be returned to the cassette and the termination of the lot.

Syntax
```
status = PrClearAll();
```

Parameters — None

Remarks — This command is valid only with a prober equipped with SMIF technology.

PrClearPipeline

Purpose — This command unloads any wafers on the Quick-Loader and Prealigner to their respective slots in their cassettes of origin (clears the pipeline).

Syntax
```
status = PrClearPipeline();
```

Parameters — None

PrError

Purpose — Use this command to obtain information regarding a prober error. It returns an encoded prober error number which must be decoded before it can be compared with the numbers listed in the prober manufacturer's error documentation. The decoding equation is given in the Remarks section below. Look up the encoded error number in the documentation supplied by the prober manufacturer.

Syntax
```
status = PrError();
```

Parameters — None

Remarks — This command is used by Keithley's execution engine. The number returned may be an error number or a return status code. The decoding equation is as follows:
prober manufacturer's error number = -(return value+1500)

PrGetNxtWafer

Purpose — This command loads the next "unprobed" wafer from the specified cassette to the chuck. If a wafer is presently on the chuck, it will be unloaded to its origin slot and cassette. When called as a function, this command returns the slot number of the chosen wafer. If there are no unprobed wafers in the specified cassette, the command generates an error message and number.

Syntax
```
status = PrGetNxtWafer(int cassette_number);
```

Parameters — cassette_number specifies a particular source cassette (INTEGER). The cassette number's value can be from 1 to a defined constant. The constant is defined in the file pointed to by KI_PRB_CONFIG (`PROBER_`*n*`_MAX_CASSETTE` where *n* = test station number). See sample in KIDAT/prbcnfg_*XXXX*.dat.

PrGetProduct

Purpose — The PrGetProduct command instructs the prober to return the currently loaded product file.

Syntax

```
status = PrGetProduct(char *file_name);
```

Parameters — file_name—variable containing the product file name currently loaded on the prober (pointer to CHAR array).

Remarks — This command is NOT used by Keithley's execution engine.

PrGetWafer

Purpose — This command loads a wafer from the specified cassette and slot to the chuck. If a wafer is presently on the chuck, it will be unloaded to its original slot and cassette. The parameters are
integers.

Syntax

```
status = PrGetWafer(int cassette_number, int slot_number);
```

Parameters — cassette_number specifies a particular source cassette (INTEGER). The cassette number's value can be from one to a defined constant. The constant is defined in the file pointed to by KI_PRB_CONFIG (PROBER_n_MAX_CASSETTE where n = test station number). See sample in KIDAT/prbcnfg_*XXXX*.dat.

slot_number specifies a particular wafer from the source cassette.

PrInit

Purpose — This command initializes the prober with the following information: probing mode, die size, first coordinates, and units.

Syntax

```
status = PrInit(int mode, double x_die_size, double y_die_size, int
x_start_coordinate, int y_start_coordinate, int units, int sub_type);
```

Parameters — mode—is the desired probing mode (if the prober supports the mode). The following mode options are defined in the prb.h file:

PR_MODE_MANUAL
PR_MODE_EXTERNAL
PR_MODE_EDGE

x_die_size, y_die_size—These parameters, representing double floating-point values, give the dimensions of the die. The values entered for these parameters will depend on the units chosen to represent the data. The units selected (millimeters or mils) are entered through the "units"
parameter.

x_start_coordinate, y_start_coordinate—These parameters are used to assign x and y coordinates to the location of the prober at alignment: this assignment will define the origin of the coordinate system for the wafer, providing a point of reference from which other locations may be identified.

units—This parameter gives the unit (Metric or English) in which the numerical data will be given. The following argument options are defined in the prb.h file:

PR_ENGLISH
PR_METRIC

sub_type—This parameter is present for legacy reasons. It has no useful function and is always set to 0. (In the past it was used to represent the secondary type of the prober).

Remarks — This command is used by Keithley's execution engine.

PrLoad

Purpose — This command forces the wafer currently on the chuck to be unloaded and the next wafer that is designated to be tested, to be loaded, profiled, and aligned.

Syntax

```
status = PrLoad();
```

Parameters — None

Remarks — This command is used by Keithley's execution engine.

PrLoadProduct

Purpose — The PrLoadProduct command instructs the prober to load a product file from the specified drive.

Syntax

```
status = PrLoadProduct (char *file_name, char *drive_name);
```

Parameters — file_name—an array that contains the name of the product file to be loaded.

drive_name—points to the location in memory that contains the identifying letter of the drive from which the product file is to be loaded. A “NULL” indicates that the default drive (usually a hard drive) is to be used.

NOTE *TEL P8 probers do not use the DriveName parameter—the probers exter-nal interface does not allow the user to select a drive.*

Remarks — Keithley’s execution engine will only run this command if the PrCheckOptions command’s return value indicates that the prober possesses all the features (or options) required for a PrLoadProduct to execute.

PrLowerBoat

Purpose — The PrLowerBoat command causes the indexer to be lowered and the desired cassette to be mapped.

Syntax

```
status = PrLowerBoat(int i_pod_number);
```

Parameters — i_pod_number—variable that contains the desired pod/cassette number.

Remarks — This command is valid only with a prober equipped with SMIF technology.

PrMove

Purpose — When the prober is running in external mode, PrMove commands the prober to move to the specified x and y location.

Syntax
```
status = PrMove(int x_location, int y_location, int ink_number);
```

Parameters — x_location, y_location—These parameters give the x and y locations of the die to which the prober should move after inking the current die.

Remarks — This command is used by Keithley's execution engine.

PrNeedleClean

Purpose — The PrNeedleClean command will direct the prober to clean the needles on the probe card by using the internal prober cleaning method(s).

Syntax
```
status = PrNeedleClean(int clean_function);
```

Parameters — clean_function is an integer describing the cleaning method to be used. (INTEGER 0-7)

PR_CLEAN_NONE—do not perform needle cleaning
PR_CLEAN_BRUSH—use prober brush
PR_CLEAN_FIBER—use prober fiber pad
PR_CLEAN_CERAMIC—use ceramic pad
PR_CLEAN_METAL—use metal pad
PR_CLEAN_STICKY—use sticky pad
PR_CLEAN_BLOW—use air to clean
PR_CLEAN_GENERIC—other unspecified cleaning station

Remarks — Each prober will support different needle cleaning types. Therefore, the availability of each cleaning function will be determined by the specific prober. Dependencies: PRB-COM

PrProfile

Purpose — The PrProfile command forces a wafer profiling.

Syntax
```
status = PrProfile();
```

Parameters — None

Remarks — This command is used by Keithley's execution engine, but only if the PrCheckOptions command returns that the required optional features are present on the prober.

PrPutNxtSlot

Purpose

This command unloads the wafer currently on the chuck, deposits it into the next empty slot of the specified cassette, and loads the next wafer that has been designated to be tested onto the chuck. If no slots are available in the specified cassette, the command returns an error message and number (when called as a function). If successful, it returns the number of the destination slot. The parameters are integers.

Syntax

```
status = PrPutNxtSlot(int cassette_number, int reason_code);
```

Parameters

cassette_number = specifies a destination cassette (INTEGER). The cassette number's value can be from one to a defined constant. The constant is defined in the file pointed to by KI_PRB_CONFIG (`PROBER_n_MAX_CASSETTE` where *n* = test station number). See sample in KIDAT/prbcnfg_*XXXX*.dat.

reason_code = specifies reason for unloading as shown in the following table.

Reason for unloading

Constants
PR_NORMAL_UNLOAD
PR_PROFILE_FAIL
PR_ALIGN_FAIL

Refer to sample prb.h file contained in Section 7.

Remarks

This command unloads the wafer presently on the chuck specified cassette and loads the next wafer if one exists. When called as a function, the return value for a successful execution of this command will correspond to one of two possible constants that are defined in the prb.h file:

Constants	Meaning
PR_WAFERCOMPLETE	new wafers successfully loaded
PR_CASSETTECOMPLETE	end of lot, no more wafers

Refer to sample prb.h file contained in Section 7.

If unsuccessful, the return value will correspond to a specific error number:<0 = various errors

PrPutWafer

Purpose

The PrPutWafer command unloads the wafer presently on the chuck to the specified cassette and slot. If the specified slot is unavailable, this command returns an error message and number. The parameters are integers.

Syntax

```
status = PrPutWafer(int cassette_number, int slot_number, int
reason_code);
```

Parameters

cassette_number—specifies a destination cassette (INTEGER). The cassette number's value can be from one to a defined constant. The constant is defined in the file pointed to by KI_PRB_CONFIG (`PROBER_n_MAX_CASSETTE` where *n* = test station number). See sample in KIDAT/prbcnfg_*XXXX*.dat.

slot_number—specifies a slot in the destination cassette (INTEGER). The cassette number's value can be from one to a defined constant. The constant is defined in the file

pointed to by KI_PRB_CONFIG (PROBER_n_MAX_SLOT where n = test station number). See sample in KIDAT/prbcnfg_*XXXX*.dat.

reason_code—specifies reason for unloading. The three valid reasons for unloading the wafer are given below; each has a corresponding numerical code which is defined in the prb.h file:

Constants
PR_NORMAL_UNLOAD
PR_PROFILE_FAIL
PR_ALIGN_FAIL
Refer to sample prb.h file contained in Section 7.

Remarks — This command is used by Keithley's execution engine, but only if the PrCheckOptions command returns that the required optional features are present on the prober.

PrQueryChuckTemp

Purpose — Queries chuck temperature

Syntax
```
status = PrQueryChuckTemp (double *chuck_temp);
```

Parameters — chuck_temp—This parameter specifies a memory location into which a value for the chuck temperature is to be inserted.

PrQueryZHeight

Purpose — The PrQueryZHeight returns the current Z-stage (chuck) height in units defined on the prober.

Syntax
```
status = PrQueryZHeight(double *z_height);
```

Parameters — z_height—The current height of the Z-stage (chuck).

Constants associated with PrAdjustZHeight
(See specific prober appendix for supported probers):
EG40
EG40_MIN_Z_ADJUST
EG40_MAX_Z_ADJUST
PR_ZFINE_RESOLUTION_EG40

P8
P8_MIN_Z_ADJUST
P8_MAX_Z_ADJUST
PR_ZFINE_RESOLUTION_P8

TSK9
TSK9_MIN_Z_ADJUST
TSK9_MAX_Z_ADJUST
PR_ZFINE_RESOLUTION_TSK9

Refer to sample prb.h file contained in Section 7.

PrReadId

Purpose The PrReadId command reads wafer ID information from the prober and places the information in a 80 character user_buffer.

Syntax
```
status = PrReadId(char *user_buf);
```

Parameters The only parameter used with this command is the name of a 80-character buffer, which contains the string returned from the prober by the prober request wafer ID command.

If ID is not entered in the prober, a null string is returned. Otherwise the length of the null terminated ID and its pointer is returned.

Remarks This command is used by Keithley's execution engine, but only if the PrCheckOptions command returns that the required optional features are present on the prober.

PrRelMove

Purpose This command moves the prober a number of units (mm or mils) in the x and y directions
(as indicated by the parameter values) relative to the present location of the probe pins. This command is used for intrasite prober moves

Syntax
```
status = PrRelMove(double x_value, double y_value);
```

Parameters x_value, y_value—these parameters specify the number of mm (or mils) the prober must move in the x and y directions.

Remarks This command is used by Keithley's execution engine but it will not be run if the prober's microprobing utilities are being used to perform the subsite probing.

PrRelReturn

Purpose This command will return the prober to the location it occupied before any PrRelMove commands (i.e. subsite moves) were executed. After any number of PrRelMove commands, PrRelReturn must be executed before the prober can be moved to another site.

Syntax
```
status = PrRelReturn();
```

Parameters None

Remarks This command is used by Keithley's execution engine.

PrStart

Purpose Starts the prober.

Syntax
```
status = PrStart();
```

Parameters None

Remarks None

PrStop

Purpose	Stops the prober.
Syntax	`status = PrStop();`
Parameters	None
Remarks	None

PrSerialPoll

Purpose	Performs a serial poll of the prober GPIB interface.
Syntax	`status = PrSerialPoll();`
Parameters	None
Remarks	This command is available but not used by Keithley's execution engine.

PrSetChuckTemp

Purpose	Sets chuck temperature.
Syntax	`status = PrSetChuckTemp(double chuck_temp);`
Parameters	chuck_temp—sets the chuck temperature expressed in °C. (DOUBLE).
Remarks	Also see "PrQueryChuckTemp," page 5-15.

PrSetDiam

Purpose	The PrSetDiam command provides the wafer diameter to the prober.
Syntax	`status = PrSetDiam(int diameter);`
Parameters	The parameter used in this command is the wafer diameter (expressed as an integer number of millimeters or inches). (INTEGER)

PrSetDieSize

Purpose	The PrSetDieSize command sets the (x,y) die size of a wafer.
Syntax	`status = PrSetDieSize(double x_die_size, double y_die_size);`
Parameters	x_die_size—a parameter that contains the size of the die, with respect to the x-coordinate, expressed in units of mm or mils. y_die_size—a parameter that contains the size of the die, with respect to the y-coordinate, expressed in units of mm or mils.

PrSetFlat

Purpose

The PrSetFlat command tells the prober in which direction to place the flat of the wafer.

Syntax

```
status = PrSetFlat(int flat_number);
```

Parameters

The only parameter used in this function is the flat_number, which is the integer number direction of the flat PrSetMode

Electroglas Specific

`EG_FLAT_POS_ZERO` :front of prober
`EG_FLAT_POS_NINETY` :left of prober
`EG_FLAT_POS_ONE_EIGHTY` :back of prober
`EG_FLAT_POS_TWO_SEVENTY` :right of prober

These integral values are defined in: KI_KULT_PATH PrEG40.h and PrEG2X.h

TSK Specific

`TSK_FLAT_POS_ZERO` :back of prober
`TSK_FLAT_POS_NINETY` :right of prober
`TSK_FLAT_POS_ONE_EIGHTY` :front of prober
`TSK_FLAT_POS_TWO_SEVENTY` :left of prober

These integral values are defined in KI_KULT_PATH PrTSK9.h

TEL P8 Specific

`P8_FLAT_POS_ZERO` :back of prober
`P8_FLAT_POS_NINETY` :right of prober
`P8_FLAT_POS_ONE_EIGHTY` :front of prober
`P8_FLAT_POS_TWO_SEVENTY` :left of prober

These integral values are defined in KI_KULT_PATH PrP8.h

T19S Specific

`T19S_FLAT_POS_ZERO` :front of prober
`T19S_FLAT_POS_NINETY` :left of prober
`T19S_FLAT_POS_ONE_EIGHTY` :back of prober
`T19S_FLAT_POS_TWO_SEVENTY` :right of prober

These integral values are defined in KI_KULT_PATH PrT19S.h

PrSetMode

Purpose

The PrSetMode command allows you to change the probing mode from the one set with the PrInit command.

Syntax

```
status = PrSetMode(int mode);
```

Parameters

mode—the desired probing mode (if the prober supports the mode). (INTEGER, 1-3)

Constants

PR_MODE_MANUAL
PR_MODE_EXTERNAL
PR_MODE_EDGE
Refer to sample prb.h file contained in Section 7.

PrSetPipeline

Purpose

This command enables/disables pipelining on the prober. Pipelining is a technique used to speed up the probing process. While enabled, the prober prepares the next wafer to be loaded on the chuck. For example: The wafer is taken from the cassette and placed on the pre-aligner (the pre-aligner finds the flat/notch and orients the wafer correctly). Once pre-aligned, the prober places the wafer on the quick loader (the wafer on the quick loader will be the next wafer probed after the wafer presently on the chuck is unloaded). This is how the second and subsequent wafers are ‘pipelined’. Note that the first wafer will go directly to the chuck.

Syntax

```
status = PrSetPipeline(int on_off);
```

Parameters

on_off—this parameter is used to relay enable/disable information to the function. To enable pipelining, on_off should be set to PR_ENABLE_PIPELINE; to disable, on_off should be set to PR_DISABLE_PIPELINE. PR_ENABLE_PIPELINE and PR_DISABLE_PIPELINE are constants defined in the prb.h file.

Remarks

To enable the pipelining, let pipe_line = PR_ENABLE_PIPELINE. To disable the pipelining, let pipe_line = PR_DISABLE_PIPELINE. (INTEGER 0 - 1) (Also see “PrClearPipeline,” page 5-10.)

PrSetQuadrant

Purpose

The PrSetQuadrant command sets the directions in which x and y coordinates will increase.

Syntax

```
status = PrSetQuadrant(int quad_number);
```

Parameters

quad_number—an integral value between 1 and 4, inclusive, where each value represents a unique directions-of-increase arrangement for the xy-axes.

PrSetRefDie

Purpose

The PrSetRefDie command assigns an (x,y) coordinate to the prober's target or reference die.

Syntax

```
status = PrSetRefDie(int x_start_position, int y_start_position);
```

Parameters

`x_start_position`—a variable that contains the x-coordinate desired to be assigned to the initial (reference) die. (INTEGER)

`y_start_position`—a variable that contains the y-coordinate desired to be assigned to the initial (reference) die. (INTEGER)

PrSetSlotStatus

Purpose

The PrSetSlotStatus command programs the prober's cassette map. This command informs the prober whether a particular slot is to be PROBED, SKIPPED, or marked as UNPROBED. In general, only UNPROBED slots will be tested.

Syntax

```
status = PrSetSlotStatus(int Cassette, int Slot, int StatusCode);
```

Parameters

Cassette—cassette number of the map to change. The cassette size is from 1 to a defined constant. The constant is defined in the file pointed to by KI_PRB_CONFIG (PROB_n_MAX_CASSETTE where n = test station number). See sample in KIDAT/prbcnfg_*XXXX*.dat. (INTEGER)

Slot—slot number of slot to change. The array size is from 1 to a defined constant. The constant is defined in the file pointed to by KI_PRB_CONFIG (PROBER_n_MAX_SLOT where n = test station number). See sample in KIDAT/prbcnfg_*XXXX*.dat. (INTEGER)

StatusCode—the value that the slot's status will change to after the PrSlotStatus command. (INTEGER, 1-3). One of a possible three arguments will be passed to the passed to this parameter: PR_SLOT_PROBED, PR_SLOT_UNPROBED, or PR_SLOT_SKIPPED.
PR_SLOT_PROBED indicates that the wafer in the slot has already been probed.
PR_SLOT_UNPROBED indicates that the wafer in the slot is designated to be probed but has not yet been probed.
PR_SLOT_SKIPPED indicates that the wafer in the slot is not to be probed.

The following is a list of the statuses to which a slot may have been set during the execution of PrCassetteMap; these constants are defined in the prb.h file.

Constants

DEFAULT_SLOT_STATUS
PR_SLOTSTATUS_UNMAPPED
PR_SLOTSTATUS_INPROCESS
PR_SLOTSTATUS_EMPTY
PR_SLOTSTATUS_UNPROBED
PR_SLOTSTATUS_PROBED
PR_SLOTSTATUS_PROBLEM
PR_SLOTSTATUS_UNSCHEDULED
Refer to sample prb.h file contained in Section 7.

PrSetTime

Purpose

This PrSetTime command changes the system default timeout value of 120 seconds on prober commands. The system will use this new value until the timeout is again changed by another PrSetTime command. Timeout errors occur when the prober takes longer than the default timeout value to respond to commands.

Syntax

```
status = PrSetTime(int new_time);
```

Parameters

`new_time`—This parameter specifies a new timeout value. The timeout value represents the integral number of seconds the prober will be given to respond to commands before the system returns a timeout error.

PrSetUnits

Purpose

The PrSetUnits command sets the prober to metric or English units.

Syntax

```
status = PrSetUnits(int Units);
```

Parameters

Units variable which defines the probing units. A value of PR_ENGLISH indicates English (imperial) units while a value of PR_METRIC indicates metric units. (INTEGER: PR_ENGLISH or PR_METRIC)

Constants

PR_ENGLISH

PR_METRIC

Refer to sample prb.h file contained in Section 7.

PrSetZHeight

Purpose

The PrSetZHeight command tells the prober to set the current z-stage (chuck) height as the new contact position.

Syntax

```
status = PrSetZHeight();
```

Parameters

NONE

Constants associated with PrAdjustZHeight
(See specific prober appendix for supported probers):
EG40
EG40_MIN_Z_ADJUST
EG40_MAX_Z_ADJUST
PR_ZFINE_RESOLUTION_EG40

P8
P8_MIN_Z_ADJUST
P8_MAX_Z_ADJUST
PR_ZFINE_RESOLUTION_P8

TSK9
TSK9_MIN_Z_ADJUST
TSK9_MAX_Z_ADJUST
PR_ZFINE_RESOLUTION_TSK9

Refer to sample prb.h file contained in Section 7.

PrSmifClamp

Purpose — The PrSmifClamp command causes the SMIF (FOUP) pod to be engaged (clamped) or disengaged (unclamped). If engaged (clamped) the cassette may NOT be removed from the indexer, conversely, if disengaged (unclamped) the cassette may be removed from the indexer.

Syntax

```
status = PrSmifClamp(int i_pod_number, int i_clamp_state);
```

Parameters — i_pod_number—this parameter identifies the pod that is to be clamped or unclamped.

i_clamp_state—this parameter identifies the desired pod status, clamped or unclamped. The argument passed to this parameter should be either SMIF_CLAMP_STATUS_UNLATCH for unclamp or SMIF_CLAMP_STATUS_LATCH for clamp. These constants are defined in the prb.h file.

Remarks — This command is valid only with a prober equipped with SMIF technology.

PrSmifLock

Purpose — The PrSmifLock command causes the SMIF (FOUP) pod to be engaged (locked) or disengaged (unlocked). If engaged (locked) the cassette may NOT be lowered, conversely, if disengaged (unlocked) the cassette may be lowered.

Syntax

```
status = PrSmifLock(int i_pod_number, int i_lock_state);
```

Parameters — i_pod_number—this parameter identifies the pod that is to be locked or unlocked.

i_lock_state—this parameter identifies the desired pod status with respect to locking (locked or unlocked). This argument passed to this parameter should be either SMIF_LATCH_STATUS_UNLATCH for unlock or SMIF_LATCH_STATUS_LATCH for lock. These constants are defined in the prb.h file.

Remarks — This command is valid only with a prober equipped with SMIF technology.

PrSmifStatus

Purpose — The PrSmifStatus command returns the status of various items on the prober, they are: operator mode, latch/lock status, cassette home status, pod present status and clamp status.

Syntax

```
status = int PrSmifStatus(int i_pod_number,
int *i_status_array, int i_status_array_size);
```

Parameters — i_pod_number—this parameter identifies the pod in question.

i_status_array —this parameter represents an array that contains information about the status of the pod and/or prober with respect to the prober features listed in the Purpose Section. The following is a list of constants that represent data entries to this array; the names of constants meaningfully convey the prober features and their corresponding statuses.

- SMIF_OPER_MODE_OFFSET
- SMIF_OPER_MODE_OFFSET - Operator mode
- SMIF_NORMAL_OPER_MODE

- SMIF_GEM_OPER_MODE
- SMIF_EXTRERNAL_OPER_MODE
- SMIF_SORTLINK_OPER_MOD

SMIF_LATCH_STATUS_OFFSET

- SMIF_LATCH_STATUS_OFFSET - Latch/Lock status,
- SMIF_LATCH_STATUS_UNKNOWN
- SMIF_LATCH_STATUS_UNLATCH

SMIF_CASS_HOME_OFFSET

- SMIF_CASS_HOME_OFFSET - indexer position
- SMIF_CASSETTE_HOME

SMIF_POD_PRESENT_OFFSET

- SMIF_CLAMP_STATUS_OFFSET *i_status_array_size* size of the status array defined by MAX_SMIF_STATUS_ITEMS.
- SMIF_POD_PRESENT_OFFSET - Pod status,

SMIF_CASSETTE_PRESENT

- SMIF_CLAMP_STATUS_OFFSET - Clamp status
- SMIF_CLAMP_STATUS_UNCLAMPED
- SMIF_CLAMP_STATUS_CLAMPED

Remarks

This command is valid only with a prober equipped with SMIF technology.

Definitions:

Clamp is defined as the availability of the pod to be removed from the indexer. If clamped, the pod may NOT be removed.

Lock and *latch* are used interchangeably, this is defined as the availability of the cassette/indexer to be lowered or mapped. If locked/latched, the cassette may NOT be mapped.

PrStatus

Purpose

This command is used to obtain status information about the following prober features: ready-for-probing, prober location with respect to the wafer, chuck position, and probing mode.

Syntax

```
status = PrStatus (int *ready, int *x_location, int *y_location,
int *chuck_position, int *prober_mode);
```

Parameters

ready—this parameter has a value of 1 when the chuck is up and the prober pins are touching the wafer; otherwise, its value is 0.

x_location, y_location—these parameters specify the xy-coordinates of the present probe site. These coordinates are given with respect to the coordinate system that was established by setting the target die coordinates using the PrInit command.

chuck_position—This parameter conveys information about the position of the chuck: a 1 indicates the chuck is up and a 0 indicates that the chuck is down.

prober_mode—This parameter indicates the prober's mode of operation. The prober may operate in one of three modes: manual, external, or edge. The argument passed to this parameter will be one of three constants that are defined in the prb.h file:

PR_MODE_MANUAL
PR_MODE_EXTERNAL
PR_MODE_EDGE

Remarks

Please note the following regarding the prober_mode parameter: some probers do not support all three modes. Please consult the documentation for your prober to ascertain which modes are supported by your prober.

PrUnLoad

Purpose

This command forces the wafer presently on the chuck to unload to its original cassette and slot.

Syntax

```
status = PrUnLoad();
```

Parameters

None

Remarks

This command is not used by Keithley's execution engine.

PrWriteRead

NOTE *PrWriteRead is to be used for serial probers only—see PrWriteReadSRQ for GPIB probers*

Purpose

This command allows you to create a string command and define a serial prober's response to the newly created command. The prober responses must be chosen from a list of available prober functions found in the prober documentation.

Syntax

```
status = PrWriteRead (char *input_buf, int input_buf_len, char
*output_buf, int output_buf, int terminator, int terminator_cnt);
```

Parameters

input_buf—this parameter is a pointer to an array whose content is a string representing the user-created command. The string's maximum length is 80 characters.

input_buf_len—this parameter is an integer that represents the actual number of characters in the name of the newly-defined command.

output_buf—this parameter is a pointer to an array that contains a string representing the prober's response to the newly created command found in input_buf.

output_buf_len—this parameter is an integer that defines the maximum length of the output buffer. This parameter is used to avoid overflowing the user output buffer.

terminator—this parameter gives the ASCII value(s) of the character(s) used to terminate the input_buf file.

terminator_cnt—this parameter represents the number of terminators that will be used to terminate the input_buf name. The value of this parameter is given by the constant NUM_TERMINATORS, that may be found in the header file for your model of prober. (e.g. KI_KULT_PATH PrXXXX.h)

Remarks

This command is only to be used for serial probers; PrWriteReadSRQ must be used for GBIP probers. When using this command, you must know which low-level commands are available for your model of prober and whether or not a reply is expected from the prober. Refer to the relevant prober documentation for information on low-level prober commands

PrWriteReadSRQ

NOTE *PrWriteReadSRQ is to be used for GPIB probers only—see PrWriteRead for serial probers.*

Purpose

This command allows you to create a string command and define a GPIB prober's response to the newly-created command; the prober responses must be chosen from a list of available prober functions found in the prober documentation. This command will also automatically poll for an srq.

Syntax

```
status = PrWriteReadSRQ (char *input_buf, int input_buf_len, char
*output_buf, int output_buf, int timeout, int *i_srq);
```

Parameters

input_buf—This parameter is a pointer to an array whose content is a string representing the user-created command. The maximum length of the string is 80 characters.

input_buf_len—This parameter is an integer that represents the actual number of characters in the name of the newly-defined command.

output_buf—This parameter is a pointer to an array which contains a string that represents the prober's response to the newly-created command found in input_buf.

output_buf_len—This parameter is an integer that defines the maximum length of the output buffer. This parameter is used to avoid overflowing the user output buffer.

timeout—This parameter gives the number of seconds the PrWriteReadSRQ function should wait for a prober response before timing out.

i_srq—This parameter is a pointer to the srq bit. This value will be the SRQ byte received from the prober. Zero and positive integer values indicate OK. Values less than (but not equal to) zero indicate errors. Use the return (from the function) to indicate status of command (success or failure). (INTEGER) For example:

```
x=PrWriteReadSRQ(...);
if (x==PR_OK);
      /* all is well */
      •
      •
      •
      /* look at the i_srq integer */
      •
      •
      •
else;
      /* generate error message to the user */
      •
      •
      •
```

Remarks

When using this command, determine which low-level commands are available on the prober and whether or not a reply is expected from the prober. Refer to the relevant prober documentation for information on prober low-level commands.

Lower level functions use the string length values to determine if a read or a write will be performed. If a string length is ≤ 0 (less than or equal to zero), the associated action will not be performed. Also, the lower level functions use the i_srq value to determine whether or not to serial poll. If the i_srq value ≥ 64 (decimal value greater than or equal to 64), a serial poll will be performed—if the i_srq value is a positive value < 64 (greater than zero but less than 64), a serial poll will NOT be performed. With this in mind:

- Set *output_buffer_maximum_length* to 0 if sending a command to the prober without returning a response in *output_buffer* (i.e., if you want to write but not read).
- Set *input_buffer_maximum_length* to 0 if returning a response from the prober without sending the command line contained in *input_buffer* to the prober (i.e., if you want to read but not write).
- Initialize the integer pointed to by *i_srq* to a decimal value ≥ 64 (decimal value greater than or equal to 64) if returning an SRQ from the prober.

Initialize the integer pointed to by i_srq to a decimal value = 0 if NOT returning an SRQ from the prober. Also, combinations of the string length values may be used as each action is independent. For example, setting output_buffer_maximum_length > 0, input_buffer_maximum_length > 0, and I_srq ≥ 64 will write to the bus, poll until SRQ of greater than or equal to 64 (or timeout) is received, and then read from the bus.

PrZParams

Purpose — Used to set one of the following height (z-axis) parameters for the prober: overtravel, clearance, up limit, down limit, or align height.

Syntax

```
status = PrZParams(int function, int value);
```

Parameters — function—this parameter identifies the z-axis parameter to be set. The following is a list of constants that may be passed as arguments to this parameter (these constants are defined in the prb.h file):

Constants	Purpose
PR_Z_TRAVEL	to set z overtravel
PR_Z_CLEARANCE	to set z clearance
PR_Z_LIMIT	to set z up limit
PR_Z_DOWN_LIMIT	to set z down limit
PR_Z_ALIGN_HEIGHT	to set z align height

Refer to sample prb.h file contained in Section 7.

value—an integer that is the value of the Z mode selected by the first parameter function.

PrZTravel

Purpose — This command sets the z travel mode from the tester.

Syntax

```
status = PrZTravel(int number);
```

Parameters — number—Selects the type of travel used for Z motion, as limited by edge sensor, profiler, or limit-to-limit setting.

Constants	Definition
PR_Z_LIMIT_MODE	Enables limits mode (limit-to-limit - Z up and Z down).
PR_Z_EDGE_MODE	Enables the edge sensor.
PR_Z_PROFILE_MODE	Enables the Z stage to be guided by the Profiler.

Refer to KI_KULT_PATH/Pr*XXXX*.h

Calling prober commands as functions

In order to return the status codes, call commands as functions. For example:

```
•
•
•
CommandStatus = PrWriteRead(command, strlen(command),response,
   20,10,  2);
if (CommandStatus <= 0)
{
/* communications error, shown below is -98, return answer as code known
for your application */
return (-98);
}
•
•
•
```

The status code returned from the above sample equals the value of variable *Command-Status*.

All automatic mode commands and most external prober commands return a function value of one (1) when they have successfully completed. The following list contains the exceptions:

In external mode, exceptions include:

- PrMove — When a successful move to a new site has been completed, a functional value of PR_MOVECOMPLETE is returned from the PrMove.
- PrLoad — When a PrLoad command is executed and a new wafer is successfully loaded, a functional value of PR_WAFERCOMPLETE is returned from PrLoad.
- PrLoad — When a PrLoad command is executed and no more wafers are available for loading, a functional value of PR_CASSETTECOMPLETE is returned from PrLoad.

The only other command that returns a negative number after proper execution is PrError. (PrError returns the prober error number.)

Acquiring prober data

The following code (Figure 5-1) is an example of how to acquire data from the prober presently in use. Pay particular attention to the **bold** code. Use this function as a template for extracting desired data from the current environment.

NOTE *Data returned will be related to the present test station number.*

Figure 5-1
KTXEADDIN/KTXEGetPrbData.c

```
/* USRLIB MODULE INFORMATION

            MODULE NAME: KTXEGetProberData
            MODULE RETURN TYPE: void
            NUMBER OF PARMS: 0
            ARGUMENTS:
            INCLUDES:
#include <stdio.h>
#include <lptdef.h>


#include <lptdef_lowercase.h>
#include <math.h>
#include "prb.h"
#include "kui_proto.h"
            END USRLIB MODULE INFORMATION
*/
/* USRLIB MODULE HELP DESCRIPTION
Example code for retrieving data from a specific test station's prober
structure.

The prober structure contains the following fields.
Please see $KIHOME/src/prb/com/getPrbStruct.c for details.
            END USRLIB MODULE HELP DESCRIPTION */
/* USRLIB MODULE PARAMETER LIST */
#include <stdio.h>
#include <lptdef.h>
#include <lptdef_lowercase.h>
#include <math.h>
#include "prb.h"
#include "kui_proto.h"

void KTXEGetProberData()
{
/* USRLIB MODULE CODE */
char C[32];
int I;
double D;

getPrbStruct(CHUCK_POSITION, &I);
printf("CHUCK_POSITION = %d\n", I);
getPrbStruct(X_POSITION, &I);
printf("X_POSITION = %d\n", I);
getPrbStruct(Y_POSITION, &I);
printf("Y_POSITION = %d\n", I);
getPrbStruct(PSXL, &I);
printf("PSXL = %d\n", I);
getPrbStruct(PSYL, &I);
```

```
printf("PSYL = %d\n", I);
getPrbStruct(MODE, &I);
printf("MODE = %d\n", I);
getPrbStruct(PROBNAME, &C);
printf("PROBNAME = %s\n", C);
getPrbStruct(X_INDEX, &I);
printf("X_INDEX = %d\n", I);
getPrbStruct(Y_INDEX, &I);
printf("Y_INDEX = %d\n", I);
getPrbStruct(X_INITIAL_LOCATION, &I);
printf("X_INITIAL_LOCATION = %d\n", I);
getPrbStruct(Y_INITIAL_LOCATION, &I);
printf("Y_INITIAL_LOCATION = %d\n", I);
getPrbStruct(X_MOVE_STEP, &I);
printf("X_MOVE_STEP = %d\n", I);
getPrbStruct(Y_MOVE_STEP, &I);
printf("Y_MOVE_STEP = %d\n", I);
getPrbStruct(UNITS, &I);
printf("UNITS = %d\n", I);
getPrbStruct(ERROR_LEVEL, &I);
printf("ERROR_LEVEL = %d\n", I);
getPrbStruct(TRANS_LOG_ENABLED, &I);
printf("TRANS_LOG_ENABLED = %d\n", I);
getPrbStruct(FAKE_LOG_ENABLED, &I);
printf("FAKE_LOG_ENABLED = %d\n", I);
getPrbStruct(IO_MODE, &I);
printf("IO_MODE = %d\n", I);
getPrbStruct(TIMEOUT, &I);
printf("TIMEOUT = %d\n", I);
getPrbStruct(GPIB_UNIT, &I);
printf("GPIB_UNIT = %d\n", I);
getPrbStruct(GPIB_SLOT, &I);
printf("GPIB_SLOT = %d\n", I);
getPrbStruct(GPIB_ADDRESS, &I);
printf("GPIB_ADDRESS = %d\n", I);
getPrbStruct(GPIB_WRITEMODE, &I);
printf("GPIB_WRITEMODE = %d\n", I);
getPrbStruct(GPIB_READMODE, &I);
printf("GPIB_READMODE = %d\n", I);
getPrbStruct(GPIB_TERMINATOR, &I);
printf("GPIB_TERMINATOR = %d\n", I);
getPrbStruct(S_PRB_OPTIONS, &C);
printf("S_PRB_OPTIONS = %s\n", C);
getPrbStruct(D_CUR_POS_X, &D);
printf("D_CUR_POS_X = %g\n", D);
getPrbStruct(D_CUR_POS_Y, &D);
printf("D_CUR_POS_Y = %g\n", D);
getPrbStruct(D_ABS_SITE_X, &D);
printf("D_ABS_SITE_X = %g\n", D);
getPrbStruct(D_ABS_SITE_Y, &D);
printf("D_ABS_SITE_Y = %g\n", D);
getPrbStruct(I_PRB_MAX_SLOTS, &I);
printf("I_PRB_MAX_SLOTS = %d\n", I);
getPrbStruct(I_PRB_MAX_CASSETTES, &I);
printf("I_PRB_MAX_CASSETTES = %d\n", I);
getPrbStruct(SHORT_TIMEOUT, &I);
printf("SHORT_TIMEOUT = %d\n", I);
getPrbStruct(SMIF_LOCK_STATUS, &I);
printf("SMIF_LOCK_STATUS = %d\n", I);
getPrbStruct(SMIF_CLAMP_STATUS, &I);
```

```
printf("SMIF_CLAMP_STATUS = %d\n", I);
getPrbStruct(SMIF_CASSETTE_STATUS, &I);
printf("SMIF_CASSETTE_STATUS = %d\n", I);
getPrbStruct(SMIF_MACHINE_STATUS, &I);
printf("SMIF_MACHINE_STATUS = %d\n", I);
getPrbStruct(SMIF_SENSE_WAFER, &I);
printf("SMIF_SENSE_WAFER = %d\n", I);
getPrbStruct(SMIF_STOP_RESUME, &I);
printf("SMIF_STOP_RESUME = %d\n", I);
getPrbStruct(D_DIE_SIZE_X, &D);
printf("D_DIE_SIZE_X = %g\n", D);
getPrbStruct(D_DIE_SIZE_Y, &D);
printf("D_DIE_SIZE_Y = %g\n", D);
getPrbStruct(D_INIT_COORD_X, &D);
printf("D_INIT_COORD_X = %g\n", D);
getPrbStruct(D_INIT_COORD_Y, &D);
printf("D_INIT_COORD_Y = %g\n", D);
getPrbStruct(I_INITIAL_SRQ, &I);
printf("I_INITIAL_SRQ = %d\n", I);
getPrbStruct(I_CURRENT_PROBER_CMD, &I);
printf("I_CURRENT_PROBER_CMD = %d\n", I);
getPrbStruct(I_LATEST_SRQ, &I);
printf("I_LATEST_SRQ = %d\n", I);
getPrbStruct(I_PROBER_COMM_MODE, &I);
printf("I_PROBER_COMM_MODE = %d\n", I);
getPrbStruct(I_PENDING_POD_EVENT, &I);
printf("I_PENDING_POD_EVENT = %d\n", I);

/* USRLIB MODULE END  */
}                       /* End KTXEGetProberData.c */
```

6
Modifying Prober Software

Introduction

NOTE *Refer to Section 1 for explanation of platform differences.*

Linux

Starting with KTE Software Version 5.1.x (the addition of proberio) use "PrWriteReadSRQ" on page 5-25 (GPIB) or "PrWriteRead" on page 5-24 (serial) instead of modifying or adding functions to the prober libraries.

The Keithley Prober Control Software is supplied as a KULT (Keithley User Library Tool) Library. This provides access to modify the source code (as a KULT library). Once `kult_copy_lib` is used to get the source and build *any* kult library, the `libnames.so` file is placed into KI_KULT_PATH (`KIHOME/usrlib`).

NOTE *Make sure to rename libprbcom.so and libprbgen_io.so (located in the KILIB) when making modifications to prober software in KULT. This allows changes to be saved properly while also providing a file containing the previous version.*

Sample names are:

```
KILIB/libprbcom.so.bak
KILIB/libprbgen_io.so.bak
```

NOTE *Prober libs are delivered into* `KI_KULT_PATH`*. Before modifying any prober specific lib, make sure to rename the original as a backup in* `KI_KULT_PATH`.

Adding a function to an existing prober driver

Names chosen for constants, functions, and libraries are critical to the correct operation of the prober driver. The PROBER_n_PROBTYPE, as defined in `(UNIX: KI_PRB_CONFIG)` or `(NT: KI_PRB_CONFIG)`, is directly related to the prober library name. The PROBER_n_PROBTYPE is case sensitive and critical for normal operation of the prober. The function name referenced in the Cnfg_nnnn.c file is the name of the actual C function used to operate the prober.

A string from the probe configuration file is taken and appended to the library name inside the code (dynamically loaded). If the exact file does not exist, the load will fail. Make sure that prb*XXXX* is a case sensitive word matching the probe config file and the library name. For example, PrRelReturn_EG40.c matches the configuration file Cnfg_EG40.c and also the library name NT: `prbEG40.lib`, `prbEG40.dll`, and for UNIX: `libprbEG40.so`.

NOTE *Throughout this section, EG4080 prober configuration files are used as samples. For prober specific sample files, refer to Section 7.*

Additionally, the order used when adding a new function to an existing prober driver is critical. This document reflects the required order.

1. Configure the appropriate file for the desired prober. Refer to Figure 6-1. Reference the environment variable KI_PRB_CONFIG to prbcnfg_*nnnn*.dat in KIDAT.
   (where '*nnnn*' is the QMO number)
2. Add a new function constant by editing `prb_func_id.h` in the `KIINCLUDE` directory. Refer to the shaded line of (last line).
3. Add a new function to the PRBGEN library (refer to Figure 6-3).
   - Change to the source directories for prbgen.
     UNIX: `KIHOME/src/prb/gen`
     NT: `KIHOME\src\prb\gen`
   - Build the PRBGEN KULT library.
     UNIX: `kult_copy_lib -lprbgen *.c`
     NT: `kultcopy prbgen`
   - Change to the source directories for prb*XXXX* (where '*XXXX*' is the prober library to be
     modified).
     UNIX: `KIHOME/src/prb/XXXX`
     NT: `KIHOME\src\prb\XXXX`
   - Build the prb*XXXX* KULT library.
     UNIX: `kult_copy_lib -lprbXXXX *.c`
     NT: `kultcopy prbXXXX`
   - Run KULT.
   - Open PRBGEN library to create a new generic function.
   - Copy an existing function to a new name. (Recommended method: copy an existing function and change the function name.)
   - Open the newly created function and edit.
   - Save the new generic function.
   - Compile the new function (within KULT).
   - Within KULT, build the prbgen KULT library.

NOTE *Pay particular attention to:*

- *Parameter quantity and types*

(Refer to the shaded lines of Figure 6-3.)

4. Add a new function to the prb*XXXX* library.
   - Using KULT, open prb*XXXX* library (the library where the new prober specific function will be created).
   - Copy an existing function to a new name (Recommended method: copy an existing function and change the logic and function name.)
   - Open the newly created function and edit.
   - Save the new function.
   - Compile the new function (within KULT).

NOTE *Pay particular attention to:*

- *Parameter quantity and types, to match the generic function*
- *Save the new function*
- *Compile the new function*
- *Build the prbXXXX library*

(Refer to the shaded lines of Figure 6-4.)

5. Using KULT open the Cnfg_*XXXX* file in the prb*XXXX* library.
   - Add a reference to the new function (see shaded lines of Figure 6-5). Add the new function in a call to '`putdrvadr`'.
   - Save the new prober specific function.
   - Compile the Cnfg_*XXXX* function within KULT.
   - Build the prb*XXXX* KULT library.

NOTE *In order to reference the new prbgen library, the following file must be backed up:*

- ***Linux:***
  backup `KILIB/libprbgen.so to KILIB/libprbgen.so.bak`

This is required because the new prbgen library exists in KI_KULT_PATH. The new function is now available.

Figure 6-1
Sample prbcnfg_nnnn.dat file (Linux)

```
# prbcnfg.dat  - EXAMPLE Prober Configuration File EG40 Prober
#
# The following line "<PRBCNFG>" is used by the engine in order to determine
# the MAX number of SLOTS and CASSETTES for a given prober at runtime.
#
<PRBCNFG>
#
# for OPTIONS "" == NULL, max 32 chars in string
#
# Example
#                  01234567890
#PROBER_1_OPTIONS=1,1,1,1,1,1
#
#
#     OcrPresent
#     AutoAlnPresent
#     ProfilerPresent
#     HotchuckPresent
#     HandlerPresent
#     Probe2PadPresent
#
#


#
# Configuration for direct GPIB probers (S530):
#  EG40
#
PROBER_1_PROBTYPE=EG40
PROBER_1_OPTIONS=0,0,0,0,1,0
PROBER_1_IO_MODE=GPIB
PROBER_1_GPIB_UNIT=0
PROBER_1_GPIB_SLOT=1
PROBER_1_GPIB_ADDRESS=5
PROBER_1_GPIB_WRITEMODE=0
PROBER_1_GPIB_READMODE=2
PROBER_1_GPIB_TERMINATOR=10
PROBER_1_TIMEOUT=300
PROBER_1_SHORT_TIMEOUT=5
PROBER_1_MAX_SLOT=25
PROBER_1_MAX_CASSETTE=1
#
#
```

Figure 6-2
Sample prb_func_id.h file

```
/* prb_func_id.h



*/
/************************************************************************

       COPYRIGHT (C) 1992  by  KEITHLEY INSTRUMENTS, INC.
       Cleveland, Ohio

       This software is furnished under a license and may
       be used and copied only in accordance with the terms
       of such license, and with the inclusion of the above
       COPYRIGHT notice.  This software or any other copies
       thereof may not be provided or otherwise made
       available to any other person.  No title to and
       ownership of the software is hereby transferred.
       The information in this software is subject to
       change without notice, and should not be construed
       as a commitment by KEITHLEY INSTRUMENTS, INC.

       KEITHLEY assumes no responsibility for the use or
       reliability of its software on equipment which is
       not supplied by KEITHLEY.

************************************************************************

 File:     $Source: /cm/test/E_Builds/S530/v420e17/COMMON/RCS/prb_func_id.h,v $
 Current $Revision: 1.13 $
 Current    $State: REL $
 Last Rev    $Date: 2000/07/14 17:15:17 $

 Change       $Log: prb_func_id.h,v $
 Change       Revision 1.13  2000/07/14 17:15:17  rybka
 Change       added new function PrCassetteMask
 Change
 Change       Revision 1.12  2000/07/11 14:20:48  rybka
 Change       added new function for needle cleaning
 Change
 Change       Revision 1.11  2000/01/25 20:43:45  rybka
 Change       added 2 new functions 66 & 67
 Change
 Change       Revision 1.10  1999/09/01 14:05:18  rybka
 Change       added SMIF specific constants
 Change
 Change       Revision 1.9  1998/12/01 20:22:40  rybka
 Change       added # 62 & 63 set/query chuck temp
 Change
 Change       Revision 1.8  1998/06/24 18:57:48  djohnson
 Change       added wrapper to only load once
 Change
 Change       Revision 1.7  1998/06/24 12:22:03  rybka
 Change       moved < extern PRBFUNCS xref_prb_func[]; defined in PRBCOM/prb_dec.c
 Change       to prb_extern.h PR5286
 Change
 Change       Revision 1.6  1998/06/22 13:40:03  rybka
 Change       moved the init of array of structs to prb_dec.
 Change
 Change       Revision 1.5  1998/06/18 17:52:11  rybka
 Change       PR5286 added a table to xref the text prober name with the function constant
 Change       this is used in KTXEAddIn/KTXEprbErrHdlr.c
 Change
 Change       Revision 1.4  1998/03/18 15:04:04  rybka
 Change       added PRWRITEREADSRQ # 61
```

Figure 6-2 (continued)
Sample prb_func_id.h file

```
 Change
 Change       Revision 1.3  1998/02/06 21:33:59  rybka
 Change       added functions 55-60 for SMIF apps
 Change
 Change       Revision 1.2  1997/04/22 13:02:39  williamson
 Change       Moved from S530 projcom area
 Change
 Change       Revision 1.2  1997/02/07 22:34:56  rybka
 Change       added support for PrAbsMove
 Change
 * Revision 1.1  1996/10/25  17:19:31  witzke
 * Initial revision
 *
 * Revision 1.2  1996/03/12  18:50:21  jain
 * PR1580 fixed the #define numbers to match the index of the prober cmds
 * array in KSOX--dispatch.c.  This will fix the problem of KITT displaying
 * incorrect prober drivers in the Prober Commnads window in KITT.  Also
 * add the #defines for the new EG functions.
 *
 * Revision 1.1  1993/02/06  17:39:49  beecher
 * Initial revision
 *
.....................................................................
*/

#ifndef _prb_func_id
#define _prb_func_id 1
#define PRAUTOALIGN          1
#define PRBEGINPROBE         2
#define PRCASSETTEMAP        3
#define PRCHECKOPTIONS       4
#define PRCHUCK              5
#define PRCLEARPIPELINE      6
#define PRDISABLETRANSLOG    7
#define PRENABLETRANSLOG     8
#define PRERROR              9
#define PRGETNXTWAFER        10
#define PRGETWAFER           11
#define PRINIT               12
#define PRINK                13
#define PRLEARN              14
#define PRLOAD               15
#define PRLOADFAILURE        16
#define PRLOADPRODUCT        17
#define PRMOVE               18
#define PRMOVNXT             19
#define PROFFLINE            20
#define PRPROFILE            21
#define PRPUTNXTSLOT         22
#define PRPUTWAFER           23
#define PRREADID             24
#define PRRELMOVE            25
#define PRRELOAD             26
#define PRRELRETURN          27
#define PRREQWAFERINFO       28
#define PRRETURNERRORMESSAGE 29
#define PRSERIALPOLL         30
#define PRSETDIAM            31
#define PRSETDIESIZE         32
#define PRSETFLAT            33
#define PRSETMATRIX          34
#define PRSETMODE            35
#define PRSETMPROBE          36
```

Figure 6-2 (continued)
Sample prb_func_id.h file

```
#define PRSETPIPELINE        37
#define PRSETQUADRANT        38
#define PRSETREFDIE          39
#define PRSETSKIPDIE         40
#define PRSETSLOTSTATUS      41
#define PRSETTIME            42
#define PRSETUNITS           43
#define PRSSLEARN            44
#define PRSSLOCATION         45
#define PRSSMOVE             46
#define PRSSMOVNXT           47
#define PRSTATUS             48
#define PRUNLOAD             49
#define PRWAIT               50
#define PRWRITEREAD          51
#define PRZPARAMS            52
#define PRZTRAVEL            53
#define PRABSMOVE            54
#define PRSMIFLOCK           55
#define PRSMIFLOCKSTATUS     56
#define PRPROBERSTATUS       57
#define PRSENSEWAFER         58
#define PRSTOP               59
#define PRSTART              60
#define PRWRITEREADSRQ       61
#define PRSETCHUCKTEMP       62
#define PRQUERYCHUCKTEMP     63
#define PRSMIFSTATUS         64
#define PRSMIFCLAMP          65
#define PRLOWERBOAT          66
#define PRCLEARALL           67
#define PRNEEDLECLEAN        68
#define PRCASSETTEMASK       69
#define PRGETPRODUCT         70
#define PRADJUSTZHEIGHT      71
#define PRQUERYZHEIGHT       72
#define PRSETZHEIGHT         73
#define PRREADCASSETTEID     74


#define PRMAXFUNCIDS         74 /*this must equal the last function value */

/* the PRBFUNCS typedef is used in KTXEAddIn/KTXEPrbErrHdlr
 to determine, based on an input string function name "PrLoad",
 the constant associated with it, PRLOAD.  This will allow users
 to define their own prober error handled.
*/

typedef struct {
char c_func_name[64];
int i_func_id;
}
PRBFUNCS;

#endif
```

Figure 6-3
Example of the PrInit.c function in the prbgen KULT library

```
/* USRLIB MODULE INFORMATION

      MODULE NAME: PrInit
      MODULE RETURN TYPE: int
      NUMBER OF PARMS: 7
      ARGUMENTS:
                  mode,       int,        Input
                  x_die_size, double,     Input
                  y_die_size, double,     Input
                  x_start_position,int,   Input
                  y_start_position,int,   Input
                  units,      int,        Input
                  subprobtype,int,        Input
      INCLUDES:
#ifdef WIN32
#include "ktemalloc.h"
#endif
#include <stdio.h>
#include <lptdef.h>
#include <lptdef_lowercase.h>
#include <math.h>
#include    "prb.h"
#include    "prb_func_id.h"
#include    "prb_extern.h"
#include    "prb_drvadr.h"
      END USRLIB MODULE INFORMATION
*/
/* USRLIB MODULE HELP DESCRIPTION
PrInit

This command initializes the prober with die size, first coordinate, units and mode information.

Syntax

status = PrInit (int mode, double x_die_size, double y_die_size, int x_start_position, int y_start_position, int units, int
sub_type);

Parameters
mode = is the desired probing mode (if the prober supports the mode).

1   Manual
2   External
3   Edge-sense

The second and third parameters of this function are the x_die_size and y_die_size, respectively. These are entered as REAL
numbers. The units used to express these values depend on the value of the units parameter specified later in this function. If
the units are metric, these parameters are input in millimeters. If the units selected are English, these parameters are input
in mils.

The fourth and fifth parameters of this function are the x_start_position and y_start_position, respectively. These are entered
as integers. These values are the x and y locations to be used to define the prober position at alignment.

The sixth parameter is the units specification parameter. This parameter is an integer. If this parameter is 0 then the units
selected are English. If this parameter is 1 then the units selected are metric.

The seventh parameter is an integer which represents the secondary type of the prober (not used).

Remarks

This command is used by Keithley's execution engine (partially supported by S530 systems).


      END USRLIB MODULE HELP DESCRIPTION */
/* USRLIB MODULE PARAMETER LIST */
#ifdef WIN32
#include "ktemalloc.h"
#endif
#include <stdio.h>
#include <lptdef.h>
#include <lptdef_lowercase.h>
#include <math.h>
#include    "prb.h"
#include    "prb_func_id.h"
#include    "prb_extern.h"
#include    "prb_drvadr.h"

int PrInit( int mode, double x_die_size, double y_die_size, int x_start_position, int y_start_position, int
units, int subprobtype )
{
/* USRLIB MODULE CODE */
/* PrInit.c */
/*****************************************************************************

        COPYRIGHT (C) 1992  by  KEITHLEY INSTRUMENTS, INC.
        Cleveland, Ohio

        This software is furnished under a license and may
        be used and copied only in accordance with the terms
        of such license, and with the inclusion of the above
        COPYRIGHT notice.  This software or any other copies
        thereof may not be provided or otherwise made
        available to any other person.  No title to and
```

Figure 6-3 (continued)
Example of the PrInit.c function in the prbgen KULT library

```
         ownership of the software is hereby transferred.
         The information in this software is subject to
         change without notice, and should not be construed
         as a commitment by KEITHLEY INSTRUMENTS, INC.

         KEITHLEY assumes no responsibility for the use or
         reliability of its software on equipment which is
         not supplied by KEITHLEY.

**************************************************************************
..........................................................................

 Function: PrInit - Dispatcher for PrInit

 Technique:
      Configures prober for the current teststation
      Gets the address of the driver for the prober type
      Executes driver

 Input Parameters:
      mode - int
      x_die_size - real
      y_die_size - real
      x_start_position - int
      y_start_position - int
      units - int
      subprobtype - int

 Output Parameters: returns status
..........................................................................*/

int confstat;
int teststation;
int status;
int  (*drvptr)(int, double, double, int, int, int, int);
extern int ProbConfig( int *);
#ifdef WIN32
_declspec (dllimport) prb_t Prb[];
extern void prb_dec(void);
#endif

if(prb_debug_print) (void)fprintf(stdout,"IN DEBUG: PrInit: mode= %d, x_die_size= %f, y_die_size = %f, x_start_position = %d,
y_start_position = %d, units = %d, subprobtype = %d\n",mode, x_die_size, y_die_size, x_start_position, y_start_position, units,
subprobtype);

      prb_dec();
      confstat = ProbConfig( &teststation );
      if ( confstat != PR_OK )
      {
                  return( confstat );
      }
      if ( Prb[teststation].trans_log_enabled ) LogCmd( "PrInit" );

      /* get prober driver */
      drvptr = getdrvadr( teststation, PRINIT );

      /* call prober function */

      status = (*drvptr)( mode, x_die_size, y_die_size, x_start_position,
                  y_start_position, units, subprobtype );


      return( status );
}

int prinit_( int *mode, float *x_die_size, float *y_die_size, int *x_start_position, int *y_start_position, int *units, int
*subprobtype )
{
    double dx, dy ;

    dx = (double) *x_die_size ;
    dy = (double) *y_die_size ;

   return( PrInit( *mode, dx, dy, *x_start_position , *y_start_position, *units, *subprobtype ));
/* USRLIB MODULE END  */
}                 /* End PrInit.c */
```

Figure 6-4
Example of the PrInit_EG40.c function in the prbEG40 KULT library

```
/* USRLIB MODULE INFORMATION

      MODULE NAME: PrInit_EG40
      MODULE RETURN TYPE: int
      NUMBER OF PARMS: 7
      ARGUMENTS:
                  mode,       int,        Input
                  x_die_size, double,     Input
                  y_die_size, double,     Input
                  x_start_position,int,   Input
                  y_start_position,int,   Input
                  units,      int,        Input
                  subprobtype,int,        Input
      INCLUDES:
#include <stdio.h>
#include <lptdef.h>
#include <lptdef_lowercase.h>
#include <math.h>
#include         <string.h>
#include         "prb.h"
#include         "prb_msg.h"
#include         "PrEG40.h"
#include         "prb_extern.h"
      END USRLIB MODULE INFORMATION
*/
/* USRLIB MODULE HELP DESCRIPTION
 Function: PrInit_EG40 - PrInit for EG40 style probers

 Technique:

 Input Parameters:

 Output Parameters:
$REVISION$

      END USRLIB MODULE HELP DESCRIPTION */
/* USRLIB MODULE PARAMETER LIST */
#ifdef WIN32
#include "ktemalloc.h"
#endif
#include <stdio.h>
#include <lptdef.h>
#include <lptdef_lowercase.h>
#include <math.h>
#include         <string.h>
#include         "prb.h"
#include         "prb_msg.h"
#include         "PrEG40.h"
#include         "prb_extern.h"

int PrInit_EG40( int mode, double x_die_size, double y_die_size, int x_start_position, int y_start_position, int units, int sub-
probtype )
{
/* USRLIB MODULE CODE */
#ifdef WIN32
      _declspec (dllimport) prb_t Prb[];
#endif
    int teststation,sendstatus, i_srq;
    char ibuf[60],obuf[60];

    if (prb_debug_print) printf("IN DEBUG: PrInit_EG40 \n");

    teststation = gettstn();/* Get index for current prober teststation */

    Prb[teststation].units=units;
    Prb[teststation].mode = mode;
    Prb[teststation].x_position=x_start_position;
    Prb[teststation].y_position=y_start_position;
    Prb[teststation].d_abs_site_x = 0.0;
    Prb[teststation].d_abs_site_y = 0.0;
    Prb[teststation].d_cur_pos_x = 0.0;  /* used to accum english only */
    Prb[teststation].d_cur_pos_y = 0.0;

    /* Set units */

    if (units != 1)
       (void)sprintf(ibuf,"SM1U%i\n",0);
    else
       (void)sprintf(ibuf,"SM1U%i\n",units);
```

Figure 6-4 (continued)
Example of the PrInit_EG40.c function in the prbEG40 KULT library

```
    sendstatus=pregio(ibuf, strlen(ibuf), obuf, 8, Prb[teststation].gpib_terminator, 2, &i_srq, Prb[teststa-
tion].short_timeout);

if ( sendstatus > 0 )
      {
        if ( strstr(obuf, "MC") != NULL )
        {
             Prb[teststation].units=units;;
        }
        else
        {
                    ProberError( SET_UNITS_FAIL, "PrInit_EG40",0);
             return( SET_UNITS_FAIL);
        }
      }
 else
      {
                ProberError( UNINTEL_RESP, "PrInit_EG40",0);
                  return( UNINTEL_RESP);
      }

/* Mode setup */

    if (mode < 1 || mode > 6)
      {
             ProberError(SET_MODE_FAIL,"PrInit_EG40",0);
       return(SET_MODE_FAIL);
      }

    Prb[teststation].mode = mode;

    if (mode==1 || mode ==2)
      {
            (void)sprintf(ibuf,"SM4P10\n");
            sendstatus = pregio(ibuf, strlen(ibuf), obuf, 60, Prb[teststation].gpib_terminator, 2, &i_srq, Prb[teststa-
tion].short_timeout);
      }
    else
     {
           if (mode==3) (void)sprintf(ibuf,"SM4P1\n"); /* Edge sense */
           if (mode==4) (void)sprintf(ibuf,"SM4P2\n"); /* Matrix */
           if (mode==5) (void)sprintf(ibuf,"SM4P3\n"); /* Circular */
           if (mode==6) (void)sprintf(ibuf,"SM4P4\n"); /* Learn */
      sendstatus = pregio(ibuf, strlen(ibuf), obuf, 60, Prb[teststation].gpib_terminator, 2, &i_srq, Prb[teststa-
tion].short_timeout);
      }

if ( sendstatus > 0 )
      {
        if ( strstr(obuf, "MC") != NULL )
          {
              Prb[teststation].x_position=x_start_position;
                   Prb[teststation].y_position=y_start_position;
                   Prb[teststation].d_abs_site_x = 0.0;
                   Prb[teststation].d_abs_site_y = 0.0;
                   Prb[teststation].d_cur_pos_x = 0.0;  /* used to accum english only */
                   Prb[teststation].d_cur_pos_y = 0.0;
               return(PR_OK);
          }
         else
           {
                   ProberError( SET_MODE_FAIL, "PrInit_EG40",0);
              return( SET_MODE_FAIL);
            }
      }
 else
      {
                ProberError( UNINTEL_RESP, "PrInit_EG40",0);
                  return( UNINTEL_RESP);
      }

/* USRLIB MODULE END  */
}                /* End PrInit_EG40.c */
```

Figure 6-5
Example of the PrInit.c function in the prbgen KULT library

```
/* USLIB MODULE INFORMATION

      MODULE NAME: Cnfg_EG40
      MODULE RETURN TYPE: int
      NUMBER OF PARMS: 1
      ARGUMENTS:
                  teststation,int,        Input
      INCLUDES:
#ifdef WIN32
#include "ktemalloc.h"
#endif
#include <stdio.h>
#include <lptdef.h>
#include <lptdef_lowercase.h>
#include <math.h>
#include        "prb.h"
#include        "prb_func_id.h"
#include        "prb_extern.h"
#include        "prbEG40_proto.h"
#include        "prb_drvadr.h"
      END USRLIB MODULE INFORMATION
*/
/* USRLIB MODULE HELP DESCRIPTION
 Function: Cnfg_EG40 - EG40 prober configuration module

 Technique:
      Configure TTY port
      Get the driver dispatch table
      Add supported commands to command table

 Input Parameters:

 Output Parameters:

............
      END USRLIB MODULE HELP DESCRIPTION */
/* USRLIB MODULE PARAMETER LIST */
#ifdef WIN32
#include "ktemalloc.h"
#endif
#include <stdio.h>
#include <lptdef.h>
#include <lptdef_lowercase.h>
#include <math.h>
#include        "prb.h"
#include        "prb_func_id.h"
#include        "prb_extern.h"
#include        "prbEG40_proto.h"
#include        "prb_drvadr.h"

int Cnfg_EG40( int teststation )
{
/* USRLIB MODULE CODE */
#ifdef WIN32
      _declspec (dllimport) prb_t Prb[];
#endif
      int     confstat;

if (prb_debug_print)
    {
        printf("IN DEBUG: Cnfg_EG40 \n");
        printf(" Prb[ teststation ].io_mode = %d \n", Prb[ teststation ].io_mode);

    }

if ( Prb[ teststation ].io_mode == SERIAL )
      confstat = Cnfg_tty( teststation );
else if ( ( Prb[ teststation ].io_mode == GPIB ) ||  ( Prb[ teststation ].io_mode == GPIBCT ) )
      confstat = Cnfg_gpib( teststation );/*configure teststation*/

      if (confstat != PR_OK ) return(confstat);

      confstat = getdsptab( teststation );
      if (confstat != PR_OK ) return(confstat);

      (void)putdrvadr( teststation, PRNEWFUNC, &PrNewFunc_xxxx );

      (void)putdrvadr( teststation, PRAUTOALIGN, &PrAutoAlign_EG40 );
      (void)putdrvadr( teststation, PRABSMOVE, &PrAbsMove_EG40 );
      (void)putdrvadr( teststation, PRCASSETTEMAP, &PrCassetteMap_EG40 );
      (void)putdrvadr( teststation, PRCHECKOPTIONS, &PrCheckOptions_EG40 );
      (void)putdrvadr( teststation, PRCHUCK, &PrChuck_EG40 );
      (void)putdrvadr( teststation, PRCLEARPIPELINE, &PrClearPipeline_EG40);
      (void)putdrvadr( teststation, PRERROR, &PrError_EG40 );
      (void)putdrvadr( teststation, PRGETNXTWAFER, &PrGetNxtWafer_EG40 );
      (void)putdrvadr( teststation, PRGETWAFER, &PrGetWafer_EG40 );
      (void)putdrvadr( teststation, PRINIT, &PrInit_EG40 );
      (void)putdrvadr( teststation, PRINK, &PrInk_EG40 );
      (void)putdrvadr( teststation, PRLOADPRODUCT, &PrLoadProduct_EG40 );
      (void)putdrvadr( teststation, PRLOAD, &PrLoad_EG40 );
      (void)putdrvadr( teststation, PRUNLOAD, &PrUnLoad_EG40 );
      (void)putdrvadr( teststation, PRMOVNXT, &PrMovNxt_EG40 );
      (void)putdrvadr( teststation, PRMOVE, &PrMove_EG40 );
      (void)putdrvadr( teststation, PRPROFILE, &PrProfile_EG40 );
      (void)putdrvadr( teststation, PRPUTNXTSLOT, &PrPutNxtSlot_EG40 );
      (void)putdrvadr( teststation, PRPUTWAFER, &PrPutWafer_EG40 );
      (void)putdrvadr( teststation, PRREADID, &PrReadId_EG40 );
      (void)putdrvadr( teststation, PRRELMOVE, &PrRelMove_EG40 );
      (void)putdrvadr( teststation, PRRELRETURN, &PrRelReturn_EG40 );
      (void)putdrvadr( teststation, PRSERIALPOLL, &PrSerialPoll_EG40 );
```

Figure 6-5 (continued)
Example of the PrInit.c function in the prbgen KULT library

```
      (void)putdrvadr( teststation, PRSETDIAM, &PrSetDiam_EG40 );
      (void)putdrvadr( teststation, PRSETDIESIZE, &PrSetDieSize_EG40 );
      (void)putdrvadr( teststation, PRSETFLAT, &PrSetFlat_EG40 );
      (void)putdrvadr( teststation, PRSETMODE, &PrSetMode_EG40 );
      (void)putdrvadr( teststation, PRSETPIPELINE, &PrSetPipeline_EG40 );
      (void)putdrvadr( teststation, PRSETQUADRANT, &PrSetQuadrant_EG40 );
      (void)putdrvadr( teststation, PRSETREFDIE, &PrSetRefDie_EG40 );
      (void)putdrvadr( teststation, PRSETSLOTSTATUS, &PrSetSlotStatus_EG40 );
      (void)putdrvadr( teststation, PRSETTIME, &PrSetTime_EG40 );
      (void)putdrvadr( teststation, PRSETUNITS, &PrSetUnits_EG40 );
      (void)putdrvadr( teststation, PRSTATUS, &PrStatus_EG40 );
      (void)putdrvadr( teststation, PRWRITEREAD, &PrWriteRead_EG40 );
      (void)putdrvadr( teststation, PRZPARAMS, &PrZParams_EG40 );
      (void)putdrvadr( teststation, PRZTRAVEL, &PrZTravel_EG40 );

      return( PR_OK );
/* USRLIB MODULE END  */
}               /* End Cnfg_EG40.c */
```

Alternative to adding a new function to a prober driver

As an alternative to adding a new function to a prober driver, use the PrWriteRead function (for GPIB probers, use PrWriteReadSRQ). This function allows you to send any selected prober string (command) and to receive the prober's response. Refer to `PrWriteRead` (`PrWriteReadSRQ`) command contained in Section 5 for detailed information.

Debugging probers

The Keithley Prober Driver Library can log all prober software and communications transaction to a file. Both high-level commands (e.g., PrInit) and resulting communications traffic are logged. This feature can be used when trying to isolate low-level prober and driver software faults and interactions.

Normally, prober transaction logging is enabled for one of the following situations:

- Before upgrading the prober's firmware, upgrading a software release from Keithley, or before writing a new command. (This provides a “before picture” of the transaction log.)
- The prober's firmware was upgraded and now programs that previously worked no longer work properly. In this case, do a before and after comparison of transmission logs.
- You wrote a new command (either adding a new function to an existing prober driver, using the PrWriteRead command or, for GPIB probers, using the PrWriteReadSRQ command), but the prober does not behave as you would expect. A transmission will show exactly what occurred at a low-level.
- You received a new software release from Keithley, and now the prober behaves differently. Again, a before and after comparison of transmission logs would prove beneficial both to you and Keithley's Technical Support personnel.

NOTE *These are just a few of the situations where a transmission log would be useful.*

Enabling the prober transaction log

NOTE *Use the transaction log to isolate driver and firmware faults as well as communications problems. If you suspect a driver software or prober firmware problem, use the transaction logging feature.*

Transaction logging commands

Enable transaction logging by setting the following environment variables (Table 6-1):

Table 6-1
Transaction logging commands

Command	Description
Linux: `setenv KI_PRB_DEBUG PRINTTRANSCMNDS`	Output transactions and commands
Linux: `setenv KI_PRB_AUDIT_LOG 'path'`	Output location

`path` examples:default (no action required)

`KILOG/prober.log`(disk file Linux)

`/dev/tty`(standard out Linux)

NOTE *Because the transaction log file is written to the disk, test programs will run slightly slower when transaction logging is enabled. This is due to the extra time needed to log each transaction. Therefore, enable transaction logging only when isolating or diagnosing prober-tester problems (recommended). Using transaction logging for everyday production is not recommended. Refer to Figure 6-5 and Figure 6-6 for an example of debug output.*

FAKE prober debug

NOTE *Earlier version of KTE (prior to KTE 4.2) used an environment variable:*
KI_PRB_FAKE_CASSMAP
This variable was replaced in version KTE 4.2 with four environment variables:
KI_PRB_FAKE_CASSMAP1, KI_PRB_FAKE_CASSMAP2, KI_PRB_FAKE_CASSMAP3, and KI_PRB_FAKE_CASSMAP4
This allows for up to four cassettes on a given prober (per the prbcfg.dat file).*

FAKE debug output is controlled by the environment variable `KI_PRB_FAKE_OUTPUT`. The environment variables `KI_PRB_FAKE_CASSMAP`*x* may be useful (where *x* is an integer from 1-4 representing the available cassette maps).

By default, the FAKE cassette map is set to `444444444444444444444444`. This represents a full complement of wafers in a 25 wafer boat. Here is an example of a 25 slot cassette where the first 5 slots have unprobed wafers and the next 20 slots are empty, set the environment variable KI_PRB_FAKE_CASSMAP equal to `444443333333333333333333`.

NOTE *Refer to Section 5, the function description for PrCassetteMap, and Figure 7-1 for details about valid slot statuses.*

To change the options, set the environment `KI_PRB_FAKE_OUTPUT` equal to ‘path’

The three output options are:

1. Output from debug goes directly to the screen.
   This is the default setup—to make sure this output option is properly configured, make sure that KI_PRB_FAKE_OUTPUT environment variable is not set.
2. Output is turned off.
   To turn off output, issue the following command:
   UNIX: `setenv KI_PRB_FAKE_OUTPUT`
   NT: `set KI_PRB_FAKE_OUTPUT=` (see Note)
   This effectively sets the environment variable to “NULL” and turns off fake debug.
3. Output is redirected.
   To redirect the output to the file fake_out.log, issue the following command:
   Linux: `setenv KI_PRB_FAKE_OUTPUT KILOG/fake_out.log`
   This instructs the environment variable to pipe output to the file pointed to by KI_PRB_FAKE_OUTPUT. The output could be similarly redirected to a serial port or other output device/file as required (e.g., `/dev/ttya, /dev/ttyb`, etc.,).

For an example of debug output, refer to Figure 6-6 and Figure 6-7.

Figure 6-6
Sample EG40 GPIB log

```
EG40 GPIB LOG
+----------------------------------------
+ S530 Prober I/O Log for Test Station 1
+ Created on 21-Jan-1998 14:40:49 by user kthmgr
+ Running
+ No Error Log File Used.
+----------------------------------------
CMD:          PrInit
TESTER:       SM1U0<LF >
PROBER:       PRQUERYGPIB: 64 (dec), 40 (hex)
PROBER:       MC<CR ><LF >
TESTER:       SM4P10<LF >
PROBER:       PRQUERYGPIB: 64 (dec), 40 (hex)
PROBER:       MC<CR ><LF >
CMD:          PrLoad
TESTER:       LO<LF >
PROBER:       PRQUERYGPIB: 64 (dec), 40 (hex)
PROBER:       MC<CR ><LF >
CMD:          PrProfile
TESTER:       PZ<LF >
PROBER:       PRQUERYGPIB: 64 (dec), 40 (hex)
PROBER:       MC<CR ><LF >
CMD:          PrAutoAlign
TESTER:       AAF0<LF >
PROBER:       PRQUERYGPIB: 64 (dec), 40 (hex)
PROBER:       MC<CR ><LF >
TESTER:       MF<LF >
PROBER:       PRQUERYGPIB: 64 (dec), 40 (hex)
PROBER:       MC<CR ><LF >
CMD:          PrMove
TESTER:       MOX-00002Y000004<LF >
PROBER:       PRQUERYGPIB: 64 (dec), 40 (hex)
PROBER:       MC<CR ><LF >
CMD:          PrMove
TESTER:       MOX-00001Y000003<LF >
PROBER:       PRQUERYGPIB: 64 (dec), 40 (hex)
PROBER:       MC<CR ><LF >
CMD:          PrMove
TESTER:       MOX000000Y000002<LF >
PROBER:       PRQUERYGPIB: 64 (dec), 40 (hex)
PROBER:       MC<CR ><LF >
CMD:          PrClearPipeline
TESTER:       ?MD11S0<LF >
PROBER:       PRQUERYGPIB: 64 (dec), 40 (hex)
PROBER:       W1D1S5E0I<CR ><LF >
TESTER:       MWD11S0D0S0<LF >
PROBER:       PRQUERYGPIB: 64 (dec), 40 (hex)
PROBER:       MC<CR ><LF >S5E0I<CR ><LF >
TESTER:       ?MD9S0<LF >
PROBER:       PRQUERYGPIB: 64 (dec), 40 (hex)
PROBER:       W0D0S0E0I<CR ><LF >
```

Figure 6-7
Sample EG40 serial log

```
EG40 Serial Log
+-----------------------------------------
+ S530 Prober I/O Log for Test Station 1
+ Created on 22-Jan-1998 08:46:49 by user kthmgr
+ Running
+ Error log file is /opt/ki/dat/prob_err.log
+-----------------------------------------
CMD:          PrInit
TESTER:       SM1U0<LF >
PROBER:       ><CR ><LF >MC<CR ><LF >
TESTER:       SM4P10<LF >
PROBER:       ><CR ><LF >MC<CR ><LF >
CMD:          PrLoad
TESTER:       LO<LF >
PROBER:       ><CR ><LF >MC<CR ><LF >
CMD:          PrProfile
TESTER:       PZ<LF >
PROBER:       ><CR ><LF >MC<CR ><LF >
CMD:          PrAutoAlign
TESTER:       AAF0<LF >
PROBER:       ><CR ><LF >MC<CR ><LF >
TESTER:       MF<LF >
PROBER:       ><CR ><LF >MC<CR ><LF >
CMD:          PrMove
TESTER:       MOX000001Y000000<LF >
PROBER:       ><CR ><LF >MC<CR ><LF >
```

Non-printable characters are indicated by symbols enclosed by greater-than and less-than signs (< and>). Non-printable characters are those characters whose ASCII values are between 0-32 (decimal). Each command group (i.e., high-level prober command and related I/O activity) is separated by blank lines in the transaction log file. All I/O activity is shown in the correct time sequence in the report (Figure 6-5 and Figure 6-6).

Table 6-2 shows a list of the non-printable ASCII characters and their log file symbols.

Table 6-2
Non-printable ASCII characters

Decimal ASCII value	Log file symbol	Decimal ASCII value	Log file symbol
000	<NUL>	016	<DLE>
001	<SOH>	017	<DC1>
002	<STX>	018	<DC2>
003	<ETX>	019	<DC3>
004	<EOT>	020	<DC4>
005	<ENQ>	021	<NAK>
006	<ACK>	022	<SYN>
007	<BEL>	023	<ETB>
008	<BS>	024	<CAN>
009	<HT>	025	<EM>
010	<LF>	026	<SUB>
011	<VT>	027	<ESC>
012	<FF>	028	<FS>
013	<CR>	029	<GS>
014	<SO>	030	<RS>
015	<SI>	031	<US>

Using the transaction log to isolate problems

The transaction log file can be used to isolate problems caused by firmware or software upgrades. For example, suppose that you received a new firmware upgrade from a prober manufacturer. Now programs that worked prior to the upgrade no longer work. In this case, you could compare before and after transaction logs generated by the same program using a file comparison utility. Normally, the only differences should be in the header of the two log files; that is, that information related to the execution time and date. If the comparison shows differences in the lines labeled **`PROBER:`**, then the new firmware is probably not 100% compatible. Similarly, if there are differences in the lines labeled **`TESTER:`**, then the driver software has probably changed. In either case, you should contact Keithley's Technical Support and provide them with copies of both I/O logs so that the problem can be resolved quickly and efficiently. In most cases, it is not wise to upgrade your prober's firmware or software without first contacting your Keithley representative. By first contacting Keithley, you can verify that the intended upgrade is compatible with Keithley's Prober Driver Library or even recommended.

The transaction log can also prove useful in debugging prober functions that you have written using the **`PrWriteRead`** or **`PrWriteReadSRQ`** commands. In this case, you could examine the transaction log to make sure that the data you wanted to send to the prober was sent correctly. You could also check the prober's response. (The **`PrWriteRead`** and the **`PrWriteReadSRQ`** commands are detailed in Section 5.)

7
Sample Files

General prober files (all supported models)

Figure 7-1
Sample prb.h

Sample prb.h file fragment		Command*
• • •		
#define DEFAULT_DEVICE_IRQ	0	
#define DEFAULT_SERIAL_TIMEOUT	120	
#define DEFAULT_SHORT_TIMEOUT	5	
/* GPIB configuration default */		
#define DEFAULT_GPIB_UNIT	0	
#define DEFAULT_GPIB_SLOT	0	
#define DEFAULT_GPIB_ADDRESS	0	
#define DEFAULT_GPIB_WRITEMODE	0	
#define DEFAULT_GPIB_READMODE	2	
#define DEFAULT_GPIB_TERMINATOR	10 /* linefeed */	
#define DEFAULT_GPIB_TIMEOUT	60	
/* Serial configuration default */		
#define DEFAULT_SERIAL_TERMINATOR	DEFAULT_GPIB_TERMINATOR	
#define NO	0	
#define YES	1	
#define NOTCFIG	-1	
#define PR_OK	1	
#define MALLOC_FAIL	0	
#define SERIAL	232	
#define GPIB	488	
#define GPIBCT	256	
#define PR_CHUCK_DOWN	0	**Pr_Chuck**
#define PR_CHUCK_UP	1	
#define PR_NOT_READY	0	
#define PR_READY	1	
#define PR_MODE_MANUAL	1	**Pr_Mode**
#define PR_MODE_EXTERNAL	2	
#define PR_MODE_EDGE	3	
#define PR_MODE_MATRIX	4	
#define PR_MODE_CIRCULAR	5	
#define PR_MODE_AUTO	6	
#define PR_MODE_LEARN	PR_MODE_AUTO	
#define PR_MOVECOMPLETE	2	
#define PR_WAFERCOMPLETE	4	
#define PR_AUTOZ_COMPLETE	6	
#define PR_CASSETTECOMPLETE	8	
#define PR_LOTEND	10	
#define PR_WAFER_REJECT_LOAD	12 /* if prober needs to have next wafer / loaded explicitly*/	
#define PR_WAFER_REJECT_NOLOAD	14 /* if prober DOES NOT need to have next wafer loaded explicitly*/	
#define PR_OPER_ACTION	16 /* if the oper needs to do something but kte does not know until the oper responds to kui query */	
#define PR_ENGLISH	0	
#define PR_METRIC	1	
#define MAXSLOTCOUNT	25	
/* Defines for PrSetSlotStatus() */		**PrSetSlotStatus**
#define PR_SLOT_SKIP	1	
#define PR_SLOT_PROBED	2	
#define PR_SLOT_UNPROBED	3	
/* Returns from PrCassetteMap() */		**PrCassetteMap**
#define PR_SLOT_STATUS_UNMAPPED	1	
#define PR_SLOT_STATUS_INPROCESS	2	
#define PR_SLOT_STATUS_EMPTY	3	
#define PR_SLOT_STATUS_UNPROBED	4	
#define PR_SLOT_STATUS_PROBED	5	
#define PR_SLOT_STATUS_PROBLEM	6	
#define PR_SLOT_STATUS_UNSCHEDULED	7	
#define DEFAULT_SLOT_STATUS	"" /* Null */	

Figure 7-1 (continued)
Sample prb.h

Sample prb.h file fragment		Command*
/* PrZTravel mode operation parameters */		**PrZTravel**
#define PR_Z_TRAVEL_LIMITS	0	
#define PR_Z_TRAVEL_EDGE	1	
#define PR_Z_TRAVEL_PROFILE	2	
/* PrZParams operation parameter function inputs */		**PrZParams**
#define PR_Z_OVERTRAVEL	1	
#define PR_Z_CLEARANCE	2	
#define PR_Z_UP_LIMIT	3	
#define PR_Z_DOWN_LIMIT	4	
#define PR_Z_ALIGN_HEIGHT	5	
/* PrPutNxtSlot and PrPutWafer Reason codes */		**PrPutNxtSlot / PrPutWafer**
#define PR_NORMAL_UNLOAD	0	
#define PR_PROFILE_FAIL	3	
#define PR_ALIGN_FAIL	4	
/* PrSetPipeLine input params */		**PrSetPipeLine**
#define PR_DISABLE_PIPELINE	0	
#define PR_ENABLE_PIPELINE	1	
/* prober quadrant numbers (from the probe pins perspective) */		**PrSetQuadrant**
#define PR_FIRST_QUADRANT	1	
#define PR_SECOND_QUADRANT	2	
#define PR_THIRD_QUADRANT	3	
#define PR_FORTH_QUADRANT	4	
/* SMIF Section */		**PrSmif commands**
/* lock and lock status */		
#define PR_SMIF_UNLOCK	1	
#define PR_SMIF_LOCK	0	
#define PR_SMIF_UNCLAMP	0	
#define PR_SMIF_CLAMP	1	
/* machine status */		
#define PR_SMIF_WAITING	'I'	
#define PR_SMIF_CARD_REPLACE	'C'	
#define PR_SMIF_LOT_PROCESS	'R'	
#define PR_SMIF_ERROR_STATE	'E'	
#define PR_SMIF_UNKNOWN	'X'	
/* cassette status */		
#define PR_SMIF_TESTING	'W'	
#define PR_SMIF_NOT_TESTING	' '	
/* smif sense wafer */		
#define PR_SMIF_NOT_SENSE_WAFER	0	
#define PR_SMIF_SENSE_WAFER	1	
/* smif stop probing */		
#define PR_SMIF_STOP_PROBING	0	
#define PR_SMIF_RESUME_PROBING	1	
/**		
Now introducing the EG4090u.... and how EG impliments SMIF		
**/		
#define MAX_SMIF_STATUS_ITEMS	6	
#define SMIF_NORMAL_OPER_MODE	0	
#define SMIF_GEM_OPER_MODE	1	
#define SMIF_EXTRERNAL_OPER_MODE	2	
#define SMIF_SORTLINK_OPER_MODE	3	
#define SMIF_LATCH_STATUS_UNKNOWN	0	
#define SMIF_LATCH_STATUS_UNLATCH	1	
#define SMIF_LATCH_STATUS_LATCH	2	
#define SMIF_CASSETTE_HOME	1	
#define SMIF_CASSETTE_PRESENT	1	
#define SMIF_CLAMP_STATUS_UNCLAMPED	0	
#define SMIF_CLAMP_STATUS_CLAMPED	1	
/*Offsets into the smif status array returned by PrSmifStatus */		
#define SMIF_OPER_MODE_OFFSET	1	
#define SMIF_LATCH_STATUS_OFFSET	2	
#define SMIF_CASS_HOME_OFFSET	3	
#define SMIF_POD_PRESENT_OFFSET	4	
#define SMIF_CLAMP_STATUS_OFFSET	5	
/* Probe needle cleaning types:		**PrNeedleClean**
brush, pad (fiber), ceramic, metal (sandpaper) sticky, blow....		
*/		
#define PR_CLEAN_NONE	0	
#define PR_CLEAN_BRUSH	1	
#define PR_CLEAN_FIBER	2	
#define PR_CLEAN_CERAMIC	3	
#define PR_CLEAN_METAL	4	
#define PR_CLEAN_STICKY	5	
#define PR_CLEAN_BLOW	6	
#define PR_CLEAN_GENERIC	7	
#define PR_MAX_CLEAN	7	**PrMaxClean**
•		
•		
•		

Detailed Explanation of prbcnfg.dat fields

The prober configuration file KI_PRB_CONFIG (i.e. KIDAT/prbcnfg_EG2X.dat) contains a field named PROBER_n_OPTIONS. This filed is used for all probers except EG models. The options field indicates the presence or non-presence of a particular option. The options are: OCR, auto alignment, profiler, hot chuck, wafer handler and probe to pad alignment. The options filed is used in the following fashion. If, PROBER_1_OPTIONS=1,0,0,0,1,1, then the following features are assumed to be present and active on the prober in question: OCR, handler, and probe to pad functions. By default only the handler option is set from the factory.

Electroglas 2001/2010/3001/4085 (EG2X driver)

Figure 7-2
EG2X prbcnfg.dat (S530KTE Sample)

```
# prbcnfg.dat  - EXAMPLE Prober Configuration File EG2X Prober
#
# The following tag, "PRBCNFG", is used by the engine in order to determine
# the MAX number of SLOTS and CASSETTES for a given prober at runtime.
#
<PRBCNFG>
#
# for OPTIONS "" == NULL, max 32 chars in string
#
# Example
#  01234567890
#PROBER_1_OPTIONS=1,1,1,1,1,1
#
#
#OcrPresent
#AutoAlnPresent
#ProfilerPresent
#HotchuckPresent
#HandlerPresent
#Probe2PadPresent
#
# NOTE: The EG2X prober driver does NOT use the "PROBER_n_OPTIONS" field.
#The PrCheckOptions function is used.
#
# Configuration for direct GPIB probers (S530):
#  EG2X
#
PROBER_1_PROBTYPE=EG2X
PROBER_1_OPTIONS=0,0,0,0,1,0
PROBER_1_IO_MODE=GPIB
PROBER_1_GPIB_UNIT=0
PROBER_1_GPIB_SLOT=1
PROBER_1_GPIB_ADDRESS=5
PROBER_1_GPIB_WRITEMODE=8
PROBER_1_GPIB_READMODE=10
PROBER_1_GPIB_TERMINATOR=10
PROBER_1_TIMEOUT=300
PROBER_1_SHORT_TIMEOUT=5
PROBER_1_MAX_SLOT=25
PROBER_1_MAX_CASSETTE=1
#
```

Figure 7-3
EG2X prbcnfg.dat (S530KTE FAKE sample)

```
# prbcnfg.dat  - EXAMPLE Prober Configuration File FAKE Prober
#
# The following tag, "PRBCNFG", is used by the engine in order to determine
# the MAX number of SLOTS and CASSETTES for a given prober at runtime.
#
<PRBCNFG>
#
# for OPTIONS "" == NULL, max 32 chars in string
#
# Example
#  01234567890
#PROBER_1_OPTIONS=1,1,1,1,1,1
#
#
#OcrPresent
#AutoAlnPresent
```

Figure 7-3 (continued)
EG2X prbcnfg.dat (S530KTE FAKE sample)

```
#ProfilerPresent
#HotchuckPresent
#HandlerPresent
#Probe2PadPresent
#
# Configuration for direct GPIB probers (S530):
#  FAKE
#
PROBER_1_PROBTYPE=FAKE
PROBER_1_OPTIONS=0,0,0,0,1,0
PROBER_1_IO_MODE=GPIB
PROBER_1_GPIB_UNIT=0
PROBER_1_GPIB_SLOT=1
PROBER_1_GPIB_ADDRESS=5
PROBER_1_GPIB_WRITEMODE=8
PROBER_1_GPIB_READMODE=10
PROBER_1_GPIB_TERMINATOR=10
PROBER_1_TIMEOUT=300
PROBER_1_SHORT_TIMEOUT=5
PROBER_1_MAX_SLOT=25
PROBER_1_MAX_CASSETTE=1
#
```

Electroglas 4060/4080/4090 (EG40 driver)

Figure 7-4
EG4X driver prbcnfg.dat (S530KTE sample)

```
# prbcnfg.dat  - EXAMPLE Prober Configuration File EG40 Prober
#
# The following tag, "PRBCNFG", is used by the engine in order to determine
# the MAX number of SLOTS and CASSETTES for a given prober at runtime.
#
<PRBCNFG>
#
# for OPTIONS "" == NULL, max 32 chars in string
#
# Example
#                 01234567890
#PROBER_1_OPTIONS=1,1,1,1,1,1
#
#
#       OcrPresent
#       AutoAlnPresent
#       ProfilerPresent
#       HotchuckPresent
#       HandlerPresent
#       Probe2PadPresent
#
#
# NOTE: The EG40 prober driver does NOT use the "PROBER_n_OPTIONS" field.
#               The PrCheckOptions function is used.
#
# Configuration for direct GPIB probers (S530):
#  EG40
#
PROBER_1_PROBTYPE=EG40
PROBER_1_OPTIONS=0,0,0,0,1,0
PROBER_1_IO_MODE=GPIB
PROBER_1_GPIB_UNIT=0
PROBER_1_GPIB_SLOT=1
PROBER_1_GPIB_ADDRESS=5
PROBER_1_GPIB_WRITEMODE=8
PROBER_1_GPIB_READMODE=10
PROBER_1_GPIB_TERMINATOR=10
PROBER_1_TIMEOUT=300
PROBER_1_SHORT_TIMEOUT=5
PROBER_1_MAX_SLOT=25
PROBER_1_MAX_CASSETTE=1
#
```

Figure 7-5
EG4X driver prbcnfg.dat (S530KTE FAKE sample)

```
# prbcnfg.dat  - EXAMPLE Prober Configuration File FAKE Prober
#
# The following tag, "PRBCNFG", is used by the engine in order to determine
# the MAX number of SLOTS and CASSETTES for a given prober at runtime.
#
<PRBCNFG>
#
# for OPTIONS "" == NULL, max 32 chars in string
#
# Example
#                   01234567890
#PROBER_1_OPTIONS=1,1,1,1,1,1
#
#
#       OcrPresent
#       AutoAlnPresent
#       ProfilerPresent
#       HotchuckPresent
#       HandlerPresent
#       Probe2PadPresent
#
# Configuration for direct GPIB probers (S530):
#  FAKE
#
PROBER_1_PROBTYPE=FAKE
PROBER_1_OPTIONS=0,0,0,0,1,0
PROBER_1_IO_MODE=GPIB
PROBER_1_GPIB_UNIT=0
PROBER_1_GPIB_SLOT=1
PROBER_1_GPIB_ADDRESS=5
PROBER_1_GPIB_WRITEMODE=8
PROBER_1_GPIB_READMODE=10
PROBER_1_GPIB_TERMINATOR=10
PROBER_1_TIMEOUT=300
PROBER_1_SHORT_TIMEOUT=5
PROBER_1_MAX_SLOT=25
PROBER_1_MAX_CASSETTE=1
#
```

Figure 7-6
"prbsrq_EG40.dat (S530KTE sample)"

```
#Keithley EG40 SRQ table
#Please NOTE: 'blank' lines are not permitted!!!!!!
#
# FORMAT of the file
#
# function name, "good srq list; bad srq list; known prober error numbers;unsolicited text messages"
#
# known prober error numbers are defined by the prober per PrError()
#
# the forth field will be used to draw a relationship between a list of srq values and text messages. In general this
# will only be applicable with EG probers.
#
# For example,
#
# PRCHECKUNSOLICITED,"70,71,72;0;0;$cmd1,cmd3,cmd2"
#
# the valid srq values (greater than 63 dec) 70,71,72 directly relate to the text messages $cmd1,cmd3,cmd2. These srq
# values are arbitrarily choosen in order to draw a relationship between a PAP (prober access point) and an unsolicited
# event.  The only constraint on the srq value is that it must be greater than 64.
#
# When the tester receives the message cmd3 this text is translated into decimal 71.  Then the 71 "srq" is checked
# against the list of available PAPs.  If a corresponding PAP exists it is executed immediately.
#
# All values are decimal
#
# Examples
#
# Good srqs are 70 and 72, Bad srqs are 76
#
# PRAUTOALIGN,"70,72;76"
#
# No good srqs , Bad srqs are 76
#
# PRAUTOALIGN,";76"
#
# Good srqs are 70 and 72, no Bad srqs
#
# PRAUTOALIGN,"70,72;"
#
# No SRQs associated with this function
#
# PRAUTOALIGN,""
#
#
#
Version,1.0
File,/opt/ki/dat/prbsrq_EG40.dat
Date,
Id,
Comment,
<EOH>
PRAUTOALIGN,""
PRBEGINPROBE,""
PRCASSETTEMAP,""
PRCHECKOPTIONS,""
PRCHUCK,""
PRCLEARPIPELINE,""
PRDISABLETRANSLOG,""
PRENABLETRANSLOG,""
PRERROR,""
PRGETNXTWAFER,""
PRGETWAFER,""
PRINIT,""
PRINK,""
PRLEARN,""
PRLOAD,""
PRLOADFAILURE,""
```

Figure 7-6 (continued)
"prbsrq_EG40.dat (S530KTE sample)"

```
#Keithley EG40 SRQ table
PRLOADPRODUCT,""
PRMOVE,""
PRMOVNXT,""
PROFFLINE,""
PRPROFILE,""
PRPUTNXTSLOT,""
PRPUTWAFER,""
PRREADID,""
PRRELMOVE,""
PRRELOAD,""
PRRELRETURN,""
PRREQWAFERINFO,""
PRRETURNERRORMESSAGE,""
PRSERIALPOLL,""
PRSETDIAM,""
PRSETDIESIZE,""
PRSETFLAT,""
PRSETMATRIX,""
PRSETMODE,""
PRSETMPROBE,""
PRSETPIPELINE,""
PRSETQUADRANT,""
PRSETREFDIE,""
PRSETSKIPDIE,""
PRSETSLOTSTATUS,""
PRSETTIME,""
PRSETUNITS,""
PRSSLEARN,""
PRSSLOCATION,""
PRSSMOVE,""
PRSSMOVNXT,""
PRSTATUS,""
PRUNLOAD,""
PRWAIT,""
PRWRITEREAD,""
PRZPARAMS,""
PRZTRAVEL,""
PRABSMOVE,""
PRSMIFLOCK,""
PRSMIFLOCKSTATUS,""
PRPROBERSTATUS,""
PRSENSEWAFER,""
PRSTOP,""
PRSTART,""
PRWRITEREADSRQ,""
PRSETCHUCKTEMP,""
PRQUERYCHUCKTEMP,""
PRSMIFSTATUS,""
PRSMIFCLAMP,""
PRLOWERBOAT,""
PRCLEARALL,""
PRNEEDLECLEAN,""
PRCASSETTEMASK,""
PRGETPRODUCT,""
PRADJUSTZHEIGHT,""
PRQUERYZHEIGHT,""
PRSETZHEIGHT,""
PRREADCASSETTEID,""
PRCHECKUNSOLICITED,""
<EOLOC>
```

Tel P8 (TELP8 driver)

Figure 7-7
TELP8 prbcnfg.dat (S530KTE sample)

```
# prbcnfg.dat  - EXAMPLE Prober Configuration File for P8 Prober
#
# The following tag, "PRBCNFG", is used by the engine in order to determine
# the MAX number of SLOTS and CASSETTES for a given prober at runtime.
#
<PRBCNFG>
#
# for OPTIONS "" == NULL, max 32 chars in string
#
# Example
#  01234567890
#PROBER_1_OPTIONS=1,1,1,1,1,1
#
#
#OcrPresent
#AutoAlnPresent
#ProfilerPresent
#HotchuckPresent
#HandlerPresent
#Probe2PadPresent
#
# Configuration for direct GPIB probers (S530):
#  P8
#
PROBER_1_PROBTYPE=P8
PROBER_1_OPTIONS=0,0,0,0,1,0
PROBER_1_IO_MODE=GPIB
PROBER_1_GPIB_UNIT=0
PROBER_1_GPIB_SLOT=1
PROBER_1_GPIB_ADDRESS=5
PROBER_1_GPIB_WRITEMODE=8
PROBER_1_GPIB_READMODE=10
PROBER_1_GPIB_TERMINATOR=10
PROBER_1_TIMEOUT=300
PROBER_1_SHORT_TIMEOUT=5
PROBER_1_MAX_SLOT=25
PROBER_1_MAX_CASSETTE=1
# only used if the P8 probe supports the "CS" command
# set equal to MASK if available and NOMASK (default) if not available
PROBE_1_P8_TYPE=NOMASK
#
```

Figure 7-8
Sample TELP8 Log

```
+----------------------------------------
+ S400 Prober I/O Log for Test Station 1
+ Created on 25-Jul-97 04:03:30 by user kthmgr4
+ Running
+ No Error Log File Used.
+----------------------------------------
CMD:          PrError
TESTER:       s<CR ><LF >
PROBER:       0000<CR ><LF >
CMD:          PrCheckOptions
CMD:          PrError
TESTER:       s<CR ><LF >
PROBER:       0000<CR ><LF >
CMD:          PrCassetteMap
TESTER:       d<CR ><LF >
PROBER:       1a````````a``a````````````2````````````````````````<CR ><LF >
CMD:          PrError
TESTER:       s<CR ><LF >
PROBER:       0000<CR ><LF >
CMD:          PrInit
TESTER:       A<CR ><LF >
PROBER:       000000<CR ><LF >
CMD:          PrRelReturn
TESTER:       I+0000001<CR ><LF >
PROBER:       PRSPOLL: 79 (dec), 4F (hex)
+----------------------------------------
+ S400 Prober I/O Log for Test Station 1
+ Created on 25-Jul-97 05:11:45 by user kthmgr4
+ Running
+ No Error Log File Used.
+----------------------------------------
CMD:          PrError
TESTER:       s<CR ><LF >
PROBER:       0000<CR ><LF >
CMD:          PrCheckOptions
```

Figure 7-8 (continued)
Sample TELP8 Log

```
CMD:          PrError
TESTER:       s<CR ><LF >
PROBER:       0000<CR ><LF >
CMD:          PrCassetteMap
TESTER:       d<CR ><LF >
PROBER:       1aaa`a``aa`aa`a``a`a``a`aa2``````````````````````````<CR ><LF >
CMD:          PrError
TESTER:       s<CR ><LF >
PROBER:       0000<CR ><LF >
CMD:          PrInit
TESTER:       A<CR ><LF >
PROBER:       000000<CR ><LF >
CMD:          PrRelReturn
TESTER:       I+0000001<CR ><LF >
PROBER:       PRSPOLL: 65 (dec), 41 (hex)
TESTER:       J+0000001<CR ><LF >
PROBER:       PRSPOLL: 66 (dec), 42 (hex)
CMD:          PrMove
TESTER:       X+00Y+04<CR ><LF >
PROBER:       PRSPOLL: 65 (dec), 41 (hex)
PROBER:       PRSPOLL: 66 (dec), 42 (hex)
CMD:          PrRelMove
TESTER:       I+0100001<CR ><LF >
PROBER:       PRSPOLL: 65 (dec), 41 (hex)
TESTER:       J+0000001<CR ><LF >
PROBER:       PRSPOLL: 66 (dec), 42 (hex)
CMD:          PrChuck
TESTER:       Z<CR ><LF >
PROBER:       PRSPOLL: 67 (dec), 43 (hex)
CMD:          PrChuck
TESTER:       D<CR ><LF >
PROBER:       PRSPOLL: 68 (dec), 44 (hex)
CMD:          PrRelReturn
TESTER:       I-0100001<CR ><LF >
PROBER:       PRSPOLL: 65 (dec), 41 (hex)
TESTER:       J+0000001<CR ><LF >
PROBER:       PRSPOLL: 66 (dec), 42 (hex)
CMD:          PrMove
TESTER:       X+02Y-02<CR ><LF >
PROBER:       PRSPOLL: 65 (dec), 41 (hex)
PROBER:       PRSPOLL: 66 (dec), 42 (hex)
CMD:          PrRelMove
TESTER:       I+0100001<CR ><LF >
PROBER:       PRSPOLL: 65 (dec), 41 (hex)
TESTER:       J+0000001<CR ><LF >
PROBER:       PRSPOLL: 66 (dec), 42 (hex)
CMD:          PrChuck
TESTER:       Z<CR ><LF >
PROBER:       PRSPOLL: 67 (dec), 43 (hex)
CMD:          PrChuck
TESTER:       D<CR ><LF >
PROBER:       PRSPOLL: 68 (dec), 44 (hex)
CMD:          PrRelReturn
TESTER:       I-0100001<CR ><LF >
PROBER:       PRSPOLL: 65 (dec), 41 (hex)
TESTER:       J+0000001<CR ><LF >
PROBER:       PRSPOLL: 66 (dec), 42 (hex)
CMD:          PrMove
TESTER:       X+00Y-02<CR ><LF >
PROBER:       PRSPOLL: 65 (dec), 41 (hex)
PROBER:       PRSPOLL: 66 (dec), 42 (hex)
CMD:          PrRelMove
TESTER:       I+0100001<CR ><LF >
PROBER:       PRSPOLL: 65 (dec), 41 (hex)
TESTER:       J+0000001<CR ><LF >
PROBER:       PRSPOLL: 66 (dec), 42 (hex)
CMD:          PrChuck
TESTER:       Z<CR ><LF >
PROBER:       PRSPOLL: 67 (dec), 43 (hex)
CMD:          PrRelReturn
TESTER:       I-0100001<CR ><LF >
PROBER:       PRSPOLL: 65 (dec), 41 (hex)
TESTER:       J+0000001<CR ><LF >
PROBER:       PRSPOLL: 66 (dec), 42 (hex)
CMD:          PrError
TESTER:       s<CR ><LF >
PROBER:       0000<CR ><LF >
CMD:          PrLoad
TESTER:       U<CR ><LF >
PROBER:       PRSPOLL: 70 (dec), 46 (hex)
CMD:          PrRelReturn
TESTER:       I+0000001<CR ><LF >
PROBER:       PRSPOLL: 65 (dec), 41 (hex)
TESTER:       J+0000001<CR ><LF >
PROBER:       PRSPOLL: 66 (dec), 42 (hex)
CMD:          PrMove
TESTER:       X+00Y+04<CR ><LF >
PROBER:       PRSPOLL: 65 (dec), 41 (hex)
PROBER:       PRSPOLL: 66 (dec), 42 (hex)
CMD:          PrRelMove
TESTER:       I+0100001<CR ><LF >
```

Figure 7-8 (continued)
Sample TELP8 Log

```
PROBER:      PRSPOLL: 65 (dec), 41 (hex)
TESTER:      J+0000001<CR ><LF >
PROBER:      PRSPOLL: 66 (dec), 42 (hex)
CMD:         PrChuck
TESTER:      Z<CR ><LF >
PROBER:      PRSPOLL: 67 (dec), 43 (hex)
CMD:         PrChuck
TESTER:      D<CR ><LF >
PROBER:      PRSPOLL: 68 (dec), 44 (hex)
CMD:         PrLoad
TESTER:      U<CR ><LF >
PROBER:      PRSPOLL: 70 (dec), 46 (hex)
```

Figure 7-9
"prbsrq_P8.dat (S530KTE sample)"

```
#Keithley P8 SRQ table
#Please NOTE: 'blank' lines are not permitted!!!!!!
#
# FORMAT of the file
#
# function name, "good srq list; bad srq list; known prober error numbers"
# known prober error numbers are defined by the prober per PrError()
#
# the forth field will be used to draw a relationship between a list of srq values and text messages.  In general this
# will only be applicable with EG probers.
#
# For example,
#
# PRCHECKUNSOLICITED,"70,71,72;0;0;$cmd1,cmd3,cmd2"
#
# the valid srq values (greater than 63 dec) 70,71,72 directly relate to the text messages $cmd1,cmd3,cmd2. These srq
# values are arbitrarily choosen in order to draw a relationship between a PAP (prober access point) and an unsolicited
# event.  The only constraint on the srq value is that it must be greater than 64.
#
# When the tester receives the message cmd3 this text is translated into decimal 71.  Then the 71 "srq" is checked
# against the list of available PAPs. If a corresponding PAP exists it is executed immediately.
#
# All values are decimal
#
# Examples
#
# Good srqs are 70 and 72, Bad srqs are 76
#
# PRAUTOALIGN,"70,72;76"
#
# No good srqs , Bad srqs are 76
#
# PRAUTOALIGN,";76"
#
# Good srqs are 70 and 72, no Bad srqs
#
# PRAUTOALIGN,"70,72;"
#
# No SRQs associated with this function
#
# PRAUTOALIGN,""
#
#
Version,1.0
File,/opt/ki/dat/prbsrq_P8.dat
Date,
Id,
Comment,
<EOH>
PRAUTOALIGN,""
PRBEGINPROBE,""
PRCASSETTEMAP,""
PRCHECKOPTIONS,""
PRCHUCK,"67,68;83;0"
PRCLEARPIPELINE,""
PRDISABLETRANSLOG,""
PRENABLETRANSLOG,""
PRERROR,""
PRGETNXTWAFER,"74,70,72;101,78;0"
PRGETWAFER,"74,70,72;101,78;0"
PRINIT,""
PRINK,""
PRLEARN,""
PRLOAD,"70,72,74;75,78,101;0"
PRLOADFAILURE,""
PRLOADPRODUCT,"89;78;0"
PRMOVE,"65,66;75,79;0"
PRMOVNXT,""
PROFFLINE,""
PRPROFILE,""
PRPUTNXTSLOT,""
PRPUTWAFER,""
PRREADID,""
PRRELMOVE,"65,66;75,79;0"
PRRELOAD,""
PRRELRETURN,"65,66;75,79;0"
PRREQWAFERINFO,""
PRRETURNERRORMESSAGE,""
PRSERIALPOLL,""
PRSETDIAM,""
PRSETDIESIZE,""
PRSETFLAT,""
PRSETMATRIX,""
PRSETMODE,""
PRSETMPROBE,""
PRSETPIPELINE,""
PRSETQUADRANT,""
PRSETREFDIE,""
PRSETSKIPDIE,""
PRSETSLOTSTATUS,""
PRSETTIME,""
PRSETUNITS,""
PRSSLEARN,""
PRSSLOCATION,""
PRSSMOVE,""
PRSSMOVNXT,""
PRSTATUS,""
PRUNLOAD,"89;0;0"
PRWAIT,""
PRWRITEREAD,""
PRZPARAMS,""
```

Figure 7-9 (continued)
"prbsrq_P8.dat (S530KTE sample)"

```
#Keithley P8 SRQ table
PRZTRAVEL,""
PRABSMOVE,"65,66;75,79;0"
PRSMIFLOCK,""
PRSMIFLOCKSTATUS,""
PRPROBERSTATUS,""
PRSENSEWAFER,""
PRSTOP,""
PRSTART,""
PRWRITEREADSRQ,""
PRSETCHUCKTEMP,"89;0;0"
PRQUERYCHUCKTEMP,""
PRSMIFSTATUS,""
PRSMIFCLAMP,""
PRLOWERBOAT,""
PRCLEARALL,"71;0;0"
PRNEEDLECLEAN,"89;0;0"
PRCASSETTEMASK,"70,89;0;0"
PRGETPRODUCT,""
PRADJUSTZHEIGHT,"67;83;0"
PRQUERYZHEIGHT,""
PRSETZHEIGHT,"89;0;0"
PRREADCASSETTEID,""
PRCHECKUNSOLICITED,"108;0;0"
<EOLOC>
```

TSK/Accretech probers (TSK9 driver)

Figure 7-10
TSK9 prbcnfg.dat (S530KTE sample)

```
# prbcnfg.dat  - EXAMPLE Prober Configuration File for TSK9 Prober
#
# The following tag, "PRBCNFG", is used by the engine in order to determine
# the MAX number of SLOTS and CASSETTES for a given prober at runtime.
#
<PRBCNFG>
#
# for OPTIONS "" == NULL, max 32 chars in string
#
# Example
#  01234567890
#PROBER_1_OPTIONS=1,1,1,1,1,1
#
#
#OcrPresent
#AutoAlnPresent
#ProfilerPresent
#HotchuckPresent
#HandlerPresent
#Probe2PadPresent
#
# Configuration for direct GPIB probers (S530):
#  TSK9
#
PROBER_1_PROBTYPE=TSK9
PROBER_1_OPTIONS=0,0,0,0,1,0
PROBER_1_IO_MODE=GPIB
PROBER_1_GPIB_UNIT=0
PROBER_1_GPIB_SLOT=1
PROBER_1_GPIB_ADDRESS=5
PROBER_1_GPIB_WRITEMODE=8
PROBER_1_GPIB_READMODE=10
PROBER_1_GPIB_TERMINATOR=10
PROBER_1_TIMEOUT=300
PROBER_1_SHORT_TIMEOUT=5
PROBER_1_MAX_SLOT=25
PROBER_1_MAX_CASSETTE=1
#
```

Figure 7-11
Sample TSK9 Log

```
+-----------------------------------------
+ S900 Prober I/O Log for Test Station 1
+ Created on 24-Apr-1998 08:10:08 by user kthmgr
+ Running
+ Prober in use: TSK9
+ No Error Log File Used.
+-----------------------------------------
CMD:          PrInit
PROBER:       PRWAITSRQCLEAR: 12 (dec), C (hex)
TESTER:       Q<CR ><LF >
PROBER:       QY132X126<CR ><LF >
CMD:          PrCassetteMap
TESTER:       w<CR ><LF >
```

Figure 7-11 (continued)
Sample TSK9 Log

```
PROBER:      w2000010001000000020000000l.0000000000000000000000000<CR ><LF >
CMD:         PrError
TESTER:      E<CR ><LF >
PROBER:      E<CR ><LF >
PROBER:      PRWAITSRQCLEAR: 4 (dec), 4 (hex)
CMD:         PrChuck
TESTER:      Z<CR ><LF >
PROBER:      PRQUERYGPIB: 67 (dec), 43 (hex)
CMD:         PrRelMove
TESTER:      AY+000000X+010000<CR ><LF >
PROBER:      PRQUERYGPIB: 65 (dec), 41 (hex)
CMD:         PrRelMove
TESTER:      AY+000000X+010000<CR ><LF >
PROBER:      PRQUERYGPIB: 65 (dec), 41 (hex)
CMD:         PrRelMove
TESTER:      AY+000000X-010000<CR ><LF >
PROBER:      PRQUERYGPIB: 65 (dec), 41 (hex)
CMD:         PrRelMove
TESTER:      AY+000000X-010000<CR ><LF >
PROBER:      PRQUERYGPIB: 65 (dec), 41 (hex)
CMD:         PrRelReturn
TESTER:      AY+000000X+010000<CR ><LF >
PROBER:      PRQUERYGPIB: 65 (dec), 41 (hex)
CMD:         PrChuck
TESTER:      D<CR ><LF >
PROBER:      PRQUERYGPIB: 68 (dec), 44 (hex)
CMD:         PrMove
TESTER:      SY-004X+003<CR ><LF >
PROBER:      PRQUERYGPIB: 66 (dec), 42 (hex)
CMD:         PrLoad
PROBER:      PRWAITSRQCLEAR: 1 (dec), 1 (hex)
TESTER:      L<CR ><LF >
PROBER:      PRQUERYGPIB: 81 (dec), 51 (hex)
PROBER:      PRSPOLL: 80 (dec), 50 (hex)
PROBER:      PRSPOLL: 70 (dec), 46 (hex)
```

The prbsrq_TSK9.dat file is used by the TSK9 driver. Use this file to define the known SRQ responses for a given prober on a command by command basis. The list for a given command is defined as follows:

- The function name
- The good srq list
- The bad srq list
- The known prober error returns. Note that the known prober error numbers are defined by the prober using the PrError() command (see Table 5-2).

Figure 7-12 and Figure 7-13 contains sample prbsrq_TSK9.dat file files.

Figure 7-12
prbsrq_TSK9.dat (S530KTE sample)

```
#Keithley TSK9 SRQ table
#Please NOTE: 'blank' lines are not permitted!!!!!!
#
# FORMAT of the file
#
# function name, "good srq list; bad srq list; known prober error numbers"
# known prober error numbers are defined by the prober per PrError()
#
# All values are decimal
#
# Examples
#
# Good srqs are 70 and 72, Bad srqs are 76
#
# PRAUTOALIGN,"70,72;76"
#
# No good srqs , Bad srqs are 76
#
# PRAUTOALIGN,";76"
#
# Good srqs are 70 and 72, no Bad srqs
#
# PRAUTOALIGN,"70,72;"
#
# No SRQs associated with this function
#
# PRAUTOALIGN,""
#
#
Version,1.0
```

Figure 7-12 (continued)
prbsrq_TSK9.dat (S530KTE sample)

```
File,/opt/ki/dat/prbsrq_TSK9.dat
Date,
Id,
Comment,
<EOH>
PRAUTOALIGN,"1,2;3,4;5"
PRBEGINPROBE,""
PRCASSETTEMAP,""
PRCHECKOPTIONS,""
PRCHUCK,"67,68;0;0"
PRCLEARPIPELINE,""
PRDISABLETRANSLOG,""
PRENABLETRANSLOG,""
PRERROR,""
PRGETNXTWAFER,"70;84,99;0"
PRGETWAFER,"70;84,99;0"
PRINIT,""
PRINK,"66,67,69,80,81;0;0"
PRLEARN,""
PRLOAD,"70,71,72,82,84,92,94,105;0;1057,1181,1184,1860"
PRLOADFAILURE,""
PRLOADPRODUCT,"98;99;0"
PRMOVE,"67,66;74;460"
PRMOVNXT,"66,67,81;0;0"
PROFFLINE,""
PRPROFILE,""
PRPUTNXTSLOT,""
PRPUTWAFER,""
PRREADID,""
PRRELMOVE,"65,67;74;0"
PRRELOAD,""
PRRELRETURN,"65,67;74;0"
PRREQWAFERINFO,""
PRRETURNERRORMESSAGE,""
PRSERIALPOLL,""
PRSETDIAM,"98;99;0"
PRSETDIESIZE,"98;99;0"
PRSETFLAT,"98;99;0"
PRSETMATRIX,""
PRSETMODE,"98;99;0"
PRSETMPROBE,""
PRSETPIPELINE,""
PRSETQUADRANT,""
PRSETREFDIE,"98;99;0"
PRSETSKIPDIE,""
PRSETSLOTSTATUS,""
PRSETTIME,""
PRSETUNITS,""
PRSSLEARN,""
PRSSLOCATION,""
PRSSMOVE,""
PRSSMOVNXT,""
PRSTATUS,""
PRUNLOAD,"71,81,82,94;84;0"
PRWAIT,""
PRWRITEREAD,""
PRZPARAMS,""
PRZTRAVEL,""
PRABSMOVE,"65,67;74;0"
PRSMIFLOCK,"98;99;0"
PRSMIFLOCKSTATUS,""
PRPROBERSTATUS,""
PRSENSEWAFER,""
PRSTOP,"85;0;0"
PRSTART,"120;121;0"
PRWRITEREADSRQ,""
PRSETCHUCKTEMP,""
PRQUERYCHUCKTEMP,""
PRSMIFSTATUS,""
PRSMIFCLAMP,"98;99;0"
PRLOWERBOAT,""
PRCLEARALL,"71,94;0;0"
PRNEEDLECLEAN,""
PRCASSETTEMASK,""
PRGETPRODUCT,""
PRADJUSTZHEIGHT,"92;0;0"
PRQUERYZHEIGHT,""
PRSETZHEIGHT,"116;0;0"
PRREADCASSETTEID,""
PRCHECKUNSOLICITED, "97,105;0;0"
<EOLOC>
```

Figure 7-13
prbsrq_FAKE.dat (S530KTE FAKE sample)

```
#Keithley FAKE SRQ table
#Please NOTE: 'blank' lines are not permitted!!!!!!
#
# FORMAT of the file
#
# function name, "good srq list; bad srq list; known prober error numbers"
# known prober error numbers are defined by the prober per PrError()
#
# All values are decimal
#
# Examples
#
# Good srqs are 70 and 72, Bad srqs are 76
#
# PRAUTOALIGN,"70,72;76"
#
# No good srqs , Bad srqs are 76
#
# PRAUTOALIGN,";76"
#
# Good srqs are 70 and 72, no Bad srqs
#
# PRAUTOALIGN,"70,72;"
#
# No SRQs associated with this function
#
# PRAUTOALIGN,""
#
#
Version,1.0
File,/opt/ki/dat/prbsrq_FAKE.dat
Date,
Id,
Comment,
<EOH>
PRAUTOALIGN,"1,2;3,4;5"
PRBEGINPROBE,""
PRCASSETTEMAP,""
PRCHECKOPTIONS,""
PRCHUCK,"67,68;0;0"
PRCLEARPIPELINE,""
PRDISABLETRANSLOG,""
PRENABLETRANSLOG,""
PRERROR,""
PRGETNXTWAFER,"70;84,99;0"
PRGETWAFER,"70;84,99;0"
PRINIT,""
PRINK,"66,67,69,80,81;0;0"
PRLEARN,""
PRLOAD,"70,71,72,82,84,92,94;0;1057,1181,1184,1860"
PRLOADFAILURE,""
PRLOADPRODUCT,"98;99;0"
PRMOVE,"67,66;74;460"
PRMOVNXT,"66,67,81;0;0"
PROFFLINE,""
PRPROFILE,""
PRPUTNXTSLOT,""
PRPUTWAFER,""
PRREADID,""
PRRELMOVE,"65,67;74;0"
PRRELOAD,""
PRRELRETURN,"65,67;74;0"
PRREQWAFERINFO,""
PRRETURNERRORMESSAGE,""
PRSERIALPOLL,""
PRSETDIAM,"98;99;0"
PRSETDIESIZE,"98;99;0"
PRSETFLAT,"98;99;0"
PRSETMATRIX,""
PRSETMODE,"98;99;0"
PRSETMPROBE,""
PRSETPIPELINE,""
PRSETQUADRANT,""
PRSETREFDIE,"98;99;0"
PRSETSKIPDIE,""
PRSETSLOTSTATUS,""
PRSETTIME,""
PRSETUNITS,""
PRSSLEARN,""
PRSSLOCATION,""
PRSSMOVE,""
PRSSMOVNXT,""
PRSTATUS,""
PRUNLOAD,"71,81,82,94;84;0"
PRWAIT,""
PRWRITEREAD,""
PRZPARAMS,""
PRZTRAVEL,""
PRABSMOVE,"65,67;74;0"
PRSMIFLOCK,"98;99;0"
```

Figure 7-13 (continued)
prbsrq_FAKE.dat (S530KTE FAKE sample)

```
PRSMIFLOCKSTATUS,""
PRPROBERSTATUS,""
PRSENSEWAFER,""
PRSTOP,"85;0;0"
PRSTART,"120;121;0"
PRWRITEREADSRQ,""
PRSETCHUCKTEMP,""
PRQUERYCHUCKTEMP,""
PRSMIFSTATUS,""
PRSMIFCLAMP,"98;99;0"
PRLOWERBOAT,""
PRCLEARALL,"71,94;0;0"
PRNEEDLECLEAN,""
PRCASSETTEMASK,""
PRGETPRODUCT,""
PRADJUSTZHEIGHT,"92;0;0"
PRQUERYZHEIGHT,""
PRSETZHEIGHT,"116;0;0"
PRREADCASSETTEID,""
PRCHECKUNSOLICITED, "97;0;0"
<EOLOC>
```

T19S GPIB (T19S driver)

Figure 7-14
T19S prbcnfg.dat (S530KTE sample)

```
# prbcnfg.dat  - EXAMPLE Prober Configuration File for T19S Prober
#
# The following tag, "PRBCNFG", is used by the engine in order to determine
# the MAX number of SLOTS and CASSETTES for a given prober at runtime.
#
<PRBCNFG>
#
# for OPTIONS "" == NULL, max 32 chars in string
#
# Example
#  01234567890
#PROBER_1_OPTIONS=1,1,1,1,1,1
#
#
#OcrPresent
#AutoAlnPresent
#ProfilerPresent
#HotchuckPresent
#HandlerPresent
#Probe2PadPresent
#
# Configuration for direct GPIB probers (S530):
#  T19S
#
PROBER_1_PROBTYPE=T19S
PROBER_1_OPTIONS=0,0,0,0,1,0
PROBER_1_IO_MODE=GPIB
PROBER_1_GPIB_UNIT=0
PROBER_1_GPIB_SLOT=1
PROBER_1_GPIB_ADDRESS=5
PROBER_1_GPIB_WRITEMODE=8
PROBER_1_GPIB_READMODE=10
PROBER_1_GPIB_TERMINATOR=10
PROBER_1_TIMEOUT=300
PROBER_1_SHORT_TIMEOUT=5
PROBER_1_MAX_SLOT=25
PROBER_1_MAX_CASSETTE=1
#
```

8
Prober AddIn Library

Overview

This AddIn library contains additional functions performed by the prober. The functions available on each prober are limited by the prober's capability (hardware and software). For example, the SofTouch AddIn function is only available for probers using the P8, TSK9, and EG40
drivers.

SofTouch

The SofTouch AddIn provides for automatic contact detection through use of the AutoZ function. In order to automatically detect contact with the wafer, the SofTouch function allows the user to define a contact test and configure the test environment. The following drivers use this function: P8, TSK9, and EG40.

To use the AutoZ function, create a global data file and also select a test function. The global data file provides for data definitions and variable types while the *test function* provides a means to determine electrically if contact has been made.

AutoZ will return (terminate) if contact is detected, the user (or prober) defined z max is exceeded, user library is invalid, or for a prober error.

Syntax

```
AutoZ(double Z_increment);
```

Parameters

Z_increment (double) — the number of Z units (microns or tenths of mils) to move the Z stage closer to the contact position between each execution of the user defined test. Define according to the specific prober (see "Prober specifics," page 8-4, for details).

Error codes

1. PR_MAX_Z_CT_EXCEEDED
2. NO_AUTOZ_LICENSE
3. KI_ABORT
4. AUTOZ_NOT_ON_PROBER
5. KI_ABORT_AUTO_Z
6. INVALID_AUTOZ_LIB
7. INVALID_AUTOZ_FUNC
8. INVALID_USER_FUNCTION
9. INVALID_RETURN_USER_FUNC
10. PR_AUTOZ_COMPLETE
11. PR_OK

AutoZ KTE setup procedure

NOTE *Refer to the KTE Software Manual (Document Number: KTE-900-01) for additional information on KTE software (e.g., KULT, global data files, etc.).*

The following steps outline the procedure for setting up AutoZ.

1. Using KULT, create a library with user test function (example provided in $KIHOME/src/AutoZ, see readme.txt).
   User test function must return 0 (failure to detect contact), 1 (contact successfully detected), or KI_ABORT_AUTO_Z (fatal condition stop AutoZ) defined in prb_msg.h.
2. Setup and configure the prober using the supplied documentation (see the specific prober appendix for details).
3. Create global data file. An example is provided in $KIHOME/src/AutoZ, see readme.txt.
4. Call AutoZ at the correct UAP point (see "Prober specifics," page 8-4, for details).
5. Add the Global Data File (*.gdf) to the Cassette Plan File (*.cpf).

Selecting a test function

The test functions success is interpreted as contact with the wafer. When selecting the test function, keep in mind the *test function* can be any function that can perform a test and return one of three responses: 1, 0, or KI_ABORT_AUTO_Z.

1 The function called performed the test with satisfactory results (contact has been made).

0 The function called performed the test with un-satisfactory results (contact has not been made).

KI_ABORT_AUTO_Z indicates conditions that are severe enough to stop the AutoZ function.

Also make sure the library containing the function is available. The test function, as well as the library, are specified in the global data file.

NOTE *The test function can not accept any input parameters.*

Global data file

Figure 8-1 contains a sample GDF. Use this file to select the test function's library (AutoZLibrary), the test function (AutoZFunction), and to limit the number of units the Z stage will move past the previous Z contact position (AutoZMax).

NOTE *AutoZ will return if the library selected does not contain the selected test (in other words, if the user library is invalid).*

Figure 8-1
Sample global data file

```
#Keithley Global Data Definition File
Version,1.1
File,$KIHOME/plans/AutoZ.gdf
Date,02/22/2001
Id,
Comment,
RevID,$Revision$
<EOH>
AutoZLibrary,CHAR_P,AutoZTest
AutoZFunction,CHAR_P,rescon
AutoZMax,DOUBLE,5.0
<EOGDF>
```

Auto Z Terminating conditions

The AutoZ function will return in one of the following ways:

Good

User function returns 1 (one) — Z contact was made, testing proceeds.

Error

- User function returns KI_ABORT _AUTO_Z, defined in prb_msg.h (fatal)
- User defined Z upper limit exceeded
- Prober defined Z upper limit exceeded
- Prober error

Prober specifics

Each prober has specific valid operation parameters. This includes but is not limited to:

- When the AutoZ function is valid (range of UAPs)
- Valid parameters
- GDF description

The specific parameters follow, arranged by prober.

EG prober specifics

- AutoZ is valid at the following UAPs:
  1. UAP_POST_INITIAL_WAFER_LOAD, perform AutoZ after first wafer load
  2. UAP_WAFER_PREPARE, start processing next wafer plan
  3. UAP_VALIDATE_OCR, after OCR and before wafer ID is logged to data file
  4. UAP_WAFER_BEGIN, start executing wafer plan
  5. UAP_SITE_CHANGE, start of next site processing
  6. UAP_SUBSITE_CHANGE, start of next subsite processing
  7. UAP_TEST_BEGIN, start of next ktm processing
  8. UAP_TEST_END, end of processing a ktm
  9. UAP_TEST_DATA_LOG, after test data has been logged
  10. UAP_HANDLE_ABORT, processing an abort condition
  11. UAP_SUBSITE_END, end of current sub-site processing
  12. UAP_SITE_END, end of current site processing
  13. UAP_WAFER_END, end of processing a wafer plan
- Z resolution: 1 tenth of a mil

TEL prober specifics

- AutoZ is valid at the following UAPs
  1. UAP_POST_INITIAL_WAFER_LOAD, perform AutoZ after first wafer load
  2. UAP_WAFER_PREPARE, start processing next wafer plan
  3. UAP_VALIDATE_OCR, after OCR and before wafer ID is logged to data file
  4. UAP_WAFER_BEGIN, start executing wafer plan
  5. UAP_SITE_CHANGE, start of next site processing
  6. UAP_SUBSITE_CHANGE, start of next subsite processing
  7. UAP_TEST_BEGIN, start of next ktm processing
  8. UAP_TEST_END, end of processing a ktm

9. UAP_TEST_DATA_LOG, after test data has been logged
10. UAP_HANDLE_ABORT, processing an abort condition
11. UAP_SUBSITE_END, end of current sub-site processing
12. UAP_SITE_END, end of current site processing
13. UAP_WAFER_END, end of processing a wafer plan

- Z resolution – 5 microns

TSK prober specifics

- AutoZ is only valid at the following UAP
  1. UAP_POST_INITIAL_WAFER_LOAD, perform AutoZ after first wafer load
- Z resolution – 1 micron

Pseudo code

Figure 8-2 illustrates the flow of events and their hierarchical structure associated with the AutoZ init function:

Figure 8-2
AutoZinit function

```
AutoZinit function

check for license

confirm autoZ is enabled on the prober

attempt to get AutoZ library from the datapool
    if NULL return error

attempt to get AutoZ function from the datapool
    if NULL return error

get user defined library from the datapool item
    if NULL return error

get user defined function from the datapool item
    if NULL return error

attempt to get AutoZ maximum Z limit from the datapool
    if NULL use default DEFAULT_MAX_Z (defined in prb.h)

open user defined contact test function
    if open failure return error
    else return success
```

Figure 8-3 illustrates the flow of events and their hierarchical structure associated with the AutoZ function:

Figure 8-3
AutoZ function

```
AutoZ function

call AutoZinit

if license not available
    return error

if auto z not enabled on prober
    return error

prchuck up
    if error return and display gui (permits aborting ktxe)
        while no errors execute user defined test function

        confirm return from user test function is either 1 or 0 or KI_ABORT_AUTO_Z
            if return from user test function is not 1, 0 or KI_ABORT_AUTO_Z return
                and display gui (permits aborting ktxe)

            if return from user test function is 1
                set current z height as contact height

            if set z height error return and display gui (permits aborting ktxe)
                else return PR_AUTOZ_COMPLETE (defined in prb.h)

        if no errors
                adjust Z height

        if error return and display gui (permits aborting ktxe)
                increment internal Z movement counter

        if internal Z movement counter exceeds user defined max return and display
            gui (permits aborting ktxe)
            end while

        if errors display gui (permits aborting ktxe)
return
```

9
Prober I/O

Summary of usage

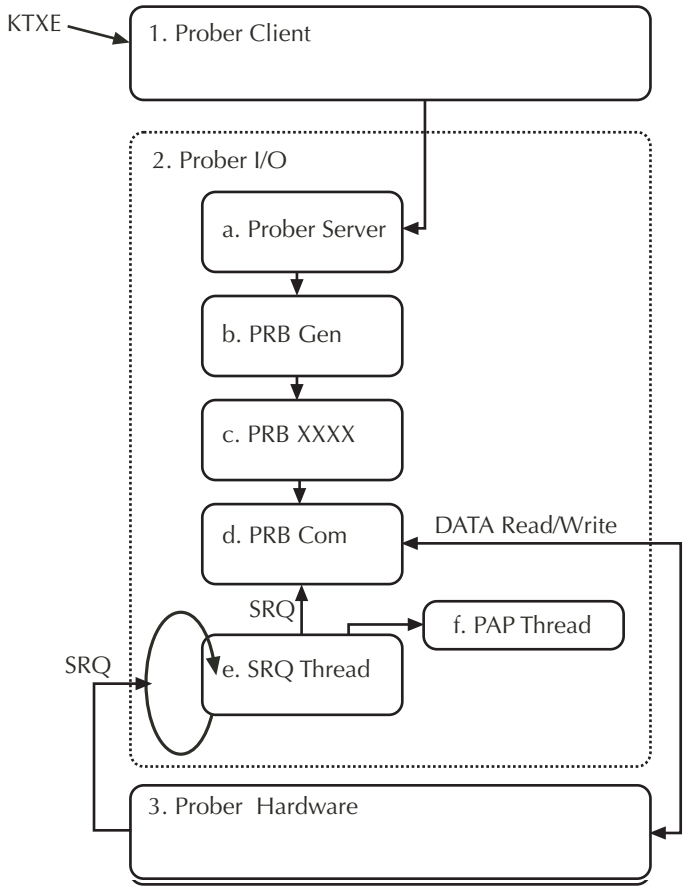
Prober I/O provides a means to quickly launch prober specific processes for unsolicited SRQs.

Prober I/O description

NOTE *Use the procedure at the end of this section to install the GPIB board and software before installing/upgrading the KTE software.* *(See "GPIB adaptor installation" on page 9-9.)*

For systems with Prober I/O functionality (KTE versions 5.1.0 and greater), prober control is through the workstation (as opposed to control through the S530 tester hardware as is the case in earlier versions of the S530). Controlling the prober through the workstation provides a specific GPIB interface for prober control. A National Instruments USB-GPIB adaptor is utilized for communications. The Prober I/O task monitors a message queue for prober commands to process. Also, command status is returned to the command through a message queue.

Figure 9-1
Prober I/O overview



1. Prober Client — Both a client-side routine and a server-side routine can exist for each prober command to be implemented.
Refer to the end of this section for details about prober commands.

2. Prober I/O
a. Prober Server — Refer to the end of this section for details about the prober server.
b. PRB Gen — Generic prober library provides the interface to the prober specific functions.
c. PRB XXXX — This library contains prober specific functions. The functions utilize the prober communication library to perform prober specific I/O.
d. PRB Com — Prober communication support library will perform file, GPIB, and serial I/O.
e. SRQ Thread — A thread that accepts every SRQ asserted and process the SRQ based on the current state. Refer to the following Active and Idle examples.
f. Pap Thread — A thread that executes if an unsolicited SRQ is received and the PAP is configured. Refer to the "Prober I/O Access Point (PAP) routines" section.

3. Prober Hardware
The physical prober hardware.

The SRQ thread determines how to handle the SRQ based on the state (active or idle) and whether the SRQ is unsolicited or not. Refer to the following pages for examples of operation (the TSK9 prober driver is used in the examples).

Example 1: Idle TSK prober

NOTE *For this example, assume that the prober is idle when material arrives and that the SRQ 97 (decimal) has been sent.*

Since the prober is idle, the prober driver is not running. The event is processed by the SRQ thread. $KIDAT/prbsrq_*probername*.dat contains a list of SRQs (good, bad, error number, and unsolicited). The SRQ thread parses this list. If SRQ 97 is listed as an unsolicited SRQ, the prober specific processes for SRQ 97 are then accomplished. (This is another thread that gets launched via the PAP method — See "Prober I/O Access Point (PAP) routines" on page 9-5.)

1. The SRQ is checked against the unsolicited list.
2. If SRQ is unsolicited:
   - If a PAP (Prober Access Point) exists for the SRQ, then the PAP code is executed.
   - Else, ignore the SRQ.
3. If SRQ is not unsolicited:
   - Ignore SRQ.

Example 2: Active TSK prober

NOTE *For this example, assume that the prober is idle when material arrives and that the SRQ 97 (decimal) has been sent.*

Since the prober is active, the prober driver is running and the event has to be processed from the running prober driver. When an SRQ is trapped:

1. The SRQ is checked against the unsolicited list.
2. If SRQ is unsolicited:
   - If a PAP (Prober Access Point) exists for the SRQ, then the PAP code is executed.
   - Else, ignore the SRQ.
3. If SRQ is not unsolicited:
   - Make the SRQ available to the calling function.

NOTE *A separate thread launches to process the PAP.*

Prober I/O configuration

New environment variables used

The prober I/O uses the KI_PROBERIO_INI environment variable to point to a .ini file that contains configuration information. Table 9-1 contains a sample .ini file.

Table 9-1
Sample .ini file

```
;
; Setup information for $KIBIN/proberio.
; NUM_TEST_STATIONS determines how many instances of proberio to start/use
; The KI_TEST_STATION Environment variable can be used to select the desired
; test station to connect with

[PROBERIO_SETUP]
NUM_TEST_STATIONS=1
QMO_NUMBER=nnnn

; section for test station #1
[PROBERIO_1]
GROUP_NAME=mbx
HOST_NAME=
TEST_STATION=1

; For systems with 300mm SECS/GEM Automation, specify here
; the PRBHOST to prober HSMS communications parameters.
[HSMS_1]
IP=nnn.nnn.nnn.nnn
PORT=nnnn
DEVICE_ID=n
GEM_PRB_TYPE=SIM
AUTO_START=DISABLED

; Proberio User access point definitions go here if needed.
;
; format:  libraryName,functionName
; PLEASE NOTE: 'functionName' must accept a single 'char* input' argument
;              This argument is supplied by the proberio process and not
;              specified in this file.
;
; Unsolicited SRQ function list format example for SRQ 97:
; PAP_SRQ_97=library_name,module_name
; List of SRQ functions should be defined between the START and END tags below
;
[PROBERIO_AP_1]
;PAP_STARTUP=library_name,module_name
;PAP_SHUTDOWN=library_name,module_name
;START_OF_SRQ_LIST
;PAP_SRQ_108=prbP8,setAutoZflag
;PAP_SRQ_105=prbTSK9,setAutoZflag_TSK9
;END_OF_SRQ_LIST


; section for test station #2 (S400 only )
[PROBERIO_2]
GROUP_NAME=mbx
HOST_NAME=
TEST_STATION=2

[HSMS_2]
IP=nnn.nnn.nnn.nnn
PORT=nnnn
DEVICE_ID=n
GEM_PRB_TYPE=SIM
AUTO_START=DISABLED

[PROBERIO_AP_2]
;PAP_STARTUP=library_name,module_name
;PAP_SHUTDOWN=library_name,module_name
;START_OF_SRQ_LIST
;END_OF_SRQ_LIST
```

Definitions for Table 9-1:

- GROUP_NAME — Used to connect to the Datahub. Default value is 'mbx.'
- HOST_NAME — Used to connect to the Datahub. Default is blank which indicates local host.
- TEST_STATION — Indicates the test station number to connect with. Defaults to 1.
- QMO_NUMBER — QMO number of the tester. This number along with the test station number is used to create a unique key for communication channels to/from the 'proberio' process.

The 'select_tester' script will set the KI_PROBERIO_INI environment variable when a tester is selected. The KTE installation process will create a $KIDAT/proberio_**nnnn**.ini file based upon the user's responses.

Prober I/O Access Point (PAP) routines

PAP_STARTUP — This routine executes during prober I/O startup. STARTUP occurs just before the prober I/O process enters the main execution loop.

PAP_SHUTDOWN — This routine executes at the beginning of the prober I/O shut-down process.

The above routines can be used to gather and report prober status information to the Automation host.

PAP_SRQ_nn — This routine executes when an unsolicited SRQ number nn is received. An entry is required for each unsolicited SRQ to be trapped and processed by prober I/O.

These items are defined in the **proberio_nnnn.ini** file. There is a limit of one function per access point or SRQ value. A single char * input argument is accepted for these routines. The prober I/O process supplies the argument for the PAP execution.

NOTE *The PAP routines are created with KULT and must be built with the 'Hide library' switch ON and 'UAP Library' switch OFF.*

Prober I/O commands

The prober I/O process will create the prbIOreq_t_nnn mailbox and register to receive commands from this mailbox. Table 9-2 contains a listing of the commands.

where: t = test station
nnnn = qmo number

Table 9-2
Prober I/O commands

Command	Description
"get prober control"	This command will allocate a prober control channel to the caller (msgQ).
"release prober"	This command will release the prober control channel so another client may connect.
"get material handling control"	This command will allocate a material handling control channel to the caller (mailbox).
"release material handling"	This command will release the material handling control channel so another client may connect.
"shutdown proberio"	This will cause the prober I/O process to terminate.
"configure logging--level--FileName"	This command is used to configure the prober I/O logging process. There are 4 levels of logging available: PRBIO_ERROR=1 PRBIO_WARNING=2 PRBIO_EVENT=4 PRBIO_DEBUG=8 The level value in the command is a bit-wise 'or' of the above values, depending upon what type(s) of messages to log. The FileName value in the command is where to log the messages. If no filename is specified, "/dev/tty" is used.

Command	Description
"reconfigure proberio"	This command will cause the prober I/O process to re-read the $KI_PRB_CONFIG file and reconfigure the prober driver options and values. PLEASE NOTE: This command does not detect changes to the KI_PRB_CONFIG environment variable, only changes to the file identified by the KI_PRB_CONFIG environment variable.

“mbx_do_xact”

This command is used from the TCL command window to send commands to the prober I/O process. The syntax of this command is as follows:

`mbx_do_xact prbIOreq_t_nnnn "COMMAND"`

where: t = test station
nnnn = qmo number
COMMAND = one of the command strings described (For example, “release prober”).

prbsrq_xxxx.dat

This file may contain a list of good SRQs, a list of bad SRQs, and a list of error codes for each available command for a given prober driver. This file also contains a function placeholder called PRCHECKUNSOLICITED. This function has a list of good SRQs that are valid unsolicited events that will be trapped. The purpose of this file is to allow an unsolicited SRQ to have specific processing associated with its value.

Example (TSK prober driver)

In this example, the load command is issued to the prober. A FOUP arrives on an empty port on the prober. The prober asserts unsolicited SRQ 97. The SRQ 97 is a valid unsolicited SRQ (based on the prbsrq_TSK9.dat). The prober is queried to ascertain the FOUP status. The status is then communicated to the datahub and ultimately to the host. Meanwhile, the SRQ thread continues to wait for the SRQ from the Load command. Once the load SRQ is received testing continues.

- Load command issued to the prober
- SRQ 97 returned (FOUP arrival)
- SRQ 97 exists in the unsolicited list
- TSK specific thread created to perform a specific task, after processing the thread exits.

Refer to the PRCHECKUNSOLICITED,"97;0;0" line contained in Table 9-3.

Table 9-3
Sample prbsrq_TSK9.dat

```
#Keithley TSK9 SRQ table
#Please NOTE: 'blank' lines are not permitted!!!!!!
#
# FORMAT of the file
#
# function name, "good srq list; bad srq list; known prober error numbers"
# known prober error numbers are defined by the prober per PrError()
#
# All values are decimal
#
# Examples
#
# Good srqs are 70 and 72, Bad srqs are 76
#
# PRAUTOALIGN,"70,72;76"
```

Table 9-3 (cont.)
Sample prbsrq_TSK9.dat

```
#
# No good srqs , Bad srqs are 76
#
# PRAUTOALIGN,";76"
#
# Good srqs are 70 and 72, no Bad srqs
#
# PRAUTOALIGN,"70,72;"
#
# No SRQs associated with this function
#
# PRAUTOALIGN,""
#
#
Version,1.0
File,/opt/ki/dat/prbsrq_TSK9.dat
Date,
Id,
Comment,
<EOH>
PRAUTOALIGN,"1,2;3,4;5"
PRBEGINPROBE,""
PRCASSETTEMAP,""
PRCHECKOPTIONS,""
PRCHUCK,"67,68;0;0"
PRCLEARPIPELINE,""
PRDISABLETRANSLOG,""
PRENABLETRANSLOG,""
PRERROR,""
PRGETNXTWAFER,"70;84,99;0"
PRGETWAFER,"70;84,99;0"
PRINIT,""
PRINK,"66,67,69,80,81;0;0"
PRLEARN,""
PRLOAD,"70,71,72,82,84,92,94,105;0;1057,1181,1184,1860"
PRLOADFAILURE,""
PRLOADPRODUCT,"98;99;0"
PRMOVE,"67,66;74;460"
PRMOVNXT,"66,67,81;0;0"
PROFFLINE,""
PRPROFILE,""
PRPUTNXTSLOT,""
PRPUTWAFER,""
PRREADID,""
PRRELMOVE,"65,67;74;0"
PRRELOAD,""
PRRELRETURN,"65,67;74;0"
PRREQWAFERINFO,""
PRRETURNERRORMESSAGE,""
PRSERIALPOLL,""
PRSETDIAM,"98;99;0"
PRSETDIESIZE,"98;99;0"
PRSETFLAT,"98;99;0"
PRSETMATRIX,""
PRSETMODE,"98;99;0"
PRSETMPROBE,""
PRSETPIPELINE,""
PRSETQUADRANT,""
PRSETREFDIE,"98;99;0"
PRSETSKIPDIE,""
PRSETSLOTSTATUS,""
PRSETTIME,""
PRSETUNITS,""
PRSSLEARN,""
PRSSLOCATION,""
PRSSMOVE,""
PRSSMOVNXT,""
PRSTATUS,""
PRUNLOAD,"71,81,82,94;84;0"
PRWAIT,""
PRWRITEREAD,""
PRZPARAMS,""
PRZTRAVEL,""
PRABSMOVE,"65,67;74;0"
PRSMIFLOCK,"98;99;0"
PRSMIFLOCKSTATUS,""
PRPROBERSTATUS,""
PRSENSEWAFER,""
PRSTOP,"85;0;0"
PRSTART,"120;121;0"
PRWRITEREADSRQ,""
```

Table 9-3 (cont.)
Sample prbsrq_TSK9.dat

```
PRSETCHUCKTEMP,""
PRQUERYCHUCKTEMP,""
PRSMIFSTATUS,""
PRSMIFCLAMP,"98;99;0"
PRLOWERBOAT,""
PRCLEARALL,"71,94;0;0"
PRNEEDLECLEAN,"89;0;0"
PRCASSETTEMASK,""
PRGETPRODUCT,""
PRADJUSTZHEIGHT,"92;0;0"
PRQUERYZHEIGHT,""
PRSETZHEIGHT,"116;0;0"
PRREADCASSETTEID,""
PRCHECKUNSOLICITED,"97;0;0"
<EOLOC>
```

FAKE probers, no probers, and real probers

Prober actions depend on the driver used. Setting KTE to “No Prober” means the prober driver is absent. This disables all prober activity and KTE will not call the Prober I/O. Either a real prober driver, such as the TSK9 or FAKE (which is the driver for a simulated prober), or the presence of a prober driver, allows KTE to make calls to the Prober I/O.

PrbGen Client/Server/IO

PRBGEN_SERVER/PRBGEN_CLIENT

The libprbgen.so and libprbgen_server.so libraries work together to implement RPC-type transactions for prober operation. Both a client-side routine and a server-side routine must exist for each prober command to be implemented. If new commands are to be added, existing routines can be used as examples.

PRBGEN_IO Modification

The prbgen_io library contains the routines to create a prober command message and send this message to the prober specific driver. The $KIHOME/src/prb/gen_io directory contains the sources for the prbgen_io library which is used by the prbgen_server library. The prbgen_io library is a KULT library and can be modified/extended using KULT.

NOTE *Only Keithley personnel should make modifications to this library*

Adding new commands to the prober driver

If a new command is added to the prober driver:

1. Add a command to the prober driver library itself. Refer to Section 6.
2. Add the prbgen_io wrapper for the new command. This wrapper branches to the prober specific function containing the arguments for the new function. The prober specific function communicates with the prober.

PRBGEN_SERVER Modification

The libprbgen_server.so library contains the routines to receive and process the prober commands requested by the client application. The $KIHOME/src/prb/gen_server directory contains the sources for the libprbgen_server.so library which is used by the prober I/O process.

NOTE *Only Keithley personnel should made modifications to this library.*

To customize or extend this library, use the following procedure:

1. Make the appropriate modifications to the source files. You may remove existing files or add new files.
2. Make sure that SRCS line in Makefile contains all the 'c' files that you want to build in the library.
3. Type "make." This will rebuild 'libprbgen_server.so'.
4. Make a backup of the existing $KILIB/libprbgen_server.so file.
5. Copy the new 'libprbgen_server.so' file to $KILIB.

PRBGEN_CLIENT Modification

The libprbgen.so library contains routines to create a prober command message and send this message to the prober I/O process. The $KIHOME/src/prb/gen directory contains the sources for the prbgen (client) library which is used by KTXE. The prbgen (client) library is a KULT library and can be modified/extended using KULT.

NOTE *Only Keithley personnel should made modifications to this library.*

Adding new commands to the prober driver

If a new command is added to the prober driver:

1. Add a command to the prober driver library itself. Refer to Section 6.
2. Add the prbgen (client) wrapper for the new command. This wrapper creates a message containing the command and arguments for the new function. The wrapper sends the command to the prober I/O process and receives the response.
3. Add the libprbgen_server.so wrapper to the new command. This wrapper extracts the command and arguments for the prober command and calls the appropriate function. A response message is created with the status of the command and this response is sent back to the prbgen (client) calling routine.

GPIB adaptor installation

Controlling the prober through the workstation requires a dedicated GPIB interface for prober control. This section provides information on installing and configuring the USB-GPIB interface adaptor.

Adaptor installation

Plug the USB-GPIB adaptor into an unused USB port on the workstation computer.

NI-488.2 software installation procedure

1. Log in as the ‘root’ user.
2. Place the NI driver file into the /usr/local/Tools directory.

3. ‘cd/usr/local/Tools’
4. ‘tar zxf ni488-2.9.0f0.tar.gz’
5. ‘cd ni488-2.9.0f0’
6. ‘./INSTALL’
7. Accept the default prompts and location for installation.
8. Test the software installation:
   a. ‘/usr/local/natinst/ni4882/bin/gpibtsw/
   b. If the test fails, click the ‘retest’ button.
   c. If the test fails again, there is a problem with the software installation.
9. Configure the driver parameters:
   a. ‘/usr/local/natinst/ni4882/bin/gpibtsw/
   b. Select the GPIB1 interface (USB-GPIB adaptor).
   c. Click the the ‘Properites’ button.
   d. Disable “Automatic Serial Polling.”

A
EG 2001/2010/3001/4085
Prober Information

Prober start-up

EG 2001/2010/3001/4085 probers have three controls associated with prober start-up: ON, OFF, and EMERGENCY OFF. To power up the prober, press the ON pushbutton. To remove power from the prober, press either the OFF or the EMERGENCY OFF pushbuttons as appropriate.

Prober setup and initialization

NOTE *Refer to the specific Prober requirements contained in Section 2 for the software version used to verify the prober driver and configuration contained in this section.*

There are two types of display and keyboard combinations. The text display uses a keyboard and a character based CRT (refer to). The touch-screen display uses a QWERTY-style keyboard and a GUI display that can be used to enter functions and settings. Although the styles of input and display differ, the configuration information is identical.

To allow the prober to communicate properly with the system, make sure communication and other settings are correctly configured. With the prober running:

Press set mode. 

Open I/O Control screen using one of the following methods:
Method 1 (for Text display), set `LINE = 07` and press enter.
Method 2 (for Touch-screen display), first press F5/Setup and then press E/EXIO.
Check the prober's I/O communication configuration and make changes as needed.
SERIAL probers: use Table A-1.
GPIB probers: use Table A-2.
Open the prober's enhanced external I/O screen. To open the prober's external I/O screen use one of the following methods:
Method 1 (for Text display), set `LINE = 02` and press enter.
Method 2 (for Touch-screen display), press M/SetMode.
Check the prober's I/O configuration. Make sure the prober's external I/O communication configuration matches Table A-3. Make any changes as needed.
Press the Enter key until the main screen appears.
Press set option. 

Check the prober's option configuration and make changes as needed. Make sure the prober's configuration matches Table A-4.
Press set mode. 

Open the probing mode screen using one of the following methods:
Method 1 (for Text display), set `LINE = 03` and press enter.
Method 2 (for Touch-screen display), press F7/Product.

Check the prober's quadrant setup and make changes as needed. Make sure the prober's configuration matches Table A-5. To change the prober's quadrant setup, use one of the following methods:

Method 1 (for Text display):
Enter the line number and make changes as required (refer to Table A-5).

Method 2 (for Touch-screen display):
1. Press C/Cntlmap.
2. Press S/Stepping.
3. Highlight the coordinate quadrant.
4. Press F9/Change.
5. Highlight 3 (SW).
6. Press F9/Change.
7. Press F10/OK.
8. Press M/Mode.
9. Highlight the coordinate quadrant.
10. Press F9/Change.
11. Highlight 3 (SW).
12. Press F9/Change.
13. Press F10/OK.

To complete the change, press F10.

NOTE *Settings contained in this document cover prober models EG2001/2010/3001/4085. Refer to the Electroglas documentation for specific instructions on modifying prober settings.*

Make sure to save the setting changes before shutting down the prober (e.g., press the S/Save button or the S-key on the keyboard). Also make a backup-copy of the settings on a floppy (recommended).

Table A-1
Serial settings

Line	Serial setting description	Value	Remarks
01	I/O Protocol	Enhanced	Selects enhanced I/O protocol.
02	External I/O Mode	MENU	See Table A-3.
03	I/O Port	SER	Selects serial RS-232 interface port.
04	Baud Rate	9600	Establishes serial interface baud rate.
05	Transmit delay in ms	0	Disables intercharacter delay between transmitted characters.
06	Terminator	CR/LF	Establishes terminator for messages between the host computer and the prober.
07	Parity	NO	Used to determine the characteristics of the parity check.
08	Number of data bits	8	Establishes the code group for each character transmission.
09	Timeout timer 1 in ms	5000	Controls data received from/transmitted to the prober (value in milliseconds).
10	Timeout timer 2 in ms	5000	Same as Timeout timer 1.

Table A-2
GPIB settings

Line	GPIB setting description	Value	Remarks
01	I/O Protocol	Enhanced	Selects enhanced I/O protocol.
02	External I/O Mode	MENU	See Table A-3.
03	I/O Port	GPIB-SP	Selects GPIB (general purpose interface bus) serial poll mode. (Do NOT use GPIB-PP mode. GPIB-PP would select parallel poll mode).
05	GPIB Address	5	Selects GPIB address.
06	Terminator	CR/LF	Establishes terminator for messages between the host computer and the prober.
07	GPIB SRQ	ENB	Enables the GPIB SRQ function in enhanced protocol.
09	Timeout timer 1 in ms	5000	Controls data received from/transmitted to the prober (value in milliseconds).
10	Timeout timer 2 in ms	5000	Same as *Timeout timer 1.*

Table A-3
Enhanced External I/O Mode Menu settings

Line	Parameter	Value	Remarks
01	MC/MF on XY motion	ENB	Message acknowledges successful completion (MC) or failure (MF) for each command created.
02	MC/MF on Z motion	ENB	Same as *MC/MF on XY motion*.
03	MC/MF on optional dev	ENB	Same as *MC/MF on XY motion*.
04	MC/MF on rest of cmd	ENB	Same as *MC/MF on XY motion*.
05	Test Start msg	ENB	Start of each command will be output to host.
06	Test Complete msg	ENB	Completion of each command will be output to host.
07	Pattern Complete msg	DIS	Completion of probe pattern will NOT be output to host.
08	Pause/Continue msg	DIS	Probe interruption will not be acknowledged.
09	Alarm msg	DIS	All alarm messages to the host computer will be repressed.
10	Wafer complete MSG	DIS	“PC” message not sent at unload.
11	Enhanced PC msg	DIS	Message is issued when wafer is unloaded.
12	Next Page		Opens next page of menu items.
01	Enhanced TS msg	DIS	Tells tester the type of test that has been started.
02	Ugly Die report	DIS	Disables sending of message: `UDX<pos>Y<pos>B<bincode>`
03	Map Transfer retries	2	Instructs the external I/O to retry sending wafers maps 2 times (twice) automatically.
04	Send Map Coord with TS	DIS	Disables sending of message: `TSX <xdie><ydie>`
05	Send EC/BC message	DIS	Disables sending of EC/BC messages (2010 only).
06	Pause pending msg	DIS	Disables pause pending message.
07	Wafer Begin msg	DIS	Disables wafer begin message.
08	Wafer Begin delay (sec)	0	Specifies start delay (interval the prober waits before sending the first TS to the tester).

NOTE *Make sure to save the setting changes before shutting down the prober (e.g., press the S/Save button or the S-key on the keyboard). Also make a backup-copy of the settings on a floppy (recommended).*

Table A-4
Option Menu settings (GPIB/serial)

Line	Parameter	Value	Remarks
01	Auto-load switch	ENB	Activate the auto-load option menu.
02	Auto-align switch	ENB	Activate the auto-align option menu.
03	Auto-profile	ENB	Activates the auto-profiler option menu.
04	Wafer to reader	DIS	Activate the ID reader menu. This menu item is not displayed if line items 1 through 3 are enabled.
05	SECS protocol option	DIS	Activate the SECS reader menu.
06	Wafer mapping option	DIS	Disables wafer mapping menu.
07	Hot chuck option	DIS	Disables hot chuck option menu.
08	Auto temp. Compensation	DIS	Disables automatic temperature compensation.
09	Ink dot inspection	DIS	Disables ink dot inspection menu.
10	Probe mark inspection	DIS	Disables probe mark inspection.

NOTE *Make sure the probers quadrant settings match Table A-5 during setup. Also check this setting if the direction of probe-movement is not as expected.*

Table A-5
Quadrant settings (GPIB/serial)

Line	Parameter	Value	Remarks
01	Probe Quad	3	Sets the prober quadrant to quadrant III (SW)
02	Coord. Quad	3	Sets the coordinate quadrant to quadrant III (SW)

Installation verification program

Use the installation verification program to exercise prober communication functions. If this fails, the installation was not properly completed (files are missing). Use the information below to perform installation verification.

Program name: `prb-EG2X-demo.cpf`

Location: `$KIHOME/src/EXAMPLE/plans`

To run:

Start KTPM.

– Then open `$KIHOME/src/EXAMPLE/plans/prb-EG2X-demo.cpf`

Start KITT.

– Then open `$KIHOME/src/EXAMPLE/pgm/prb-demo.ktm`

Start WDU.

– Then open `$KIHOME/src/EXAMPLE/pgm/prb-demo.wdf`

EG2X driver configuration

Make sure the $KI_PRB_CONFIG file is set to the correct prober. For example, if an EG2X prober is being used, a file in $KIDAT called prbcnfg_EG2X.dat should exist. The environment variable, KI_PRB_CONFIG, should be set to $KIDAT/prbcnfg_EG2X.dat.

Command list

The following table lists commands supported by EG 2001/2010/3001/4085 probers. For detailed information about a specific command, refer to Section 5.

Table A-6
EG 2001/2010/3001/4085 Supported commands

Commands		
PrAbsMove	PrMove	PrSetPipeline
PrAutoAlign	PrProfile	PrSetQuadrant
PrCassetteMap	PrPutNxtSlot	PrSetRefDie
PrCheckOptions	PrPutWafer	PrSetSlotStatus
PrChuck	PrReadId	PrSetTime
PrClearPipeline	PrRelMove	PrSetUnits
PrError	PrRelReturn	PrStatus
PrGetNxtWafer	PrSerialPoll	PrUnLoad
PrGetWafer	PrSetDiam	PrWriteRead
PrInit	PrSetDieSize	PrWriteReadSRQ
PrLoad	PrSetFlat	PrZParams
PrLoadProduct	PrSetMode	PrZTravel

Error symbols and returned values table

After calling a prober command as a function, a value or symbol is returned. Table A-7 contains error symbols and return values associated with specific commands.

Table A-7
Error symbols and returned values

Return Value	Commands																		Error Symbol
	PrAbsMove	PrAutoAlign	PrCassetteMap	PrCheckOptions	PrChuck	PrClearPipeline	PrError[5]	PrGetWafer	PrGetNxtWafer[1]	PrInit	PrLoad	PrLoadProduct	PrMove	PrMovNxt	PrProfile	PrPutNxtSlot[2]	PrPutWafer[3]	PrReadId[4]	
0, +1	√	√	√	√	√	√		√		√	√	√			√				PR_OK
+2																			PR_MOVECOMPLETE
+4																			PR_WAFERCOMPLETE
+8																			PR_CASSETTECOMPLETE
+10											√								PR_LOTEND
+12																			PR_WAFER_REJECT_LOAD
+14																			PR_WAFER_REJECT_NOLOAD
+16																			PR_OPER_ACTION
-1001																			BAD_TST_NUM
-1002									√										BAD_CONFIG_DAT
-1003																			FEAT_NOT_SUPPORT
-1004	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	PORT_ASS_FAIL
-1005										√									SET_UNITS_FAIL
-1006																			INVAL_MODE
-1007																			CLR_WAF_MAP
-1008						√		√	√	√						√	√		SET_MODE_FAIL
-1009																			SET_DIE_FAIL
-1010										√									SET_PRESET_FAIL
-1011		√			√			√	√		√		√	√		√	√		BAD_MODE
-1012														√					TST_COMPL_FAIL
-1013	√	√	√	√	√	√	√	√	√		√		√	√	√	√	√	√	UNINTEL_RESP
-1014	√	√				√			√				√	√	√	√			MOVE_FAIL
-1015														√					UNEXPE_ERROR
-1016								√									√		LOAD_ERROR
-1017					√														BAD_CHUCK
-1018																			CHUCK_NOT_UP
-1019																			INK_FAIL
-1020				√		√	√		√									√	TIME_GONE
-1021																			BAD_LEARN_FUNCT
-1022																			SET_DIAM_FAIL
-1023																			SET_MATRIX_FAIL
-1024																			BAD_Z_PARAM
-1025																			NO_RESPONSE
-1026																			TOO_MANY_NAKS
-1027			√													√	√		INVAL_PARAM
-1028												√							ERR_LOAD_FILE
-1029																			ALIGN_FAIL
-1030																			GPIB_ERROR
-1031																			CHK_PORT_ASS
-1032																			CHK_DEV_PROT
-1033																			PR_MEM_ALLOC_ERR
-1034																			PR_NO_CASSETTE
-1035																			SET_FLAT_FAIL
-1036																			SET_REF_DIE_FAIL
-1037																			Z_PARAM_FAIL
-1038																			ERR_OPEN_PRBCNFG
-1039																			NO_CNFG_DATA
-1040																			NO_PRB_TYPE
-1041																			INVAL_PRB_MODE
-1042																			INVAL_PRB_OPTIONS
-1043																			INVAL_DEVICE_NAME
-1044																			INVAL_DEVICE_IRQ
-1045																			INVAL_BAUD_RATE
-1046																			INVAL_TIMEOUT
-1047																			INVAL_GPIB_UNIT
-1048																			INVAL_GPIB_SLOT
-1049																			INVAL_GPIB_ADDRESS
-1050																			INVAL_GPIB_WRITE_MODE
-1051																			INVAL_GPIB_READ_MODE
-1052																			INVAL_GPIB_TERMINATOR
-1053																			INVAL_KULT_PATH
-1054																			INVAL_PRB_LIB
-1055																			INVAL_PRB_CNFG_FUNC
-1056																			INVAL_PRB_MAX_SLOTS

[1]**PrGetNxtWafer:** Returns the slot number.
[2]**PrPutNxtSlot:** Returns the slot number.
[3]**PrPutWafer:** Returns the slot number.
[4]**PrReadId:** Returns the number of characters received (>0).
[5]**Prober error:** Add 1500 to the error number returned and then refer to the prober manufacturer for error handling.

Table A-7 (continued)

Error symbols and returned values

Return Value	Commands																		Error Symbol
	PrAbsMove	PrAutoAlign	PrCassetteMap	PrCheckOptions	PrChuck	PrClearPipeline	PrError[5]	PrGetWafer	PrGetNxtWafer[1]	PrInit	PrLoad	PrLoadProduct	PrMove	PrMovNxt	PrProfile	PrPutNxtSlot[2]	PrPutWafer[3]	PrReadId[4]	
-1057																			INVAL_PRB_MAX_CASSETTES
-1058																			SET_QUAD_FAIL
-1059																			NO_UNPROBED_WAFERS
-1060																			SET_TEMPERATURE_FAIL
-1061																			QUERY_TEMPERATURE_FAIL
-1081																			INVAL_PRB_P8_OPTION
-1082																			NI_UNIX_ERROR
-1083																			NI_BOARD_NOT_CIC
-1084																			NI_NO_LISTENER
-1085																			NI_BOARD_NOT_ADDRESSES
-1086																			NI_INVALID_FUNCTION_ARG
-1087																			NI_BOARD_NOT_SYS_CONTROLLER
-1088																			NI_IO_ABORTED_TIMO
-1089																			NI_BOARD_NOT_PRESENT
-1090																			NI_DMA_HW_PROBLEM
-1091																			NI_DMA_HW_TIMO
-1092																			NI_NO_CAPABILITY
-1093																			NI_FILE_SYS_ERROR
-1094																			NI_GPIB_BUS_ERROR
-1095																			NI_SRQ_BYTE_Q_OVERFLOW
-1096																			NI_SRQ_STUCK_ON
-1097																			NI_TABLE_PROBLEM
-1500 & on							√												Prober error[5]

[1]**PrGetNxtWafer:** Returns the slot number.
[2]**PrPutNxtSlot:** Returns the slot number.
[3]**PrPutWafer:** Returns the slot number.
[4]**PrReadId:** Returns the number of characters received (>0).
[5]**Prober error:** Add 1500 to the error number returned and then refer to the prober manufacturer for error handling.

Table A-7 (continued)

Error symbols and returned values

Return Value	Commands																		Error Symbol
	PrRelMove	PrRelReturn	PrSerialPoll[6]	PrSetDiam	PrSetDieSize	PrSetFlat	PrSetMode	PrSetPipeline	PrSetQuadrant	PrSetRefDie	PrSetSlotStatus	PrSetTime	PrSetUnits	PrStatus	PrUnload	PrWriteread[7]	PrZParams	PrZTravel	
0, +1	√	√		√	√	√	√	√	√	√	√	√	√	√	√		√	√	PR_OK
+2																			PR_MOVECOMPLETE
+4																			PR_WAFERCOMPLETE
+8																			PR_CASSETTECOMPLETE
+10																			PR_LOTEND
+12																			PR_WAFER_REJECT_LOAD
+14																			PR_WAFER_REJECT_NOLOAD
+16																			PR_OPER_ACTION
-1001																			BAD_TST_NUM
-1002																			BAD_CONFIG_DAT
-1003																			FEAT_NOT_SUPPORT
-1004	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	PORT_ASS_FAIL
-1005													√						SET_UNITS_FAIL
-1006							√												INVAL_MODE
-1007							√												CLR_WAF_MAP
-1008							√	√			√								SET_MODE_FAIL
-1009					√														SET_DIE_FAIL
-1010																			SET_PRESET_FAIL
-1011															√				BAD_MODE
-1012																			TST_COMPL_FAIL
-1013	√	√				√		√	√		√			√	√	√	√	√	UNINTEL_RESP
-1014	√	√																	MOVE_FAIL
-1015																			UNEXPE_ERROR
-1016																			LOAD_ERROR
-1017																			BAD_CHUCK
-1018																			CHUCK_NOT_UP
-1019																			INK_FAIL
-1020			√					√			√			√		√			TIME_GONE
-1021																			BAD_LEARN_FUNCT
-1022				√															SET_DIAM_FAIL
-1023																			SET_MATRIX_FAIL
-1024																	√		BAD_Z_PARAM
-1025																			NO_RESPONSE
-1026																			TOO_MANY_NAKS
-1027								√			√								INVAL_PARAM
-1028																			ERR_LOAD_FILE
-1029																			ALIGN_FAIL
-1030																			GPIB_ERROR
-1031																			CHK_PORT_ASS
-1032																			CHK_DEV_PROT
-1033																			PR_MEM_ALLOC_ERR
-1034																			PR_NO_CASSETTE
-1035																			SET_FLAT_FAIL
-1036																			SET_REF_DIE_FAIL
-1037																			Z_PARAM_FAIL
-1038																			ERR_OPEN_PRBCNFG
-1039																			NO_CNFG_DATA
-1040																			NO_PRB_TYPE
-1041																			INVAL_PRB_MODE
-1042																			INVAL_PRB_OPTIONS
-1043																			INVAL_DEVICE_NAME
-1044																			INVAL_DEVICE_IRQ
-1045																			INVAL_BAUD_RATE
-1046																			INVAL_TIMEOUT
-1047																			INVAL_GPIB_UNIT
-1048																			INVAL_GPIB_SLOT
-1049																			INVAL_GPIB_ADDRESS

[5]**Prober error:** Add 1500 to the error number returned and then refer to the prober manufacturer for error handling.
[6]**PrSerialPoll:** Returns the SRQ status byte (>0).
[7]**PrWriteRead:** Returns the number of characters received (>0).

Table A-7 (continued)

Error symbols and returned values

Return Value	Commands																		Error Symbol
	PrRelMove	PrRelReturn	PrSerialPoll[6]	PrSetDiam	PrSetDieSize	PrSetFlat	PrSetMode	PrSetPipeline	PrSetQuadrant	PrSetRefDie	PrSetSlotStatus	PrSetTime	PrSetUnits	PrStatus	PrUnload	PrWriteread[7]	PrZParams	PrZTravel	
-1050																			INVAL_GPIB_WRITE_MODE
-1051																			INVAL_GPIB_READ_MODE
-1052																			INVAL_GPIB_TERMINATOR
-1053																			INVAL_KULT_PATH
-1054																			INVAL_PRB_LIB
-1055																			INVAL_PRB_CNFG_FUNC
-1056																			INVAL_PRB_MAX_SLOTS
-1057																			INVAL_PRB_MAX_CASSETTES
-1058																			SET_QUAD_FAIL
-1059																			NO_UNPROBED_WAFERS
-1060																			SET_TEMPERATURE_FAIL
-1061																			QUERY_TEMPERATURE_FAIL
-1081																			INVAL_PRB_P8_OPTION
-1082																			NI_UNIX_ERROR
-1083																			NI_BOARD_NOT_CIC
-1084																			NI_NO_LISTENER
-1085																			NI_BOARD_NOT_ADDRESSES
-1086																			NI_INVALID_FUNCTION_ARG
-1087																			NI_BOARD_NOT_SYS_CONTROLLER
-1088																			NI_IO_ABORTED_TIMO
-1089																			NI_BOARD_NOT_PRESENT
-1090																			NI_DMA_HW_PROBLEM
-1091																			NI_DMA_HW_TIMO
-1092																			NI_NO_CAPABILITY
-1093																			NI_FILE_SYS_ERROR
-1094																			NI_GPIB_BUS_ERROR
-1095																			NI_SRQ_BYTE_Q_OVERFLOW
-1096																			NI_SRQ_STUCK_ON
-1097																			NI_TABLE_PROBLEM
-1500 & on																			Prober error[5]

[5]**Prober error:** Add 1500 to the error number returned and then refer to the prober manufacturer for error handling.
[6]**PrSerialPoll:** Returns the SRQ status byte (>0).
[7]**PrWriteRead:** Returns the number of characters received (>0).

B

EG4060/4080/4090/5|300 Prober Information

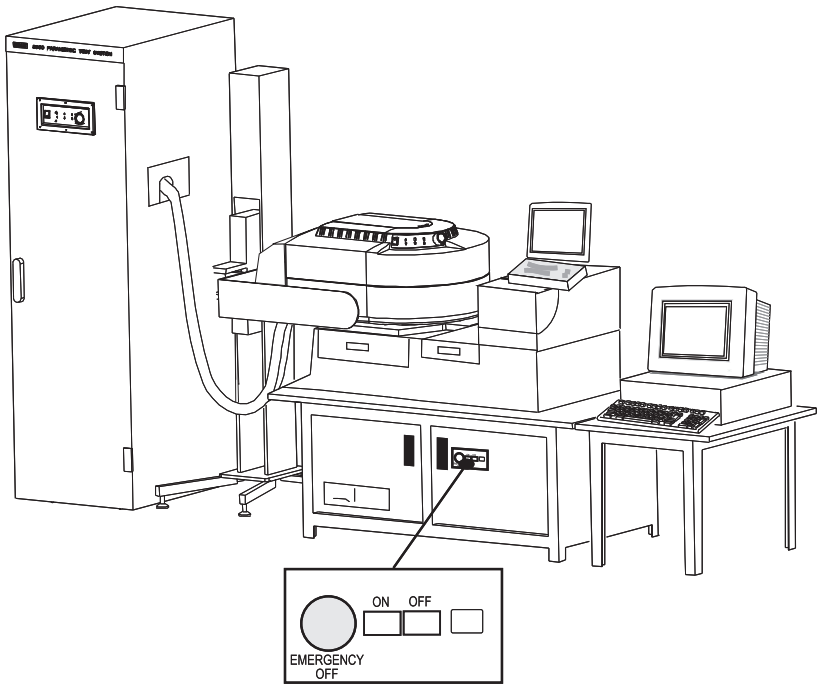
Prober GPIB connection

On Electroglas probers, make sure to use the GPIB connection labeled “IEEE-488” (do not connect to the connection labeled “Tester”). See Section 2 of this manual for additional connection information.

Prober start-up

EG 4060/4080/4090/5|300 probers have three controls associated with prober start-up: ON, OFF, and EMERGENCY OFF. Figure B-1 shows typical control location (refer to the documentation provided with the prober for specific control location). To power up the prober, press the ON pushutton. To remove power from the prober, press either the OFF or the EMERGENCY OFF pushbuttons as appropriate.

Figure B-1
Typical Electroglas prober controls



EG4080 Prober with S600 System shown above

Prober setup and initialization

NOTE *Refer to the specific Prober requirements contained in Section 2 for the software version used to verify the prober driver and configuration contained in this section.*

To allow the prober to communicate properly with the system, make sure communication and other settings are correctly configured. With the prober running...

1. Press the F5/Setup screen-button on the main window (or F5 on the keyboard). This displays the setup parameters.
2. Press the E/EXIO screen-button (or E). This opens the “Host-A: External I/O window” (Figure B-2). For an *IEEE-488 (GPIB) interface*: Change the settings to match Figure B-2 and Table B-1.
3. For a *Serial interface*: Change the settings to match Figure B-3 and Table B-2.
4. Press the M/SetMode screen-button (Figure B-3) (or M). This opens the “Enhanced External I/O Mode Menu” window (Figure B-4).
5. Change the settings to match Table B-3.
6. Press the F10/OK screen-button (Figure B-4) (or F10). This closes the window and opens the “Host-A: External I/O window”.
7. Press the S/Save screen-button (or S).
8. Press the F10/OK screen-button (or F10) to open the previous window.

NOTE *Settings contained in this document cover prober models EG4060/4080/4090/5|300. Refer to the Electroglas documentation for specific instructions on modifying the prober settings.*

Figure B-2
Sample GPIB settings window

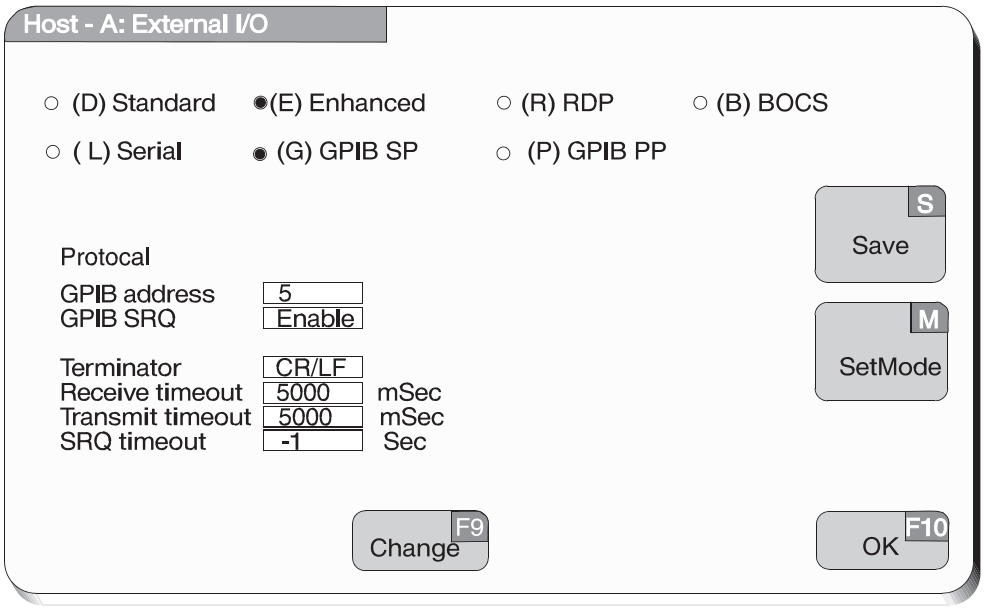


Table B-1
GPIB settings

Setting type	Parameter	Value
Host-A: External I/O settings	(D) Standard	De-Selected ◎
	(L) Serial	De-Selected ◎
	(E) Enhanced	Selected ◉
	(G) GPIB_SP	Selected ◉
	(R) RDP	De-Selected ◎
	(P) GPIB_PP	De-Selected ◎
	(B) BOCS	De-Selected ◎
Protocol settings	GPIB address:	5
	GPIB SRQ:	Enable
	Terminator	CRLF
	Receive Timeout:	5000 (msec)
	Transmit Timeout:	5000 (msec)
	SRQ Timeout:	-1 (msec)

NOTE *Make sure to save the setting changes before shutting down the prober (e.g., press the S/Save button or S on the keyboard). Also, it is recommended to make a backup-copy of the settings on a floppy.*

Figure B-3
Sample serial settings window

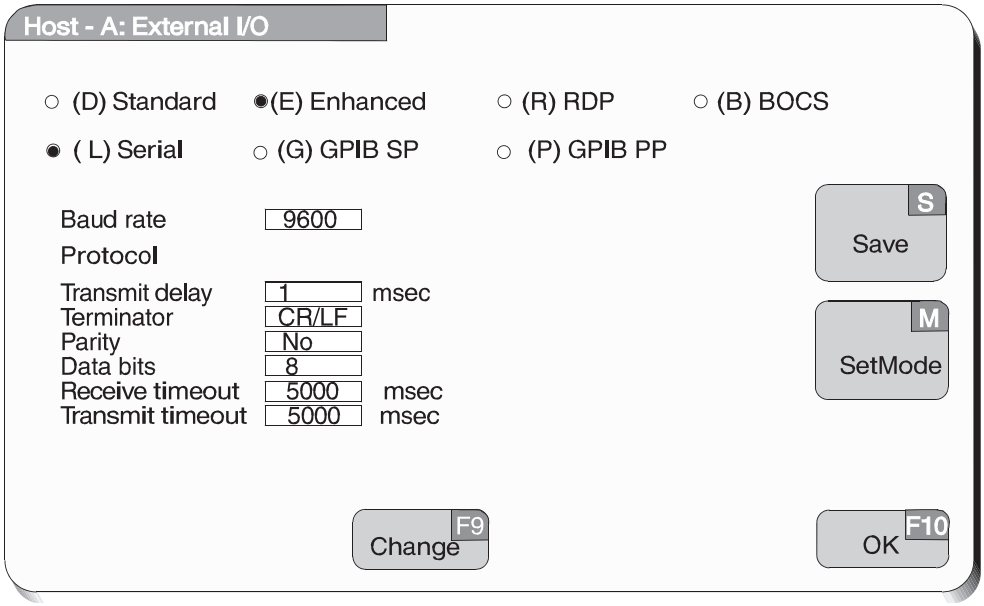


Table B-2
Serial settings

Setting type	Parameter	Value
Host-A: External I/O settings	(D) Standard	De-Selected ◎
	(L) Serial	Selected ◉
	(E) Enhanced	Selected ◉
	(G) GPIB_SP	De-Selected ◎
	(R) RDP	De-Selected ◎
	(P) GPIB_PP	De-Selected ◎
	(B) BOCS	De-Selected ◎
Protocol settings	BAUD Rate:	9600
	Transmit delay	1
	Terminator	CR/LF
	Parity	No
	Data Bits	8
	Receive Timeout:	5000 (msec)
	Transmit Timeout:	5000 (msec)

Figure B-4
Enhanced External I/O Mode Menu window

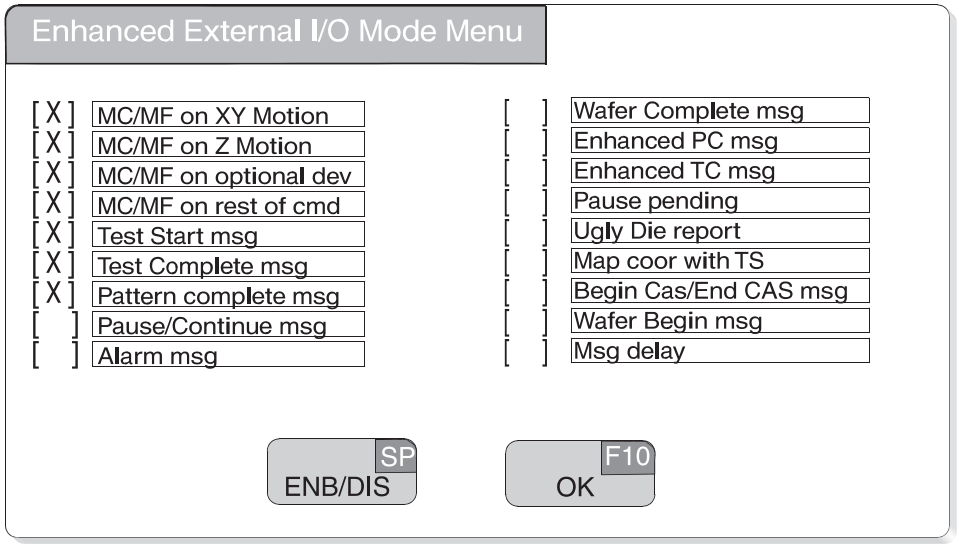


Table B-3
Enhanced external I/O mode menu settings

Parameter	Value	Parameter	Value
MC/MF on XY motion	Selected [X]	Wafer complete MSG	De-Selected [ ]
MC/MF on Z motion	Selected [X]	Enhanced PC msg	De-Selected [ ]
MC/MF on optional dev	Selected [X]	Enhanced TS msg	De-Selected [ ]
MC/MF on rest of cmd	Selected [X]	Pause pending msg	De-Selected [ ]
Test Start msg	Selected [X]	Ugly Die report	De-Selected [ ]
Test Complete msg	Selected [X]	Map Coord with TS	De-Selected [ ]
Pattern Complete msg	Selected [X]	Begin Cas/End Cas msg	De-Selected [ ]
Pause/Continue msg	De-Selected [ ]	Wafer Begin msg	De-Selected [ ]
Alarm msg	De-Selected [ ]	msg delay	De-Selected [ ]

Quadrant setup

To set the prober to the desired probing quadrant, make sure the probe quadrant settings (stepping and mode) are correctly configured. With the prober running:

1. Press the F7/Product screen-button on the main window (or F7 on the keyboard) (Figure B-5).
2. Press the C/Cntlmap (or the C key on the keyboard).
3. Press S/Stepping (or the S key on the keyboard) (Figure B-6).
4. Highlight the coordinate quadrant (Figure B-7).
5. Press F9/Change (or the F9 on the keyboard).
6. Highlight 3 (SW).
7. Press F9/Change (or the F9 on the keyboard).
8. Press FI0/OK (or FI0 on the keyboard).
9. Press M/Mode (or M on the keyboard) (Figure B-6).
10. Highlight the probing quadrant (Figure B-8).
11. Press F9/Change (or the F9 on the keyboard).
12. Highlight 3 (SW).
13. Press F9/Change (or the F9 on the keyboard).
14. Press the F10/OK (or FI0 on the keyboard).
15. To complete the change, press FI0/Close (or FI0 on the keyboard) (Figure B-6).

Figure B-5
Main window

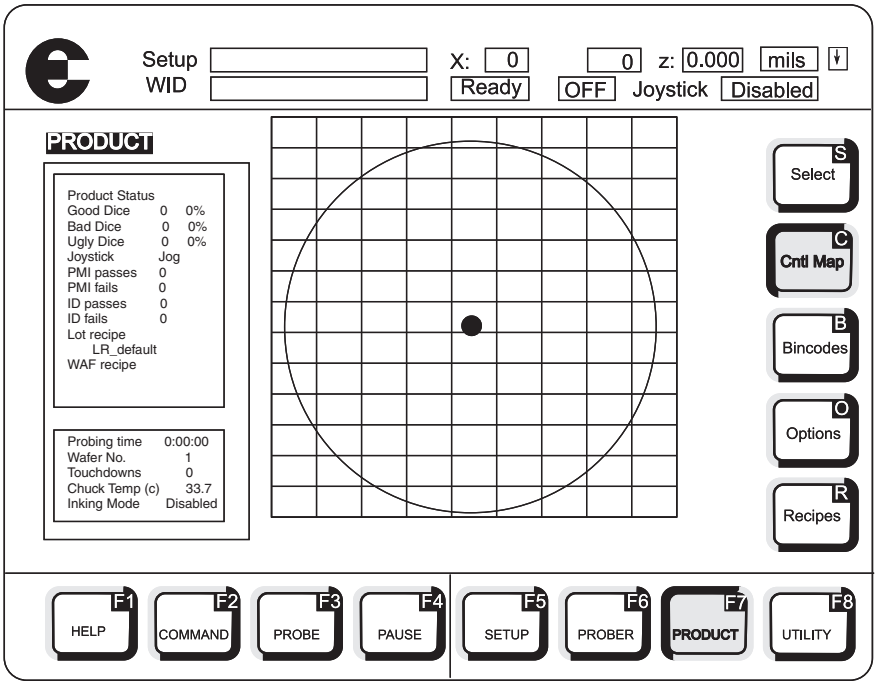


Figure B-6
Control map editor

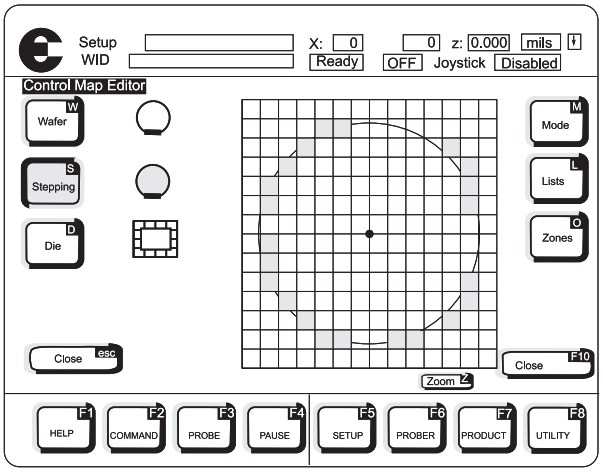


Figure B-7
Wafer layout

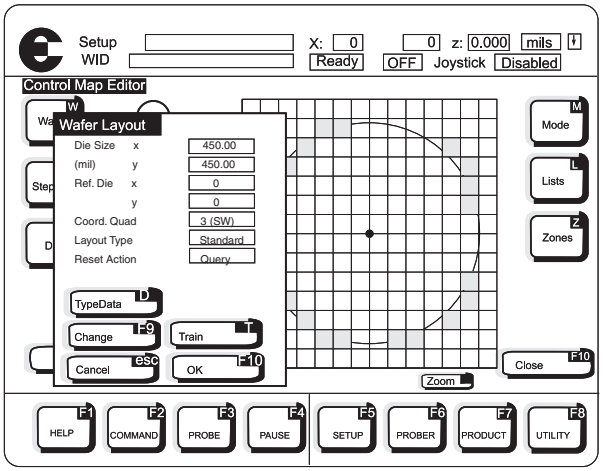
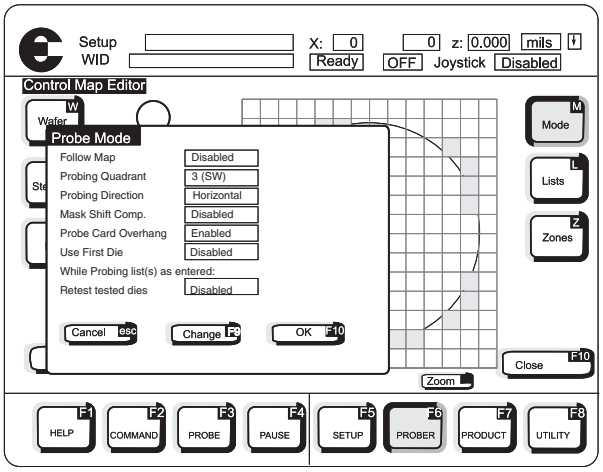


Figure B-8
Probe mode



Installation verification program

Use the installation verification program to exercise prober communication functions. If this fails, the installation was not properly completed (files are missing). Use the information below to perform installation verification.

Program name: `prb-EG40-demo.cpf`
Location: `$KIHOME/src/EXAMPLE/plans`
To run:
Start KTPM.
- Then open `$KIHOME/src/EXAMPLE/plans/prb-EG40-demo.cpf`

Start KITT.
- Then open `$KIHOME/src/EXAMPLE/pgm/prb-demo.ktm`

Start WDU.
- Then open `$KIHOME/src/EXAMPLE/pgm/prb-demo.wdf`

EG40 Driver configuration.

– Make sure the $K_PRB_CONFIG file is set to the correct prober. For example, if an EG4080 prober is being used, a file in $KIDAT called prbcnfg_EG40.dat should exist. The environment variable, KI_PRB_CONFIG, should be set to $KIDAT/prbcnfg_EG40.dat.

Updating prbsqr_EG40.dat

The prbsrq_EG40.dat file is used by the EG40 driver. Use this file to define the known SRQ responses for a given prober on a command-by-command basis. The list for a given command is defined as follows:

- The function name
- The good SRQ list
- The bad SRQ list
- The known prober error returns. Note that the known prober error numbers are defined by the prober using the PrError() command (see Table 5-2 on page 3).

Refer to Figure 7-6 for a sample prbsrq_EG40.dat (S530 sample). To add a function:

1. Open prbsrq_EG40.dat using a text editor (such as notepad).
2. Save a backup with a new file name (e.g., prbsrq_EG40.bak).
3. Place the new function and a comma somewhere after the <EOH> symbol (functions currently in the file are in alphabetical order).
4. Inside quotes, add comma-separated good SRQ's, a semicolon, comma separated bad SRQ's, a semicolon, and known prober error numbers. The use of semicolons to designate between the three different fields is required.
5. Save the file as prbsrq_EG40.dat. An example function PRLOAD is contained in Figure B-9:

Figure B-9
Example updating prbsqr_EG40.dat

```
•
•
•
<EOH>
•
•
•
PRLOAD,"70,72,74;75,78,101;0"
<EOLOC>\
```

This function (PRLOAD) has:

- Good SRQ's of 70, 72, and 74.
- Bad SRQ's of 75, 78, and 101.
- And no known prober errors.

Command list

Table B-3 lists commands supported by EG 4060/4080/4090/5|300 probers. For detailed information about a specific command, refer to Section 5.

Table B-4

EG 4060/4080/4090/5|300 supported commands

Commands		
PrAbsMove	PrMove	PrSetPipeline
PrAdjustZHeight	PrNeedleClean	
PrAutoAlign	PrProfile	PrSetQuadrant
PrCassetteMap	PrPutNxtSlot	PrSetRefDie
PrCheckOptions	PrPutWafer	PrSetSlotStatus
	PrQueryZHeight	
PrChuck	PrReadId	PrSetTime
PrClearPipeline	PrRelMove	PrSetUnits
		PrSetZHeight
		PrSmifClamp
		PrSmifLock
		PrSmifStatus
PrError	PrRelReturn	PrStatus
PrGetNxtWafer	PrSerialPoll	PrUnLoad
PrGetProduct	PrSetDiam	PrWriteRead
PrGetWafer	PrSetDieSize	PrWriteReadSRQ
PrInit	PrSetFlat	PrZParams
PrLoad	PrSetMode	PrZTravel
PrLoadProduct		

Error symbols and returned values table

After calling a prober command as a function, a value or symbol is returned. Table B-5 contains error symbols and return values associated with specific commands.

Table B-5
Error symbols and returned values

Return Value	Commands																				Error Symbol
	PrAbsMove	PrAdjustZHeight	PrAutoAlign	PrCassetteMap	PrCheckOptions	PrChuck	PrClearPipeline	PrError	PrGetWafer	PrGetNxtWafer[1]	PrInit	PrLoad	PrLoadProduct	PrMove	PrMovNxt	PrNeedleClean	PrProfile	PrPutNxtSlot[2]	PrPutWafer[3]	PrQueryZHeight	PrReadId[4]	
0, +1	√		√	√	√	√	√		√		√	√	√				√					PR_OK
+2																						PR_MOVECOMPLETE
+4																						PR_WAFERCOMPLETE
+8																						PR_CASSETTECOMPLETE
+10												√										PR_LOTEND
+12																						PR_WAFER_REJECT_LOAD
+14																						PR_WAFER_REJECT_NOLOAD
+16																						PR_OPER_ACTION
-1001																						BAD_TST_NUM
-1002										√												BAD_CONFIG_DAT
-1003																						FEAT_NOT_SUPPORT
-1004	√		√	√	√	√	√	√	√	√	√	√	√	√	√		√	√	√		√	PORT_ASS_FAIL
-1005											√											SET_UNITS_FAIL
-1006																						INVAL_MODE
-1007																						CLR_WAF_MAP
-1008							√		√	√	√							√	√			SET_MODE_FAIL
-1009																						SET_DIE_FAIL
-1010											√											SET_PRESET_FAIL
-1011			√			√			√	√		√		√	√			√	√			BAD_MODE
-1012															√							TST_COMPL_FAIL
-1013	√		√	√	√	√	√	√	√	√		√		√	√		√	√	√		√	UNINTEL_RESP
-1014	√		√				√			√				√	√		√	√				MOVE_FAIL
-1015															√							UNEXPE_ERROR
-1016									√										√			LOAD_ERROR
-1017						√																BAD_CHUCK
-1018																						CHUCK_NOT_UP
-1019																						INK_FAIL
-1020					√		√	√		√											√	TIME_GONE
-1021																						BAD_LEARN_FUNCT
-1022																						SET_DIAM_FAIL
-1023																						SET_MATRIX_FAIL
-1024																						BAD_Z_PARAM
-1025																						NO_RESPONSE
-1026																						TOO_MANY_NAKS
-1027				√														√	√			INVAL_PARAM
-1028													√									ERR_LOAD_FILE
-1029																						ALIGN_FAIL
-1030																						GPIB_ERROR
-1031																						CHK_PORT_ASS
-1032																						CHK_DEV_PROT
-1033																						PR_MEM_ALLOC_ERR
-1034																						PR_NO_CASSETTE
-1035																						SET_FLAT_FAIL
-1036																						SET_REF_DIE_FAIL
-1037																						Z_PARAM_FAIL
-1038																						ERR_OPEN_PRBCNFG
-1039																						NO_CNFG_DATA
-1040																						NO_PRB_TYPE
-1041																						INVAL_PRB_MODE
-1042																						INVAL_PRB_OPTIONS
-1043																						INVAL_DEVICE_NAME
-1044																						INVAL_DEVICE_IRQ
-1045																						INVAL_BAUD_RATE
-1046																						INVAL_TIMEOUT
-1047																						INVAL_GPIB_UNIT
-1048																						INVAL_GPIB_SLOT
-1049																						INVAL_GPIB_ADDRESS

[1]**PrGetNxtWafer:** Returns the slot number.
[2]**PrPutNxtSlot:** Returns the slot number.
[3]**PrPutWafer:** Returns the slot number.
[4]**PrReadId:** Returns the number of characters received (>0).
[5]**Prober error:** Add 1500 to the error number returned and then refer to the prober manufacturer for error handling.

Table B-5 (continued)
Error symbols and returned values

Return Value	Commands																						Error Symbol
	PrAbsMove	PrAdjustZHeight	PrAutoAlign	PrCassetteMap	PrCheckOptions	PrChuck	PrClearPipeline	PrError	PrGetWafer	PrGetNxtWafer[1]	PrInit	PrLoad	PrLoadProduct	PrMove	PrMovNxt	PrNeedleClean	PrProfile	PrPutNxtSlot[2]	PrPutWafer[3]	PrQueryZHeight	PrReadId[4]	
-1050																						INVAL_GPIB_WRITE_MODE
-1051																						INVAL_GPIB_READ_MODE
-1052																						INVAL_GPIB_TERMINATOR
-1053																						INVAL_KULT_PATH
-1054																						INVAL_PRB_LIB
-1055																						INVAL_PRB_CNFG_FUNC
-1056																						INVAL_PRB_MAX_SLOTS
-1057																						INVAL_PRB_MAX_CASSETTES
-1058																						SET_QUAD_FAIL
-1059																						NO_UNPROBED_WAFERS
-1060																						SET_TEMPERATURE_FAIL
-1061																						QUERY_TEMPERATURE_FAIL
-1062																						SMIF_LOCK_FAIL
-1063																						INVALID_ARRAY_SIZE
-1064																						NO_SRQ_FOUND
-1065																						UNKNOWN_SRQ_FOUND
-1066																						INVALID_SRQ_FILE
-1067																√						INVALID_NEEDLE_CLEANING_STATION
-1068																√						ERROR_CLEANING_NEEDLES
-1069		√																				ERROR_MOVING_ZFINE
-1070																				√		INVALID_Z_SET_RETURNED
-1071																						INVALID_Z_SET_HT
-1072																						PR_MAX_Z_CT_EXCEEDED
-1073																						INVALID_USER_FUNCTION
-1074																						INVALID_RETURN_USER_FUNC
-1075																						INVALID_AUTOZ_LIB
-1076																						INVALID_AUTOZ_FUNC
-1077																						AUTOZ_NOT_ON_PROBER
-1078																						NO_AUTOZ_LICENSE
-1079																						NONEEDLE_CLEANING_LICENSE
-1080																						KI_ABORT_AUTO-Z
-1081																						INVAL_PRB_P8_OPTION
-1082																						NI_UNIX_ERROR
-1083																						NI_BOARD_NOT_CIC
-1084																						NI_NO_LISTENER
-1085																						NI_BOARD_NOT_ADDRESSES
-1086																						NI_INVALID_FUNCTION_ARG
-1087																						NI_BOARD_NOT_SYS_CONTROLLER
-1088																						NI_IO_ABORTED_TIMO
-1089																						NI_BOARD_NOT_PRESENT
-1090																						NI_DMA_HW_PROBLEM
-1091																						NI_DMA_HW_TIMO
-1092																						NI_NO_CAPABILITY
-1093																						NI_FILE_ SYS_ERROR
-1094																						NI_GPIB_BUS_ERROR
-1095																						NI_SRQ_BYTE_Q_OVERFLOW
-1096																						NI_SRQ_STUCK_ON
-1097																						NI_TABLE_PROBLEM
-1500 & on								√														Prober error[5]

[1] **PrGetNxtWafer:** Returns the slot number.
[2] **PrPutNxtSlot:** Returns the slot number.
[3] **PrPutWafer:** Returns the slot number.
[4] **PrReadId:** Returns the number of characters received (>0).
[5] **Prober error:** Add 1500 to the error number returned and then refer to the prober manufacturer for error handling.

Table B-5 (continued)
Error symbols and returned values

Return Value	Commands																					Error Symbol
	PrRelMove	PrRelReturn	PrSerialPoll[6]	PrSetDiam	PrSetDieSize	PrSetFlat	PrSetMode	PrSetPipeline	PrSetQuadrant	PrSetRefDie	PrSetSlotStatus	PrSetTime	PrSetUnits	PrSetZHeight	PrSmifClamp	PrSmifLock	PrSmifStatus	PrStatus	PrUnload	PrWriteread[7]	PrZParams	PrZTravel	
0, +1	√	√		√	√	√	√	√	√	√	√	√	√					√	√		√	√	PR_OK
+2																							PR_MOVECOMPLETE
+4																							PR_WAFERCOMPLETE
+8																							PR_CASSETTECOMPLETE
+10																							PR_LOTEND
+12																							PR_WAFER_REJECT_LOAD
+14																							PR_WAFER_REJECT_NOLOAD
+16																							PR_OPER_ACTION
-1001																							BAD_TST_NUM
-1002																							BAD_CONFIG_DAT
-1003																							FEAT_NOT_SUPPORT
-1004	√	√	√	√	√	√	√	√	√	√	√	√	√					√	√	√	√	√	PORT_ASS_FAIL
-1005													√										SET_UNITS_FAIL
-1006							√																INVAL_MODE
-1007							√																CLR_WAF_MAP
-1008							√	√			√												SET_MODE_FAIL
-1009					√																		SET_DIE_FAIL
-1010																							SET_PRESET_FAIL
-1011																			√				BAD_MODE
-1012																							TST_COMPL_FAIL
-1013	√	√				√		√	√		√							√	√	√	√	√	UNINTEL_RESP
-1014	√	√																					MOVE_FAIL
-1015																							UNEXPE_ERROR
-1016																							LOAD_ERROR
-1017																							BAD_CHUCK
-1018																							CHUCK_NOT_UP
-1019																							INK_FAIL
-1020			√					√			√							√		√			TIME_GONE
-1021																							BAD_LEARN_FUNCT
-1022				√																			SET_DIAM_FAIL
-1023																							SET_MATRIX_FAIL
-1024																					√		BAD_Z_PARAM
-1025																							NO_RESPONSE
-1026																							TOO_MANY_NAKS
-1027								√			√												INVAL_PARAM
-1028																							ERR_LOAD_FILE
-1029																							ALIGN_FAIL
-1030																							GPIB_ERROR
-1031																							CHK_PORT_ASS
-1032																							CHK_DEV_PROT
-1033																							PR_MEM_ALLOC_ERR
-1034																							PR_NO_CASSETTE
-1035																							SET_FLAT_FAIL
-1036																							SET_REF_DIE_FAIL
-1037																							Z_PARAM_FAIL
-1038																							ERR_OPEN_PRBCNFG
-1039																							NO_CNFG_DATA
-1040																							NO_PRB_TYPE
-1041																							INVAL_PRB_MODE
-1042																							INVAL_PRB_OPTIONS
-1043																							INVAL_DEVICE_NAME
-1044																							INVAL_DEVICE_IRQ
-1045																							INVAL_BAUD_RATE
-1046																							INVAL_TIMEOUT
-1047																							INVAL_GPIB_UNIT
-1048																							INVAL_GPIB_SLOT
-1049																							INVAL_GPIB_ADDRESS

[5]**Prober error:** Add 1500 to the error number returned and then refer to the prober manufacturer for error handling.
[6]**PrSerialPoll:** Returns the SRQ status byte (>0).
[7]**PrWriteRead:** Returns the number of characters received (>0).

Table B-5 (continued)
Error symbols and returned values

Return Value	Commands																						Error Symbol
	PrRelMove	PrRelReturn	PrSerialPoll[6]	PrSetDiam	PrSetDieSize	PrSetFlat	PrSetMode	PrSetPipeline	PrSetQuadrant	PrSetRefDie	PrSetSlotStatus	PrSetTime	PrSetUnits	PrSetZHeight	PrSmifClamp	PrSmifLock	PrSmifStatus	PrStatus	PrUnload	PrWriteread[7]	PrZParams	PrZTravel	
-1050																							INVAL_GPIB_WRITE_MODE
-1051																							INVAL_GPIB_READ_MODE
-1052																							INVAL_GPIB_TERMINATOR
-1053																							INVAL_KULT_PATH
-1054																							INVAL_PRB_LIB
-1055																							INVAL_PRB_CNFG_FUNC
-1056																							INVAL_PRB_MAX_SLOTS
-1057																							INVAL_PRB_MAX_CASSETTES
-1058																							SET_QUAD_FAIL
-1059																							NO_UNPROBED_WAFERS
-1060																							SET_TEMPERATURE_FAIL
-1061																							QUERY_TEMPERATURE_FAIL
-1062															√	√							SMIF_LOCK_FAIL
-1063																	√						INVALID_ARRAY_SIZE
-1064																							NO_SRQ_FOUND
-1065																							UNKNOWN_SRQ_FOUND
-1066																							INVALID_SRQ_FILE
-1067																							INVALID_NEEDLE_CLEANING_STATION
-1068																							ERROR_CLEANING_NEEDLES
-1069																							ERROR_MOVING_ZFINE
-1070																							INVALID_Z_SET_RETURNED
-1071														√									INVALID_Z_SET_HT
-1072																							PR_MAX_Z_CT_EXCEEDED
-1073																							INVALID_USER_FUNCTION
-1074																							INVALID_RETURN_USER_FUNC
-1075																							INVALID_AUTOZ_LIB
-1076																							INVALID_AUTOZ_FUNC
-1077																							AUTOZ_NOT-ON_PROBER
-1078																							NO_AUTOZ_LICENSE
-1079																							NO_NEEDLE_CLEANING_LICENSE
-1080																							KI_ABORT_AUTO_Z
-1081																							INVAL_PRB_P8_OPTION
-1082																							NI_UNIX_ERROR
-1083																							NI_BOARD_NOT_CIC
-1084																							NI_NO_LISTENER
-1085																							NI_BOARD_NOT_ADDRESSES
-1086																							NI_INVALID_FUNCTION_ARG
-1087																							NI_BOARD_NOT_SYS_CONTROLLER
-1088																							NI_IO_ABORTED_TIMO
-1089																							NI_BOARD_NOT_PRESENT
-1090																							NI_DMA_HW_PROBLEM
-1091																							NI_DMA_HW_TIMO
-1092																							NI_NO_CAPABILITY
-1093																							NI_FILE_SYS_ERROR
-1094																							NI_GPIB_BUS_ERROR
-1095																							NI_SRQ_BYTE_Q_OVERFLOW
-1096																							NI_SRQ_STUCK_ON
-1097																							NI_TABLE_PROBLEM
-1500 & on																							Prober error[5]

[5]**Prober error:** Add 1500 to the error number returned and then refer to the prober manufacturer for error handling.
[6]**PrSerialPoll:** Returns the SRQ status byte (>0).
[7]**PrWriteRead:** Returns the number of characters received (>0).

C
TEL Prober Information

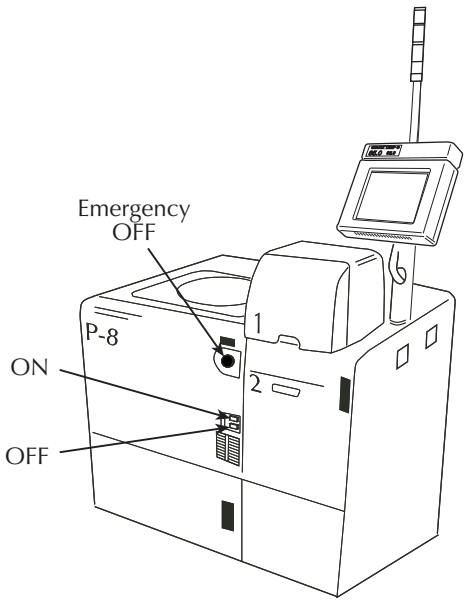
TEL models:

P8-series, P12-series, T19S, and Precio

Prober start-up

TEL P8 probers have three controls associated with prober start-up: ON, OFF, and EMERGENCY Off. Figure C-1 shows control location. To power up the prober, press the ON pushbutton. To remove power from the prober, press either the OFF or the EMERGENCY OFF pushbuttons as appropriate.

Figure C-1
TEL P8 controls



Prober setup and initialization

NOTE *Refer to the specific Prober requirements contained in Section 2 for the software version used to verify the prober driver and configuration contained in this section.*

Wafer(s) must be loaded onto the chuck PRIOR to starting KTXE. The prober does not allow the remote loading of the first wafer from cassette. Also, the prober initialization process can not take place until the first wafer(s) are loaded.

Use the following information to properly setup the prober. After powering-up and initializing the prober, make sure the following parameters are properly set:

MAIN MENU SETUP

OPERATION PARAMETER	**Setting**
1. Index Unit	English
2. GPIB	*menu*
1. GP-IB.....	Use
2. Stop at reference die.....	Yes
3. Receive parameters.....	Y/N (see Note 1)
4. Random wafer testing.....	No
5. Watch Time for GP-IB Timer (Sec).....	0
6. Contact Check.....	Y/N (see Note 2)
7. Cassette Map.....	*
8. SRQ Timeout.....	0

*User preferred setting.

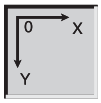
Note 1: Refer to the paragraph titled "Load first wafer remotely" on page C-4.

Note 2: If Contact Check is set to Y, at startup the prober will output an SRQ (decimal 108). This SRQ indicates the SofTouch (contact check) is enabled.

DIAGNOSTICS ADJUSTMENTS

AGING Aging Parameters	**Setting**
1. Tester communications.....	Yes
2. Tester results with no tester communication	Use
3. Wafer alignment.....	Yes
4. Loading check.....	No
5. Probe alignment.....	Yes
6. Consecutive probing.....	No

NOTE *Test Limitations Settings (below) only applies to software versions Pap00-R011.02-2 and newer.*

TEST LIMITATIONS Settings	**Setting**
DIE Coordinate Axis	

HARDWARE OPTIONS GPIB Parameter Settings	**Setting**
1. GP-IB Address	5
2. Terminator	CR-LF
3. I, J Coordinates	*
4. Z Position after XY stage	Position before drive
5. Drive off wafer with X, Y, I, J, b	Yes
6. Count pass/fail with the r,Q command	Yes
7. Output T-Start signal with Q,r command	No
8. Process Q and r the same way after T-COMP Signal is input	No

HARDWARE OPTIONS GPIB Parameter Settings	**Setting**
9. Use SRQ 43 or C3 After Z when using profiler	43
10. Output SRQ at Carrier End	Yes
11. Initial wafer SRQ	46+4A
12. Output SRQ of Command/Parameter Error	Yes
13. Auto Unload with Alignment Error signal or with the u, U Command	Auto
14. SRQ 4E Output at Lot End	Yes
15. SRQ 4D Output count at consecutive failures	Number of dies generated
16. Correspond to p1, p2 command	No
17. Output SRQ 4EH for Assist During PMI	No
18. Output SRQ 4EH for Assist During IDI	No
19. Z Position at Initial Die	Separate
20. Output SRQ 62H at the StopSW make testing stop	No
21. Adapt to the IG Version 2.4	No
22. Output SRQ 47H at the Off Wafer With X, Y, I, J, b	No
23. Output SRQ 5dH before execute PMI	No
24. Output SRQ41(H),42(H)at the off wafer with I,J...	Yes

Installation verification program

Use the installation verification program to exercise prober communication functions. If this fails, the installation was not properly completed (files are missing). Use the information below to perform installation verification.

Program name: `prb-P8-demo.cpf`
Location: `$KIHOME/src/EXAMPLE/plans`
To run:
Start KTPM.

– Then open `$KIHOME/src/EXAMPLE/plans/prb-P8-demo.cpf`

Start KITT.

– Then open `$KIHOME/src/EXAMPLE/pgm/prb-demo.ktm`

Start WDU.

– Then open `$KIHOME/src/EXAMPLE/pgm/prb-demo.wdf`

P8 driver configuration

Load first wafer remotely

NOTE *This applies to the P8 driver only.*

In order to load the lot's first wafer from KTXE, perform the following configuration:

1. Add uap write lot info - KTXEOperatorLoadAlign (or equivalent).
2. Add uap prober init - KTXEGetProductFile.

3. Configure the prober to "receive commands."

After the tester and prober have been configured correctly, use the following sequence to test a lot of wafers:

1. Place the cassette on the prober.
2. Press the Run button.
3. Press the start button (the product file will be determined by the test program).

Updating prbsqr_P8.dat

The prbsrq_P8.dat file is used by the P8 driver. Use this file to define the known SRQ responses for a given prober on a command-by-command basis. The list for a given command is defined as follows:

- The function name
- The good SRQ list
- The bad SRQ list
- The known prober error returns. Note that the known prober error numbers are defined by the prober using the PrError() command (see Table 5-2 on page 5-3).

Refer to Figure 7-9 for a sample prbsrq_P8.dat (S530 sample). To add a function:

1. Open prbsrq_P8.dat using a text editor (such as notepad).
2. Save a backup with a new file name (e.g., prbsrq_P8.bak).
3. Place the new function and a comma somewhere after the <EOH> symbol (functions currently in the file are in alphabetical order).
4. Inside quotes, add comma-separated good SRQ's, a semicolon, comma separated bad SRQ's, a semicolon, and known prober error numbers. The use of semicolons to designate between the three different fields is required.
5. Save the file as prbsrq_P8.dat. An example function PRLOAD is contained in Figure C-2:

Figure C-2
Example updating prbsqr_P8.dat

```
•
•
•
<EOH>
•
•
•
PRLOAD,"70,72,74;75,78,101;0"
<EOLOC>
```

This function (PRLOAD) has:

- Good SRQ's of 70, 72, and 74.
- Bad SRQ's of 75, 78, and 101.
- And no known prober errors.

Environment variable KI_PRB_CONFIG

Make sure the $KI_PRB_CONFIG file is set to the correct prober. For example, if a P8 prober is being used, a file in $KIDAT called prbcnfg_P8.dat should exist. The environment variable, KI_PRB_CONFIG, should be set to $KIDAT/prbcnfg_P8.dat.

Command list

Table C-1 commands supported by the TEL P8 prober. For detailed information about a specific command, refer to Section 5.

Table C-1
TEL P8 Supported commands

Command	
PrAbsMove	PrQueryZHeight
PrAdjustZHeight	PrReadId
PrCassetteMap	PrRelMove
PrCheckOptions	PrRelReturn
PrChuck	PrSerialPoll
PrError	PrSetTime
PrInit	PrSetZHeight
PrLoad	PrStatus
PrLoadProduct	PrUnload
PrMove	PrWriteRead
PrNeedleClean	PrWriteReadSRQ

Error symbols and returned values table

After calling a prober command as a function, a value or symbol is returned. Table C-2 contains error symbols and return values associated with specific commands.

Table C-2
Error symbols and returned values

Return Value	Commands													Error Symbol
	PrAbsMove	PrAdjustZHeight	PrCassetteMap	PrCheckOptions	PrChuck	PrError	PrInit	PrLoad	PrLoadProduct	PrMove	PrNeedleClean	PrQueryZHeight	PrReadId[4]	
0, +1	√		√	√	√	√	√		√					PR_OK
+2														PR_MOVECOMPLETE
+4								√						PR_WAFERCOMPLETE
+8								√						PR_CASSETTECOMPLETE
+10								√						PR_LOTEND
+12														PR_WAFER_REJECT_LOAD
+14														PR_WAFER_REJECT_NOLOAD
+16														PR_OPER_ACTION
-1001														BAD_TST_NUM
-1002														BAD_CONFIG_DAT
-1003														FEAT_NOT_SUPPORT
-1004	√		√	√	√	√	√	√	√	√			√	PORT_ASS_FAIL
-1005							√							SET_UNITS_FAIL
-1006														INVAL_MODE
-1007														CLR_WAF_MAP
-1008							√	√						SET_MODE_FAIL
-1009														SET_DIE_FAIL
-1010														SET_PRESET_FAIL
-1011														BAD_MODE
-1012														TST_COMPL_FAIL
-1013	√				√			√					√	UNINTEL_RESP
-1014	√				√			√					√	MOVE_FAIL
-1015														UNEXPE_ERROR

[1]**PrGetNxtWafer:** Returns the slot number.
[2]**PrPutNxtSlot:** Returns the slot number.
[3]**PrPutWafer:** Returns the slot number.
[4]**PrReadId:** Returns the number of characters received (>0).
[5]**Prober error:** Add 1500 to the error number returned and then refer to the prober manufacturer for error handling.

Table D-2 (continued)
Error symbols and returned values

Return Value	Commands													Error Symbol
	PrAbsMove	PrAdjustZHeight	PrCassetteMap	PrCheckOptions	PrChuck	PrError	PrInit	PrLoad	PrLoadProduct	PrMove	PrNeedleClean	PrQueryZHeight	PrReadId[4]	
-1016								√						LOAD_ERROR
-1017					√									BAD_CHUCK
-1018														CHUCK_NOT_UP
-1019														INK_FAIL
-1020					√	√		√					√	TIME_GONE
-1021														BAD_LEARN_FUNCT
-1022														SET_DIAM_FAIL
-1023														SET_MATRIX_FAIL
-1024														BAD_Z_PARAM
-1025														NO_RESPONSE
-1026														TOO_MANY_NAKS
-1027			√											INVAL_PARAM
-1028									√					ERR_LOAD_FILE
-1029														ALIGN_FAIL
-1030														GPIB_ERROR
-1031														CHK_PORT_ASS
-1032														CHK_DEV_PROT
-1033														PR_MEM_ALLOC_ERR
-1034			√											PR_NO_CASSETTE
-1035														SET_FLAT_FAIL
-1036														SET_REF_DIE_FAIL
-1037														Z_PARAM_FAIL
-1038														ERR_OPEN_PRBCNFG
-1039														NO_CNFG_DATA
-1040														NO_PRB_TYPE
-1041														INVAL_PRB_MODE
-1042														INVAL_PRB_OPTIONS
-1043														INVAL_DEVICE_NAME
-1044														INVAL_DEVICE_IRQ
-1045														INVAL_BAUD_RATE
-1046														INVAL_TIMEOUT
-1047														INVAL_GPIB_UNIT
-1048														INVAL_GPIB_SLOT
-1049														INVAL_GPIB_ADDRESS
-1050														INVAL_GPIB_WRITE_MODE
-1051														INVAL_GPIB_READ_MODE
-1052														INVAL_GPIB_TERMINATOR
-1053														INVAL_KULT_PATH
-1054														INVAL_PRB_LIB
-1055														INVAL_PRB_CNFG_FUNC
-1056														INVAL_PRB_MAX_SLOTS
-1057														INVAL_PRB_MAX_CASSETTES
-1058														SET_QUAD_FAIL
-1059														NO_UNPROBED_WAFERS
-1060														SET_TEMPERATURE_FAIL
-1061														QUERY_TEMPERATURE_FAIL
-1062														SMIF_LOCK_FAIL
-1063														INVALID_ARRAY_SIZE
-1064														NO_SRQ_FOUND
-1065														UNKNOWN_SRQ_FOUND
-1066														INVALID_SRQ_FILE
-1067											√			INVALID_NEEDLE_CLEANING_STATION
-1068											√			ERROR_CLEANING_NEEDLES
-1069		√												ERROR_MOVING_ZFINE
-1070												√		INVALID_Z_HT_RETURNED
-1071														INVALID_Z_SET_HT
-1072														PR_MAX_Z_CT_EXCEEDED
-1073														INVALID_USER_FUNCTION
-1074														INVALID_RETURN_USER_FUNC

[1]**PrGetNxtWafer:** Returns the slot number.
[2]**PrPutNxtSlot:** Returns the slot number.
[3]**PrPutWafer:** Returns the slot number.
[4]**PrReadId:** Returns the number of characters received (>0).
[5]**Prober error:** Add 1500 to the error number returned and then refer to the prober manufacturer for error handling.

Table D-2 (continued)
Error symbols and returned values

Return Value	Commands													Error Symbol
	PrAbsMove	PrAdjustZHeight	PrCassetteMap	PrCheckOptions	PrChuck	PrError	PrInit	PrLoad	PrLoadProduct	PrMove	PrNeedleClean	PrQueryZHeight	PrReadId[4]	
-1075														INVALID_AUTOZ_LIB
-1076														INVALID_AUTOZ_FUNC
-1077														AUTOZ_NOT_ON_PROBE
-1078														NO_AUTOZ_LICENSE
-1079														NO_NEEDLE_CLEANING_LICENSE
-1080														KI_ABORT_AUTO_Z
-1081														INVAL_PRB_P8_OPTION
-1082														NI_UNIX_ERROR
-1083														NI_BOARD_NOT_CIC
-1084														NI_NO_LISTENER
-1085														NI_BOARD_NOT_ADDRESSES
-1086														NI_INVALID_FUNCTION_ARG
-1087														NI_BOARD_NOT_SYS_CONTROLLER
-1088														NI_IO_ABORTED_TIMO
-1089														NI_BOARD_NOT_PRESENT
-1090														NI_DMA_HW_PROBLEM
-1091														NI_DMA_HW_TIMO
-1092														NI_NO_CAPABILITY
-1093														NI_FILE_SYS_ERROR
-1094														NI_GPIB_BUS_ERROR
-1095														NI_SRQ_BYTE_Q_OVERFLOW
-1096														NI_SRQ_STUCK_ON
-1097														NI_TABLE_PROBLEM
-1500 & on						√								Prober error[5]

[1]**PrGetNxtWafer:** Returns the slot number.
[2]**PrPutNxtSlot:** Returns the slot number.
[3]**PrPutWafer:** Returns the slot number.
[4]**PrReadId:** Returns the number of characters received (>0).
[5]**Prober error:** Add 1500 to the error number returned and then refer to the prober manufacturer for error handling.

Table D-2 (continued)
Error symbols and returned values

Return Value	Commands									Error Symbol
	PrRelMove	PrRelReturn	PrSerialPoll[6]	PrSetTime	PrSetZHeight	PrStatus	PrUnload	PrWriteread[7]	PrWritereadSRQ[7]	
0, +1	√	√		√		√	√			PR_OK
+2										PR_MOVECOMPLETE
+4										PR_WAFERCOMPLETE
+8										PR_CASSETTECOMPLETE
+10										PR_LOTEND
+12										PR_WAFER_REJECT_LOAD
+14										PR_WAFER_REJECT_NOLOAD
+16										PR_OPER_ACTION
-1001										BAD_TST_NUM
-1002										BAD_CONFIG_DAT
-1003										FEAT_NOT_SUPPORT
-1004	√	√	√	√		√	√	√	√	PORT_ASS_FAIL
-1005										SET_UNITS_FAIL
-1006										INVAL_MODE
-1007										CLR_WAF_MAP
-1008										SET_MODE_FAIL
-1009										SET_DIE_FAIL
-1010										SET_PRESET_FAIL
-1011							√			BAD_MODE
-1012										TST_COMPL_FAIL
-1013							√			UNINTEL_RESP
-1014	√	√								MOVE_FAIL
-1015										UNEXPE_ERROR
-1016										LOAD_ERROR
-1017										BAD_CHUCK
-1018										CHUCK_NOT_UP
-1019										INK_FAIL
-1020			√					√	√	TIME_GONE
-1021										BAD_LEARN_FUNCT
-1022										SET_DIAM_FAIL
-1023										SET_MATRIX_FAIL
-1024										BAD_Z_PARAM
-1025										NO_RESPONSE
-1026										TOO_MANY_NAKS
-1027										INVAL_PARAM
-1028										ERR_LOAD_FILE
-1029										ALIGN_FAIL
-1030										GPIB_ERROR
-1031										CHK_PORT_ASS
-1032										CHK_DEV_PROT
-1033										PR_MEM_ALLOC_ERR
-1034										PR_NO_CASSETTE
-1035										SET_FLAT_FAIL
-1036										SET_REF_DIE_FAIL
-1037										Z_PARAM_FAIL
-1038										ERR_OPEN_PRBCNFG
-1039										NO_CNFG_DATA
-1040										NO_PRB_TYPE
-1041										INVAL_PRB_MODE
-1042										INVAL_PRB_OPTIONS
-1043										INVAL_DEVICE_NAME
-1044										INVAL_DEVICE_IRQ
-1045										INVAL_BAUD_RATE
-1046										INVAL_TIMEOUT
-1047										INVAL_GPIB_UNIT
-1048										INVAL_GPIB_SLOT
-1049										INVAL_GPIB_ADDRESS
-1050										INVAL_GPIB_WRITE_MODE
-1051										INVAL_GPIB_READ_MODE
-1052										INVAL_GPIB_TERMINATOR
-1053										INVAL_KULT_PATH
-1054										INVAL_PRB_LIB
-1055										INVAL_PRB_CNFG_FUNC

[5]**Prober error:** Add 1500 to the error number returned and then refer to the prober manufacturer for error handling.
[6]**PrSerialPoll:** Returns the SRQ status byte (>0).
[7]**PrWriteRead:** Returns the number of characters received (>0).

Table D-2 (continued)

Error symbols and returned values

Return Value	Commands									Error Symbol
	PrRelMove	PrRelReturn	PrSerialPoll[6]	PrSetTime	PrSetZHeight	PrStatus	PrUnload	PrWriteread[7]	PrWritereadSRQ[7]	
-1056										INVAL_PRB_MAX_SLOTS
-1057										INVAL_PRB_MAX_CASSETTES
-1058										SET_QUAD_FAIL
-1059										NO_UNPROBED_WAFERS
-1060										SET_TEMPERATURE_FAIL
-1061										QUERY_TEMPERATURE_FAIL
-1062										SMIF_LOCK_FAIL
-1063										INVALID_ARRAY_SIZE
-1064										NO_SRQ_FOUND
-1065										UNKNOWN_SRQ_FOUND
-1066										INVALID_SRQ_FILE
-1067										INVALID_NEEDLE_CLEANING_STATION
-1068										ERROR_CLEANING_NEEDLES
-1069										ERROR_MOVING_ZFINE
-1070										INVALID_Z_HT_RETURNED
-1071					√					INVALID_Z_SET_HT
-1072										PR_MAX_Z_CT_EXCEEDED
-1073										INVALID_USER_FUNCTION
-1074										INVALID_RETURN_USER_FUNC
-1075										INVALID_AUTOZ_LIB
-1076										INVALID_AUTOZ_FUNC
-1077										AUTOZ_NOT_ON_PROBE
-1078										NO_AUTOZ_LICENSE
-1079										NO_NEEDLE_CLEANING_LICENSE
-1080										KI_ABORT_AUTO_Z
-1081										INVAL_PRB_P8_OPTION
-1082										NI_UNIX_ERROR
-1083										NI_BOARD_NOT_CIC
-1084										NI_NO_LISTENER
-1085										NI_BOARD_NOT_ADDRESSES
-1086										NI_INVALID_FUNCTION_ARG
-1087										NI_BOARD_NOT_SYS_CONTROLLER
-1088										NI_IO_ABORTED_TIMO
-1089										NI_BOARD_NOT_PRESENT
-1090										NI_DMA_HW_PROBLEM
-1091										NI_DMA_HW_TIMO
-1092										NI_NO_CAPABILITY
-1093										NI_FILE_SYS_ERROR
-1094										NI_GPIB_BUS_ERROR
-1095										NI_SRQ_BYTE_Q_OVERFLOW
-1096										NI_SRQ_STUCK_ON
-1097										NI_TABLE_PROBLEM
-1500 & on										Prober error[5]

[5]**Prober error:** Add 1500 to the error number returned and then refer to the prober manufacturer for error handling.
[6]**PrSerialPoll:** Returns the SRQ status byte (>0).
[7]**PrWriteRead:** Returns the number of characters received (>0).

D
TSK/Accretech Prober Information

TSK model: A-PM-90A

TSK/Accretech models: UF190, UF200, UF300, and UF3000

Prober start-up

TSK probers have four controls associated with prober start-up: ON, OFF, EMERGENCY off, and LOADER STOP (Figure D-1 shows location).

With a properly installed and setup prober and the OFF switch lamp lit, pushing in the ON switch turns on the prober (the ON switch's lamp will light). To shut down the power supply, push in the OFF switch. The ON switch lamp goes off, and the power supply is removed.

If the prober cannot function (trouble cannot be resolved with normal operations), cycle power to the prober (push in the OFF switch for power-down and then push in the ON switch). The prober will be reset to the state before the startup. (In this resetting, each cassette elevator moves up to the cassette removal height, and wafers remaining on the wafer transfer section of Loader unit are sent back to the inspection tray one by one). With a wafer on the chuck, the chuck moves close to the operation panel's access cover. A prompt for manual removal of the wafer appears on the screen.

If an abnormal and dangerous condition has occurred in the loader unit (actions or the chuck movement), push the EMERGENCY off switch. The prober power supply is immediately shut off, and every movement of prober stops. (At this time, the ON switch lamp goes off, and OFF switch lamp lights up. Note that when starting the prober after an emergency shutdown the ON switch will have no effect.)

To reset the prober power supply circuit condition produced by the EMERGENCY switch:

1. Turn off the main circuit breaker (located behind prober's rear panel).
2. Wait 5 seconds (minimum).
3. Turn main circuit breaker on again. The ON switch will become effective.

Pressing the S. Reset button cycles the prober. To start the system, after making sure the prober has been correctly installed and set up, press the ON button. Reset the main circuit breaker located on the rear of the prober to start the prober after a power loss or after EMERGENCY off has been pressed (Figure D-2). Refer to the appropriate prober manufacturer documentation for detailed operating instructions.

Figure D-1
TSK A-PM-90A controls

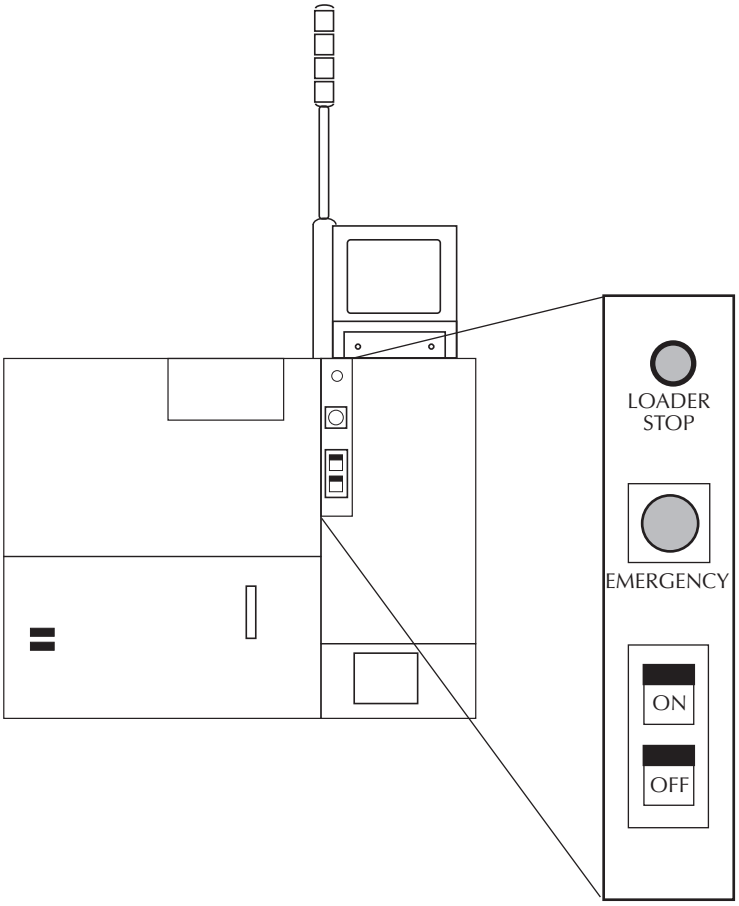
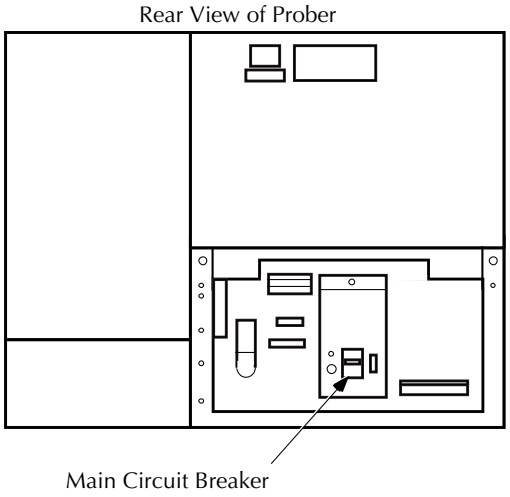


Figure D-2
Main circuit breaker (rear view of prober)



Prober configuration

NOTE *Refer to the specific Prober requirements contained in Section 2 for the software version used to verify the prober driver and configuration contained in this section.*

In order for the GPIB interface to control the prober, the probe must be setup properly. Use the following procedure to set up the prober for GPIB operation and Keithley tester control (these steps MUST be performed after the prober boots up).

1. With the Prober Configuration Data Settings page displayed, press SYSTEM MODE CHANGE.
2. Select ENGINEER (MANAGER or SYSTEM may alternatively be selected, but OPERATOR may not).
3. Select PROBER CONFIGURATION CHANGE (instead of START).
4. Select TESTER INTERFACE SETTINGS.
5. Use the previous page/next page keys to display:
   “Perform Automatic Z Up at 1st Die?: 0” (Select 0:No)
   0: No 1: Yes
6. Use the previous page/next page keys to display and set the following:
   “Prober GP-IB Device Address : 5” (Set address to 5 unless using a different GPIB address)
   “GPIB command for wafer ID: 1” (Select 1:Yes)
   0 (Enable the 'b' command for wafer ID)
   W (Used for needle cleaning)
   “Enable Auto reject after loading: 1” (Select 1:Yes)
7. Use the previous page/next page keys to display and set the following:
   “Allow to travel out of prober area by S” (Select 1:Yes)
   0: No 1: Yes
   This allows the prober to move off the wafer via a PrMove call.
8. Use the previous page/next page keys to display and set the following:
   “And Test Spec” (Select NO)
9. Press “Data Store/Restore.”
10. Select “Data store.”
11. Press execute twice.
12. When “Processing done” appears, press exit and then settings (or end).

Prober setup and initialization

Use the following procedure to set up the prober. Menus and sub-menus are shown below in all capital letters, menu items are shown in upper-lower case, and values are boldface. The three dots indicate menus or menu items nonessential for standard system setup (S530). For a complete listing of the available menu items, refer to the manufacturer's documentation.

1. With the prober and system installed and running (after initialization), change the prober settings listed on the following page.
2. Label a DOS formatted 3-1/2" 1.44 MB floppy disk "Original configuration setup". Insert the floppy into the 3-1/2" drive and save the settings using Data Store (on the prober's touch screen).
3. Save the settings using Data Store to the prober's hard drive. This must be accomplished for each product file (see Note).

NOTE ***Critical:*** *Make sure DEVICE PARAMETER:Probing Mode is set to 2: Regular & Sample for each product file.*

Quadrant setup is accomplished in SETUP SEQUENCE SETTINGS > ***x*** *and* ***y*** *coordinate increments (Table E-1).*

Installation verification program

Use the installation verification program to exercise prober communication functions. If this fails, the installation was not properly completed (files are missing). Use the information below to perform installation verification.

Program name: `prb-TSK9-demo.cpf`
Location: `$KIHOME/src/EXAMPLE/plans`

To run:
Start KTPM.
- Then open `$KIHOME/src/EXAMPLE/plans/prb-TSK9-demo.cpf`

Start KITT.
- Then open `$KIHOME/src/EXAMPLE/pgm/prb-demo.ktm`

Start WDU.
- Then open `$KIHOME/src/EXAMPLE/pgm/prb-demo.wdf`

Table D-1
TSK Prober settings

```
MAIN MENU
      •
      •
      •
      OPERATION SETTINGS
            Alignment mode                          Auto
            Stop after needle align                 Continue
            Stop before probing                     Continue
            Stop after probing                      Continue
            Load/Unload position                    Cassette
            NEEDLE HEIGHT MODE=AUTO
            NEEDLE HEIGHT METHOD=GPIB
            Confirm Auto Needle Height Adjustment NO
            Auto Unload at Align fail
            Alignment Sequence Settings
            Recovery at Alignment Error
            0:Alarm Stop  1:Wafer Reject
            Note: Make sure that the SRQ 76 is enabled.
            •
            •
            •
            PROBER MODE SETTINGS
                     •
                     •
                     •
                     SETUP SEQUENCE SETTINGS
                          X =                              128     Suggested value.
                          Y =                              128     Suggested value.
                          Coordinate origin                  0     center of wafer
                          X coordinate increment             0     to right
                          Y coordinate increment             0     to back
                          Wafer ID Settings
                             Perform Wafer ID Reading
                             0:no  1:yes
                          •
                          •
                          •
                          NEEDLE HEIGHT SETTINGS
                               Method of Needle Height Setting
                                   1 - Auto (SofTouch)
                                   2 - Manual
                               Assort. Of Auto Needle Height Settings
                                   1 - GPIB (SofTouch)
                                   2 - Normal operation
                               Confirm of Auto Needle ht adj.
                                   0 - No
                     EXTERNAL CONTROL SETTINGS
                          METHOD SETTINGS             YES
                          •
                          •
                          •
                          GPIB INTERFACE SETTINGS
                                 GPIB Text Delimiter                  0
                                 Time for STB handshaking             3000ms
                                 Perform STB code handshaking         NO:0
                                 Output EOI                           YES:1
                                 STB code settings
                                      64 GPIB initial setting done                  0:0
                                      75 Prober initial setting done                0
                                      100 Test done received                        0
                                 •
                                 •
                                 •
      DEVICE PARAMETER CHANGE
            •
            •
            •
            Probing Mode           2:Regular & Sample (This needs to be set for each product file)
            •
            •
            •
            UNIT SYSTEM SETTINGS
                 Unit system        metric (Suggested value.)
                 Die size           metric (Suggested value.)
       UTILITIES
            •
            •
            •
            GPIB MANUAL TEST
                 Test end
                 Test start
                 Exit
            •
            •
            •
            System Reset (If GPIB command execution error occurs)
            •
            •
            •
```

TSK9Driver configuration.

– Make sure the $KI_PRB+CONFIG file is set to the correct prober. For example, if a TSK9 prober is being used, a file in #KIDAT called prbcnfg_TSK9.dat should exist. The environment variable, KI_PRB_CONFIG, should be set to $KIDAT/prbcnfg_TSK9.dat.

Command list

The following table lists commands supported by TSK probers. For detailed information about a specific command, refer to Section 5.

Table D-2
TSK Supported commands

Command	
PrAbsMove	PrRelMove
PrAdjustZHeight	
PrCassetteMap	PrRelReturn
PrCheckOptions	PrSerialPoll
PrChuck	PrSetDiam
PrError	PrSetDieSize
PrGetNxtWafer	PrSetFlat
PrGetProduct	PrSetMode
PrGetWafer	PrSetRefDie
PrInit	PrSetTime
PrLoad	PrStatus
PrLoadProduct	PrUnload
PrMove	PrWriteRead
PrNeedleClean	
PrQueryZHeight	
PrSetZHeight	
PrSmifStatus	
PrReadId	PrWriteReadSRQ

Updating prbsqr_TSK9.dat

The prbsrq_TSK9.dat file is used by the TSK9 driver. Use this file to define the known SRQ responses for a given prober on a command-by-command basis. The list for a given command is defined as follows:

- The function name
- The good SRQ list
- The bad SRQ list
- The known prober error returns. Note that the known prober error numbers are defined by the prober using the PrError() command (see Table 5-2 on page 5-3).

Refer to Figure 7-26 for a sample prbsrq_TSK9.dat (S530 sample). To add a function:

1. Open prbsrq_TSK9.dat using a text editor (such as notepad).
2. Save a backup with a new file name (e.g., prbsrq_TSK9.bak).
3. Place the new function and a comma somewhere after the <EOH> symbol (functions currently in the file are in alphabetical order).
4. Inside quotes, add comma-separated good SRQ's, a semicolon, comma separated bad SRQ's, a semicolon, and known prober error numbers. The use of semicolons to designate between the three different fields is required.
5. Save the file as prbsrq_TSK9.dat.

An example function PRLOAD is contained in Figure D-3:

Figure D-3
Example updating prbsqr_TSK9.dat

```
•
•
•
<EOH>
•
•
•
PRLOAD,"70,71,72,82,84,92,94,105;0;1057,1181,1184,1860"
<EOLOC>
```

This function (PRLOAD) has:

- Good SRQ's of 70-72, 82, 84, 92, 94, and 105.
- Bad SRQ's of 0.
- And known prober errors of 1057, 1181, 1184, and 1860.

Error symbols and returned values table

After calling a prober command as a function, a value or symbol is returned. Table D-3 contains error symbols and return values associated with specific commands.

Table D-3
Error symbols and returned values

Return Value	Commands												Error Symbol
	PrAbsMove	PrCassetteMap	PrCheckOptions	PrChuck	PrError	PrGetWafer	PrGetNxtWafer	PrInit	PrLoad	PrLoadProduct	PrMove	PrReadId[4]	
0, +1	√	√	√	√	√			√	√	√			PR_OK
+2											√		PR_MOVECOMPLETE
+4									√				PR_WAFERCOMPLETE
+8									√				PR_CASSETTECOMPLETE
+10									√				PR_LOTEND
+12									√				PR_WAFER_REJECT_LOAD
+14						√	√		√				PR_WAFER_REJECT_NOLOAD
+16													PR_OPER_ACTION
-1001													BAD_TST_NUM
-1002													BAD_CONFIG_DAT
-1003													FEAT_NOT_SUPPORT
-1004													PORT_ASS_FAIL
-1005								√					SET_UNITS_FAIL
-1006													INVAL_MODE
-1007													CLR_WAF_MAP
-1008								√					SET_MODE_FAIL
-1009													SET_DIE_FAIL
-1010													SET_PRESET_FAIL
-1011													BAD_MODE
-1012													TST_COMPL_FAIL
-1013		√			√		√					√	UNINTEL_RESP
-1014	√			√							√		MOVE_FAIL
-1015											√		UNEXPE_ERROR
-1016					√	√	√		√				LOAD_ERROR
-1017				√									BAD_CHUCK
-1018													CHUCK_NOT_UP
-1019													INK_FAIL
-1020				√	√	√	√		√			√	TIME_GONE
-1021													BAD_LEARN_FUNCT
-1022													SET_DIAM_FAIL
-1023													SET_MATRIX_FAIL
-1024													BAD_Z_PARAM
-1025													NO_RESPONSE
-1026													TOO_MANY_NAKS
-1027		√								√			INVAL_PARAM
-1028										√			ERR_LOAD_FILE
-1029						√	√		√				ALIGN_FAIL
-1030	√			√						√	√		GPIB_ERROR
-1031													CHK_PORT_ASS
-1032													CHK_DEV_PROT
-1033													PR_MEM_ALLOC_ERR
-1034		√					√						PR_NO_CASSETTE
-1035													SET_FLAT_FAIL
-1036													SET_REF_DIE_FAIL
-1037													Z_PARAM_FAIL
-1038													ERR_OPEN_PRBCNFG
-1039													NO_CNFG_DATA
-1040													NO_PRB_TYPE
-1041													INVAL_PRB_MODE
-1042													INVAL_PRB_OPTIONS
-1043													INVAL_DEVICE_NAME
-1044													INVAL_DEVICE_IRQ
-1045													INVAL_BAUD_RATE

[1]**PrGetNxtWafer:** Returns the slot number.
[2]**PrPutNxtSlot:** Returns the slot number.
[3]**PrPutWafer:** Returns the slot number.
[4]**PrReadId:** Returns the number of characters received (>0).
[5]**Prober error:** Add 1500 to the error number returned and then refer to the prober manufacturer for error handling.

Table E-3 (continued)

Error symbols and returned values

Return Value	Commands												Error Symbol
	PrAbsMove	PrCassetteMap	PrCheckOptions	PrChuck	PrError	PrGetWafer	PrGetNxtWafer	PrInit	PrLoad	PrLoadProduct	PrMove	PrReadId[4]	
-1046													INVAL_TIMEOUT
-1047													INVAL_GPIB_UNIT
-1048													INVAL_GPIB_SLOT
-1049													INVAL_GPIB_ADDRESS
-1050													INVAL_GPIB_WRITE_MODE
-1051													INVAL_GPIB_READ_MODE
-1052													INVAL_GPIB_TERMINATOR
-1053													INVAL_KULT_PATH
-1054													INVAL_PRB_LIB
-1055													INVAL_PRB_CNFG_FUNC
-1056						√							INVAL_PRB_MAX_SLOTS
-1057						√	√						INVAL_PRB_MAX_CASSETTES
-1058													SET_QUAD_FAIL
-1059							√						NO_UNPROBED_WAFERS
-1060													SET_TEMPERATURE_FAIL
-1061													QUERY_TEMPERATURE_FAIL
-1081													INVAL_PRB_P8_OPTION
-1082													NI_UNIX_ERROR
-1083													NI_BOARD_NOT_CIC
-1084													NI_NO_LISTENER
-1085													NI_BOARD_NOT_ADDRESSES
-1086													NI_INVALID_FUNCTION_ARG
-1087													NI_BOARD_NOT_SYS_CONTROLLER
-1088													NI_IO_ABORTED_TIMO
-1089													NI_BOARD_NOT_PRESENT
-1090													NI_DMA_HW_PROBLEM
-1091													NI_DMA_HW_TIMO
-1092													NI_NO_CAPABILITY
-1093													NI_FILE_SYS_ERROR
-1094													NI_GPIB_BUS_ERROR
-1095													NI_SRQ_BYTE_Q_OVERFLOW
-1096													NI_SRQ_STUCK_ON
-1097													NI_TABLE_PROBLEM
-1500 & on					√								Prober error[5]

[1]**PrGetNxtWafer:** Returns the slot number.
[2]**PrPutNxtSlot:** Returns the slot number.
[3]**PrPutWafer:** Returns the slot number.
[4]**PrReadId:** Returns the number of characters received (>0).
[5]**Prober error:** Add 1500 to the error number returned and then refer to the prober manufacturer for error handling.

Table E-3 (continued)
Error symbols and returned values

Return Value	Commands													Error Symbol
	PrRelMove	PrRelReturn	PrSerialPoll[6]	PrSetDiam	PrSetDieSize	PrSetFlat	PrSetMode	PrSetRefDie	PrSetTime	PrStatus	PrUnload	PrWritetead[7]	PrWritereadSRQ[7]	
0, +1	√	√		√	√	√	√	√	√	√	√			PR_OK
+2														PR_MOVECOMPLETE
+4														PR_WAFERCOMPLETE
+8														PR_CASSETTECOMPLETE
+10											√			PR_LOTEND
+12														PR_WAFER_REJECT_LOAD
+14											√			PR_WAFER_REJECT_NOLOAD
+16														PR_OPER_ACTION
-1001														BAD_TST_NUM
-1002														BAD_CONFIG_DAT
-1003														FEAT_NOT_SUPPORT
-1004														PORT_ASS_FAIL
-1005														SET_UNITS_FAIL
-1006														INVAL_MODE
-1007														CLR_WAF_MAP
-1008							√							SET_MODE_FAIL
-1009				√	√									SET_DIE_FAIL
-1010														SET_PRESET_FAIL
-1011														BAD_MODE
-1012														TST_COMPL_FAIL
-1013										√				UNINTEL_RESP
-1014	√	√												MOVE_FAIL
-1015														UNEXPE_ERROR
-1016											√			LOAD_ERROR
-1017														BAD_CHUCK
-1018														CHUCK_NOT_UP
-1019														INK_FAIL
-1020			√							√	√	√	√	TIME_GONE
-1021														BAD_LEARN_FUNCT
-1022				√										SET_DIAM_FAIL
-1023														SET_MATRIX_FAIL
-1024														BAD_Z_PARAM
-1025														NO_RESPONSE
-1026														TOO_MANY_NAKS
-1027					√	√	√		√		√			INVAL_PARAM
-1028														ERR_LOAD_FILE
-1029											√			ALIGN_FAIL
-1030	√	√								√	√	√	√	GPIB_ERROR
-1031														CHK_PORT_ASS
-1032														CHK_DEV_PROT
-1033														PR_MEM_ALLOC_ERR
-1034														PR_NO_CASSETTE
-1035						√								SET_FLAT_FAIL
-1036								√						SET_REF_DIE_FAIL
-1037														Z_PARAM_FAIL
-1038														ERR_OPEN_PRBCNFG
-1039														NO_CNFG_DATA
-1040														NO_PRB_TYPE
-1041														INVAL_PRB_MODE
-1042														INVAL_PRB_OPTIONS
-1043														INVAL_DEVICE_NAME
-1044														INVAL_DEVICE_IRQ
-1045														INVAL_BAUD_RATE
-1046														INVAL_TIMEOUT
-1047														INVAL_GPIB_UNIT
-1048														INVAL_GPIB_SLOT
-1049														INVAL_GPIB_ADDRESS
-1050														INVAL_GPIB_WRITE_MODE

[5]**Prober error:** Add 1500 to the error number returned and then refer to the prober manufacturer for error handling.
[6]**PrSerialPoll:** Returns the SRQ status byte (>0).
[7]**PrWriteRead:** Returns the number of characters received (>0).

Table E-3 (continued)
Error symbols and returned values

Return Value	Commands													Error Symbol
	PrRelMove	PrRelReturn	PrSerialPoll[6]	PrSetDiam	PrSetDieSize	PrSetFlat	PrSetMode	PrSetRefDie	PrSetTime	PrStatus	PrUnload	PrWriteread[7]	PrWritereadSRQ[7]	
-1051														INVAL_GPIB_READ_MODE
-1052														INVAL_GPIB_TERMINATOR
-1053														INVAL_KULT_PATH
-1054														INVAL_PRB_LIB
-1055														INVAL_PRB_CNFG_FUNC
-1056														INVAL_PRB_MAX_SLOTS
-1057														INVAL_PRB_MAX_CASSETTES
-1058														SET_QUAD_FAIL
-1059														NO_UNPROBED_WAFERS
-1060														SET_TEMPERATURE_FAIL
-1061														QUERY_TEMPERATURE_FAIL
-1081														INVAL_PRB_P8_OPTION
-1082														NI_UNIX_ERROR
-1083														NI_BOARD_NOT_CIC
-1084														NI_NO_LISTENER
-1085														NI_BOARD_NOT_ADDRESSES
-1086														NI_INVALID_FUNCTION_ARG
-1087														NI_BOARD_NOT_SYS_CONTROLLER
-1088														NI_IO_ABORTED_TIMO
-1089														NI_BOARD_NOT_PRESENT
-1090														NI_DMA_HW_PROBLEM
-1091														NI_DMA_HW_TIMO
-1092														NI_NO_CAPABILITY
-1093														NI_FILE_SYS_ERROR
-1094														NI_GPIB_BUS_ERROR
-1095														NI_SRQ_BYTE_Q_OVERFLOW
-1096														NI_SRQ_STUCK_ON
-1097														NI_TABLE_PROBLEM
-1500 & on														Prober error[5]

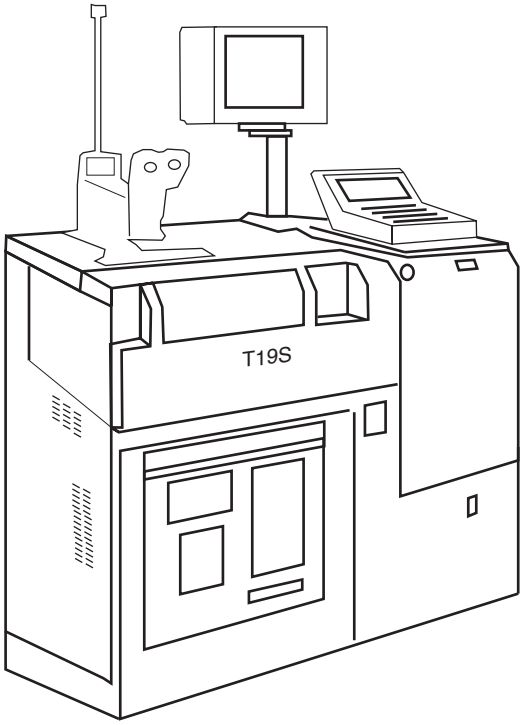
[5]**Prober error:** Add 1500 to the error number returned and then refer to the prober manufacturer for error handling.
[6]**PrSerialPoll:** Returns the SRQ status byte (>0).
[7]**PrWriteRead:** Returns the number of characters received (>0).

E
T19S (GPIB Driver) Prober Information

Prober start-up

This appendix covers T19S type probers using a serial interface (Figure E-1). Refer to the prober manufacturer's documentation for specific start-up information.

Figure E-1
T19S prober



Additional Documentation

In addition to this manual, refer to the T19S User's Guide.

Prober setup

NOTE *Refer to the specific Prober requirements contained in Section 2 for the software version used to verify the prober driver and configuration contained in this section.*

Use the information in this section to set up the prober.

Prober DIP switch settings

In order for the prober to properly communicate with the system, specific DIP settings are required. This section details the specific settings for a GPIB system.

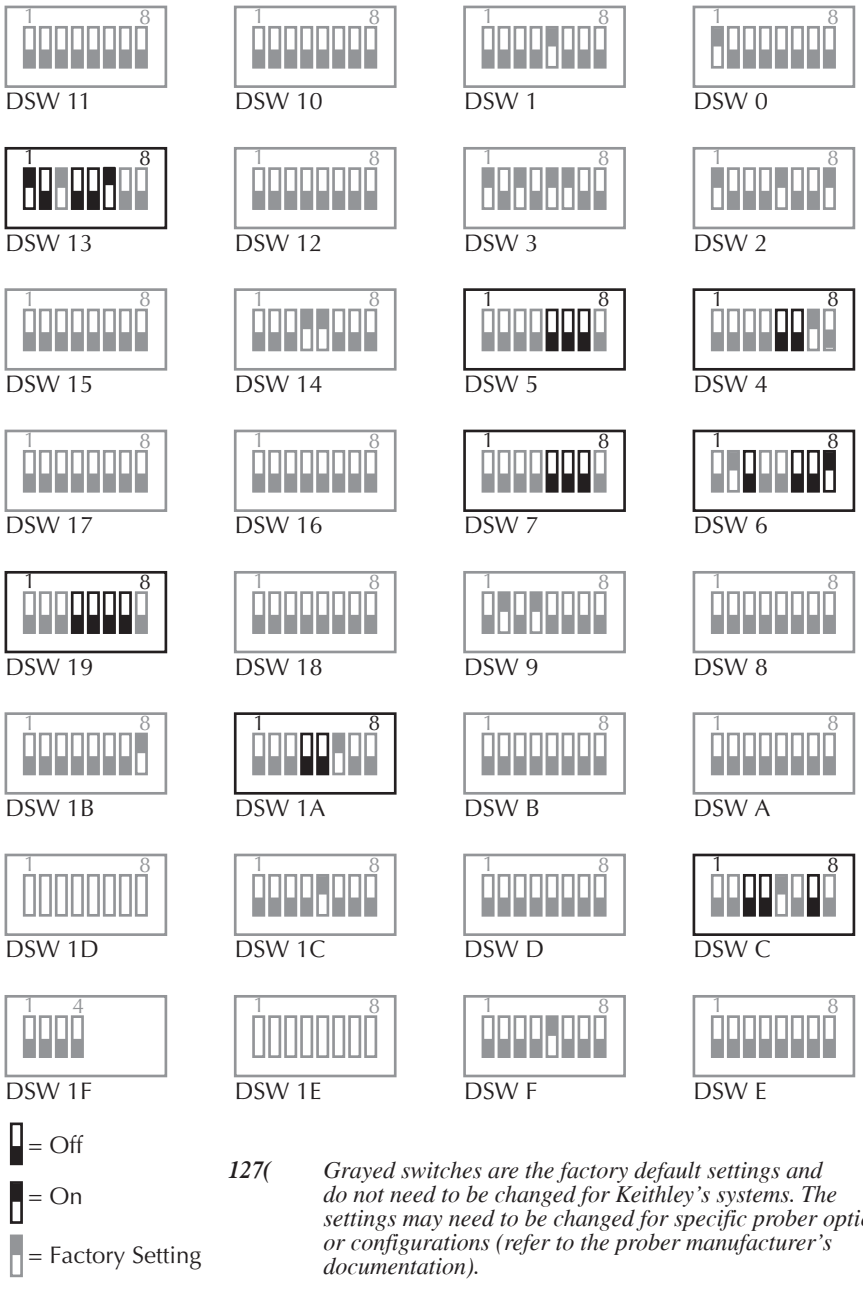
Make sure all DIP switches are properly set according to the plant specific configuration and options for the prober. Then, make changes to the DIP switches for Keithley's parametric test system. Refer to Figure E-2 through Figure E-4 for DIP switch settings. If the

optional line printer is installed, prober function L1 will print the present state of all the probers DIP switches. To use this function, enter:

```
[CLR][L][1][ENT]
```

and follow the instructions on the prober display.

Figure E-2
Bus Distribution board



127(*Grayed switches are the factory default settings and do not need to be changed for Keithley's systems. The settings may need to be changed for specific prober options or configurations (refer to the prober manufacturer's documentation).*

Table E-1 details uses for specific Bus Distribution board switches (Figure E-2).

Table E-1
Bus distribution switches

Switch Number	DIP	Description (* = default)
DSW4	Switch 5	Select positive Y coordinate direction: On = Up, Chuck movement from back to front. Off = Down, Chuck movement from front to back.
	Switch 6	Select positive X coordinate direction: On = Up, Chuck movement from left to right. Off = Down, Chuck movement from right to left.
DSW5	Switch 5	Issues TTL TEST START after a Q, r, or d command is received from the host and an SPSC47H (Index Complete) is issued from the prober: On = Send TTL TEST START. *Off = No TTL TEST STMLT sent. Used with GPIB / TTL communications (see DSW 19-5 and DSW 19-7).
	Switch 6	Determines when pass / fail & real-time wafer map data is processed: On = Processed after receiving a C or C command. *Off = Processed after receiving a Q, r, or d command. Used with GPIB communications only.
	Switch 7	Communication termination code: On = EOI termination. *Off = CR/LF termination. GPIB only.
NOTE *This switch (DSW6-3) may be used to make left-hand probers use right-hand prober setup file without changing parameters. Used with DSW 3-7, DSW 4-5, DSW 4-6, DSW 7-5, DSW 7-6. Notch pre-alignment can be enabled via autoloader DSW 1-8.*		
DSW6	Switch 3	Wafer flat/notch modification: On = Wafer will be loaded onto the main chuck rotated 180° from value defined in setup file parameter 3FLAT ORIENTATION. *Off = Wafer will be loaded onto the main chuck with the flat (notch location defined in setup file parameter 3 FLAT ORIENTATION. Occasionally used to make left hand probers use right hand prober setup file without changing Parameter 3. Used with DSW 3-7, DSW 4-5, DSW 4-6, DSW 7-5, DSW 7-6. Notch pre-alignment can be enabled via autoloader DSW 1-8.
	Switch 6	Use of SPSCs 4 EH (Assist Error) and 48H (End of Lot): *On = SPSC 48H is issued at end of lot (EOL). Off = SPSC 48H, then 4EH issued at EOL. GPIB. If DSW 1A-5 and autoloader DSW 10-6 are not both off, no EOL message or SPSC 48H sent by the prober. In v2.6X and v3.1X, 48H will not appear if DSW 13-5 on and a U, u, or V command is used to unload the wafer.
	Switch 7	Issue SPSC 4AH (Initial Wafer): On = 4AH issued from every new lot. *Off = 4AH issued for new lot only if `Wafer Name Received = Yes` (U6). GPIB. If On, 4AH issued whether `Wafer Name Receive=Yes` or `No`. If Off, 4AH not issued if `No`. If `Yes`, 4AH issued in response to N or H commands only.
	Switch 8	Indexing outside the defined probe area: On = Off-wafer indexing permitted. *Off = Off-wafer indexing not permitted, and SPSC 4FH (Off Wafer) is set. GPIB. Affects prober response to X, Y, and J commands when directed to index off wafer.
DSW7	Switch 5	Y location from which probing will begin (top row or bottom row of wafer): On = Start probing from bottom of wafer (closest to the front of the prober) *Off = Start probing from top of wafer (closest to rear of prober)
	Switch 6	X location from which probing will begin (left or right side of the wafer): On = Start probing from rightmost edge of wafer on the chuck. *Off Start probing from left most edge of wafer on the chuck. Location independent of DSW 6-3, DSW 4-5, DSW 4-6, and DSW 3-7 settings. DSW 7-5 and DSW 7-6 are used with each other to determine in which quadrant of the wafer probing will begin.

Table E-1 (cont.)
Bus distribution switches

	Switch 7	Selection of reference die location: *On = Use X/Y value declared in setup Parameter 60 as reference die. Off = Top row has coordinate Y = 5, leftmost column has coordinate X = 5. U21 ALIGN REFERENCE MASK must be set to NO, and setup menu 21 must be set to CENT or TARGET for the reference die in Parameter 60 to be accurately maintained.
DSWC	Switch 3	Wafer flat/notch modification: On = Wafer will be loaded onto the main chuck rotated 180° from value defined in setup file parameter 3 FLAT ORIENTATION
	Switch 4	Control Z height from host: On = Enable. *Off = Disable. GPIB only.
	Switch 7	Issue SPSC 50H (error) upon any loader error: On = Enable. *Off = Disable. GPIB only.
DSW13	Switch 1	SRQs issued by prober: On = Enable. *Off = Disable. GPIB only.
	Switch 2	Require a TTL `TEST COMPLETE` from tester before a Q or r causes the prober to index: On = Enable. *Off = Disable. GPIB / TTL.
	Switch 4	Issue TTL `TEST START` along with END OF LOT SIGNAL: On = Enable. *Off = Disable. GPIB / TTL.
	Switch 5	Wafer unload/load control: On = Unload wafer at end of wafer. *Off Requires GPIB or RDP command before prober will handle wafer.
	Switch 6	In real time, display GPIB response and SPSC values to LCD: ON = Display GPIB responses and SPSCs. *Off = Do not display GPIB response or SPSCs to LCD. GPIB only. Requires software version 2.52 or higher. DSW 13-6 set to Off save 6 ms / command in communication time.
DSW19	Switch 4	Display `WAFER NAME RECEIVE` on LCD (online / offline): On = Enable. *Off = Disable. GPIB only. See the KLA 1007 GPIB Communications Manual.
	Switch 5	Issue `TEST START` after receiving a Q or r command but before sending an SPSC 47H (Index Complete): On = Enable. *Off = Disable. GPIB only. DSW 5-5 must be in the opposite state of this setting.
	Switch 6	Display W-g command while first wafer is being aligned: On = Enable. *Off = Disable. GPIB only.
	Switch 7	Issue TTL TEST START before all SPSCs except 4AH (Initial Wafer): On = Enable. *Off = Disable. GPIB only.
DSW1A	Switch 4	Initialization diagnostic: locates chuck center on power up: On = Bypass chuck center search. *Off = Perform chuck center search.
	Switch 5	Used with autoloader for wafer cycling. LOT END signal generation: On = No LOT END signal. *Off = Issue LOT END signal. DSW 10-6 set to On.

Table E-2 details uses for specific Front Panel board switches (Figure E-3).

Table E-2
Front panel board switches

Switch Number	DIP	Description (* = default)
DSW10	Switch 8	Carrier Type: *On = Plastic, Off = Metal.

Table E-3 details uses for specific Autoloader board switches (Figure E-4).

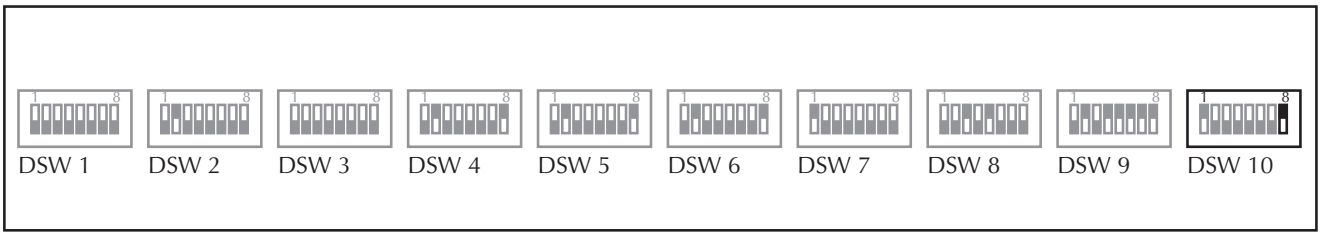
Table E-3
Autoloader board switches

Switch Number	DIP	Description (* = default)
DSW1	Switch 8	Wafer pre-align: On = Notched wafer pre-align. *Off: Flat wafer pre-align.
DSW10	Switch 3	Wafer return mode: On = Return wafers to consecutive slots. *Off = Return wafers to original slots.
	Switch 4	Wafer fetch mode: On = Fetch wafers from top of cassette. *Off = Fetch wafers from bottom of cassette.

Figure E-3
Front Panel board

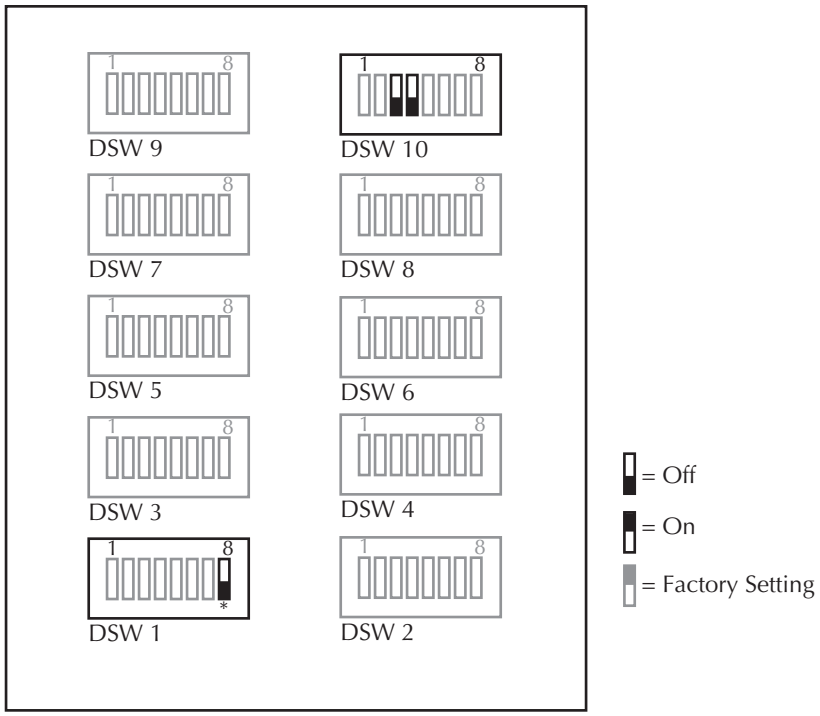
127(Grayed switches are the factory default settings and do not need to be changed for Keithley's systems. The settings may need to be changed for specific prober options or configurations (refer to the prober manufacturer's documentation).

127(DSW 10-8 On for plastice cassettes
DSW 10-8 Off for metal cassettes



= Off
= On
= Factory Setting

Figure E-4
Autoloader board



127(*DSW 1-8: Off=flat; On=notch*

Grayed switches are the factory default settings and do not need to be changed for Keithley's systems. The settings may need to be changed for specific prober options or configurations (refer to the prober manufacturer's documentation).

Menu setups

Make sure when setting up the prober that the following line items have been properly set.

NOTE *Settings that are not listed in this section are either device dependent or user dependent. Refer to the appropriate prober manufacturer's documentation for information on these functions.*

```
02 WAFER SIZE?
03 FLAT ORIENTATION? (degrees)
04 X INDEX
05 Y INDEX
13 INSPECTION? NO
30 INKER OFFSET? 0
```

In addition to these items, the following user settings are required.

```
U003
AUTO STOP SEQUENCE ?N

U005
Z UP WITH SW TEST START

U006
REMOTE COMM. OPERATION ?Y
SELECT PROTOCOL ?GPIB
STOP AT REFERENCE DIE ?Y
WAFER NAME RECEIVE ?N
TTL INTERFACE ON ?N

U010
INDEX SIZE? INCH, METRIC

U021
ALIGN REFERENCE MASK ?N
```

GPIB setup

Set the GPIB interface board (supplied with the prober) to the desired GPIB address (5 is Keithley's default setting). Refer to the prober manufacturer for information on setting the GPIB address.

Once the GPIB address has been set, setup the tester for GPIB communication. This can be accomplished by editing KIDAT/prbcnfg_T19S.dat for the prober communication settings (refer to Section 7 for sample files).

Installation verification program

Use the installation verification program to exercise prober communication functions. If this fails, the installation was not properly completed (files are missing). Use the information below to perform installation verification.

Program name: `prb-T19S-demo.cpf`
Location: `$KIHOME/src/EXAMPLE/plans`

To run:
Start KTPM.
- Then open `$KIHOME/src/EXAMPLE/plans/prb-T19S-demo.cpf`

Start KITT.
- Then open `$KIHOME/src/EXAMPLE/pgm/prb-demo.ktm`

Start WDU.
- Then open `$KIHOME/src/EXAMPLE/pgm/prb-demo.wdf`

T19S Driver configuration.
- Make sure the $KI_PRB_CONFIG file is set to the correct prober. For example, if a T19S prober is being used, a file in $KIDAT called prbcnfg_T19S.dat should exist. The environment variable, KI_PRB_CONFIG, should be set to $KIDAT/prbcnfg_T19S.dat.

Command list

The following table lists commands supported by T19S (GPIB driver) probers. For detailed information about a specific command, refer to Section 5.

Table E-4
T19S GPIB driver supported commands

Command	
PrAbsMove	PrMovNxt
PrCassetteMap	PrRelMove
PrCheckOptions	PrRelReturn
PrChuck	PrSerialPoll
PrError	PrSetTime
PrInit	PrStatus
PrInk	PrWriteRead
PrLoad	PrWriteReadSRQ
PrMove	

Error symbols and returned values table

After calling a prober command as a function, a value or symbol is returned. Table E-5 contains error symbols and return values associated with specific commands.

Table E-5
Error symbols and returned values

Return Value	Commands																Error Symbol
	PrAbsMove	PrCassetteMap	PrCheckOptions	PrChuck	PrError	PrInit	PrLoad	PrMove	PrMovNxt	PrRelMove	PrRelReturn	PrSerialPool	PrSetTime	PrStatus	PrWriteRead [2]	PrWriteReadSRQ [2]	
0, +1	√	√	√	√	√	√	√										PR_OK
+2								√									PR_MOVECOMPLETE
+4							√										PR_WAFERCOMPLETE
+8							√										PR_CASSETTECOMPLETE
+10							√										PR_LOTEND
+12																	PR_WAFER_REJECT_LOAD
+14																	PR_WAFER_REJECT_NOLOAD
+16																	PR_OPER_ACTION
-1001																	BAD_TST_NUM
-1002																	BAD_CONFIG_DAT
-1003																	FEAT_NOT_SUPPORT
-1004																	PORT_ASS_FAIL
-1005						√											SET_UNITS_FAIL
-1006																	INVAL_MODE
-1007																	CLR_WAF_MAP
-1008						√											SET_MODE_FAIL
-1009																	SET_DIE_FAIL
-1010																	SET_PRESET_FAIL
-1011								√									BAD_MODE
-1012																	TST_COMPL_FAIL
-1013	√			√	√	√	√	√	√	√	√			√	√	√	UNINTEL_RESP
-1014	√							√	√	√	√						MOVE_FAIL
-1015	√			√	√		√			√	√						UNEXPE_ERROR
-1016																	LOAD_ERROR
-1017				√													BAD_CHUCK
-1018																	CHUCK_NOT_UP
-1019								√									INK_FAIL
-1020	√				√	√	√	√	√	√	√			√	√	√	TIME_GONE
-1021																	BAD_LEARN_FUNCT
-1022																	SET_DIAM_FAIL
-1023																	SET_MATRIX_FAIL
-1024																	BAD_Z_PARAM
-1025																	NO_RESPONSE
-1026																	TOO_MANY_NAKS
-1027		√											√				INVAL_PARAM
-1028						√											ERR_LOAD_FILE
-1029																	ALIGN_FAIL
-1030																	GPIB_ERROR
-1031																	CHK_PORT_ASS
-1032																	CHK_DEV_PROT
-1033																	PR_MEM_ALLOC_ERR
-1034																	PR_NO_CASSETTE
-1035																	SET_FLAT_FAIL
-1036																	SET_REF_DIE_FAIL
-1037																	Z_PARAM_FAIL
-1038																	ERR_OPEN_PRBCNFG
-1039																	NO_CNFG_DATA

[1] **PrReadId:** Returns the number of characters received (>0).
[2] **PrWriteRead / PrWriteReadSRQ:** Returns the number of characters received (>0).
[3] **Prober error:** Add 1500 to the error number returned and then refer to the prober manufacturer for error handling.

Table E-5 (continued)

Error symbols and returned values

Return Value	Commands																Error Symbol
	PrAbsMove	PrCassetteMap	PrCheckOptions	PrChuck	PrError	PrInit	PrLoad	PrMove	PrMovNxt	PrRelMove	PrRelReturn	PrSerialPool	PrSetTime	PrStatus	PrWriteRead [2]	PrWriteReadSRQ [2]	
-1040																	NO_PRB_TYPE
-1041																	INVAL_PRB_MODE
-1042																	INVAL_PRB_OPTIONS
-1043																	INVAL_DEVICE_NAME
-1044																	INVAL_DEVICE_IRQ
-1045																	INVAL_BAUD_RATE
-1046																	INVAL_TIMEOUT
-1047																	INVAL_GPIB_UNIT
-1048																	INVAL_GPIB_SLOT
-1049																	INVAL_GPIB_ADDRESS
-1050																	INVAL_GPIB_WRITE_MODE
-1051																	INVAL_GPIB_READ_MODE
-1052																	INVAL_GPIB_TERMINATOR
-1053																	INVAL_KULT_PATH
-1054																	INVAL_PRB_LIB
-1055																	INVAL_PRB_CNFG_FUNC
-1056																	INVAL_PRB_MAX_SLOTS
-1057																	INVAL_PRB_MAX_CASSETTES
-1058																	SET_QUAD_FAIL
-1059																	NO_UNPROBED_WAFERS
-1060																	SET_TEMPERATURE_FAIL
-1061																	QUERY_TEMPERATURE_FAIL
-1081																	INVAL_PRB_P8_OPTION
-1082																	NI_UNIX_ERROR
-1083																	NI_BOARD_NOT_CIC
-1084																	NI_NO_LISTENER
-1085																	NI_BOARD_NOT_ADDRESSES
-1086																	NI_INVALID_FUNCTION_ARG
-1087																	NI_BOARD_NOT_SYS_CONTROLLER
-1088																	NI_IO_ABORTED_TIMO
-1089																	NI_BOARD_NOT_PRESENT
-1090																	NI_DMA_HW_PROBLEM
-1091																	NI_DMA_HW_TIMO
-1092																	NI_NO_CAPABILITY
-1093																	NI_FILE_SYS_ERROR
-1094																	NI_GPIB_BUS_ERROR
-1095																	NI_SRQ_BYTE_Q_OVERFLOW
-1096																	NI_SRQ_STUCK_ON
-1097																	NI_TABLE_PROBLEM
-1500 & on					√												Prober error [3]

[1] **PrReadId:** Returns the number of characters received (>0).
[2] **PrWriteRead / PrWriteReadSRQ:** Returns the number of characters received (>0).
[3] **Prober error:** Add 1500 to the error number returned and then refer to the prober manufacturer for error handling.

F
GPIB Adapter Installation

GPIB adapter installation

NI-488.2 software installation procedure

To install the software, follow the instructions in the National Instruments Driver installation notes that are included with the board.

GPIB Configuration

1. Start ibconf with the command:

   /usr/local/natinst/ni4882/bin/gpibexplorer <return>

2. Select the GPIB-USB interface
3. Select the properties button
4. Select the "advanced" tab
5. Uncheck the "Automatic Serial Polling" checkbox

A

Acquiring prober data 5-28
Alignment errors 4-7
Alternative to adding a new prober function 6-14
ATTENTION heading 1-3
Auditing Tester/Prober communications 4-2

C

Cable pinouts
    IEEE-488 contact designation 2-5
Calling prober commands as functions 5-27
CAUTION heading 1-3
Chuck movement 3-4
CMNDS command 4-2
Command format 5-2
command summary and syntax 5-2
Configuration files
    (kth.ini and prbcnfg_xxxx.dat) 2-6
Connection 2-4
    IEEE-488 connection 2-4

D

Debugging probers 6-14
determining available prober options/commands . 3-6
DIP switch settings
    T19S E-2
DisableTransactionLogging 4-2
DisableTransactionLogging command 4-2
DisplayErrorMsg User Defined (stdout) 4-2

E

Electroglas prober information A-2, B-2
EnableErrorLogging(x) command 4-2
EnableTransactionLogging command 4-2
Enabling the prober transaction log 6-15
Environment variable KI_PRB_CONFIG C-5
Environment variables used to debug probers 4-2
Error detection when writing code 4-8
Error Level -- KI_PRB_ERROR_LEVEL 0-3 4-2
Error messages and recovery procedures 4-3
Error return values 4-4
Example
    Error detection when writing code 4-8
    Handling alignment errors 4-7
    return values 4-6
Execution engine 3-6
    exit 3-8
External mode 3-5

F

FAKE prober debug 6-16

G

General prober files 7-2
GPIB Configuration F-2

H

Handling alignment errors 4-7

I

Initializing prober 3-6

K

KI_PRB_DEBUG command 4-2
KTXE prober interaction 3-6

M

Manual contents 1-2
manual organization 1-2
Mechanical interfacing 2-4
Modifying prober software 6-2
    Adding a function to an existing prober driver ... 6-3
    Linux 6-2

N

NI-488.2 software installation F-2
Noise problems 2-4
Non-printable ASCII characters 6-18

P

Parameter data type 5-2
Positive return value 4-3
PrAbsMove 5-3, 5-7
PrAdjustZHeight 5-3, 5-7
PrAutoAlign 5-3, 5-8
prbcnfg.dat fields 7-4
PrCassetteMap 5-3, 5-8
PrCassetteMask 5-3
PrCheckOptions 5-3, 5-9
    command 3-6
    definition 3-9
PrChuck 5-3, 5-9
PrClearAll 5-3
PrClearPipeline 5-3, 5-10
PrError 5-3, 5-10
PrGetNxtWafer 5-3, 5-10
PrGetProduct 5-3
PrGetWafer 5-4, 5-11
PrInit 5-4, 5-11
PRINT command 4-2
PrLoad 5-4, 5-12
PrLoadProduct 5-4, 5-12
PrLowerBoat 5-4
PrMove 5-4, 5-13
PrNeedleClean 5-4
Probe die 3-3
Prober GPIB connection B-2
Prober interactions in KTXE 3-9
Prober mode of operation 3-5
Prober requirements 2-2
    Electroglas probers 2-2
    General (all prober models) 2-2
    Tel P8 probers 2-3

TSK probers 2-4
PrProfile 5-4, 5-13
PrPutNxtSlot 5-4, 5-14
PrPutWafer 5-4
PrQueryChuckTemp 5-4, 5-15
PrQueryZHeight 5-4, 5-15
PrReadId 5-4, 5-16
PrRelMove 5-4, 5-16
PrRelReturn 5-4, 5-16
PrSerialPoll 5-5, 5-17
PrSetChuckTemp 5-5, 5-17
PrSetDiam 5-5, 5-17
PrSetDieSize 5-5, 5-17
PrSetFlat 5-5, 5-18
PrSetMode 5-5, 5-19
PrSetPipeline 5-5, 5-19
PrSetQuadrant 5-5, 5-19
PrSetRefDie 5-5, 5-20
PrSetSlotStatus 5-5, 5-20
PrSetTime 5-5, 5-21
PrSetUnits 5-5, 5-21
PrSetZHeight 5-5
PrSmifClamp 5-5
PrSmifLock 5-5
PrSmifStatus 5-5
PrStart 5-5, 5-16
PrStatus 5-6
PrStop 5-5, 5-17
PrUnLoad 5-6, 5-24
PrWriteRead 5-6
PrWriteReadSRQ 5-6
PrZParams 5-6, 5-26
PrZTravel 5-6, 5-27

Q

Quick reference table 5-3

R

Return values 4-3
Ring Carrier/Top plate
Electroglas 2-2, 2-3
Tel P8 2-3
TSK 2-4

S

Safety symbols and terms 1-3
Sample
EG2X prbcnfg.dat 7-4
EG40 GPIB log 6-17
EG40 serial log 6-18
EG4X driver prbcnfg.dat 7-5, 7-6
prbcnfg_nnnn.dat 6-5
probe die locations 3-3
T19S prbcnfg.dat 7-15
TELP8 Log 7-8
TELP8 prbcnfg.dat 7-8
TSK9 Log 7-11
TSK9 prbcnfg.dat 7-11
Software version
Electroglas 2-2, 2-3
Tel P8 2-3

T

T19S Prober information E-2
Target die 3-2
Tel P8 prober information C-2
Trademarks 1-2
TRANS command 4-2
transaction logging 6-15
TSK Prober information D-2
Typical cycle of execution 3-8

U

UAP_PROBER_INIT (definition) 3-9
Useful commands 4-2
Using a UAP for error recovery 4-8
Using the transaction log to isolate problems 6-19

V

Verbose System Defined (stdout) 4-2

W

Wafer loop 3-7
WARNING heading 1-3

Keithley Instruments, Inc.
Corporate Headquarters • 28775 Aurora Road • Cleveland, Ohio 44139 • 440-248-0400 • Fax: 440-248-6168 • 1-888-KEITHLEY • www.keithley.com

KEITHLEY
A Tektronix Company

A Greater Measure of Confidence