CalPower
Via Acquanera, 29 22100 COMO
tel. 031.526.566 (r.a.) fax 031.507.984
info@calpower.it www.calpower.it

DEVELOPER'S GUIDE

KLR SERIES

POWER SUPPLY

2400 WATT PROGRAMMABLE POWER SUPPLY

KEPCO INC.
An ISO 9001 Company.

CE

MODEL
KLR SERIES
DEVELOPER'S GUIDE

ORDER NO.	REV. NO

IMPORTANT NOTES:

1) This manual is valid for the following Model and associated serial numbers:

FIRMWARE VERSION	NOTE
10.07 and higher	

2) A Change Page may be included at the end of the manual. All applicable changes and revision number changes are documented with reference to the equipment serial numbers. Before using this Instruction Manual, check your equipment serial number to identify your model. If in doubt, contact your nearest Kepco Representative, or the Kepco Documentation Office in New York, (718) 461-7000, requesting the correct revision for your particular model and serial number.

3) The contents of this manual are protected by copyright. Reproduction of any part can be made only with the specific written permission of Kepco, Inc.

Data subject to change without notice.

P/N 243-1298

KEPCO®
THE POWER SUPPLIER™

KEPCO, INC. ● 131-38 SANFORD AVENUE ● FLUSHING, NY. 11355 U.S.A. ● TEL (718) 461-7000 ● FAX (718) 767-1102
email: hq@kepcopower.com ● World Wide Web: http://www.kepcopower.com

TABLE OF CONTENTS

TABLE OF CONTENTS

TABLE OF CONTENTS

TABLE OF CONTENTS

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

SECTION 1 - INTRODUCTION

This manual contains instructions for digital programming of the KLR series of 1200W output power, stabilized voltage or current, d-c power supplies manufactured by KEPCO, Inc., Flushing, New York, U.S.A. Basic operation of the front panel controls as well as analog programming of the KLR Series is covered in the KLR User Manual. (When analog programming is in use, the unit will still respond to digital queries related to status and readback.) Refer to the KLR User Manual for all other installation and operating instructions.

1.1 GENERAL DESCRIPTION

The KLR is a power supply class of instrument. The KLR supports all Power supply class functionality including measurement and trigger. The KLR has additional functionality beyond the class requirements, including a 250 step list, state storage, and hyperbolic power capability. The KLR Power Supply Series can be digitally programmed using SCPI commands and queries sent from a computer via one of two interfaces. Standard models (also referred to as -1200 models) include the IEEE 488.2 (GPIB) and RS 232 interface. E-Series models (formerly referred to as -1.2K models) include the IEEE 488.2 (GPIB) and LAN interfaces.

1.1.1 DRIVERS

Three instrument drivers are available which allow remote operation via virtual front panels.

- IVI-COM driver supplied with the unit [E-Series Models only] This is a state-of-the-art driver which provides wrappers to allow use within Visual C, Visual Basic, LabView and LabWindows/CVI environments to fully control the KLR power supply. (see Section 3).

- LabView G driver supplied with the unit. This driver is written in native LabView code using the VISA write and read functions. (see Section 4).

- VXI *plug&play* driver supplied with the unit. This driver is written in C using VISA write and read functions. It is compliant with the VXI *plug&play* specification and can be used in an ANSI compatible C program such as LabWindows /CVI. It is also portable to Linux and Apple operating systems, however verification of these environments has not been completed. (see Section 5).

The drivers (and the raw programming examples found in this manual), do not communicate directly to the KLR. They use the operating system or vendor specific interface drivers to handle the actual communication of the serial, Ethernet or GPIB cables. Vendors such as National instruments, Agilent technologies and Keithley have created a VISA (Virtual Instrument System Architecture) library. The VISA library uses consistent commands to open a connection, write and read data, read the interrupt status and perform standard functions such as lock, unlock and device clear.

The VISA libraries “open” command creates the handle for all other functions. The open command requires a resource string that provides the physical address of the unit and also specifies the type of interface that will be used. Table 1-1 provides the resource strings for the four possible interfaces to the KLR. All KLR Models include the GPIB interface. The serial interface is included only in standard Models; the LAN interface is included only in E-Series Models.

These drivers require a helper application (visa32.dll) to be installed on the computer being used. VISA uses resource strings (see Table 1-1) to address the unit. The IVI-COM, and VXI *plug&play* drivers require that all calls to an instrument be made through the VISA library. The

LabView G driver also uses VISA calls, allowing it to work on all ports of the KLR. The examples given in Programming, Section 6, PAR. 6.8, all utilize VISA calls so they are universally applied.

TABLE 1-1. VISA RESOURCE STRING CORRESPONDING TO INTERFACE

INTERFACE	VISA RESOURCE STRING	COMMENT
GPIB	GPIB::xx::INSTR	The GPIB address replaces xx.
SERIAL	ASRLy::INSTR	The com port number replaces y.
LAN-SCPI-RAW	TCIP::192.168.0.100::5025::SOCKET	This is the fastest LAN interface, similar to the serial port with automatic XON XOFF protocol support.
LAN-VXI-11	TCIP::192.168.0.100::INSTR	This LAN interface requires a more complex handshake for data and is inherently slower than a socket interface. It is similar to the GPIB interface where you tell the device when to take data and when it is acceptable to receive data.

1.1.2 COMMUNICATION

Communication between a KLR and a computer system may be via one of three methods: IEEE 488.2, RS 232 and LAN. These three interfaces communicate by sending formatted strings to the KLR which are then parsed to perform specific actions.

- IEEE 488.2 (GPIB) Interface (see Section 7).
- RS 232 Interface [standard models only] (see Section 8).
- LAN interface [E-Series models only] (see Section 9).

1.1.3 PROGRAMMING

SCPI and IEEE 488 common commands/queries are the building blocks used to control the KLR power supply. These sections are provided to allow a user to write their own program to control the KLR power supply or to use the various interactive tools provided by Measurement Computing, National Instruments, Agilent Technologies and Microsoft to send strings to a device over the RS 232, GPIB or LAN interfaces. SCPI commands and queries are supported by all three interfaces.

- Description of SCPI Syntax (see Section 6).
- IEEE 488 Common commands supported (see Appendix A).
- Listing of SCPI commands supported (see Appendix B).

1.2 DRIVER OVERVIEW

The three drivers for KLR all have common functional groups Each group contains similar functions which work together. The four different environments, IVI-COM, LabView G, VXI Plug&Play and Programming all have common functionality.

The KLR power supply, like most instrument power supplies, has four subsystems: output, trigger, status and measurement. In additional to these subsystems, the KLR has a storage system and a list system. The commands to use these subsystems have been grouped by function as detailed in the following paragraphs. These functional groupings are also used in the examples that are presented in Sections 3 (IVI-COM Driver), 4 (LabView G Driver), and 5 (VXI *plug&play* Driver).

1.2.1 INITIALIZATION

When an instrument is first accessed, it is desirable to have it start at a standard state and to verify that the instrument is the correct one. These functions utilize standard commands which are common to all power supply class instruments such as *IDN? *CLS and *RST. In the various drivers, this functional group also includes an open and close functionality.

1.2.2 OUTPUT CONTROL

The power supply class instrument has specific requirements to support voltage and current settings, output enable and protection. These functions are common to all power supplies in the power supply class.

1.2.3 MEASUREMENT

The Power supply has the ability to measure the voltage and current at the output and display this information on the front panel. The measurement system provides the ability to read this back from the unit.

1.2.4 LIMIT MODELS (USER-DEFINED VOLTAGE/CURRENT LIMITS)

The KLR Power Supply can be programmed to user-defined values that can be lower than the maximum values. For example, although the KLR 40-60 is factory set to 40V and 60A (2400 Watts), arbitrary limits, e.g., 40A@30V or 30A@35V can be established. Once the limits are set, setting values exceeding the limit values will not be accepted. The limit model settings are password protected.

1.2.5 SAVE/RECALL (STORAGE OF USER-PROGRAMMED ACTIVE SETTINGS)

The KLR Power Supply contains 40 memory locations that can be used to store active settings. Values are stored in the nonvolatile memory and are retained when the unit is turned off. The stored settings can then be recalled and applied as desired. See PAR. A.13 for **`*SAV`** (Save) and A.13 for **`*RCL`** (Recall). The unit has the ability to read or set the storage locations without affecting the output of the power supply. These setting are cleared when a calibration is performed. Recalling a location which has never been loaded with data will cause a -221,"Settings conflict" error and the output is unchanged.

If a limit model setting (PAR. 1.2.4) is changed and a stored parameter is outside the range established by the new limit model, when that location is recalled, the parameter recalled appears at the output as the default minimum (zero V or minimum A), but the stored value is unchanged, and no error message is issued.

1.2.6 LIST (USER-PROGRAMMED SEQUENCES)

Using the LIST functions, up to 250 locations are available for programming the KLR output. These locations enable the user to program the output using multiple command sequences which may be initiated by a single command (see PARs B.9 through B.29). The repeatable user-determined sequences are stored in volatile memory and are retained until reprogrammed, the power supply is turned off, a calibration is performed, a **`LIST:CLEar`** command is received, or limit model settings are changed.

Each location defines values for the active channel (either output voltage or output current), a dwell time duration (between 0.010 and 655.36 seconds) for the programmed settings, and the state of the internal relay. By programming the output to change in small increments, complex outputs can be generated.

NOTES: 1. When programming sequential voltage levels, it is important to set the Overvoltage to accommodate the highest voltage of the sequence. Otherwise, when going from higher to lower voltage levels, the overvoltage protection will trip and shut down the unit because the overvoltage setting registers faster than the power supply can attain the lower voltage.

2. To operate the internal relay using the LIST commands, first configure the internal relay to LIST using the UTIL menu from the front panel (see KLR User Manual).

3. The message **dLST** appears in the front panel status display while a LIST program is running.

4. Removing the unit from remote digital programming mode will immediately terminate a running LIST program.

5. The LIST capability is disabled for master /slave configurations and all LIST commands will generate error message -221,"Configuration conflict."

1.2.7 STATUS

The KLR status system consists of the standard register configuration as defined by the IEEE 488.2 and SCPI standards. This configuration allows the errors to be reported and causes interrupts to be sent to the computer. The drivers provide the ability to set and clear these registers but the interrupt functionality is not supported by the drivers. Using interrupts requires special programming techniques that are environment-specific. See PAR. 1.2.7.1 through 1.2.7.5 for details on the process required to implement this in a VISA environment.

The serial poll response of the KLR power supply provides summary bits of the status and error reporting system. (The simplest status report is the "command valid" reporting and data availability. The successful decoding of a command string generates no error and is indicated by the bit 3 of the serial poll response being a zero.) The setting of bit 4 in the status byte indicates data is available to the controller in response to a command query message.

1.2.7.1 STATUS REPORTING STRUCTURE

The status reporting of the KLR uses four status register sets, illustrated in Figure 1-1. These register sets are the Questionable, Operation, Event Status and Status Byte register sets. The Questionable and Operation registers are 16 bit registers whose inputs are unique to each instrument, while the Event Status and Status Byte registers are 8 bit registers with standard inputs defined by IEEE 488.2. Each of these four register sets is comprised of event and enable registers, with the Questionable and Operation sets adding condition and transition registers.

The Questionable and Operation condition registers hold unlatched events reported in real-time by the instrument, viewable at any time using the appropriate query. The contents of the Questionable and Operation condition registers are latched in the corresponding event registers, which hold records of specific event occurrences as determined by the accompanying transition register; for KLR, all transition register bits are configured to record low-to-high (0 to 1) bit states only, so that a 1 in the condition register is always stored as a 1 in the event register. Since the event register bits are reset when read, the event register provides a record of changes in status since the last time the event register was read.

For all four register sets the contents of each event register is gated by an accompanying enable register. The user must set the corresponding enable bit value to one (high) in order to include the specific event bit in the reported status.

For example, if an overvoltage error is detected, bit 0 of the Questionable Status Condition register is set. The 0 to 1 transition causes bit 0 to be stored as a 1 in the corresponding Event register. If bit 0 of the Questionable Status Enable register has bit 0 set, bit 3 of the Status Byte (STB) register is asserted. If bit 3 of the Service Request Enable (SRE) register is also set to 1, then bit 6 of the STB is set to 1 (true), causing the power supply to assert the SRQ line to the host computer.

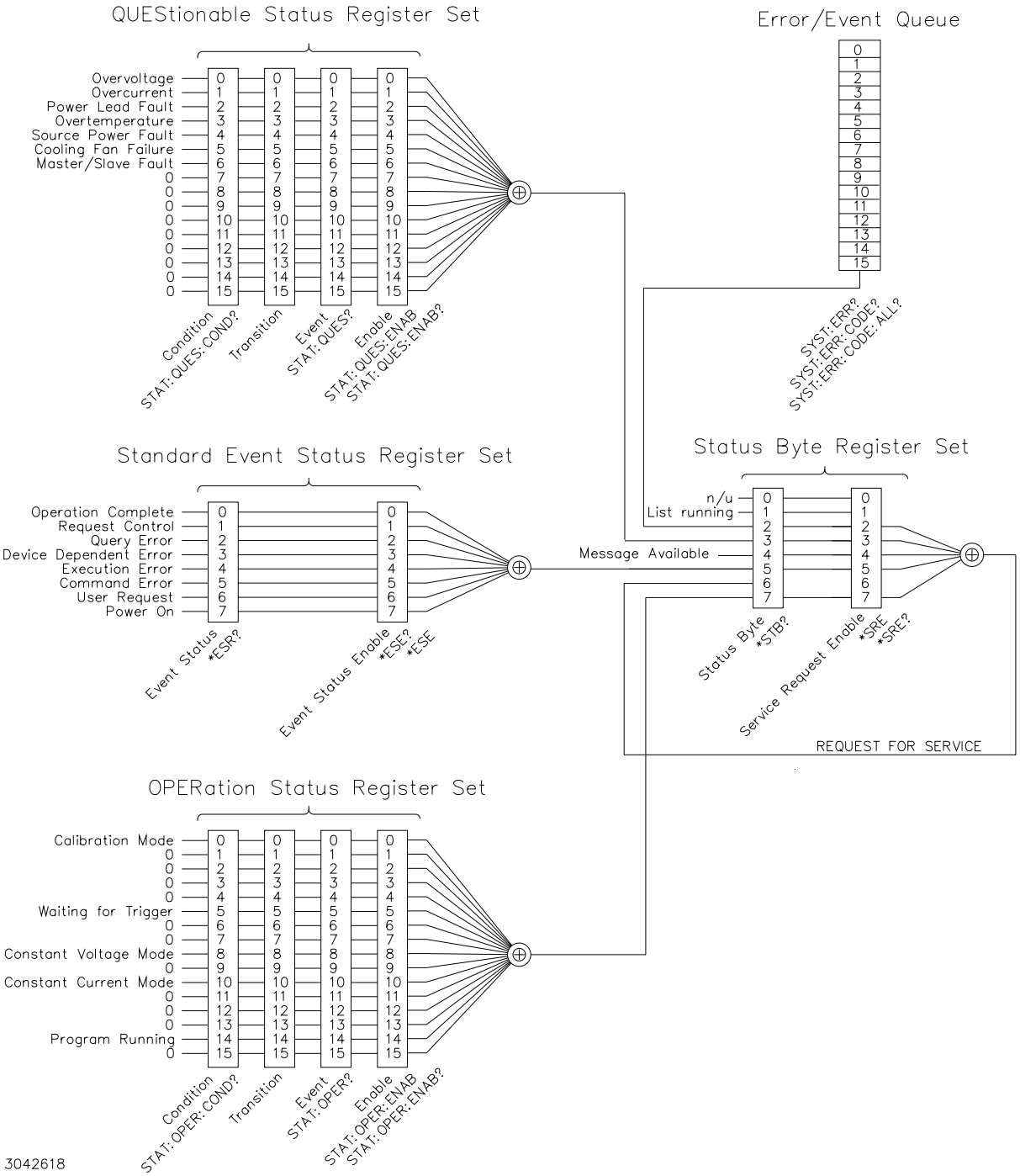


FIGURE 1-1. STATUS REPORTING STRUCTURE

When the service request is executed, the STB register bits are read: bit 3 set indicates Questionable Status; bit 2 set indicates the error/event queue is not empty; bit 4 set indicates that a message is available in the output buffer. bit 5 set indicates event status; bit 7 set indicates operation status. The host computer program might then issue a query to read one of the indicated register set.

Reading an Event or Condition register clears all of the bits found in that register. Event register contents are also cleared when the *CLS command is received. Condition register bits are only cleared when the corresponding condition is not present.

1.2.7.2 STATUS BYTE REGISTER SET

The Status Byte register set is comprised of the Status Byte (STB) and the Service Request Enable (SRE) registers. The STB register is read by issuing the *STB? query (see Appendix A, PAR. A.15). The SRE register is read using the *SRE? query (see Appendix A, PAR. A.14) and modified using the *SRE command (see Appendix A,.PAR. A.13), except that bit 6 (request for service) of STB is not masked by SRE.

- 0 - Not Used - always 0.
- 1 - LIST Running - 1 indicates that a LIST sequence is presently running. Read (and clear) Operation Event register using STAT:OPER?
- 2 - Error Queue — 1 indicates error queue is not empty. Read error queue using SYST:ERR?
- 3 - Questionable Status Summary — 1 indicates a Questionable Status register bit has been set (see Appendix B, Table B-5 for details). Read (and clear) Questionable Event register using STAT:QUES?.
- 4 - Message Available — 1 indicates a message is waiting in the output buffer.
- 5 - Event Status Summary — 1 indicates an Event Status register bit has been set. Read (and clear) Questionable Event register using *ESR?.
- 6 - Service Request — Either RQS (Request for Service) or MSS (Master Status Summary (see Appendix A, PAR. A.13 for details).
- 7 - Operation Status Summary — 1 indicates an Operation Status register bit has been set (see Appendix B, Table B-5 for details). Read (and clear) Operation Event register using STAT:OPER?.

1.2.7.3 STANDARD EVENT STATUS REGISTER SET

The Standard Event Status register set is comprised of the Event Status (ESR) and Event Status Enable (ESE) registers. The ESR register is read by issuing the *ESR? Query (see Appendix A, PAR. A.5). The ESE register is read using the *ESE? query (see Appendix A, PAR. A.4) and modified using the *ESE command (see Appendix A, PAR. A.3).

- 0 - Operation complete — 1 indicates the operation is complete and the unit is ready to accept another command, or that query results are ready to be transferred.
- 1 - Request Control — Not used (always zero).
- 2 - Query Error — 1 indicates a query error has occurred (see Appendix B, Table B-5 for details).
- 3 - Device Dependent Error — 1 indicates device dependent error has occurred (see Appendix B, Table B-5 for details).

- 4 - Execution Error — 1 indicates execution error has occurred (parameter exceeded allowable range) (see Appendix B, Table B-5 for details).
- 5 - Command Error — 1 indicates a command syntax error has occurred (see Appendix B, Table B-5 for details).
- 6 - User Request — Not used (always zero).
- 7 - Power On — set once upon power-up, however ESE bit 7 set to 0 prevents Status Byte bit 5 from being set.

1.2.7.4 OPERATION STATUS REGISTER SET

The Operation Status register set is comprised of condition, transition, event and enable registers (see Figure 1-2). Appendix B, PAR's B.58 through B.61 provide detailed explanations of the queries/commands for reading and modifying these registers as applicable; the transition register cannot be modified.

The Operation condition registers record conditions which are a part of the instrument's normal operation. The definition of each of these bits (condition register) is as follows:

- 0 through 4 — Not used (always zero).
- 5 - Waiting for Trigger Summary — 1 indicates the unit is waiting for trigger
- 6 and 7 — Not used (always zero).
- 8 - Constant Voltage — 1 indicates the instrument is in constant voltage mode.
- 9 — Not used (always zero).
- 10 - Constant Current — 1 indicates the instrument is in constant current mode.
- 11 through 13 — Not used (always zero).
- 14 - Program Running — 1 indicates the program is running.
- 15 — Not used (always zero).

1.2.7.5 QUESTIONABLE STATUS REGISTER SET

The Questionable Status register set is comprised of condition, transition, event and enable registers (see Figure 1-2). Appendix B, PAR's B.63 through B.66 provide detailed explanations of the queries/commands for reading and modifying these registers as applicable; the transition register cannot be modified.

The Questionable Condition register (see Figure 1-1) contains status bits representing data/signals which give an indication of the quality of various aspects of the signal.

A bit set in the Questionable Condition register indicates that the data currently being acquired or generated is of questionable quality due to some condition affecting the parameter associated with that bit.

- 0 - Overvoltage Error — 1 indicates an overvoltage fault has been detected.
- 1 - Overcurrent Error — 1 indicates an overcurrent fault has been detected.
- 2 - Power Lead Error — 1 indicates that output lead connections are not complete.
- 3 - Overtemperature Error — 1 indicates a thermal error has been detected.
- 4 - Input Power Error — 1 indicates input mains error.
- 5 - Fan Error — 1 indicates inoperative fan.
- 6 - Master/Slave Error — 1 indicates communication error between master and slave units.
- 7 - 15 — Not used (always zero).

1.2.7.6 ERROR/EVENT QUEUE

The Error/Event queue is a FIFO (first in first out) buffer that stores errors as they occur. As it is read, each error is removed from the queue and the next error message is made available. When all errors have been read, the query returns 0,"No error".

If more than 15 errors are accumulated, it will overflow. When overflow occurs, the oldest errors stay in the queue, but the most recent errors are discarded. The last error in the queue will be -350,"Too many errors." Error messages are defined in Appendix B, Table B-5.

1.2.8 TRIGGER

The KLR unit has the ability to utilize both a software trigger and an external (hardware) trigger. Model 1-2.K units have an additional software trigger capability accessed via Port 5044. When a trigger event is properly enabled and received, the output voltage and current are set to the preset trigger levels.

The TRIG:SOUR command (see PAR. B.107) determines whether the external (hardware) (EXT) or software (INT) trigger is selected.

If external trigger mode is selected, the hardware trigger is activated by grounding the EXT_TRG line (J2 pin 14 to pin 9, 11, 13 or 15); this creates the trigger event at which time the trigger values (preset values **VOLT:TRIG** and **CURR:TRIG**) will become the setpoint values (Vset and Cset) for the power supply. Refer to PARs. B.44, B.56, and B.94. The **VOLT:TRIG** and **CURR:TRIG** values are reset to the default minimum (zero V or minimum A) when the unit is calibrated or if a limit model setting is changed.

If software trigger is selected, trigger events are activated by *TRG (see PAR. A.16) or a GPIB <GET> command.

SECTION 2 - COMMUNICATION

2.1 INTRODUCTION

Figure 2-1 shows all paths used to communicate with the KLR. These include local control from the front panel, GPIB, LAN (Ethernet) for E-Series models only, and RS 232 for standard models only.

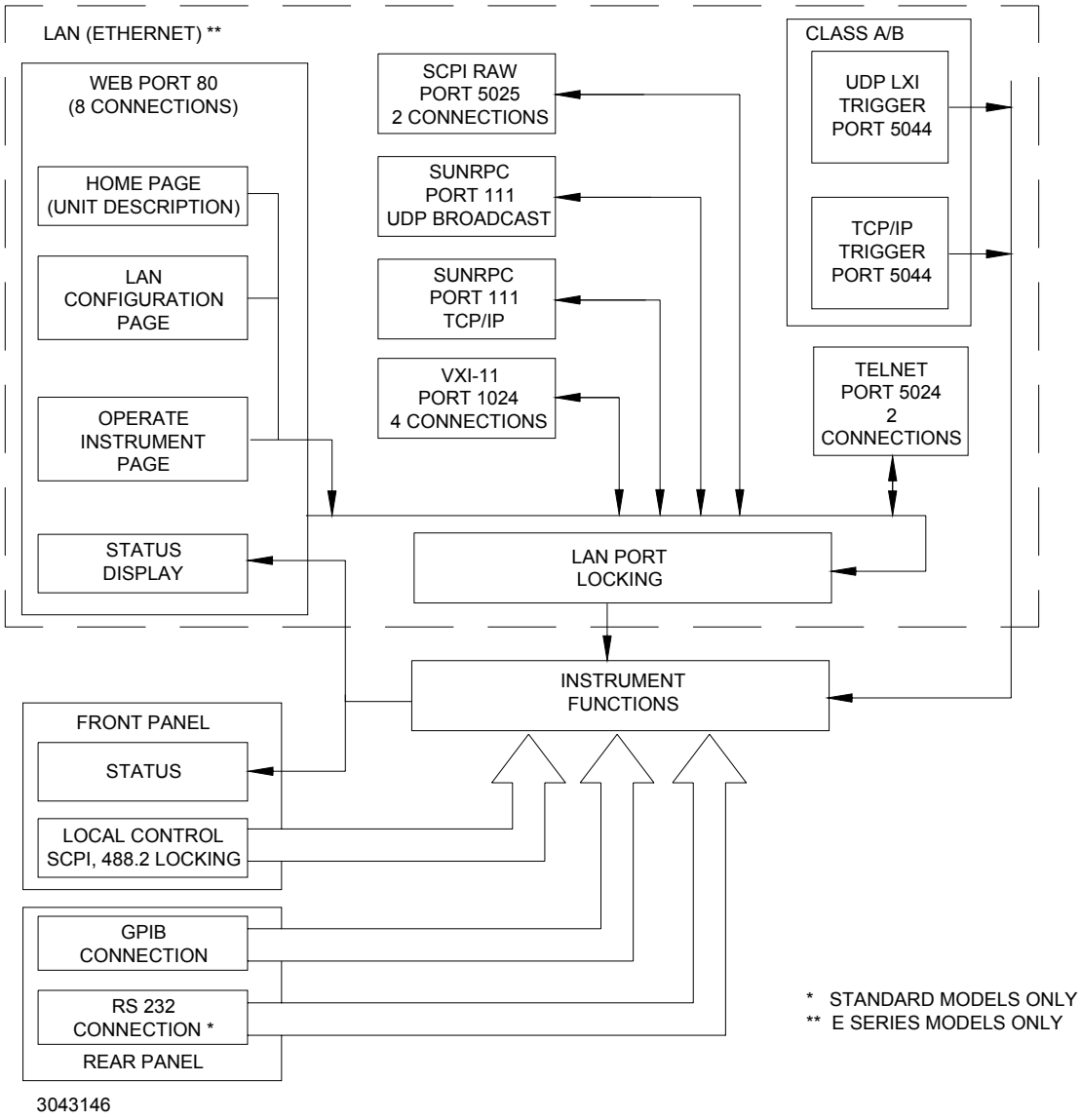


FIGURE 2-1. KLR COMMUNICATION, BLOCK DIAGRAM

2.2 FRONT PANEL (LOCAL) CONTROL

Front panel control is the default upon power up (see the KLR User Manual). The unit automatically enters remote mode upon receipt of a remote command. The front panel can be locked and unlocked in all environments as shown in Table 2-1.

TABLE 2-1. FRONT PANEL LOCKOUT COMMANDS

Environment	Command	Description
Programming	SYSTem:KLOCk	Follow with on or off. Setting on locks the front panel.
VXI plug&play	Kp_KLR_SetResetKeyblock	Requires the instrument session and a Boolean value of 0 for off and 1 to lock the keyboard.
LabView G	Kepco Serial initialize with options	The input defaults to lock the keyboard.
IVI-COM		Requires the instrument session and a Boolean value of 0 for off and 1 to lock the keyboard.

2.3 DIGITAL CONTROL VIA LAN [E-SERIES MODELS ONLY]

Figure 2-1 shows all the ports available to the LAN interface. These port are described in Section 9, LAN INTERFACE [E-Series MODELS ONLY], and include:

- Port 80 - Web Interface. (See KLR User Manual for details on how to use the web interface.) This port supports up to eight connections. For details as to how to access, operate and configure the unit using the web interface, refer to the KLR User Manual. In addition, the web interface is compatible with **\\LXI\instrument.xml** and **\\LXI\instrument.xsd** required by Version 1.2 of the LXI specification.

- Port 111 - SUNRPC (UDP Broadcast and TCP/IP). The SUNRPC port is used for discovery when sent as a UDP broadcast message. The only command supported is the GETPORT. The ports that can be requested are detailed in the VXI specification and repeated in the LXI specifications. The SUNRPC port can be also used with the TCP/IP protocol. Again, the only command supported is the GETPORT. Port 111 is the method (called the Discovery process) used by National Instruments and Agilent to find the resource. KLR E-Series Models support the discovery process implemented by Version 1.1 of the LXI Specification using this port and a VXI-11 protocol, as well as the XML protocol proposed by Version 1.2 of the LXI specification.

- Port 1024 - VXI-11. This port supports up to two connections. VXI-11 uses one standard port (1024) and two assigned by the instrument when connections are opened. The 1024 port is open at all times to accept connection requests. The VXI-11 port requires the use of a lock which prevents other VXI-11 connections from gaining access to the instrument. This port is accessed using the VISA resource string ending in INST, e.g.:

  TCIP0::192.168.1.100::INST

- Port 5024 - Telnet. This port supports up to two connections. the Telnet command provided with all windows operating systems can be run with the following command line (use Start - Run) **TELENET IPADDRESS PORT** e.g.:

  TELNET 192.168.0.100 5024

  When the program is run, entering commands found in Appendix A and B of this Developer's Guide allow the operation of the unit via the Telnet port.

- Port 5025 - SCPI Raw. This port supports up to two connections. The SCPI-Raw port provides faster access than the VXI-11 port (1024) as it has little overhead. This port is accessed using the VISA resource string, e.g.:

  TCIPO::192.168.1.100::5025:SOCKET

- Port 5044 - Trigger Port (UDP LXI and TCP/IP).

2.4 DIGITAL CONTROL VIA GPIB

KLR Power Supplies may be programmed over the IEEE 488 standard communication bus (General Purpose Interface Bus, GPIB) control bus using SCPI (Standard Commands for Programmable Instruments). Use the **SYST:COMM:GPIB:ADDR?** query to read the current GPIB address. Use the **SYST:COMM:GPIB:ADDR** command to change it. Refer to Section 6, PROGRAMMING THE KLR USING SCPI COMMANDS, and Section 7, IEEE 488.2 (GPIB) INTERFACE. for additional details on using the GPIB interface.

2.5 DIGITAL CONTROL VIA RS 232 [STANDARD MODELS ONLY]

KLR standard models may be programmed over the RS 232 control bus using SCPI (Standard Commands for Programmable Instruments) (see Section 6). All power supply functions available from the front panel can be programmed via remote commands, as well as some that are not available from the front panel: Save/Recall (see PAR. 1.2.5) and List (see PAR. 1.2.6). Refer to Section 8, RS 232C INTERFACE [STANDARD MODELS ONLY] for additional details on using the RS 232 interface.

The KLR Power Supply may be operated via an RS232-C terminal, or from a PC using a terminal emulation program. Refer to KLR User Manual for RS 232 connections.

All RS 232 parameters may be changed using SCPI commands (see Appendix B) as follows:

- To enable RS 232, refer to PAR's B.86 and B.87
- For baud rate, refer to PAR's B.82 and B.83
- For prompt, refer to PAR's B.90 and B.91;
- For echo, refer to PAR's B.84 and B.85;
- For XON/XOFF, refer to PAR's B.88 and B.89.

The default settings are as follows:

- Baud rate: 38400
- Parity: None
- Data Bits 8
- Stop Bits 1
- Echo OFF
- XON ON (enabled)

NOTE: Kepco strongly recommends the XON XOFF method for data transfer via RS 232 protocol for all Kepco products. If this method is not selected, it is the user's responsibility to ensure completion of any response by the power supply prior to issuance of subsequent commands.

The XON XOFF method allows the KLR Power Supply to control when the command originator is allowed to send data. The command originator can only send data after the XON (transmission on) character (011_H) has been received; the command originator stops sending data after receiving the XOFF (transmission off) character (013_H), and waits until the XON character is received before sending additional data.

Control characters, either CR or LF, are returned as XOFF CR if echo mode is on, and as XOFF if echo mode is off. XOFF stops data from the command originator and the KLR returns the normal sequence of CR LF (if echo mode is enabled).

Prior to use, the RS 232 port must be initialized be sending either CR, LF or ESC.

The RS 232 port of the KLR must be configured properly to work with the three drivers (see Table 2-2). The baud rate is not automatically detected and must be set correctly by the user using either the front panel, LabView G, VXI *plug&play* or Programming environment. The IVI-COM driver can not modify RS 232 setup. In the VXI *plug&play* environment the program must use the Kepco Serial Initialize routine instead of the standard KpKLRInit routine.

TABLE 2-2. RS 232 SETUP

Environment	Method
IVI-COM	Automatically enabled
LabView	Automatic - uses Kepco Serial Initialize
VXI *plug&play*	KpKLR_Serial Initialize

SECTION 3 - IVI-COM DRIVER

3.1 INTRODUCTION

The IVI-COM (Interchangeable Virtual Instrument) driver provided with the instrument allows remote programming of the unit via either the LAN, GPIB or RS 232 ports.

The IVI-COM (Interchangeable Virtual Instrument) driver provided with the instrument can be used with many Kepco power supplies. The KLR does not support all features found in the driver, specifically those related to the Bipolar type power supplies. It allows remote programming of the KLR via either the LAN (E-Series models only), GPIB (all models) or RS 232 (standard models only) ports.

This driver supports the IviDCPwr class-compliant interface and fulfills all requirements imposed by IVI-4.4: IviDCPwr, Revision 2.0. The driver implements all of the class capabilities from the IviDCPwr specification for which the instrument has matching functionality. The Kepco KLR IVI-COM driver does not support the configuring and resetting of overvoltage and overcurrent conditions.

3.1.1 SPECIFICATION COMPLIANCE

The IVI Foundation provides several specifications outlining the criteria for compliance with the IVI-COM driver architecture. The chief aim of these specifications is to ensure compatibility with specific IVI-COM infrastructure components, thereby increasing the degree of driver and instrument interchangeability. This IVI-COM driver complies with the following IVI standards:

- • IVI 3.1 - IVI Driver Architecture Specification
- • IVI 3.2 - Inherent Capabilities Specification
- • IVI 3.3 - Standard Cross-Class Capabilities Specification
- • IVI 3.4 - Guidelines for API Style
- • IVI 3.5 - IVI Factory Specification
- • IVI 3.6 - IVI Configuration Server Specification

The driver is built upon the National Instruments VISA-COM interface.

3.1.2 RANGE CHECKING AND COERCION

The driver will predominantly rely upon the instrument to perform range checking and coercion.

3.1.3 MULTITHREADING SUPPORT

Multithread safety is provided using standard operating system primitives for object locking. This driver uses method-level locking, so that the driver can operate properly in multithreaded applications.

3.1.4 CONTEXT-SENSITIVE HELP

The driver includes both HTML 1.x and HTML 2.0 help files which document the following driver elements:

- Interfaces
- Methods
- Method parameters
- Properties

3.1.5 INSTALLER

A stand-alone installer, file KepcoDCPwr.0.1.0.0.msi, using Microsoft MSI technology may be downloaded from the Kepco website at www.kepcopower.com/drivers.

NOTE: The filename of the installer, either on our web site or sent via e-mail, will have .delme added to the end of the file. After downloading the file, it must be renamed, deleting .delme. The .delme extension allows the driver to pass through most firewalls. The .msi extension indicates the file is a system update file and many networks prohibit them. Once it is renamed, it can be installed on the computer by double clicking the file.

This installer complies with all of the installer requirements of Section 6 of IVI-3.1: Driver Architecture Specification.

The installer checks to insure the IVI foundation class components of the proper revision are installed. It also verifies the VISA-COM driver is properly registered. If these checks fail, the installer provides instructions on how to repair the problem. The installer will not install the Kepco driver unless the IVI foundation class is installed. If VISA-COM problems are detected, they are reported with suggestions for repair, but do not stop the driver from installing and registering the files.

3.1.6 RIGHTS

The IVI-COM driver was written by Kepco for use on Kepco products. It was written using Pacific Mindworks Nimbus system. It is intended for use with Kepco power supplies. Disassembly and/or modification of the IVI-COM driver is expressly forbidden.

3.2 IVI-COM INSTRUMENT DRIVER FUNCTIONS

Table 3-1 is a list of the KLR IVI-COM driver functions. The installed help file provides detailed information on these functions and all functions in the driver. This driver is both a DCPWR class driver and a Kepco Specific driver. The DCPWR class driver requires the class specific attribute to be accessed as a member in a structure. The standard functions such as enabling the output, setting the voltage level, setting the current level are handled in this manner.

A setting in Table 3-1 is identified by the variable being in all caps. Functions use both upper and lower case letters in the function name. When a setting is to be modified, then the functions KpDCPwr_SetAttributeViBoolean, or KpDCPwr_SetAttributeViReal64 are used to update the variable and the functions KpDCPwr_GetAttributeViBoolean or KpDCPwr_GetAttributeViReal64 are used to get the variable. When one of these class-specified variables are modified, its value is maintained in the driver. When the get or read of the variable is called for, the internal state variable is returned. This behavior can be modified by making the KPDCPWR_ATTR_CACHE attribute false. It defaults to a true state as required by the IVI conventions.

TABLE 3-1. KLR IVI-COM DRIVER FUNCTIONS

Function Name	Purpose
Initialization	
KpDCPwr_init	Initialize Power supply, create session handle.
KpDCPwr_InitWithOptions	Initialize power supply, allowing user to determine reset and identification options.
KpDCPwr_close	Close session handle.
KpDCPwr_LockSession	Prevents other instrument tasks from accessing unit.
KpDCPwr_UnlockSession	Allows other instrument tasks to access unit.
KpDCPwr_reset	Sends reset to unit setting output off
KpDCPwr_revision_query	Gets the revision level of the power supply
KpDCPwr_AttachToExistingCOMSession	Allows second usage of driver
KpDCPwr_serial_query	Gets the serial number of the power supply
KpDCPwr_cal_date_query	Gets the calibration date of the power supply
Output Control Functions	
KPDCPWR_ATTR_OUTPUT_ENABLED	Boolean is used to set the output on and off with this attribute
KPDCPWR_ATTR_VOLTAGE_LEVEL	Sets or reads back the output voltage of power supply.
KPDCPWR_ATTR_CURRENT_LIMIT	Sets and read back the output current level.
KPDCPWR_ATTR_POSITIVE_CURRENT_LIMIT	Sets the current limit level.
KpDCPwr_MeasureCurrent	Measure current function.
KpDCPwr_MeasureVoltage	Measure Voltage function.
KPDCPWR_ATTR_OUTPUT_MODE	Indicates if unit is in voltage or current mode of operation, read only on the KLR.
KPDCPWR_ATTR_POSITIVE_CURRENT_PROTECTION_LIMIT	Sets the current protection level.
KPDCPWR_ATTR_POSITIVE_VOLTAGE_PROTECTION_LIMIT	Sets the voltage protection level.
Output trigger	
KPDCPWR_ATTR_TRIGGERED_CURRENT_LEVEL	Sets the current trigger level.
KPDCPWR_ATTR_TRIGGERED_VOLTAGE_LEVEL	Sets the triggered voltage level of the power supply.
KPDCPWR_ATTR_TRIGGERED_CURRENT_LIMIT	Sets the triggered current protect level of the KLR.
KPDCPWR_ATTR_TRIGGER_SOURCE	Sets and views the trigger source.
KPDCPWR_ATTR_INIT_CONTINUOUS	Enables continuous or single trigger events.
KpDCPwr_ConfigureTriggerSource	Allows selection of the trigger source from External or software triggers.
KpDCPwr_ConfigureTriggeredCurrentLimit	Sends the current trigger level to the power supply.
KpDCPwr_ConfigureTriggeredVoltageLevel	Send the voltage trigger level to the power supply.
KpDCPwr_Abort	Sends abort to the KLR, preventing a trigger event from changing the power supply output.

TABLE 3-1. KLR IVI-COM DRIVER FUNCTIONS (CONTINUED)

Function Name	Purpose
List (supported in standalone configurations only)	
KpDCPwr_ClearLists	Empties all lists. Prepares the KLR to receive a new list.
KpDCPwr_SetVoltageListValues	Sends the new voltage point to the KLR.
KPDCPWR_ATTR_VOLTAGE_LIST_POINTS	Provides the location of the next point to be entered into voltage list.
KpDCPwr_SetCurrentListValues	Sends the new current point to the KLR.
KPDCPWR_ATTR_CURRENT_LIST_POINTS	Provides the location of the next point to be entered into current list.
KpDCPwr_SetListDwellTimes	Sends the new dwell point to the KLR.
KPDCPWR_ATTR_LIST_DWELL_POINTS	Provides the location of the next point to be entered into dwell list.
KpDCPwr_SetListControl	Sends the new relay point to the KLR.
KPDCPWR_ATTR_LIST_CONTROL_POINTS	Provides the location of the next point to be entered into control (relay) list.
KPDCPWR_ATTR_LIST_SKIP	Provides ability to skip a number of points when a list is repeatedly executed.
KpDCPwr_ExecuteVoltageList	Causes a list to be executed.
KpDCPwr_HaltVoltageList	Causes a list to be stopped; useful when list is executed continuously.
KpDCPwr_SetListCycles	Causes the list to be executed a number of times. Sending 0 will make list execute forever.
KPDCPWR_ATTR_LIST_CYCLES	Provides number of cycles for a list to be executed. The clear list sets this variable to 1.
Utility	
KpDCPwr_EnablePasswordState	Enables Password controlled conditions.
KPDCPWR_ATTR_POSITIVE_CURRENT_RATED_LIMIT	Limit model current limit
KPDCPWR_ATTR_POSITIVE_VOLTAGE_RATED_LIMIT	Limit model voltage limit
KpDCPwr_DisablePasswordState	Turn off password enable
KpDCPwr_ChangePassword	Allows for changing the password of the power supply.

3.3 EXAMPLES USING C

The examples illustrated in the following paragraphs are installed in the ivi\drivers\kepco\examples folder during installation of the IVI-COM driver. Figures showing the complete code for each example are provided

3.3.1 SETTING THE OUTPUT TO A VALUE AND MAKING A MEASUREMENT

The filename for this example is *C output example.txt*, located in the ivi\drivers\kepco\examples folder (see Figure 3-1 for complete code).

All programs written in C++ must open the driver and include some standard H files.

```
/*~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
 This program demonstrates how to set a voltage and current
 and measure the output voltage using the IVI-COM driver.
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~*/

#include <stdafx.h>
#include <stdio.h>
#include <conio.h>
#include <atlbase.h>

#import "IviDriverTypeLib.dll" no_namespace
#import "IviDCPwrTypeLib.dll" no_namespace
#import "KepcDCPwr.dll" no_namespace
```

The driver dll's must be loaded and initialized to operate:

```
    hr = CoInitialize(NULL);
    if (FAILED(hr))
        exit(1);

    // create a safe pointer for interface access
    IKLRPtr driverPtr=NULL;
    hr = driverPtr.CreateInstance(__uuidof(KepcoDCPwr));
    if (FAILED(hr))
        exit(1);
```

Once initialized, communication with the KLR must be established. The following code shows how to open the driver at GPIB address 6. The id query is set to true. Failure to id query to true will reduce the functionality of the driver to the DCPWR class capabilities, preventing functions such as setting the limit model, list and other advanced features of the KLR from being accessed.

```
// open the instrument for communication
hr = driverPtr->Initialize(
"GPIB0::6::INSTR",  //Visa address,(not applicable if simulation=true)
VARIANT_TRUE,       // ID query
VARIANT_TRUE,       // Reset
LPCTSTR("Cache=true,  InterchangeCheck=false,       QueryInstrStatus=true,
          Simulate=false"));  //IVI options
```

The driver is a linked list of pointers to properties and functions. To read a property, it is necessary to locate the property. The following shows how to read the firmware revision of the KLR. To read this property the identification must be performed first.

```
    bstrInstrFwRev = driverPtr->Identity->InstrumentFirmwareRevision;
```

Accessing properties requires many pointer operations. The examples below use pointer redirection to make the operations a little clearer.

```
// get pointers to the needed interfaces
IKLROutputPtr outputPtr;
outputPtr = driverPtr->Outputs->GetItem(OLESTR("Output"));

IKLRMeasurementPtr measurementPtr;
measurementPtr = driverPtr->Measurements->GetItem(OLESTR("Measurement"));
```

The setting of voltage and current is then very simple.

```
// set voltage
hr = outputPtr->VoltageLevel(75.0);
if (FAILED(hr))
    exit(1);



// set the current limit
hr = outputPtr->CurrentLimit(10);
if (FAILED(hr))
    exit(1);

// enable the output
outputPtr->Enabled = true;

Sleep(500);
```

Note that a measurement requires accessing a function, not a variable.

```
// measure the voltage
double measVoltage;
measVoltage = measurementPtr->Measure(KepcoDCPwrMeasurementVoltage);
```

```
/*~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
 This program demonstrates how to set a voltage and current
 measure the output voltage
 using the IVI-COM driver.
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~*/

#include <stdafx.h>
#include <stdio.h>
#include <conio.h>
#include <atlbase.h>

#import "IviDriverTypeLib.dll" no_namespace
#import "IviDCPwrTypeLib.dll" no_namespace
#import "KepcDCPwr.dll" no_namespace


int main(int argc, char* argv[])
{
HRESULT hr;

try
{
     hr = CoInitialize(NULL);
     if (FAILED(hr))
          exit(1);

     // create a safe pointer for interface access
     IKLRPtr driverPtr=NULL;
     hr = driverPtr.CreateInstance(__uuidof(KepcoDCPwr));
     if (FAILED(hr))
          exit(1);
```

FIGURE 3-1. EXAMPLE OF SETTING THE OUTPUT AND TAKING A MEASUREMENT, WRITTEN IN C++ (SHEET 1 OF 2)

```
    // open the instrument for communication
    hr = driverPtr->Initialize(
         L"GPIB0::6::INSTR",  //Visa address,(not applicable if simulation=true)
         VARIANT_TRUE,    // ID query
         VARIANT_TRUE,     // Reset
         LPCTSTR("Cache=true,  InterchangeCheck=false,  QueryInstrStatus=true,  Simulate=false"));
             //IVI options
    if (FAILED(hr))
         exit(1);

    // Get InstrumentFirmwareRevision property.
    BSTR bstrInstrFwRev;
    bstrInstrFwRev = driverPtr->Identity->InstrumentFirmwareRevision;
    wprintf(L"InstrumentFirmwareRevision: %s\n", bstrInstrFwRev);


    // get pointers to the needed interfaces
    IKLROutputPtr outputPtr;
    outputPtr = driverPtr->Outputs->GetItem(OLESTR("Output"));


    IKLlnMeasurementPtr measurementPtr;
    measurementPtr = driverPtr->Measurements->GetItem(OLESTR("Measurement"));

    // set voltage
    hr = outputPtr->VoltageLevel(75.0);
    if (FAILED(hr))
         exit(1);

    // enable OV protection and set the limit.
    hr = protectionPtr->ConfigureOVP(10);
    if (FAILED(hr))
         exit(1);


    // set the current limit
    hr = outputPtr->CurrentLimit(10);
    if (FAILED(hr))
         exit(1);

    // enable the output
    outputPtr->Enabled = true;

    Sleep(500);

    // measure the voltage
    double measVoltage;
    measVoltage = measurementPtr->Measure(KepcoDCPwrMeasurementVoltage);

    //print out voltage measurement
    printf("measured voltage : %f V \n",measVoltage);

}
catch(_com_error err)
{
    printf("%s", err.Description());
}


return 0;
}
```

FIGURE 3-1. EXAMPLE OF SETTING THE OUTPUT AND TAKING A MEASUREMENT, WRITTEN IN C++ (SHEET 2 OF 2)

3.3.2 USING A LIST TO PERFORM A SERIES OF OPERATIONS

The filename for this example is *Clistexample.txt*, located in the ivi\drivers\kepco\examples folder (see Figure 3-2 for complete code).

This example is written in C# and applies to a KLR 75-32 (either standard or E-Series model). Step sizes may need to be readjusted for other models. This example will not work for units configured for master/slave operation.

Since the IVI-COM driver is a native application for the C# and the VB.NET environments, it is the easiest to use. More information about the various programming environments can be found in the kepcoDCPwr driver help file which is automatically installed during driver installation.

Initializing the driver is easy; only the driver name is required.

```
using System;
using KepcoDCPwr.Interop;

       // Create driver instance.
       KepcoDCPwr KLRdriver;
       KepcoDCPwr.Interop.Output outputPtr;
       KepcoDCPwr.Interop.IList  listPtr;
       KLRdriver = new KepcoDCPwrClass();
```

The connection between the host and the KLR is then established by the following code:

```
KLRdriver.Initialize("GPIB0::6::INSTR",
    true,
    true,
        "Cache=true,InterchangeCheck=false,QueryInstrStatus=true,Simu-
           late=false");//IVI options


           // get references to the needed interfaces
           outputPtr = KLRdriver.Outputs.get_Item(driver.Outputs);
           listPtr = KLRdriver.List.get_Item(driver.Lists);
```

The list functionality uses arrays. The following code creates these arrays and then fills them with data.

```
           //create arrays for the ListPoints method
           double[] voltList = new double [6];
           double[] currList = new double [6];
           double[] dwellTime = new double [6];

           voltList[0] = 10;
           voltList[1] = 20;
           voltList[2] = 30;
           voltList[3] = 40;
           voltList[4] = 50;
           voltList[5] = 60;

           currList[0] = 2;  // select max current for each step
           currList[1] = 2;
           currList[2] = 2;
           currList[3] = 2;
```

```
currList[4] = 2;
currList[5] = 2;

dwellTime[0] = 1;
dwellTime[1] = 2;
dwellTime[2] = 3;
dwellTime[3] = 4;
dwellTime[4] = 5;
dwellTime[5] = 0.01;
```

The following command uses the IVI-COM driver to send the lists to the KLR. Note that the arrays are passed by reference to the driver.

```
listPtr.ListPoints(ref voltList, ref currList, ref dwellTime);
```

The list is set to perform the required number of executions, then started:

```
// Set the number of executions.
listPtr.count = 2;

// initiate
outputPtr.listInitiate();
```

The driver should be closed at the end using the following code. The conditional statement allows the close to be added to the end of the program without getting any runtime errors.

```
//' Close driver if initialized.

if (KLRdriver.Initialized == true ) KLRdriver.Close();
```

```
/*~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
'This program executes a 6 point current and voltage list.
' It also specifies 6 different dwell times.
'~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~*/

using System;
using KepcoDCPwr.Interop;

namespace ListExample
{
/// <summary>
/// Summary description for Class1.
/// </summary>
class Class1
{
     /// <summary>
     /// The main entry point for the application.
     /// </summary>
     [STAThread]
     static void Main(string[] args)
     {
          // Create driver instance.
          KepcoDCPwr KLRdriver;
          KepcoDCPwr.Interop.IOutput outputPtr;
          KepcoDCPwr.Interop.IList  listPtr;
          KLRdriver = new KepcoDCPwrClass();


          try
          {
```

FIGURE 3-2. EXAMPLE OF USING A LIST, WRITTEN IN C# (SHEET 1 OF 2)

```
            // Initialize driver
            KLRdriver.Initialize("GPIB0::6::INSTR",
                true,
                true,
                "Cache=true,InterchangeCheck=false,QueryInstrStatus=true,Simu-
               late=false");//Optional ivi options

            System.Console.WriteLine("Done.\n");

            // get references to the needed interfaces
            outputPtr = KLRdriver.Outputs.get_Item(driver.Outputs);
            listPtr = KLRdriver.List.get_Item(driver.Lists);

            //create arrays for the ListPoints method
            double[] voltList = new double [6];
            double[] currList = new double [6];
            double[] dwellTime = new double [6];

            voltList[0] = 10;
            voltList[1] = 20;
            voltList[2] = 30;
            voltList[3] = 40;
            voltList[4] = 50;
            voltList[5] = 60;

            currList[0] = 2;  // select max current for each step
            currList[1] = 2;
            currList[2] = 2;
            currList[3] = 2;
            currList[4] = 2;
            currList[5] = 2;

            dwellTime[0] = 1;
            dwellTime[1] = 2;
            dwellTime[2] = 3;
            dwellTime[3] = 4;
            dwellTime[4] = 5;
            dwellTime[5] = 0.01;

            //' call ListPoints to set the list values and set the voltage and current modes to
            LIST
            listPtr.ListPoints(ref voltList, ref currList, ref dwellTime);

            // enable the output
            outputPtr.Enabled = true;

            // Set the number of executions.
            listPtr.count = 2;

            // initiate the list in the output system
            outputPtr.listInitiate();


            KLRdriver.Systems.WaitForOperationComplete(10000);

            //Read instrument errors
            ReadInstrumentError(driver);

            //' Close driver if initialized.
            if (KLRdriver.Initialized == true ) KLRdriver.Close();

            Console.WriteLine();
            Console.Write("Press Enter to Exit ");
            Console.ReadLine();
        }
        catch (Exception e)
        {
            Console.WriteLine();
            Console.WriteLine("Exception Error:");
            Console.WriteLine("  " + e.Message );

            KLRdriver.Close();

            Console.WriteLine();
            Console.Write("Press Enter to Exit ");
            Console.ReadLine();
        }
    }

}
}
```

FIGURE 3-2. EXAMPLE OF USING A LIST, WRITTEN IN C# (SHEET 2 OF 2)

3.3.3 SETTING LIMIT MODEL

This example is a series of files found on the CD in the
ivi\drivers\kepcodcpwr\example\cviVirtual directory.
In CVI you just need to open the .prj file to access the entire project

Note: This project file establishes a link to the kepcodcpwr.fp file by using the Edit menu -> project -> Function panel file. To create another project this step is critical to proper operation of the driver under the CVI environment.

In CVI each C file in the project needs to invoke the various include files.

```
#include "KpDCPwr.h"
#include <ansi_c.h>
#include <cvirte.h>
#include <visatype.h>
#include "KpDCPwrErrorHandler.h"
#include "KpDCPwrExample.h"
```

The CVI environment is aware of the IVI directory structure and will automatically locate the directories containing the files without adding any path statements. It is an advantage of using CVI over other C environments as it is IVI aware.

When the resource string is entered and the user clicks the appropriate button, the driver is invoked and the password is established with the following code.

```
sprintf(initOptions, "QueryInstrStatus=1, Simulate=%d", simulate);
       // Initialize the driver
checkErr(KpDCPwr_InitWithOptions(resourceName, VI_TRUE, VI_TRUE,
              initOptions, &vi));
       // Enable the password state
checkErr(KpDCPwr_EnablePasswordState(vi, "7532"));
```

The user is then provided another form. When the data is entered, the standard CVI logic is invoked and the variables are read from the form, then sent to the KLR.

The setting the KLR's voltage is performed by two lines of code. The first line is the standard CVI method of getting a variable from the form and the second line is the function in the IVI driver that sets an attribute to a specific value.

```
// Get the desired voltage limit from the UI
GetCtrlVal(panelHandle, PANEL_VOLTAGE_LIMIT, &voltageLimit);
// Set the voltage limit
checkErr(KpDCPwr_SetAttributeViReal64(vi, VI_NULL,
          KPDCPWR_ATTR_POSITIVE_VOLTAGE_RATED_LIMIT, voltageLimit));
```

The setting of a maximum current level for the limit model is performed using the same two functions but the attribute is changed on the set function.

```
 // Get the desired current limit from the UI
 GetCtrlVal(panelHandle, PANEL_CURRENT_LIMIT, &currentLimit);
 // Set the current limit
 checkErr(KpDCPwr_SetAttributeViReal64(vi, VI_NULL, //
```

The functions above used a function checkErr which is part of the project file *KpDCPwrErrorHandler.c*. This file has calls to two functions: *KpDCPwr_GetError* and *KpDCPwr_error_query*. These two functions check for errors and provide readable text of the error messages.

```
KpDCPwr_GetError(vi, &errorCode, bufferSize, pos);
// Display the error from the instrument, if that is the source
if (errorCode == KPDCPWR_ERROR_INSTRUMENT_STATUS)
{
KpDCPwr_error_query(vi, &errorCode, buffer);
}
MessagePopup("Error!", buffer);
```

3.4 EXAMPLES USING VISUAL BASIC

The following paragraphs are example of how to use code written in Visual Basic to set the output and take a measurement, and how to use a list to perform as series of operations

3.4.1 SETTING THE OUTPUT TO A VALUE AND TAKING A MEASUREMENT

The following example, written in Visual Basic, sets the output to a value and then takes a measurement. The filename for this example is *VBOutputExample.txt*, located in the ivi\drivers\kepco\examples folder (see Figure 3-3 for complete code).

Note that Visual Basic has its own methods to support the IVI-COM driver.

```
Public driver As KepcoDCPwr
Public outputPtr As IKLROutput
Public protectionPtr As IKLRProtection
Public measurementPtr As IKLRMeasurement

        Set driver = New KepcoDCPwr
```

Once the driver is initialized, the connection between host and KLR is established using a standard VISA resource string. It is important that the initialize function be passed as true since its second argument allows KLR-specific features to be enabled, however in this example, no KLR-specific feature such as list is required.

```
        ' initialize the driver - note ID device is true
        driver.Initialize "GPIB0::6::INSTR", _
                                True, _
                                True, _
                                 "Cache=true,InterchangeCheck=false,Query-
          InstrStatus=true,Simulate=false"

        Dim result As Boolean
        result = driver.Initialized

        Set outputPtr = driver.Outputs.Item(driver.Outputs)
        Set protectionPtr = driver.Protections.Item(driver.Protections)
        Set measurementPtr = driver.Measurements.Item(driver.Measurements)
```

Using the pointers it becomes very easy to set voltage and current. Each parameter is simply loaded with the value to be set using individual statements.

```
' set voltage level
outputPtr.VoltageLevel 75

' enable OV protection and set OV level
protectionPtr.ConfigureOVP 15


' set current level
outputPtr.CurrentLimit 10

' enable the output
outputPtr.Enabled = True
```

Measuring the output is performed by accessing a function and not a parameter.

```
' Measure the voltage
Dim measVoltage As Double
measVoltage = measurementPtr.Measure(KepcoDCPwrMeasurementVoltage)
```

Closing the driver is the last step in any program:

```
driver.Close
```

```
'/*~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
' This program demonstrates how to set the output voltage and current
'   measure the voltage
' using the IVI-COM driver.
'/*~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Option Explicit

Public driver As KepcoDCPwr
Public outputPtr As IKLROutput
Public protectionPtr As IKLRProtection
Public measurementPtr As IKLRMeasurement

Private Sub cmdExit_Click()
Unload Me
End Sub

Private Sub cmdStart_Click()

        Set driver = New KepcoDCPwr

        On Error GoTo ErrorHandler


        ' initialize the driver - note ID device is true
        driver.Initialize "GPIB0::6::INSTR", _
                          True, _
                          True, _
                               "Cache=true,InterchangeCheck=false,QueryInstrStatus=true,Simu-
                                late=false"

        Dim result As Boolean
        result = driver.Initialized


        Set outputPtr = driver.Outputs.Item(driver.Outputs)
        Set protectionPtr = driver.Protections.Item(driver.Protections)
        Set measurementPtr = driver.Measurements.Item(driver.Measurements)

        ' set voltage level
        outputPtr.VoltageLevel 75

        ' enable OV protection and set OV level
        protectionPtr.ConfigureOVP 15


        ' set current level
        outputPtr.CurrentLimit 10

    ' enable the output
        outputPtr.Enabled = True

        Delay 10

        ' Measure the voltage
        Dim measVoltage As Double
        measVoltage = measurementPtr.Measure(KepcoDCPwrMeasurementVoltage)
```

FIGURE 3-3. SETTING THE OUTPUT AND TAKING A MEASUREMENT EXAMPLE, WRITTEN IN VISUAL BASIC (SHEET 1 OF 2)

```
          ' display the measured voltage
        StatusTextBox.Text = StatusTextBox.Text & "Measured Voltage is " & measVoltage & " at chan-
             nel " & channel
        StatusTextBox.Refresh

         ReadInstrumentError driver

        driver.Close

         StatusTextBox.Text = StatusTextBox.Text & "Driver closed." & vbCrLf
        StatusTextBox.Refresh
Exit Sub
ErrorHandler:
    MsgBox Err.Description
    driver.Close
    Exit Sub

End Sub


' Wait the specified number of seconds
Private Sub Delay(DelayTime As Single)
   Dim Finish As Single
   Finish = Timer() + DelayTime
   Do
   Loop Until Finish <= Timer()
End Sub
```

FIGURE 3-3. SETTING THE OUTPUT AND TAKING A MEASUREMENT EXAMPLE, WRITTEN IN VISUAL BASIC (SHEET 2 OF 2)

3.4.2 USING A LIST TO PERFORM A SERIES OF OPERATIONS

The filename for this example is *VBListexample.txt*, located in the ivi\drivers\kepco\examples folder (see Figure 3-6 for complete code).

```
'/*~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
'This program executes a 6 point current and voltage list.
' It also specifies 3 different dwell times.
'~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~*/
Option Explicit

Public driver As KepcoDCPwr
Public outputPtr As IKepcoKLROutput

Private Sub cmdExit_Click()
Unload Me
End Sub

Private Sub cmdStart_Click()

    Set driver = New KepcoDCPwr

    On Error GoTo errorHandler

        ' initialize the driver

        driver.Initialize "GPIB0::6::INSTR", _
                           False, _
                           True, _
                          "Cache=true,InterchangeCheck=false,QueryInstrStatus=true,Simulate=false"

        Dim result As Boolean
        result = driver.Initialized


        ' get references to the needed interfaces
        Set outputPtr = driver.Outputs.Item(driver.Outputs)
      Set listPtr =driver.list
```

FIGURE 3-4. EXAMPLE OF USING A LIST, WRITTEN IN VISUAL BASIC (SHEET 1 OF 2)

```
        Delay (100)


        ReadInstrumentError driver

        driver.Close

    Exit Sub
errorHandler:
    MsgBox Err.Description
    driver.Close
    Exit Sub
End Sub

Private Sub ReadInstrumentError(agDrvr As IIviDriver)
    ' Read instrument error queue until its empty.
    Dim errCode As Long
    errCode = 999
    Dim errMsg As String

    StatusTextBox.Text = StatusTextBox.Text & vbCrLf
    While errCode <> 0
        agDrvr.Utility.ErrorQuery errCode, errMsg
        StatusTextBox.Text = StatusTextBox.Text & "ErrorQuery: " & errCode & ", " & errMsg & vbCrLf
    Wend
End Sub

' Wait the specified number of seconds
Private Sub Delay(DelayTime As Single)
   Dim Finish As Single
   Finish = Timer() + DelayTime
   Do
   Loop Until Finish <= Timer()
End Sub
```

FIGURE 3-4. EXAMPLE OF USING A LIST, WRITTEN IN VISUAL BASIC (SHEET 2 OF 2)

3.4.3 SETTING LIMIT MODEL

The filename for this example is *VBlimit model Example.txt*, located in the ivi\drivers\kepco\examples folder (see Figure 3-5 for complete code). This example is very similar to the other examples in this section. The difference is that functions in the driver are used to perform the operation instead of setting the value to a specific state. Access to the functions is identical to the method used to access the open and close of the driver. The functions used are:

KpDCPwr_SetAttributeViReal64 which sets the various attributes to a specific value.

KpDCPwr_EnablePasswordState which sends the string to the unit to enable the ability to set the limit model.

```
'/*~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
' This program demonstrates how to set the limit model
' using the IVI-COM driver.
'/*~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Option Explicit

Public driver As KepcoDCPwr

Private Sub cmdExit_Click()
Unload Me
End Sub

Private Sub cmdStart_Click()

        Set driver = New KepcoDCPwr

        On Error GoTo ErrorHandler


        ' initialize the driver - note ID device is true
        driver.Initialize "GPIB0::6::INSTR", _
                                True, _
                                True, _
                                    "Cache=true,InterchangeCheck=false,QueryInstrStatus=true,Simu-
             late=false"

        Dim result As Boolean
        result = driver.Initialized


' This will now set the limit model to a 36-32 unit.

     driver.KpDCPwr_SetAttributeViReal64 KPDCPWR_ATTR_POSITIVE_VOLTAGE_RATED_LIMIT, 36
     driver.KpDCPwr_SetAttributeViReal64 KPDCPWR_ATTR_POSITIVE_CURRENT_RATED_LIMIT, 32

        driver.Close

        StatusTextBox.Text = StatusTextBox.Text & "Driver closed." & vbCrLf
        StatusTextBox.Refresh
Exit Sub
ErrorHandler:
    MsgBox Err.Description
    driver.Close
    Exit Sub

End Sub


' Wait the specified number of seconds
Private Sub Delay(DelayTime As Single)
   Dim Finish As Single
   Finish = Timer() + DelayTime
   Do
   Loop Until Finish <= Timer()
End Sub
```

FIGURE 3-5. SETTING LIMIT MODEL, WRITTEN IN VISUAL BASIC

3.5 EXAMPLES USING LABVIEW

These LabView examples show the use of the IVI-COM driver with LabView 8. The procedure used to create the examples is found in the KepcoDcPwr help file that was installed in the IVI portion of your Start menu. Step by step instructions are detailed in:

- Getting Started with the IVI-COM Driver - Development Environments - Using LabView

3.5.1 SETTING THE OUTPUT TO A VALUE

Figure 3-6 illustrates the use of the IVI-COM driver with LabView to set the KLR output to a value. The first two blocks show the automation controller and kepcodriver functions. These are followed by the initialize method of the driver,

Referring to Figure 3-6, the initialize method makes the connection to the KLR using the visa resource string. The driver supports a simulation mode of operation which is set false in the example. The next three blocks are property nodes which are used to set the voltage, current and output enable of the power supply. These are the properties of the power supply which directly affect the power supply output. In LabView each property must be set separately, but multiple outputs of the property node are allowed. The final two blocks are the close functions. One block is the method to close (disable the output). The second block is the ActiveX close reference which performs the driver close, breaking the connection to the KLR, and the instrumentation close which releases the memory used by the driver.

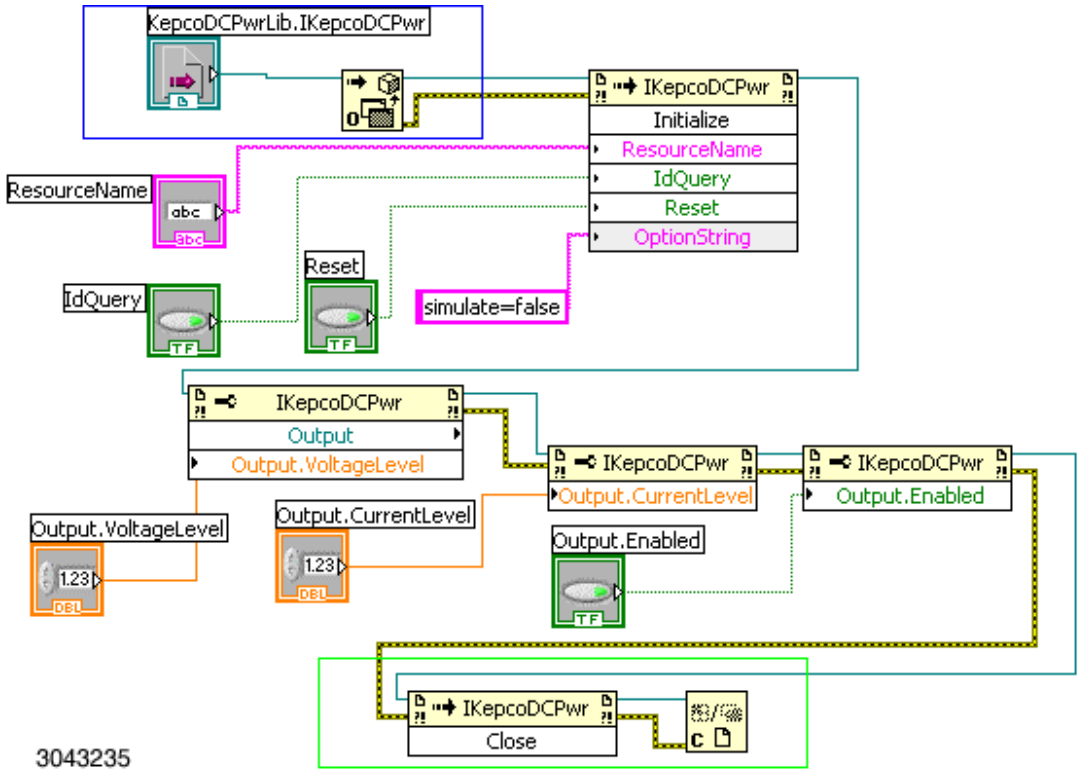


FIGURE 3-6. SETTING THE OUTPUT USING LABVIEW WITH IVI-COM DRIVER

3.5.2 GETTING A VOLTAGE AND CURRENT READING FROM THE POWER SUPPLY

Figure 3-7 illustrates the use of the IVI-COM driver with LabView to measure the KLR output.

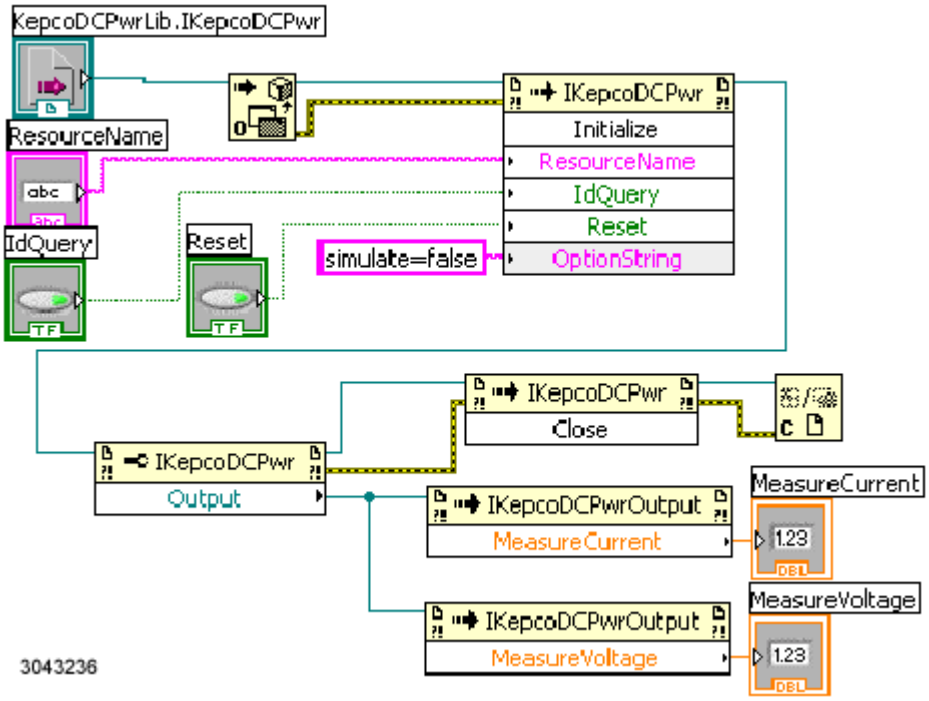


FIGURE 3-7. MEASURING VOLTAGE AND CURRENT USING LABVIEW WITH IVI-COM DRIVER

3.5.3 CHANGING THE LIMIT MODEL

The limit model Vi is very similar to the output setting and measurement examples; it has the identical open and close functionality. The middle three blocks set the password, then cause the voltage limit and the current limit to be applied to the KLR. This is done using the Max Positive Current Limit and Max Positive Voltage Limit properties. As indicated in the earlier table these are the setting nodes that are required to change the KLR limit model.

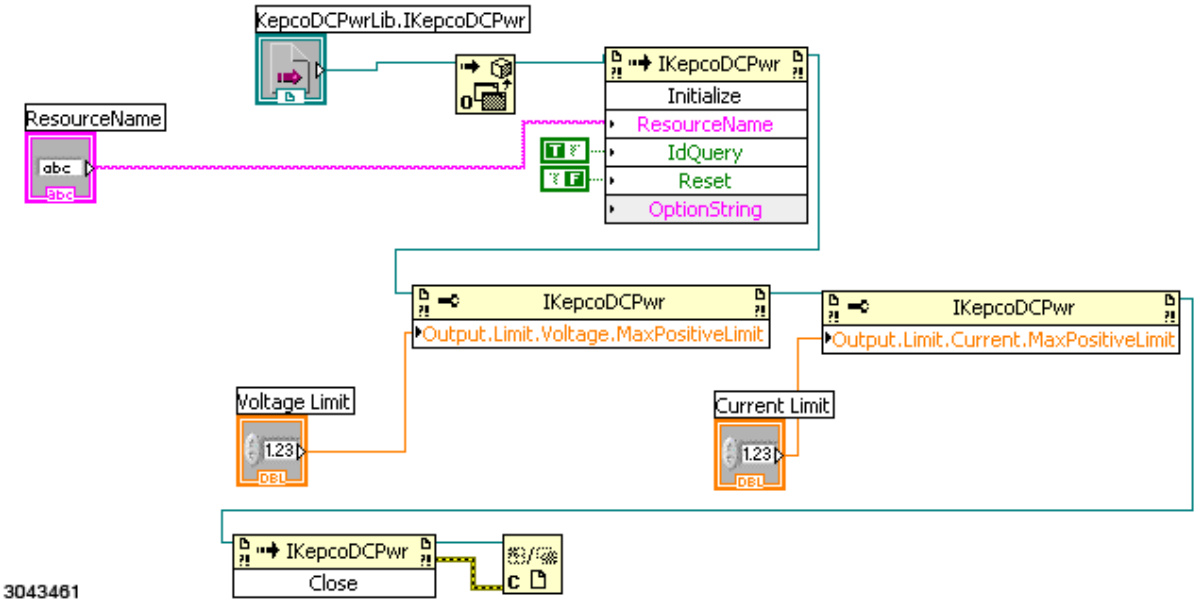


FIGURE 3-8. SETTING THE LIMIT MODEL USING LABVIEW WITH IVI-COM DRIVER

SECTION 4 - LABVIEW G DRIVER

4.1 INTRODUCTION

The LabView G driver can be used for remote programming over all ports of the KLR (Serial and GPIB ports [and LAN port for E-Series models only]. It is designed to operate in LabView 8.5 environments only.

NOTE: The LabView5 driver available from the Kepco website is recommended to be used only with LabView 5 and supports remote programming only via the GPIB port of the KLR. A LabView 6i version provides access to GPIB- and LAN-based KLR units. All three LabView G Drivers can be downloaded from the Kepco web site at:
www.kepcopower.com/drivers/.
The older versions are found at:
www.kepcopower.com/drivers/drivers-dl3a.htm.

The LabView G driver uses VISA calls to communicate with the power supply. Since LabView automatically installs the VISA libraries, the proper version of the library should already be installed on your computer.

The Version 8 LabView driver is available in two formats: a standalone version that includes the LabView runtime program, and a version that requires that LabView be installed on the computer.

NOTE: The LabView G driver is modifiable by the user. Once any utility or function is modified, it is the user's responsibility to verify there are no unintended errors introduced.

4.2 LABVIEW G INSTRUMENT DRIVER

The LabView G driver is a multipurpose driver. It is capable of operating all Kepco Power supplies using LabView Version 8 except for multiple output units such as MST. The LabView G driver does not report model-specific errors but, using case structures, passes the function and an appropriate model- specific response to the user's program.

4.3 LABVIEW G INSTRUMENT DRIVER FUNCTIONS

Kepco's KLR LabView G instrument driver provides programming support for Kepco's KLR Power Supply. It contains functions for opening, configuring, taking measurements from, testing, calibrating and closing the instrument. To successfully use this module, the instrument must be connected to either the LAN or the GPIB.

The driver supports all functions of the KLR except the instrument state commands of Store, Recall and Memory Read. The driver is also supplied with some simple examples and an interactive demo program. Users that are modifying the driver are encouraged to copy parts of this program to maximize usage of tested functional sections. Table 4-1 lists the functions that are available.

TABLE 4-1. KLR LABVIEW G DRIVER FUNCTIONS

Function Name	Purpose
KepcoDCPWR initialize with options	Makes Visa Connection. Creates Open handle Provides ability to issue Reset and parse IDN
KepcoDCPWR initialize	Makes Visa Connections, No Reset, IDN is parsed and Kepco model is properly identified.
Kepco Serial initialize with options	Makes Visa Connection. Creates Open handle Provides ability to issue Reset and parse IDN
KepcoDCPWR close	Closes the driver, freeing resources
Recognize Model	Sends *IDN? And verifies KEPCO string returned
KepcoDCPWR IDN parser	Parse the IDN string – determining model
KepcoDCPWR Revision Query	Uses IDN parser to determine firmware revision
KepcoDCPWR Reset	Sends *RST to Power supply, set output off, volt=0, curr min, over voltage and over current to max values
KepcoDCPWR Reset with Defaults	Supplied for compatibility purposes uses above
KepcoDCPWR Self-test	Sends *TST? To unit and returns result- o= all okay.
Output Control functions	
KepcoDCPWR Configure Voltage Level	Set voltage output setting to a specific value
KepcoDCPWR Configure Current Level	Sets current output setting to a specific level
KepcoDCPWR Query Voltage Level	Queries unit and returns voltage level setting
KepcoDCPWR Query Current Level	Queries unit and returns Current level setting
KepcoDCPWR Configure Voltage Limit	Set virtual voltage setting to a specific value
KepcoDCPWR Configure Current Limit	Sets virtual current setting to a specific level
KepcoDCPWR Query Voltage Limit	Queries unit and returns limit model voltage level setting
KepcoDCPWR Query Current Limit	Queries unit and returns limit model Current setting
KepcoDCPWR Configure OVP	Set over voltage setting to a specific value
KepcoDCPWR Configure OCP	Sets Over current setting to a specific level
KepcoDCPWR Configure Output State	Sets output on or off
KepcoDCPWR Query Output State	Returns Output on(1) or off(0).
Measurement	
KepcoDCPWR Measure [MSR]	Returns the voltage read back 9 default or Current read back value from KLR.
List Functions (supported in standalone configurations only)	
KepcoDCPWR Configure Current Mode	Sets unit to List or Fixed mode of operation. Fixed operation is default
KepcoDCPWR Configure Voltage Mode	Sets unit to List or Fixed mode of operation. Fixed operation is default
KepcoDCPWR Configure User Sequence	Initializes list from an array of either voltage and dwell or Current and dwell to execute the list a number of times.

TABLE 4-1. KLR LABVIEW G DRIVER FUNCTIONS (CONTINUED)

Function Name	Purpose
KepcoDCPWR Clear All Lists	Clears the list if not executing.
KepcoDCPWR Configure List	Loads a specific list from an array of numbers.
KepcoDCPWR Configure List Count	Initializes the list count variable
Trigger Functions	
KepcoDCPWR Configure Triggered Current Level	Establishes the current level upon receipt of a valid trigger
KepcoDCPWR Configure Triggered Voltage Level	Establishes the voltage level upon receipt of a valid trigger
KepcoDCPWR Configure Triggered Source	Establishes the source for a trigger and arms the trigger system.
KepcoDCPWR initiate[TRG]	Issue a software trigger to the KLR.
KepcoDCPWR Abort [TRG]	Clears the trigger system.
Error and status reporting	
KepcoDCPWR Error-Query	Provides entry from the error queue of the power supply.

4.4 USING THE DRIVER FUNCTIONS

The interactive example uses various driver functions of the LabView G driver. The following two paragraphs details the inputs and outputs of two of typical VI's used in the interactive example: KepcoDCPwr Configure Voltage Level.vi (PAR. 4.4.1) and KepcoDCPwr Measure [MSR].vi (PAR. 4.4.2). An overall block is provided for each function which shows inputs and outputs, followed by detailed descriptions of inputs and outputs. A detailed block diagram with associated text is also provided to show how the function is implemented.

4.4.1 KepcoDCPwr Configure Voltage Level.vi

This VI configures the DC voltage level the power supply is expected to generate.

4.4.1.1 CONNECTOR PANEL

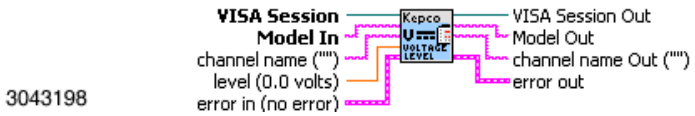


FIGURE 4-1. KepcoDCPwr Configure Voltage Level.vi CONNECTOR PANEL

4.4.1.2 CONTROLS AND INDICATORS

See Table 4-2.

TABLE 4-2. KepcoDCPwr Configure Voltage Level.vi INPUT/OUTPUT DESCRIPTIONS

DESCRIPTION	SYMBOL
channel name - (**“”**) Passes the name of the channel on which to configure the voltage level. It is not required for most supplies and is provided for compatibility. Valid Channel Names: 1 - 27 Default Value: “”	
level (0.0 volts) - Passes the DC voltage the power supply is to generate. The driver uses this value to set the Voltage Level. The level is not checked by the driver function but is checked by the KLR or power supply. Default Value: 0.0 volts	
error in (no error) - The **error in** cluster can accept error information wired from VI’s previously called. Use this information to decide if any functionality should be bypassed in the event of errors from other VI’s. The pop-up option Error (or Explain Warning) gives more information about the error displayed (see Table 4-3).	
VISA Session - The **ViSession** handle is obtained from the *KepcoDCPwr Init* or *KepcoDCPwr InitWithOptions* function. The handle identifies a particular instrument session. Default Value: None	
Model In - **Model In** is the type of Kepco power supply. This defines the capabilities and requirements in other Kepco functions to insure proper operation of the command. This function works on All Kepco power Supplies: BOP-HP, BOP-LP, KLR, STANDARD, MULTI and MULTI_PLUS. The MULTI type power supplies require a channel number to operate correctly. The constant is created by the Recognize Kepco model utility.	
error out - The **error out** cluster passes error or warning information out of a VI to be used by other VI’s. The pop-up option **Explain Error** (or Explain Warning) gives more information about the error displayed (see Table 4-3).	
VISA Session Out - The **ViSession** handle is obtained from the *KepcoDCPwr Init* or *KepcoDCPwr InitWithOptions* function. The handle identifies a particular instrument session. Default Value: None	
Model Out - The **Model Out** handle is obtained from the *KepcoDCPwr Init* or *KepcoDCPwr InitWithOptions* function. The handle identifies a particular instrument. Default Value: None	
channel name out - (**“”**) Passes the name of the channel on which to configure the OVP limit. Valid Channel Names: 1-27; Default Value: “”	

TABLE 4-3. ERROR (OR EXPLAIN WARNING) CODES

	**status** - The boolean is either TRUE (X) for an error, or FALSE (check mark) for no error or a warning. The pop-up option **Explain Error** (or Explain Warning) gives more information about the error displayed.
	**code** - The **code** input identifies the error or warning. The pop-up option **Explain Error** (or Explain Warning) gives more information about the error displayed.
	**source** - The string describes the origin of the error or warning. The pop-up option **Explain Error** (or Explain Warning) gives more information about the error displayed.

4.4.1.3 BLOCK DIAGRAM DESCRIPTION

Figure 4-2 shows that the level input is converted to a string that is compatible with the power supply, floating with 5 places to the right of the decimal (%.5f). Then, using the VISA write function, the string is sent to the power supply. The case statement in the middle of Figure 4-2 shows that all Kepco power supplies support this same functionality.

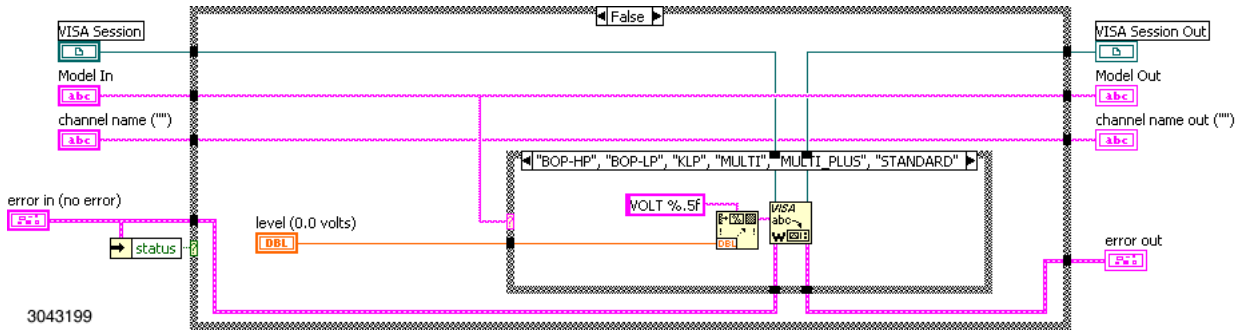


FIGURE 4-2. KepcoDCPwr Configure Voltage Level.vi BLOCK DIAGRAM

Figure 4-3 shows that when the error is true (not equal to zero), the function is a null function, passing the inputs (VISA session, Model, Channel and error) through the function without modification.

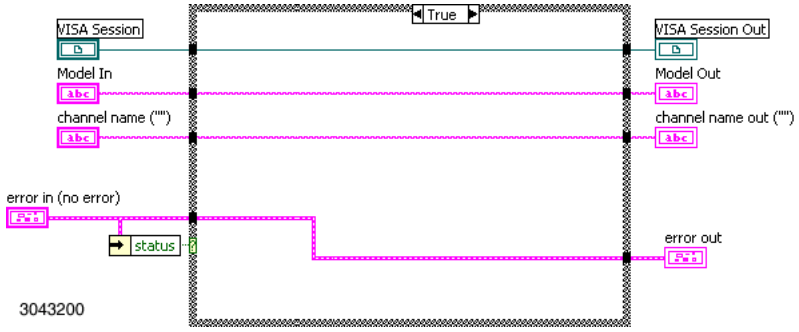


FIGURE 4-3. KepcoDCPwr Configure Voltage Level.vi ERROR BLOCK DIAGRAM

4.4.2 KepcoDCPwr Measure [MSR].vi

This VI takes a measurement of the output signal and returns the measured value (see Figure 4-4.

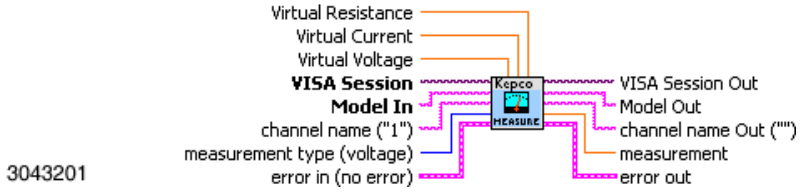


FIGURE 4-4. KepcoDCPwr Measure [MSR].vi CONTROL PANEL

This VI has the standard inputs of all of the functions of the LabView G Driver. It also has three orange inputs at the top of the VI (Virtual Resistance, Virtual Current, Virtual Voltage). These inputs are used in simulation. They do not need to be connected, but provide the ability to verify the operation without using the power supply. These simulation inputs are only used when error in is true (indicating an error has occurred).

TABLE 4-4. KepcoDCPwr Measure [MSR].vi INPUT/OUTPUT DESCRIPTIONS

DESCRIPTION	SYMBOL
channel name - (**"1"**) Passes the name of the channel on which to configure the voltage level. It is not required for most supplies and is provided for compatibility. Valid Channel Names: 1 Default Value: "1"	
error in (no error) - The **error in** cluster can accept error information wired from VI's previously called. Use this information to decide if any functionality should be bypassed in the event of errors from other VI's. The pop-up option **Explain Error** (or Explain Warning) gives more information about the error displayed (see Table 4-3).	
VISA Session - The **ViSession** handle is obtained from the *KepcoDCPwr Init* or *KepcoDCPwr InitWithOptions* function. The handle identifies a particular instrument session. Default Value: None	
Model Out - The **Model Out** handle is obtained from the *KepcoDCPwr Init* or *KepcoDCPwr InitWithOptions* function. The handle identifies a particular instrument. Default Value: None	
channel name out - (**""**) Passes the name of the channel on which to configure the OVP limit. Valid Channel Names: 1; Default Value: ""	

Figure 4-5 is a block diagram which shows the method used by the VI to perform the simulated current measurement. The query MEAS:CURR? is transmitted to the power supply which replies with the current measurement.

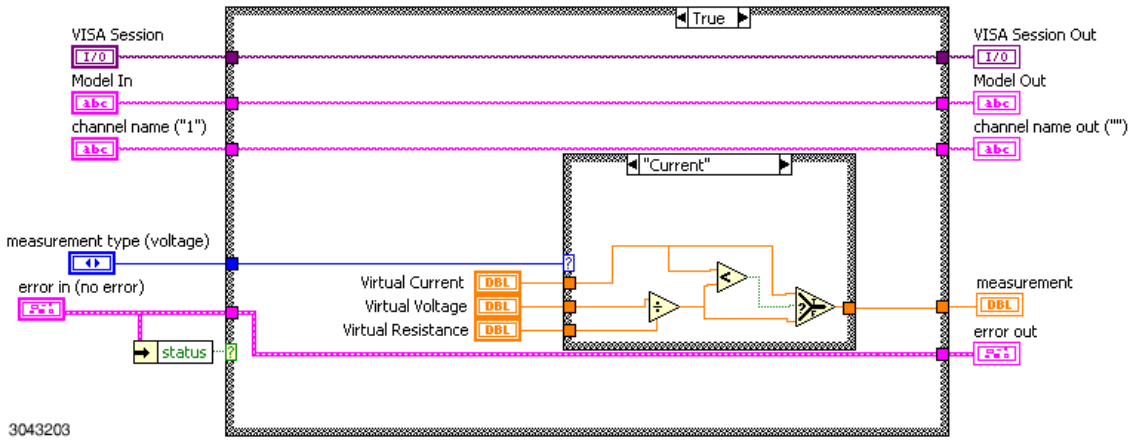


FIGURE 4-5. KepcoDCPwr Measure [MSR].vi BLOCK DIAGRAM

4.5 EXAMPLE OF SETTING THE OUTPUT

Figure 4-6 shows the use of the Kepco subvi's to initialize the driver and the device, set the output on or off, establish a voltage and current setting, get a measurement and unit status, and close the session to release the memory. The file is KepcoDCPwr Voltage Application Example.vi It is located in program files\national Instruments\labview\drivers\kepco. The subvi's used to implement this example are described in the following paragraphs. Although the LabView G driver provides a variety of functions (see Table 4-1), only the ones shown in Figure 4-6, needed for this example.

4.5.1 INITIALIZATION FUNCTION

The initialization function use three subvi's:

- *KepcoDCPWR initialize* - Initializes the driver.
- *KepcoDCPWR Query Voltage Limit* - Verifies the limit model voltage setting.
- *KepcoDCPWR Configure Voltage Mode* - Sets Mode to fixed, thereby stopping list execution.

The initialization function uses five inputs:

- Recognize device - T-F (True-false) input. If true, IDN string is parsed, looking for proper model (KLR). If false, model is automatically set to KLR for simulation mode.
- Resource string - The VISA resource string, e.g., GPIB::6::INSTR
- ID Query - T-F input. If true, allows the unit to be identified by sending *IDN query. If false, program enters simulation mode and no commands are actually sent to the power supply.
- Error in - Described in Table 4-2.
- Reset device - T-F input. If true, sends *RST to unit which turns the output off, sets voltage to zero and current to minimum, and disables execution of a programmed list.

The initialization function provides three outputs:

- Visa Session - a handle defining the connection.
- Model - a string providing the model type for the functions.
- Error out - If the function did not fail, this is a 0.

The first block of Figure 4-6, *KepcoDCPWR initialize* function, performs the open, creating the connection between the LabView program and the power supply using the VISA.dll supplied with LabView. The *KepcoDCPWR initialize* function has two optional settings: reset unit and identify unit. The reset unit option sends *RST to the unit which sets the output to 0 volts, minimum current, turns the output off, and stops list execution. The identify unit option sends an *IDN query to the power supply.

The recognize unit utility function parses the response to determine the proper model number output. If the identify unit flag is not set, the MODEL OUT of the initialize function reports Standard as the power supply type. The Standard model is a unit that only supports setting voltage, current and output on/off, and performing measurements; all other functions are bypassed. The function *KepcoDCPWR initialize with options* (not used in the example) uses *KepcoDCPWR initialize* except that Kepco-recommended options are preset: the reset and identify flags set.

All subsequent functions have at least two inputs: VISA session and Model In. VISA session is the output of *viOpen*. Model In is a string; for the KLR the string is *KLR*.

The second block, *KepcoDCPWR Query Voltage Limit*, is needed to verify the limit model settings. It is recommended that the response from this block be checked to insure that the power supply is configured to supply the voltage and current needed. For example, if the limit model for a KLR 75-32 was set to 36 Volts, 32 Amperes, the response will be 36 to this query, indicating that the program will not respond to a command to set the output to 50 Volts.

The third subVi is *KepcoDCPWR Configure Voltage Mode*. This block sets the mode to Fixed. It is recommended that the mode be set to Fixed in order to stop a list if it is running, however if the Reset Device input into the first block is true, this block is not needed since *RST also stops a list.

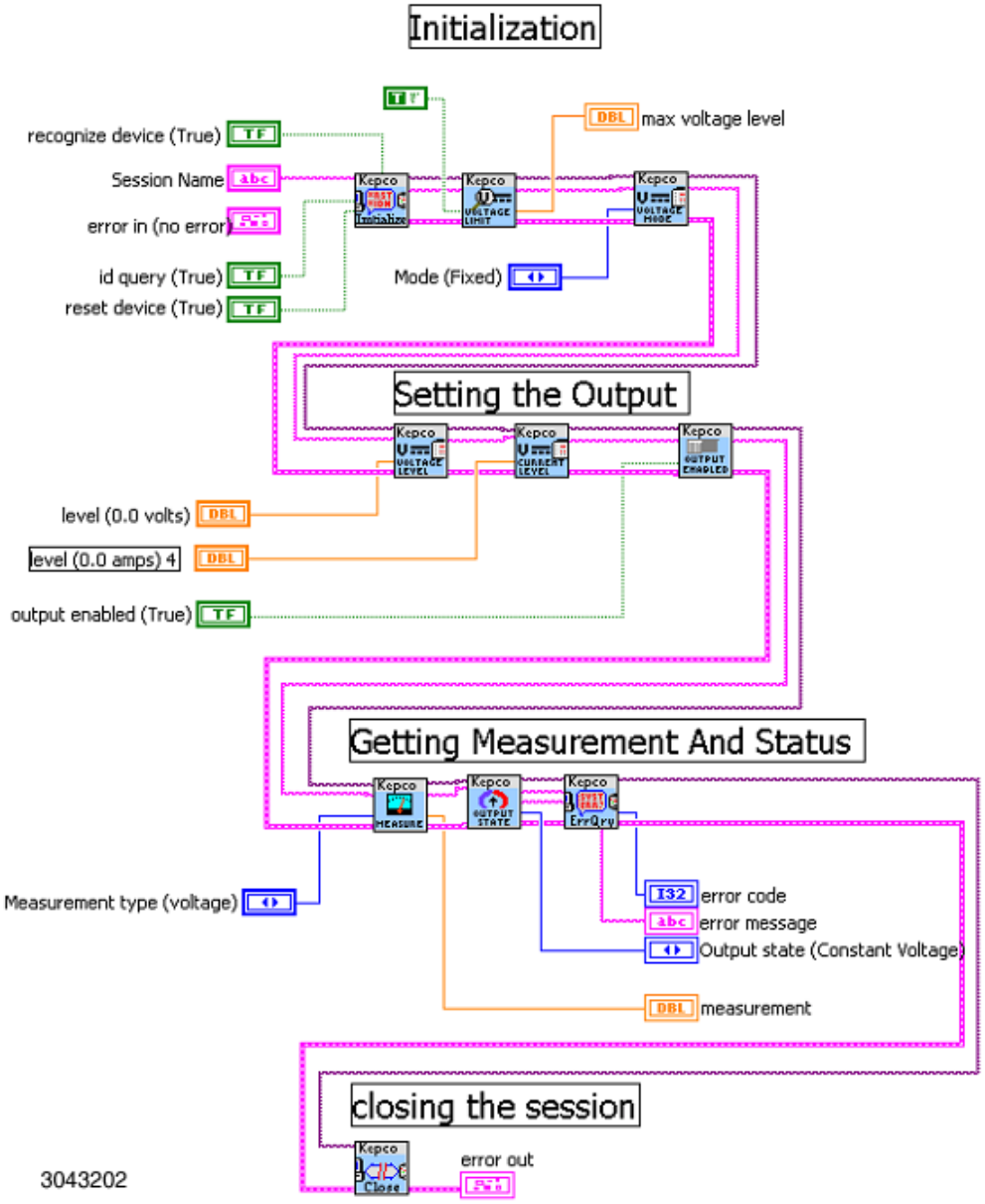


FIGURE 4-6. SETTING THE OUTPUT, OVERALL BLOCK DIAGRAM

4.5.2 SET THE OUTPUT ON OR OFF

The following three functions are all that are required to set a voltage, current and change the output state:

- *KepcoDCPWR Configure Voltage Level* - Sets the voltage output of the power supply
- *KepcoDCPWR Configure Current Level* - Sets the current output of the power supply
- *KepcoDCPWR Configure Output State* - Turns the output on

4.5.3 GET MEASUREMENT AND STATUS

Three functions are needed to retrieve a measurement (either voltage or current), determine the output state and determine if there have been any errors.

- *KepcoDCPWR Measure [MSR]* - Measures the output (voltage in voltage mode, current in current mode). This function is explained in detail in paragraph 4.4.2.
- *KepcoDCPWR Query Output State* - Queries the output state and returns 1 for output on, 0 for output off.
- *KepcoDCPWR Error-Query* - Retrieves the first error from the error queue. The internal logic of this subvi changes the “0- no error” response to a null string, so no error returns nothing, but if an error is detected returns an error string is returned.

4.5.4 CLOSE CONNECTION

The last section is the close function, *KepcoDCPWR close*. This function is required to remove the connection to the instrument and clean up RAM usage in LabView. It is very important to release the connections in E-Series models as there are only four connections possible over VXI-11 (Port 1024) and only two connections possible over SCPI Raw (Port 5025). The *KepcoDCPWR close* function releases the connection for other programs to utilize the device.

When the LabView program completes, it must close the driver, however the output is not automatically turned off. Therefore it is recommended that the *KepcoDCPWR Configure Output State* function precede the *KepcoDCPWR close* function. This is not shown in the simplified example (Figure 4-6), however it is illustrated in the full block diagram showing implementation of the ramp function, Figure 4-12.

4.6 INTERACTIVE DEMONSTRATION PROGRAM

The interactive demo program allows the user to operate the power supply and provides samples of how to implement the functions.

4.6.1 KepcoDCPwr Interactive Example.vi

KepcoDCPwr Interactive Example.vi is the main demonstration program. The file is located in program files\national Instruments\labview\drivers\kepco. Either double-click on the file or open the file using File > Open from within LabView. When the program is opened, the Front Panel (Figure 4-7) is displayed.

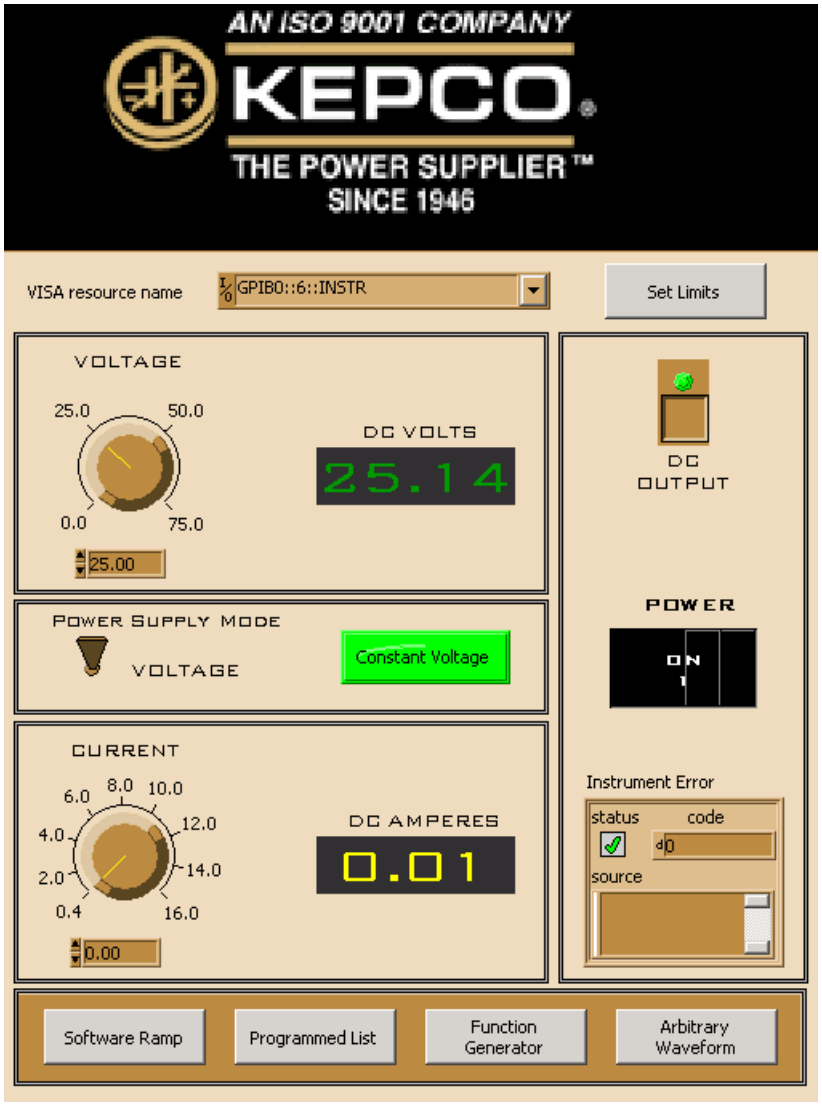


FIGURE 4-7. FRONT PANEL WINDOW

NOTE: When accessed via TCP/IP the four buttons at the bottom of the panel (Software Ramp, Programmed List, Function Generator and Arbitrary Waveform) are not visible.

Below the logo is the VISA Resource Name field. With the application stopped, click on this field to display a drop-down menu. Click Refresh causes the LabView program to initiate a new search and provide an updated list of all devices found on the GPIB, Serial ports and network connections. You may enter a valid resource name directly or, for demo mode, enter a blank resource name.

The VOLTAGE and CURRENT knobs are used to set the voltage and current of the power supply. Once the run button is clicked, the VOLTAGE and CURRENT knob markings will show the limit model minimum and maximum values. The corresponding DC VOLTS and DC AMPERES displays report the measured voltage and current of the power supply and are updated once every 250 milliseconds. As the knob rotates, the number in the counter below the knob changes; if a number is entered in the counter, the knob rotates to the proper position.

If the instrument is found, the demonstration program runs in normal mode. If the instrument is not found (the Instrument Error box at the lower right shows **`instrument not found`**), the demonstration program runs in simulation mode. In simulation mode the resistance value at the bottom of the front panel is a simulated load used to measure the DC output. The simulated load can be set by the user to control the DC VOLTS and DC AMPERES displays.

The POWER SUPPLY MODE switch is used only to select current or voltage mode ramps when using the Software Ramp button. The indicator to the right of the switch shows whether the unit is primed to produce voltage ramps (green) or current ramps (yellow).

The DC OUTPUT switch can be clicked to enable/disable the output. The green indicator within the switch lights when the output is enabled.

The Set Limits button at the top right is used to set the limit model of the KLR.

The four buttons at the bottom of the front panel invoke standalone functions which can be used to 1) create a computer-timed software ramp (Software Ramp), 2) create a ramp using the list functionality (Programmed List), 3) create functions and files for executing complex waveforms (Function Generator) or 4) create an arbitrary series of points (Arbitrary Waveforms).

4.6.2 SOFTWARE TIMED RAMP EXAMPLE

When the Software Ramp button on the front panel is depressed, the panel shown in Figure 4-8 is displayed. The figure illustrates a current ramp; the voltage ramp panel is similar. The {Power Supply Mode switch on the front panel (Figure 4-7) determines whether the current ramp or voltage ramp is displayed. It is also possible to run this program by selecting either:

KepcoDCPwr Software Timed Ramp Example.vi for a voltage ramp
KepcoDCPwr Current Software Timed Ramp Example.vi for a current ramp.

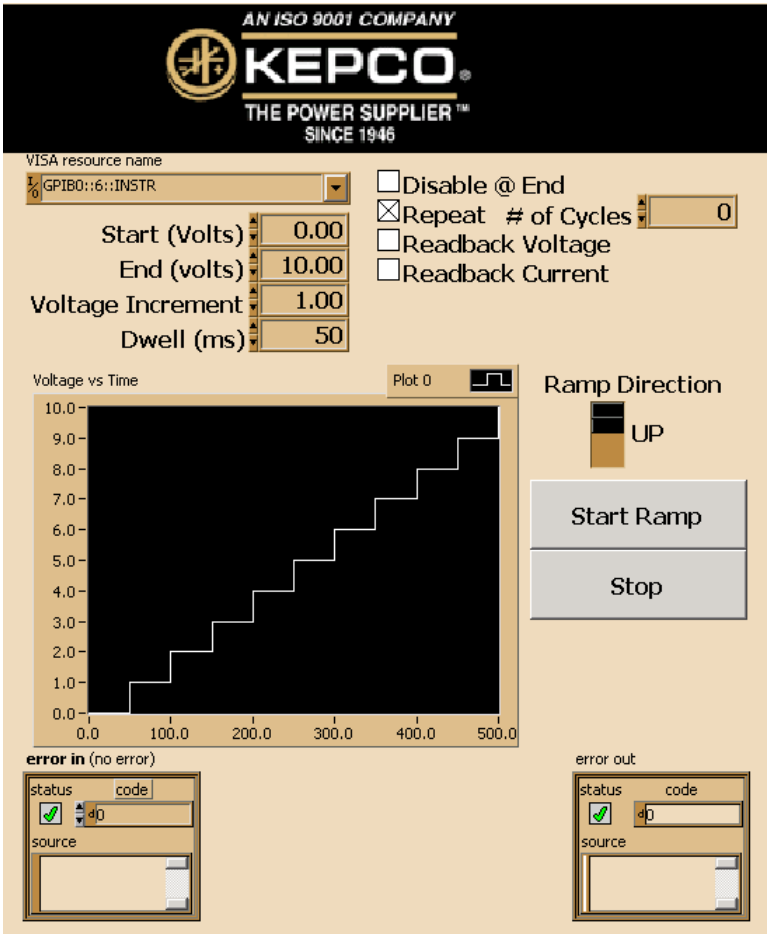


FIGURE 4-8. SOFTWARE RAMP PANEL, CURRENT RAMP EXAMPLE

These files are located in the *kepco DcPWr* folder created when the Kepco LabView G driver is installed and can be executed by selecting File > Run from within LabView. When invoking either of these files directly, either a network address or a GPIB address must be set before clicking Start Ramp (this is automatic when executing Software Ramp via the Front Panel.

The four boxes labeled Start (beginning of ramp), End (end of ramp), Voltage Increment [or Current Increment] (determines the size of steps, affecting the smoothness of the ramp) and Dwell (length of time from start to end of ramp) are used to build the ramp. As soon as these values are entered, the Current [or Voltage] Vs Time Graph shows what the settings will produce when applied to the output.

The check boxes on the right are effective once the Start Ramp button is clicked.

- Disable @ End. Checked causes the output to turn off when the list is complete.

- # of Cycles. Checked to enter how many ramps will be created; setting of 0 causes ramps to continue indefinitely. Unchecked to create one ramp.

- Readback Voltage, Readback Current. Once Start Ramp is clicked, checked causes the DC VOLTS and DC AMPERES indicators on the Ramp Function Panel (see Figure 4-9) to show measured voltage and current; unchecked causes the indicators to show the commanded values of voltage and current, respectively.

When Start Ramp is clicked, information from the check boxes and # of Cycles is passed to the example program and used to start ramp execution (see PAR. 4.6.3).

The Error in and Error out boxes are standard error callouts, showing 0 for no error, and an error code and associated string if there is an error.

4.6.3 SOFTWARE TIMED RAMP EXECUTION

Clicking the Start Ramp button on the Software Ramp panel (Figure 4-8) causes the Ramp Function Panel (Figure 4-9) to be displayed as the ramp is being generated. Checking the Readback Voltage or Readback Current checkboxes above the DC VOLTS or DC AMPERES indicators cause the Voltage and Current indicators to show measured voltage and current vs.commanded values of voltage and current if unchecked.

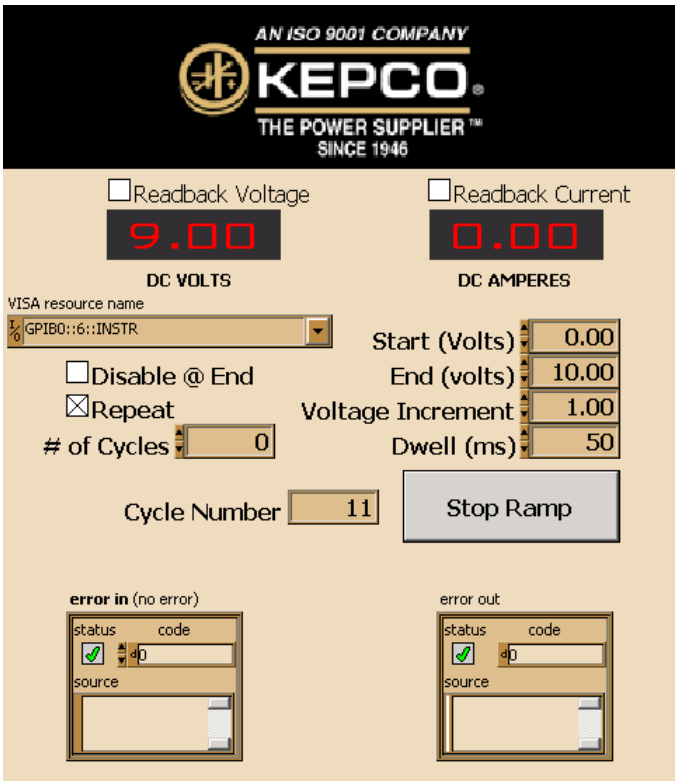


FIGURE 4-9. RAMP FUNCTION PANEL, CURRENT RAMP

The information from the Software Ramp Panel is passed to either *KepcoDCPwr Software Timed Ramp Example.vi* for a voltage ramp or *KepcoDCPwr Current Software Timed Ramp Example.vi* for a current ramp. The computer sends the proper commands to the KLR that generate the ramp. The checkboxes for Readback Voltage, Readback Current can be altered while the ramp is being generated to allow the indicators to display either measured or commanded voltage or current. The Disable @ End and Repeat checkboxes can also be altered while the ramp is being generated.

The actual function that creates the ramp, a series of points, is shown in Figure 4-12. The logic for each point is very simple: send the voltage to the unit with the voltage level command, measure the voltage using the Kepco measure vi and do it again for current. A delay is initiated at the start of the function so the time it takes for the commands to execute does not affect the overall ramp operation (unless the dwell time is less than 0.01 second on an older PC or about 0.008 second on a higher performance computer).

4.6.4 FUNCTION GENERATOR (Supported in Standalone Configurations Only)

NOTE: Complex, dynamic waveshapes typically require a power supply output that can both source and sink current, however KLR models without R Option can only source current, and depend on the load to sink current. KLR Models with R Option can sink current for brief periods.

When the Function Generator button on the Front Panel is clicked, the function generator panel (Figure 4-10) is displayed. The frequency slider determines the frequency of the generated waveform. The number of points determines the limits for frequency adjustment. For example, if 10 points are selected, the maximum frequency is 100 Hz, the lowest frequency is 0.0015 Hz. The number of points determines the smoothness of the waveform because it is constructed incrementally, but it also limits the maximum frequency. For example, if the 250 (maximum) points are selected, the frequency is 0.00001 Hz.

The Amplitude and Offset sliders are to the right of the Frequency slider. When the requested output exceeds the capability of the KLR, the output is clipped by the function generator. Below these sliders is the Waveform Type dropdown selector, either Sine, Sawtooth, Triangle or Square. Below the Waveform Type dropdown is a visual display of the points that have been specified. To the left of the displayed waveform is the Duty Cycle slider which affects square wave-shapes only. The Phase slider shifts the start point of the waveform. As these sliders are changed, the digital display beneath the slider shows the setting. Changes to the slider settings immediately change the Commanded Waveform display to reflect the new parameters.

Above the Start button, two additional parameters required for the execution of the waveform are displayed. The value of the non-changing parameter, current (in Amperes) for a voltage waveform (shown in Figure 4-13) or voltage (in Volts) for a current waveform must be specified. The # of Cycles specifies how many waveforms to produce; if set to 0 the waveform repeats continuously.

Once the Start button is clicked, the commands are sequentially sent to the KLR to build the waveform by incrementally changing the output. While the waveform is being generated, periodic measurements of the KLR output are made and shown in the Measured Waveform display. To stop the waveform click on the Stop button, which is red while the waveform is running.

Figure 4-13 is a block diagram that shows how the function generator was implemented using list commands in this example. The *KepcoDCPwr Clear All Lists.vi* was executed first. This VI clears the previous list, allowing the new list to be sent to the KLR. Lists are not deleted from the KLR except when a limit model setting is changed or when the Clear All function is executed. The *KepcoDCPwr Configure User Sequence.vi* does the actual downloading of the points to the KLR unit. The *KepcoDCPwr Configure Voltage Mode.vi* causes the list to be executed.

The *KepcoDCPwr Configure User Sequence.vi* is shown in Figure 4-11. This logic is specifically configured to generate a voltage list where two arrays of voltage and dwell times are supplied, along with the operating current and number of cycles for the list to execute. In the left-most case statement there is a command to send one entry for the control and current lists of the internal arbitrary waveform generator. The last two functions in the diagram send the dwell and then the voltage list to the KLR. These functions, along with the individual write of the LIST:COUN command, set up the arbitrary waveform generator for the selected number of repetitions.

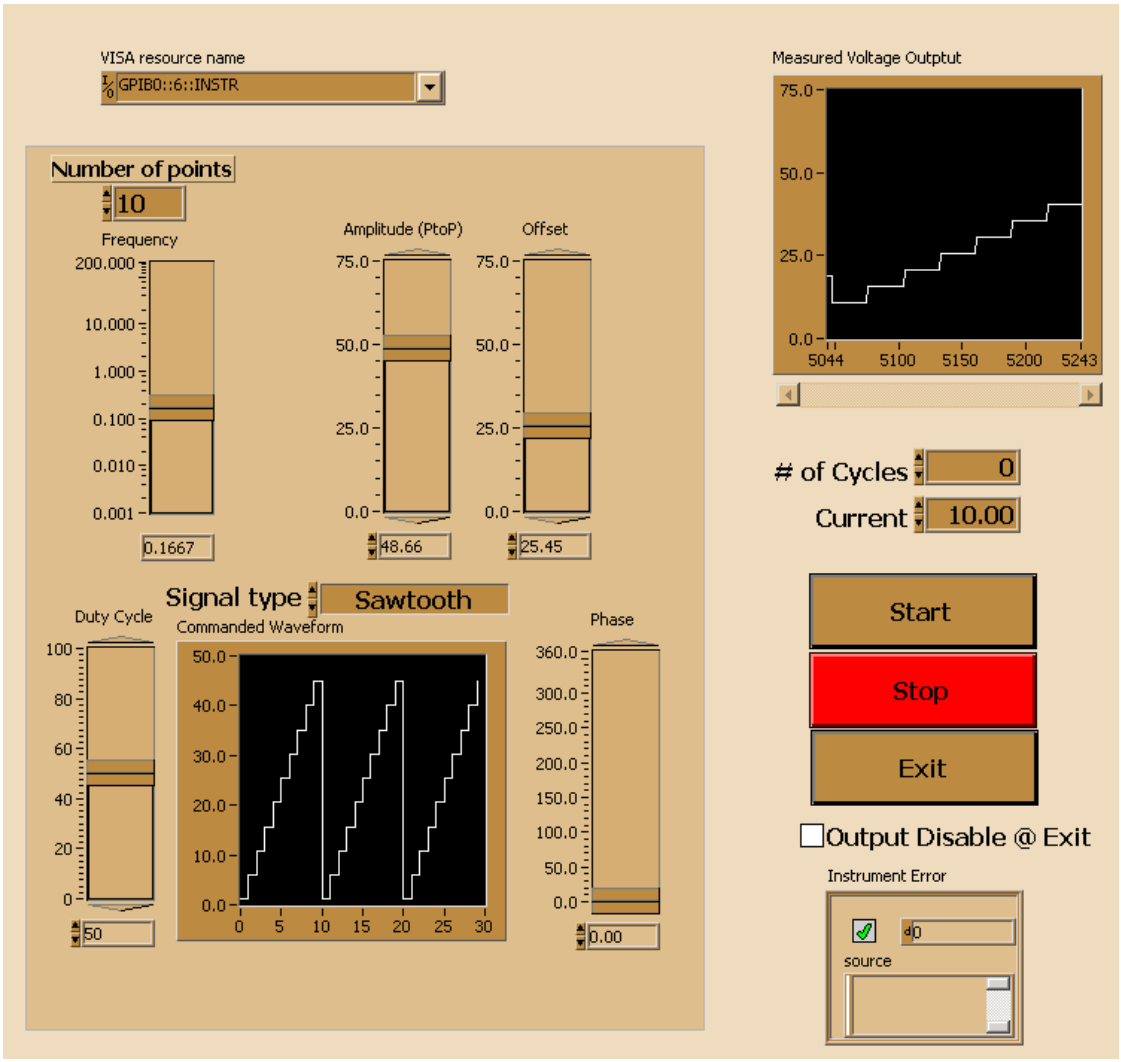


FIGURE 4-10. FUNCTION GENERATOR PANEL

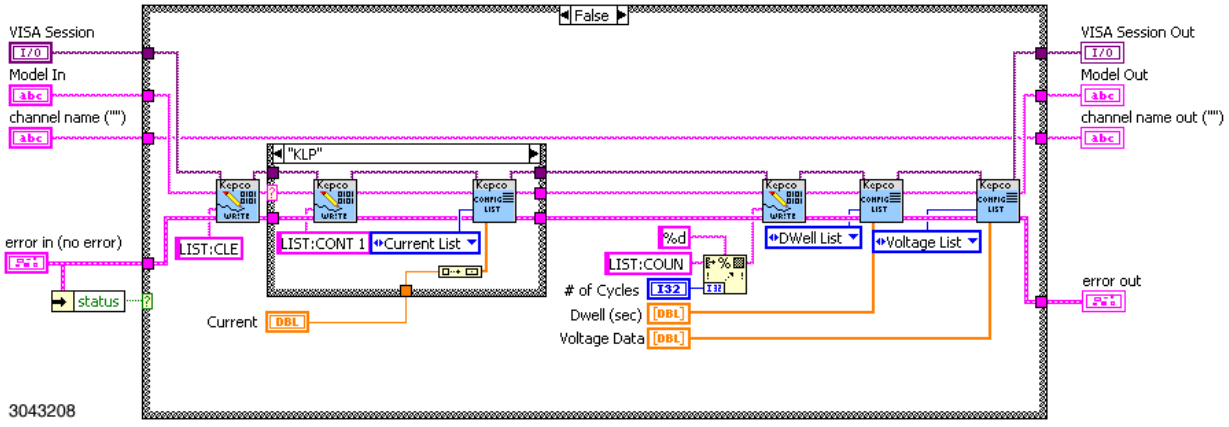


FIGURE 4-11. CONFIGURE User Sequence.vi BLOCK DIAGRAM

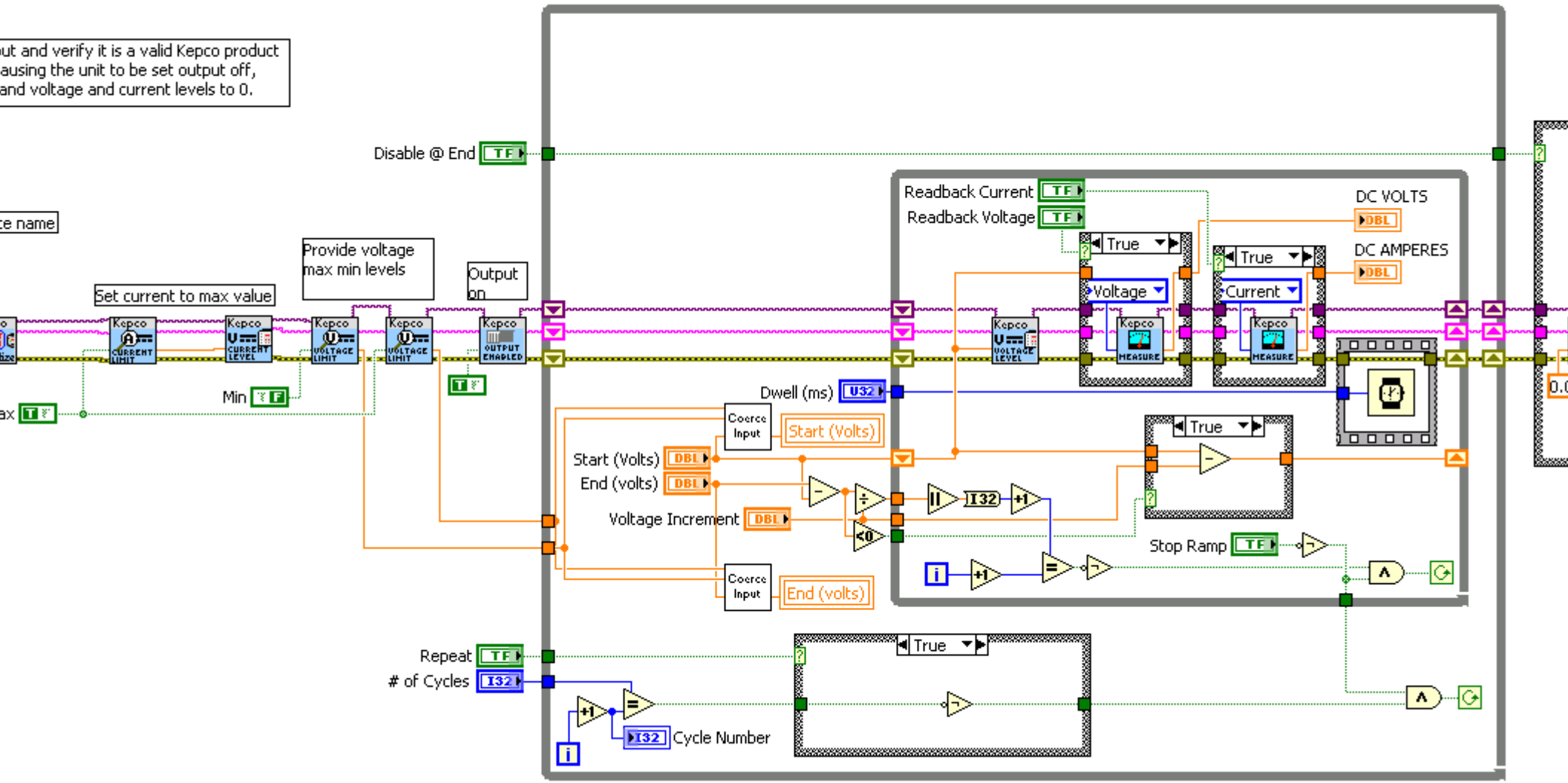


FIGURE 4-12 RAM

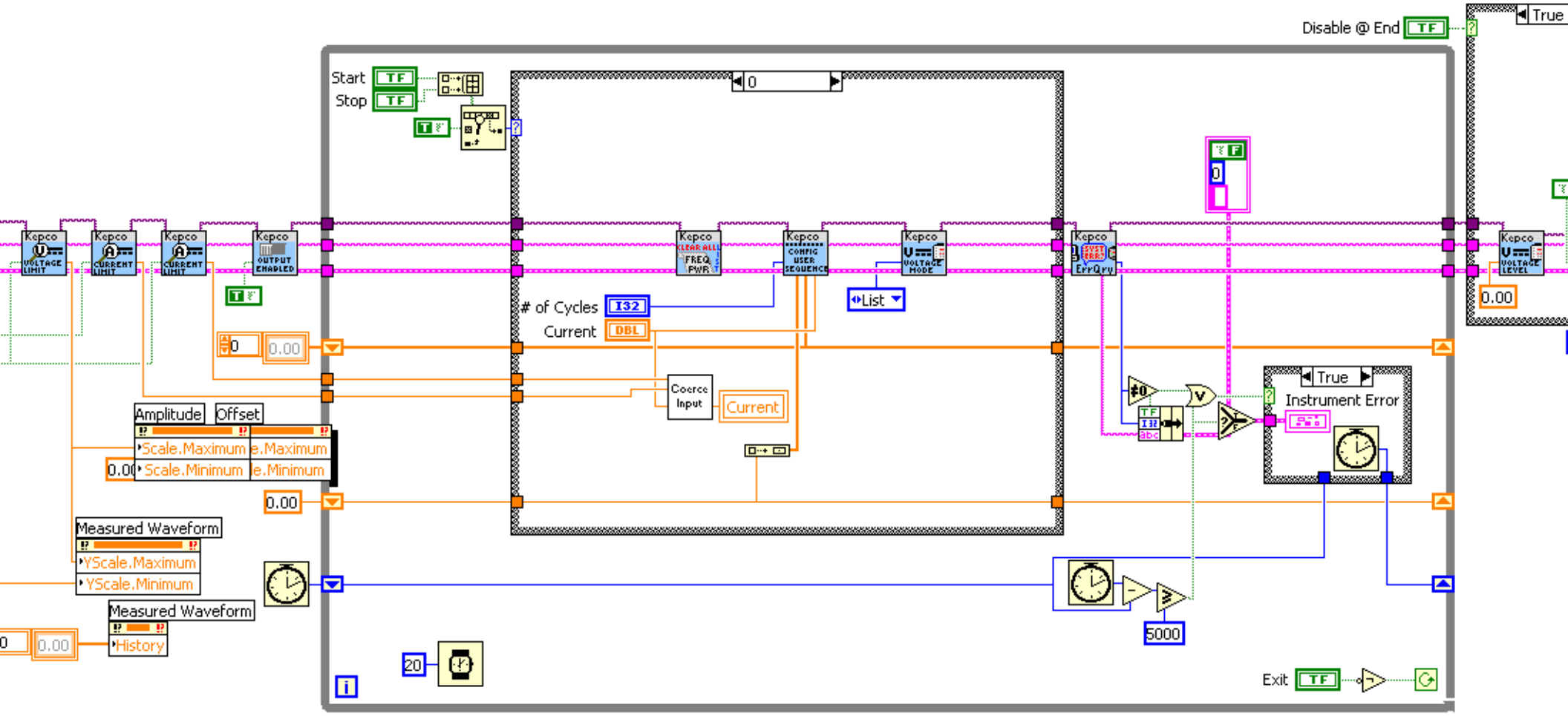


FIGURE 4-13. FUNCTION GE
BLOCK DIAGR

SECTION 5 - VXI *plug&play* DRIVER

5.1 INTRODUCTION

The VXI *plug&play* driver can be used with all KLR models for remote programming.

The driver can be used in a program created by a user with any C compiler to program all ports accessible via a resource string of the KLR power supply.

5.2 VXI *PLUG&PLAY* INSTRUMENT DRIVER

The VXI *plug&play* instrument driver provides support for the VXI *plug&play* environment. This driver can be downloaded from Kepco's web site at www.kepcopower.com/drivers/. The driver includes an executable demonstration program and two files which provide the source code for the driver and the functional prototypes. In addition, the examples found in this section are also included with the driver as separate C files.

5.3 VXI *PLUG&PLAY* INSTRUMENT DRIVER FUNCTIONS

Kepco's KLR VXI *plug&play* instrument driver provides programming support for Kepco's KLR Power Supply via all remote interfaces. It contains functions for opening, configuring, taking measurements from, testing, calibrating and closing the instrument. To successfully use this module, the instrument must be connected to the appropriate port using the proper resource string for the initialize function (see Table 1-1).

The initialize function opens the session, and communicates with the KLR to validate it as a supported model and determines the voltage and current limits of the unit. This driver validates the model to insure it is the proper driver for the unit. The driver differs from the IVI and LabView G driver in that it will only operate on a Kepco KLR product and has not support for generic operation or operating other Kepco supplies.

Table 5-1 lists the functions that are available.

TABLE 5-1. KLR VXI *PLUG&PLAY* DRIVER FUNCTIONS

Purpose	Function Name	Description
INITIALIZE FUNCTION		
Initialize Unit	Kpklr_Init	A resource string is passed to this function and the driver is opened. The handle returned is used in all other functions. This function uses Kpklr_reset and also performs an *IDN? query to verify the unit is a Kepco, KLR unit.
Reset	Kpklr_psReset	Resets the instrument to a known state and sends initialization commands to the instrument.
Close	Kpklr_close	This function takes the instrument off-line.
SETTING AND MONITORING OUTPUT FUNCTIONS		
Set Value	Kpklr_SetValue	Sets the output voltage, the output current of the power supply, depending on the slide selection position. The values are checked against the maximum acceptable values for the corresponding power supply: 0 - voltage, 1 - current, 2 - voltage protection, 3 - current protection
Get Value	Kpklr_GetValue	Gets the output voltage, the output current, voltage protection or current protection of the power supply depending on slide selection position0 - voltage, 1 - current, 2 - voltage protection, 3 - current protection.

TABLE 5-1. KLR VXI *PLUG&PLAY* DRIVER FUNCTIONS (CONTINUED)

Purpose	Function Name	Description
Measure Current and Voltage Output Values	KpklrMeasVoltCurr	Measures the values of output (voltage and current).
SETTING AND MONITORING OUTPUT FUNCTIONS (CONTINUED)		
Get Source Mode	KpklrGetSourceMode	Gets the operating mode of the power supply.
Output On/Off	KpklrOutputOnOff	Sets the output on or off.
Get Output Status	Kpklr_getOutputState	Returns the output status (on or off).
UTILITY FUNCTIONS		
Relay Mode	KpklrGetRelayMode	Gets operating mode of instrument relay. 0 = Fault (factory default), 1 = Manual, 2 = List Program
Get Relay Status	KpklrGetRelayStatus	Gets on/off status of instrument relay.
Set Relay Status	KpklrSetRelayStatus	Sets relay on (energized) or off (de-energized) if Relay Mode is set to Manual (1).
Set/reset Front Panel Lock	KpklrSetResetKeyblock	Sets or resets front panel controls lockout.
Front Panel Lock Status	KpklrKeybLockStat	Gets status of front panel lockout
Revision Query	KpklrRevisionQuery	Returns the revision numbers of the instrument driver and instrument firmware version from the *idn? query. This instrument driver's Revision Number is "Rev 1.0, 9/99, CVI 5.1" and the KLR firmware version is Firmware Version "1.0". This data is necessary when requesting technical support.
Model Query	KpklrModelQuery	Return the model number of the KLR power supply.
Serial Number Query	KpklrSerialnQuery	Returns the serial number of the KLR power supply. The serial number is extracted from the answer to the *idn? query.
Query SCPI Version	KpklrGetScpiVersion	Returns the power supply answer to the Query SCPI Version command. Checks the Standard Commands for Programmable Instruments (SCPI) language version.
Self-Test	KpklrselfTest	Runs the instrument's self test routine and returns the test result(s).
Error-Query	KpklrerrorQuery	Reads an error code from the instrument's error queue.
Error Message	KpklrerrorMessage	Takes the Status Code returned by the instrument driver functions, interprets it and returns it as a user readable string.
Get System Setting	KpklrSYST_Set	Gets System Settings
Get Trigger Value	KpklrGetTrigValue	Gets the trigger voltage or the trigger current level, depending on the switch position.
Set Trigger Value	KpklrSetTrigValue	Sets the trigger voltage or trigger current level, depending on the switch position.
Trigger	KpklrTrig	Triggers the instrument once. The output will go to the trigger voltage and current values.
LIMIT MODEL FUNCTIONS		
Set Max Voltage/Current	KpklrSetMaxLimValue	Sets limit to maximum voltage/current.
Get Max Voltage/Current	KpklrGetMaxLimValue	Gets maximum voltage/current limit values.
Security Code	KpklrCalCode	Allows the user to change the password to prevent accidental or unauthorized calibrations of the instrument. The password is stored in non-volatile memory, and does not change when power has been off or after a reset. To change the password, the instrument must already be in calibration status, ensuring that the user knows the current password. If the password is lost, call the factory for support.
Calibration Dump	KpklrCalDump	Instrument sends calibration data in Intel Hex format.

TABLE 5-1. KLR VXI *PLUG&PLAY* DRIVER FUNCTIONS (CONTINUED)

Purpose	Function Name	Description
Calibration Restore	Kpklr_CalRestore	instrument receives calibration data in Intel hex format and replaces the Working Calibration.
ARBITRARY WAVEFORM FUNCTIONS (Standalone configuration only)		
Clear List	Kpklr_ListCl	Clears the list to prepare for adding points.
Add Volt Step Value	Kpklr_ListVolt	Add a single voltage entry to the end of a list.
Add Current Step Value	Kpklr_ListCurrent	Add a single current entry to the end of a list.
Add Step Time	Kpklr_ListDwell	Add a single time entry to the end of a list.
Add relay control step value	Kpklr_ListControl	Add a single relay control entry to the end of a list.
Query How Many Points	Kpklr_Points	Determines the maximum number of points on the list.
Add waveform to a list	Kpbop_ListData	Add waveform to a list.
Set list execution cycles	Kpklr_ListCount	A list can be run from 1- 65535 times. It can also be run continuously if the value is set to 0. The KLR defaults to 1 on a list clear.
Run/stop LIST program	Kpklr_Prog	Runs or stops LIST program.
LIST Program Run Status	Kpklr_Prog_Stat	Status of LIST program (running or stopped)
STATUS FUNCTIONS		
Set SRQ Condition	Kpklr_SetSrqCond	Sets SRQ Condition Bit.
Display Status Message	Kpklr_getDisplay	Displays message in Status Display.
Initialize Status Registers	Kpklr_InitReg	Clears status Operation and Event registers
Get Questionable Event	Kpklr_StatQues	Tests and returns contents of Questionable Event register.
Wait for Operation to Complete	Kpklr_WaitOPC~	Waits for operation to complete.
Wait for SRQ	Kpklr_WaitSrq	Waits for SRQ

5.4 USING THE VXI *plug&play* DRIVER

The example programs are provided to illustrate how to operate the KLR via a remote interface. The three examples show the following functions:

- Setting Voltage and Current (PAR. 5.4.1)
- Using the Power Supply to Create a Voltage Ramp (PAR. 5.4.2)
- Measurements

5.4.1 EXAMPLE 1: SETTING VOLTAGE AND CURRENT

The simple routine shown in Figure 5-1 sets the power supply to a specific value, then leaves it at this value. It opens the driver, calls the proper functions and then closes the driver. Four possible id_str variable statements have been provided, depending on the interface being used. The id_str statements for the interfaces not used must be commented out, otherwise errors will result.

```
#include <formatio.h>
#include <visa.h>
#include "kp_KLR.h"

/* Order of the H files is important.
visa.h uses some of the formation functions
kp:KLR.h needs definitions from vias.h to compile correctly
***** end of required h files */

Void main{

/* call function to set power supply to 15 volts at 12 amperes */
Set_powersupply(15,12);
}

/* function Set_powersupply

Sets power supply to the supplied voltage and current.
Also sets the output on as setting to a voltage and current
implies output being placed on */

ViStatus Set_powersupply (ViReal64 voltage, ViReal64 current){

ViByte ps_type;
ViSession KLR_Session;

ViStatus power_supply_status = VI_SUCCESS;

/* select one of the following ViChar id_str{}strings depending upon port to be used and
comment out the rest */

ViChar id_str{}="GPIB::06::INSTR"; // change 06 to the GPIB address of unit.

/* For LAN ports ( - change 192.168.0.100 to IP address found via front panel interface
*/

ViChar id_str()="TCIP::192.168.0.100::INSTR"; // VXI-11 instrument- via LAN port 1024
ViChar id_str()="TCIP::i92.168.0.100::5025::SOCKET"; // SCPI-RAW port (5025) LAN instrument

/* For RS 232 change 0 to actual communications port in use */
ViChar id_str()="ASRL0::INSTR"; // serial port

if (
(power_supply_status = Kpklr_init ( &id_str, 1, &ps_type, &KLR_Session)) !=Visuccess)
return (power_supply_status); // could not open power supply

/* init the power supply operation
Power supply handle is now created if unit is KLR
     ************************************/

Kpklr_Set_Volt_Curr (KLR_Session, voltage, current); //voltage and current
/*
The above function sets both the voltage and current setpoint of the power supply
*/


0..._OutputOnOff ( KLR_Session, 1); // output on


Kpklr_close(&KLR_Session); // free up memory and close VISA session

return (power_supply_status);
} // end of main

// if not created as a project, include the driver code at end of program

#include "Kp_klr.c"
```

FIGURE 5-1. EXAMPLE 1: SETTING VOLTAGE AND CURRENT

5.4.2 EXAMPLE 2: USING THE POWER SUPPLY TO CREATE A VOLTAGE RAMP

A ramp is created by sending a series of points to the power supply at a defined interval. The example shown in Figure 5-2 initializes the power supply to the starting voltage and current and turns the output on. The “for” loop provides the actual voltage changes to the power supply once every 500 milliseconds. The loop uses the set value function to only update the voltage setting during the loop. This speeds up the process and follows the Kepco’s recommended programming practice of only changing one variable during the loop. The current setting is set for the worst case output setting of the power supply to insure it remains in voltage mode of operation and has the cleanest and fastest rise times.

```
#include <formatio.h>
#include <visa.h>
#include "kp_KLR.h"
/* Order of the H files is important
visa.h uses some of the formation functions
kp:KLR.h needs definitions from visa.h to compile correctly
***** end of required h files */

ViByte ps_type;
ViSession KLR_Session;

void main{
/* these three variables determine the ramp
start voltage, end voltage and rate of change */

ViReal64 voltage_start=0;
ViReal64 voltage_end= 10;
ViReal64 voltage_change=.5;
ViReal64 voltage=0; // used during the ramp startup
ViReal64 current=16; // current during the ramp execution
Unsigned int i;


ViStatus power_supply_status = VI_SUCCESS;

/* select one of the following strings depending upon port to be used */
ViChar id_str{}="GPIB::06::INSTR"; // change 06 to the GPIB address of unit.

/* LAN addresses - change 192.168.0.100 to IP address found via front panel interface
*/
ViChar id_str()="TCIP::192.168.0.100::INSTR"; // VXI-11 instrument- via LAN port 1024
ViChar id_str()="TCIP::i92.168.0.100::5025::SOCKET"; // SCPI-RAW port (5025) LAN instrument
/* change 0 to actual communications port in use */
ViChar id_str()="ASRL0::INSTR"; // serial port

if (
(power_supply_status = Kpklr_init ( &id_str, 1, &ps_type, &KLR_Session)) !=Visuccess)
                              return (power_supply_status); // could not open power supply

/* init the power supply operation
Power supply handle is now created provided product is KLR
                 ************************************/
```

FIGURE 5-2. EXAMPLE 2: USING THE POWER SUPPLY TO CREATE A VOLTAGE RAMP (SHEET 1 OF 2)

```
Kpklr_Set_Volt_Curr (KLR_Session, voltage, current); //voltage and current
/*
The above function sets both the voltage and current setpoint of the power supply

*/

Kpklr_OutputOnOff ( KLR_Session, 1); // output on

for (i=0;i<10;i++){            // generates the ramp ten times
         for (voltage-voltage_start; voltage <=voltage_end; voltage+=voltage_change){
                  Kpklr_SetValue (KLR_Session, voltage,0); // set the voltage level only
                  Delay(500);                              // delay for 500 milliseconds

                                 }
 Kpklr_SetValue (KLR_Session, voltage_start,0); // set the voltage level only
 delay (1000);                // wait for unit to return to zero (no load)
}                             // end of all ramps

Kpklr_OutputOnOff ( KLR_Session, 0); // output off

Kpklr_close( KLR_Session); // free up memory and close VISA session

return (power_supply_status);
} // end of main

// if not created as a project, include the driver code at end of program

#include "Kp_KLR.c"
```

FIGURE 5-2. EXAMPLE 2: USING THE POWER SUPPLY TO CREATE A VOLTAGE RAMP (SHEET 2 OF 2)

5.4.3 EXAMPLE 3: USING THE POWER SUPPLY TO TAKE MEASUREMENTS

Figure 5-3 is an example of using the power supply to take measurements

```
#include <formatio.h>
#include <visa.h>
#include "kp_KLR.h"
/* Order of the H files in important
visa.h uses some of the formation functions
kp:KLR.h needs definitions from visa.h to compile correctly
***** end of required h files */

/* select one of the following strings depending upon port to be used */
ViChar id_str{}="GPIB::06::INSTR"; // change 06 to the GPIB address of unit.
/* LAN addresses - change 192.168.0.100 to IP address found via front panel interface
*/
ViChar id_str()="TCIP::192.168.0.100::INSTR"; // VXI-11 instrument- via LAN port 1024
ViChar id_str()="TCIP::i92.168.0.100::5025::SOCKET"; // SCPI-RAW port (5025) LAN instrument
/* change 0 to actual communications port in use */
ViChar id_str()="ASRL0::INSTR"; // serial port
ViByte ps_type;
ViSession KLR_Session;

Void main{

ViReal64 meas_volt,meas_curr; // variables for the measurement results

/* call function to set power supply to 15 volts at 12 amperes */
         if (Kpklr_init ( &id_str, 1, &ps_type, &KLR_Session) !=Visuccess)
                  return; // could not open power supply
         Set_powersupply(15,12);
         delay (100);

            Kpklr_MeasVoltCurr (KLR_session, *meas_volt, *meas_curr);
            Kpklr_close(KLR_Session); // free up the memory and close visa session

}


ViStatus Get_powersupply (ViReal64 *voltage, ViReal64 *current){

ViStatus power_supply_status = VI_SUCCESS;

return (power_supply_status);
} // end of readback
```

FIGURE 5-3. EXAMPLE 3: USING THE POWER SUPPLY TO TAKE MEASUREMENTS (SHEET 1 OF 2)

```
/* function Set_powersupply

         Sets power supply to the supplied voltage and current
         Also sets the output on as setting to a voltage and current
         Implies output being placed on

*/



ViStatus Set_powersupply (ViReal64 voltage, ViReal64 current){

ViStatus power_supply_status = VI_SUCCESS;


Kpklr_Set_Volt_Curr (KLR_Session, voltage, current); //voltage and current
/*
The above function sets both the voltage and current setpoint of the power supply

*/


Kpklr_OutputOnOff ( KLR_Session, 1); // output on




return (power_supply_status);
} // end of main

// if not created as a project, include the driver code at end of program

#include "Kp_KLR.c"
```

FIGURE 5-3. EXAMPLE 2: USING THE POWER SUPPLY TO TAKE MEASUREMENTS (SHEET 2 OF 2)

5.5 DEMONSTRATION PROGRAM USING THE VXI *PLUG&PLAY* DRIVER

The demonstration program is intended to illustrate the use of the VXI *plug&play* functions included with the KLR power supply. The demonstration program is installed under Windows by running SETUP.EXE. The program as written presents a virtual front panel for control of a single KLR power supply

The following paragraphs describe the windows and the associated controls and indicators provided with the demonstration program. For additional details regarding operation of the KLR, refer to the operating instructions for local and remote mode found in the KLR User Manual.

5.5.1 INSTRUMENT SETUP

NOTE: Please ensure that all programming connections (GPIB or RS 232 as applicable) are in place before switching the power supply on. Before running the program, verify the GPIB address and adjust if necessary. For RS 232 communications the power supply must be set to 38400 baud. Ethernet programming is not supported.

After the program is installed, double clicking KLRCTRL.exe starts the program and opens the Instrument Setup window (see Figure 5-4). Select the appropriate communications method (TYPE) from the Instrument Resource list provided. If GPIB is selected, enter the corresponding GPIB address in the box labeled ADDRESS. If RS 232 is selected, enter the number of the communications port used (e.g., for COM1 enter "1"). Click the CONNECT button to establish communication with the power supply. If successful, the power supply model number is displayed in the box labeled POWER SUPPLY TYPE FOUND. If not, the Error indicator is red and an error message is displayed in the box. The most common errors are incorrect GPIB address or improper baud rate setting (RS 232). If the power supply displayed is correct, click the Continue button to open the main panel (Figure 5-5).

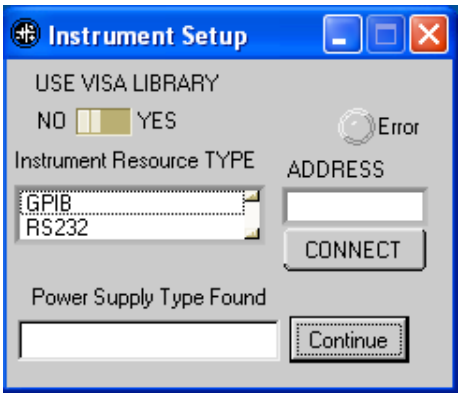


FIGURE 5-4. INSTRUMENT SETUP WINDOW

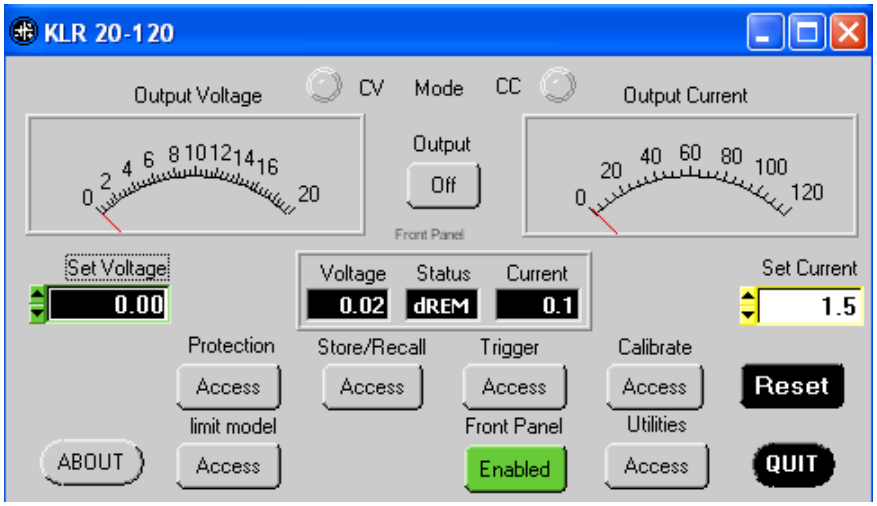


FIGURE 5-5. MAIN PANEL WINDOW

5.5.2 MAIN PANEL

The main panel window allows remote access to many power supply parameters without having to operate the local controls and read the corresponding displays, and provides a demonstration of the programming flexibility of the KLR power supply. The main panel is a real-time display of output values and programmed parameters.

Various functions accessible via the panel (e.g voltage and current settings, protection limits, trigger settings, etc.) may be altered either by clicking on the arrows to left of the applicable display or by using the mouse to highlight the setting, typing in a new value and then exiting the field either by using the Tab key or by clicking the mouse button a second time. Refer to the indicated paragraphs of the KLR User Manual for programming details.

The analog meters read actual output voltage and current; Two windows beneath the meters are provided to enter voltage and current setpoints. The **Front Panel box** shows a digital readout of the voltage and current displayed on the analog meters (this is identical to the readouts on the KLR front panel DC VOLTS and DC AMPERES displays) as well as showing messages appearing on the front panel Status Display. The operating mode (CV or CC) Mode indicators light to indicate whether the power supply is operating in either CV (constant voltage) or CC (constant current) mode.

The **Output** button applies the programmed settings to the output terminals when set to ON or keeps the output voltage at zero and current at minimum when set to OFF. When the output is enabled the button color changes from gray to green.

Set Voltage and **Set Current** windows are used to program the output voltage and current for the unit. These settings may be changed by the user (see KLR User Manual, “Setting Voltage or Current”).

The **Protection** button opens the Protection Window (Figure 5-6) which displays the present overvoltage and overcurrent settings; these settings may be changed by the user. These entries are automatically limited by the Limit Model settings as described in the KLR User Man-

ual, “Viewing/Changing Overvoltage or Overcurrent Protection Values,” and may affect prior output settings.

3043213

FIGURE 5-6. PROTECTION WINDOW

The **Store/Recall** button displays the Store/Recall window (Figure 5-7) which can be used to store or recall up to 40 different power supply settings. After selecting a location, the Store button saves the current power supply settings for Voltage Setpoint, Current Setpoint, and Output State (On or Off) at the selected location. Whenever the same location is selected, the Recall button restores the saved settings.

3043214

FIGURE 5-7. STORE/RECALL WINDOW

The **Trigger** button opens the Trigger window (Figure 5-8) which allows setting of trigger voltage and current values. Clicking the Trigger button within the Trigger window causes the power supply output to be programmed to the settings stored in the Trigger Voltage and Trigger Current displays.

3043215

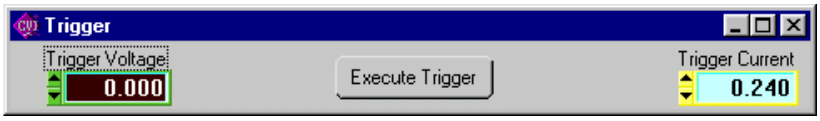


FIGURE 5-8. TRIGGER WINDOW

The **Calibrate** button opens the Calibration Window (Figure 5-9), and is used to recalibrate the unit (see Section 4 of the KLR User Manual). This button is disabled for master/slave configurations; units must be configured for Standalone to be calibrated.

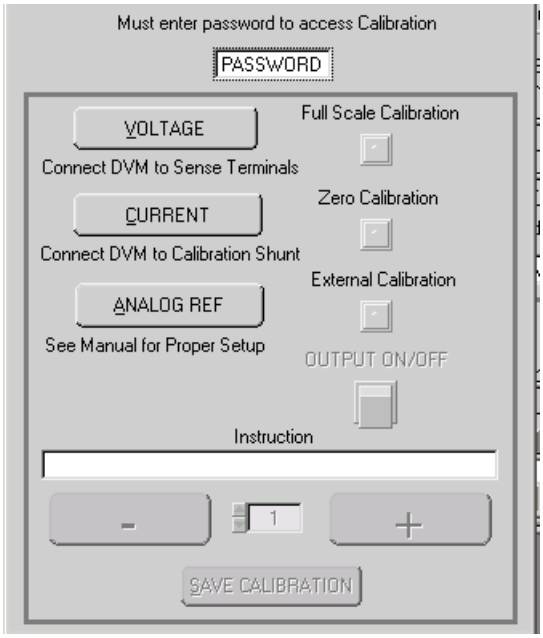


FIGURE 5-9. CALIBRATION WINDOW

The **ABOUT** button displays the model, serial number, firmware version number and driver version number. Click OK to close the window

The **Limit Model** button opens the Limit Model window (Figure 5-10) which displays the present limit model settings; these limits may be changed by the user. To change these limits the Calibration Password (see KLR User Manual) must be entered. Once either limit is altered, the output is automatically disabled and the previous settings for voltage and current are lost.

FIGURE 5-10. LIMIT MODEL WINDOW

The **Front Panel** button allows all front panel controls to be locked during remote operation. When the front panel is locked the button color changes from green to red.

The **Utilities** button opens the Utilities window (Figure 5-12).

Calibration Data (hex format) may be archived and restored using the **Dump** and **Restore** buttons, respectively. When the **Dump** button is clicked, the program will prompt the user for a location and file name for all three calibration locations (see KLR User Manual). The default file names and locations may be altered by the user. In each case the user may elect not to save one or more calibration locations by clicking the Cancel button.

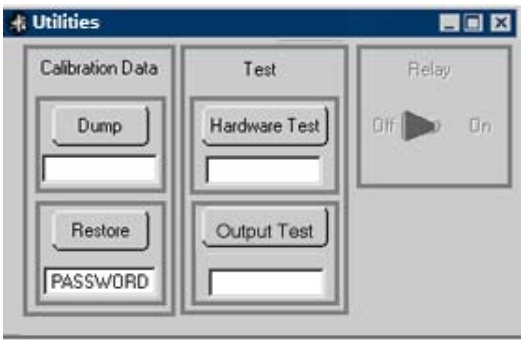


FIGURE 5-11. UTILITIES WINDOW

To restore the contents of the Working location to a previously dumped set of values, enter the calibration password in the box below the **Restore** button and then click the button. If the correct password is entered, the program will open a dialog box permitting the user to select from previously saved calibration data sets. Highlight the desired replacement calibration data and click **Select** to replace the Working calibration; click **Cancel** to exit without changing the Working calibration.

The **Hardware Test** button tests the validity of system parameters (e.g., CRC).

The **Output Test** button verifies basic performance by first testing maximum voltage output, then testing minimum voltage output. Note that this test must be performed at no load (open circuit at output power terminals). When Output Test is running, the button changes to **Running** (see Figure 5-12.); do not click the button while the test is running.

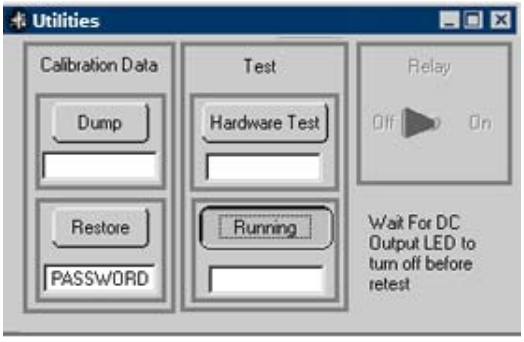


FIGURE 5-12. UTILITIES WINDOW WITH OUTPUT TEST RUNNING

The **Reset** button resets the unit to the power up defaults: output voltage set to zero, current set to minimum current, and output off.

The **QUIT** button on the Main panel (Figure 5-5) is used to exit the sample VXI *plug&play* application.

5.5.3 POWER SUPPLY EVENTS

A Source Power Loss error will cause the Power Supply Event Window (Figure 5-13) to open. This allows the user an opportunity to correct the error condition and continue or quit the VXI *plug&play* application. If this window opens when operating a master/slave configuration, power to both units must be turned off, then reapplied before pressing either of the two buttons, otherwise the VXI *plug&play* application will terminate incorrectly.

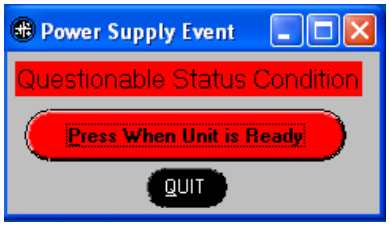


FIGURE 5-13. POWER SUPPLY EVENT WINDOW

SECTION 6 - PROGRAMMING THE KLR USING SCPI COMMANDS

6.1 SCPI PROGRAMMING

SCPI (Standard Commands for Programmable Instruments) is a programming language conforming to the protocols and standards established by IEEE 488.2 (reference document *ANSI/ IEEE Std 488.2, IEEE Standard Codes, Formats, Protocols, and Common Commands*). SCPI commands are sent to the KLR Power Supply as ASCII output strings within the selected programming language (PASCAL, C, BASIC, etc.) in accordance with the manufacturer's requirements for the particular GPIB controller card used.

Different programming languages (e.g., BASIC, C, PASCAL, etc.) have different ways of representing data that is to be put on the control bus. It is up to the programmer to determine how to output the character sequence required for the programming language used. If the IEEE 488.2 (GPIB) control bus is used, address information (GPIB address) must be included before the command sequence. (See PAR.7.2.1 to establish the KLR Power Supply GPIB address.)

6.2 SCPI MESSAGES

There are two kinds of SCPI messages: program messages from controller to power supply, and response messages from the power supply to the controller. Program messages consist of one or more properly formatted commands/queries and instruct the power supply to perform an action; the controller may send a program message at any time. Response messages consist of formatted data; the data can contain information regarding operating parameters, power supply state, status, or error conditions.

6.3 COMMON COMMANDS/QUERIES

Common commands and queries are defined by the IEEE 488.2 standard to perform overall power supply functions (such as identification, status, or synchronization) unrelated to specific power supply operation (such as setting voltage/current). Common commands and queries are preceded by an asterisk (*) and are defined and explained in Appendix A. Refer also to syntax considerations (PARs 6.5 through 6.7).

6.4 SCPI SUBSYSTEM COMMAND/QUERY STRUCTURE

Subsystem commands/queries are related to specific power supply functions (such as setting output voltage, current limit, etc.) Figure 6-2 is a tree diagram illustrating the structure of SCPI subsystem commands used in the KLR Power Supply with the "root" at the left side, and specific commands forming the branches. The following paragraphs introduce the subsystems; subsystem commands are defined and explained in Appendix B.

6.4.1 DISPLAY SUBSYSTEM

This subsystem returns the character string displayed in the Status display.

6.4.2 TRIGGER SUBSYSTEM

This subsystem enables the trigger system. When an internal trigger is enabled, the triggering action will occur upon receipt of a GPIB <GET>, *TRG or TRIGger command. When an external trigger is enabled, the triggering action occurs when a ground is applied to J2, pin 14. If a trigger circuit is not enabled, all trigger commands are ignored. Abort sets the trigger subsystem to the ready-to-arm state.

6.4.3 ABORT SUBSYSTEM

This subsystem cancels any pending trigger levels previously stored and resets them to the immediate value. ABORt also resets the WTG bit in the Operation Condition register. ABORt is executed upon initial power-up of the power supply.

6.4.4 LIST SUBSYSTEM (Supported in Standalone Configurations Only)

The LIST subsystem is represented by the 250 memory locations (groups of settings) which are stored in the volatile memory. Each setting contains values for: Current, Voltage, Dwell, and Relay. The range for the first two values is constrained by the limit model and protection limit settings. The range for the dwell time is between 0.01 and 655.36 seconds. If the relay is configured to LIST using the UTIL menu on the front panel, it can be set to 1 (on, energized) or 0 (off, de-energized). Even when the relay is not configured to run a list, LIST:CONT (either 0 or 1) must be executed once, otherwise the list will not execute.

6.4.5 MEASURE SUBSYSTEM

This query subsystem returns the voltage and current measured at the power supply's output terminals.

6.4.6 OUTPUT SUBSYSTEM

This subsystem controls the power supply's output state: on or off.

6.4.7 STATUS SUBSYSTEM

This subsystem programs the power supply status register. The power supply has two groups of status registers: Operation and Questionable. Each group consists of three registers: Condition, Enable, and Event.

6.4.8 STORAGE SUBSYSTEM

The KLR has 40 memory locations where the current state of the power supply can be stored and then recalled. The storage system saves five settings, Voltage, Current, Voltage protection level, Current protection level and output on/off state. The advantage to using these 40 locations instead of sending five commands is processing time. The Saved data is binary data and requires no processing before applying it to the output, thereby reducing the time needed to change from one state to another to less a millisecond. The *SAV command saves the five power supply setting to a location. The *RCL command recalls the fives settings from a location and applies them it to the power supply output. The MEM:LOC command allows the data to be loaded into a location without actually programming the KLR. The MEM:LOC? query allows the five settings to be retrieved prior to applying them to the output, providing a method of verifying that the settings are correct before applying them to a unit under test.

6.4.9 SYSTEM SUBSYSTEM

This subsystem controls the RS 232 port, GPIB address, passwords, security, language, keyboard lockout, and compatibility with older Kepco equipment.

6.4.10 [SOURCE:]VOLTAGE AND [SOURCE:]CURRENT SUBSYSTEMS

These subsystems program the output voltage and current of the power supply. An R-Option (Rapid Output Discharge Circuit installed) unit configured for Standalone operation can generate a voltage or current transient from 0.04 to 655.36 seconds in length.

6.4.11 CALIBRATE SUBSYSTEM

CAL commands and queries are used to perform calibration of the unit via the control interface. These commands must be issued in a specific sequence in order to properly calibrate the unit. To use these commands, refer to Kepco's website (www.kepcopower.com/drivers) and download the LabWindows/CVI Version 5 driver for KLR. This file provides remote calibration capability and uses the following supported commands and queries:

CAUTION: The CAL commands are provide for reference only. Use of these commands in any way, other than in the supplied remote calibration application, may result in irrecoverable corruption of the internal calibration files.

CAL:CEXT command
CAL:CGA command
CAL:CURR:LEV command
CAL:CURR[:DATA] command
CAL:DPOT command
CAL:SAVE command
CAL:STAT command and query
CAL:VEXT command
CAL:VGA command
CAL:VOLT:LEV command
CAL:VOLT[:DATA] command
CAL:ZERO command

6.5 PROGRAM MESSAGE STRUCTURE

SCPI program messages (commands from controller to power supply) consist of one or more ***message units*** ending in a ***message terminator*** (required by Kepco power modules). The message terminator is not part of the syntax; it is defined by the way your programming language indicates the end of a line ("newline" character). The message unit is a keyword consisting of a single command or query word followed by a message terminator (e.g., CURR?<newline> or TRIG<end-of-line>). The message unit may include a data parameter after the keyword separated by a space; the parameter is usually numeric (e.g., CURR 5<newline>), but may also be a string (e.g., OUTP ON<newline>). Figure 6-1 illustrates the message structure, showing how message units are combined. The following subparagraphs explain each component of the message structure.

NOTE: An alternative to using the message structure for multiple messages defined in the following paragraphs is to send each command as a separate line. In this case each command must use the full syntax shown in Appendix B.

6.5.1 KEYWORD

Keywords are instructions recognized by a decoder within the KLR, referred to as a "parser." Each keyword describes a command function; all keywords used by the KLR are listed in Figure 6-2.

Each keyword has a long form and a short form. For the long form the word is spelled out completely (e.g. STATUS, OUTPUT, VOLTAGE, and TRIGGER are long form keywords). For the short form only the first three or four letters of the long form are used (e.g., STAT, VOLT, OUTP, and TRIG). The rules governing short form keywords are presented in Table 6-1.

You must use the rules above when using keywords. Using an arbitrary short form such as ENABL for ENAB (ENABLE) or IMME for IMM (IMMEDIATE) will result in an error. Regardless of which form chosen, you must include all the letters required by that form.

To identify the short form and long form in this manual, keywords are written in upper case letters to represent the short form, followed by lower case letters indicating the long form (e.g., IMMediate, EVENt, and OUTPut). The parser, however, is not sensitive to case (e.g., outp, OutP, OUTPUt, ouTPut, or OUTp are all valid).

TABLE 6-1. RULES GOVERNING SHORTFORM KEYWORDS

IF NUMBER OF LETTERS IN LONGFORM KEYWORD IS:	AND FOURTH LETTER IS A VOWEL?	THEN SHORT FORM CONSISTS OF:	EXAMPLES
4 OR FEWER	(DOES NOT MATTER)	ALL LONG FORM LETTERS	MODE
5 OR MORE	NO	THE FIRST FOUR LONG FORM LETTERS	MEASure, OUTPut, EVENt
	YES	THE FIRST THREE LONG FORM LETTERS	LEVel, IMMediate, ERRor

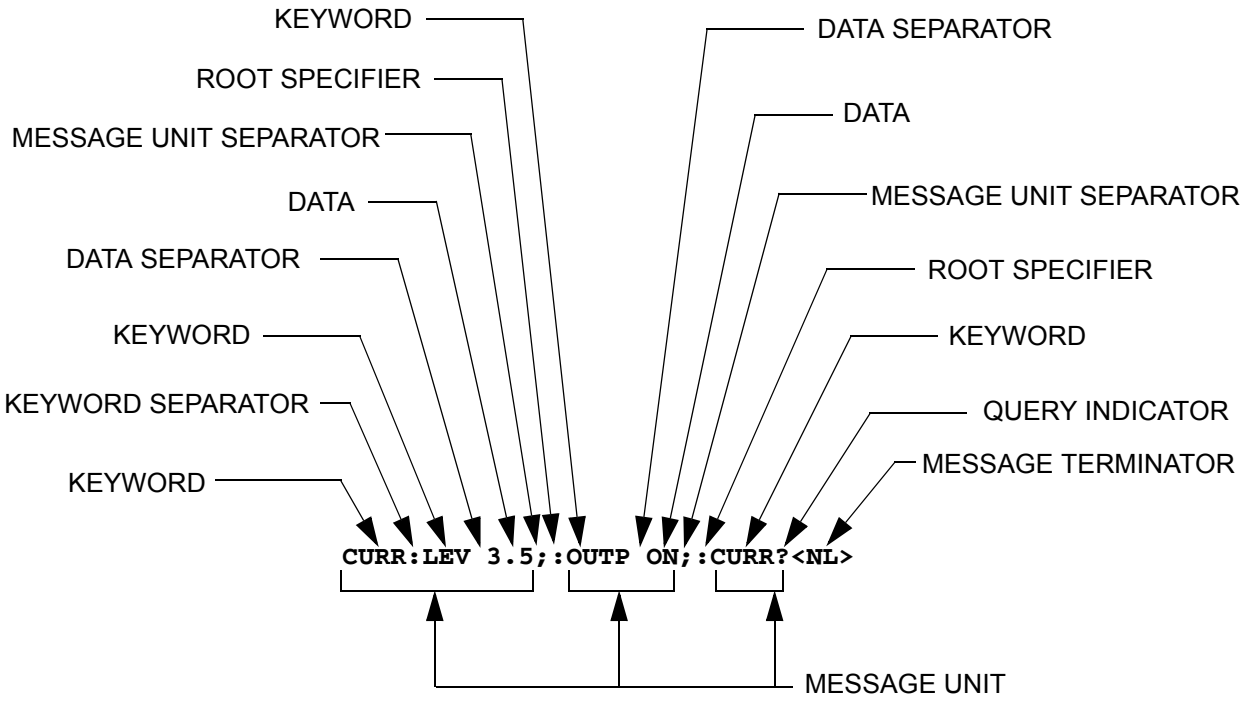


FIGURE 6-1. MESSAGE STRUCTURE

6.5.2 KEYWORD SEPARATOR

If a command has two or more keywords, adjacent keywords must be separated by a colon (:) which acts as the keyword separator (e.g., CURR:LEV:TRIG). The colon can also act as a root specifier (paragraph 6.5.7).

6.5.3 QUERY INDICATOR

The question mark (?) following a keyword is a query indicator. This changes the command into a query. If there is more than one keyword in the command, the query indicator follows the last keyword. (e.g., VOLT? and MEAS:CURR?).

6.5.4 DATA

Some commands require data to accompany the keyword either in the form of a numeric value or character string. Data always follows the last keyword of a command or query (e.g., VOLT:LEV:TRIG 14 or SOUR:VOLT? MAX

6.5.5 DATA SEPARATOR

Data must be separated from the last keyword by a space (e.g., VOLT:LEV:TRIG 14 or SOUR:VOLT? MAX

6.5.6 MESSAGE UNIT SEPARATOR

When two or more message units are combined in a program message, they must be separated by a semicolon (;) (e.g., VOLT 15;MEAS:VOLT? and CURR 12; CURR:TRIG 12.5).

6.5.7 ROOT SPECIFIER

The root specifier is a colon (:) that precedes the first keyword of a program message. This places the parser at the root (top left, Figure 6-2) of the command tree. Note the difference between using the colon as a keyword separator and a root specifier in the following examples:

VOLT:LEV:IMM 16 Both colons are keyword separators.

:CURR:LEV:IMM 4 The first colon is the root specifier, the other two are keyword separators.

VOLT:LEV 6;:CURR:LEV 15 The second colon is the root specifier, the first and third are keyword separators

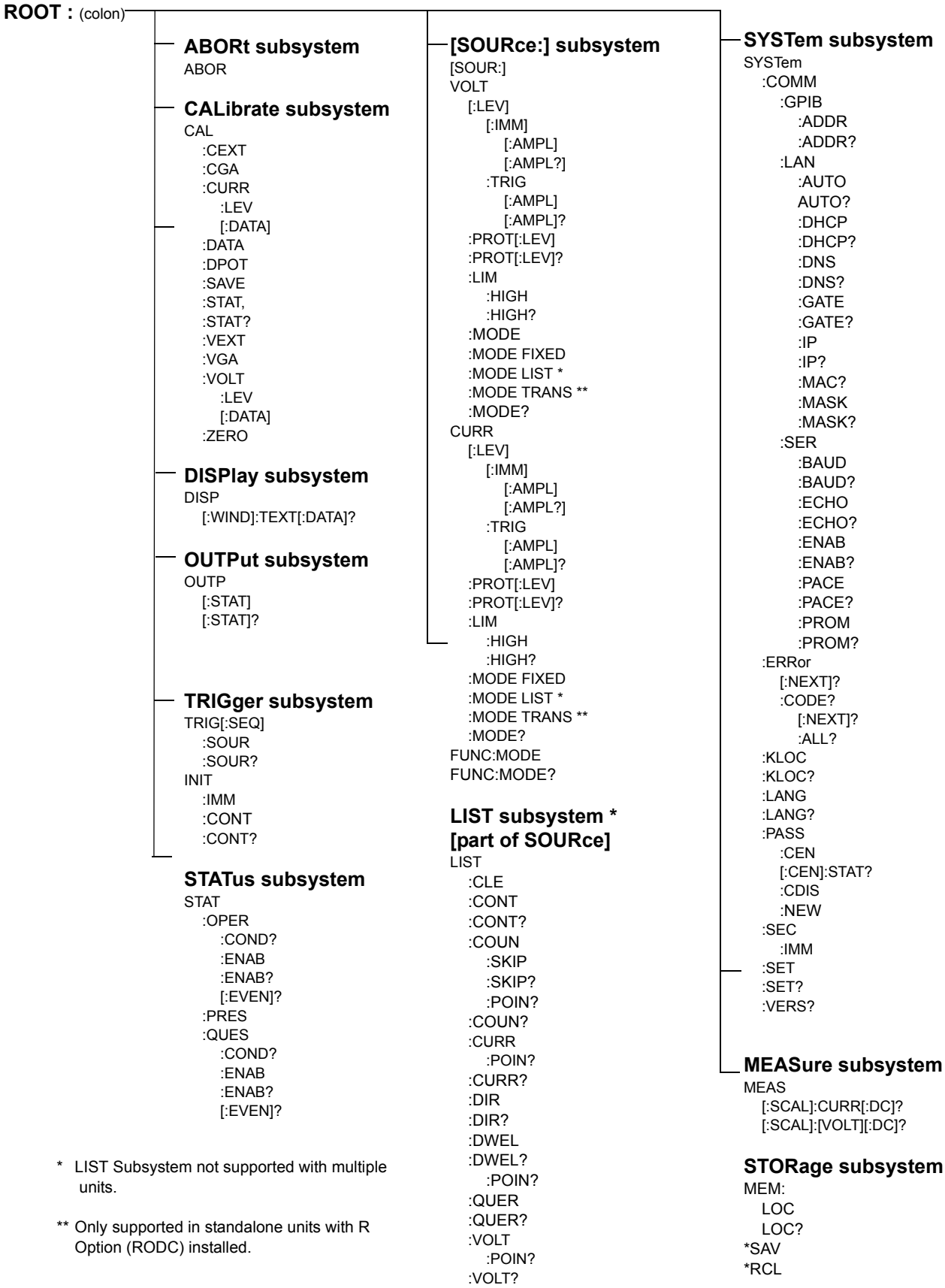
:INIT ON;:TRIG;:MEAS:CURR?;VOLT? The first three colons are root specifiers.

6.5.8 MESSAGE TERMINATOR

The message terminator defines the end of a message. One message terminator is permitted:

- new line (<NL>), ASCII 10 (decimal) or 0A (hex)

NOTE: Kepco power supplies ***require*** a message terminator at the end of each program message. The examples shown in this manual assume a message terminator will be added at the end of each message. Where a message terminator is shown it is represented as <NL> regardless of the actual terminator character.



* LIST Subsystem not supported with multiple units.

** Only supported in standalone units with R Option (RODC) installed.

FIGURE 6-2. TREE DIAGRAM OF SCPI COMMANDS USED WITH KLR POWER SUPPLY

6.6 UNDERSTANDING THE COMMAND STRUCTURE

Understanding the command structure requires an understanding of the subsystem command tree illustrated in Figure 6-2. The “root” is located at the top left corner of the diagram. The parser goes to the root if:

- a message terminator is recognized by the parser
- a root specifier is recognized by the parser

Optional keywords are enclosed in brackets [ ] for identification; optional keywords can be omitted and the power supply will respond as if they were included in the message. The root level keyword [SOURce] is an optional keyword. Starting at the root, there are various branches or paths corresponding to the subsystems. The root keywords for the KLR Power Supply are :ABORt, :CALibrate, :DISPlay, :LIST, :MEASure, :OUTPut, [:SOURce], :STATus, :STORage, :SYSTem and :TRIGger. Because the [SOURce] keyword is optional, the parser moves the path to the next level, so that VOLTage, CURRent, and FUNCtion commands are at the root level.

Each time the parser encounters a keyword separator, the parser moves to the next indented level of the tree diagram. As an example, the STATus branch is a root level branch that has three sub-branches: OPERation, PRESet, and QUEStionable. The following illustrates how SCPI code is interpreted by the parser:

STAT:PRES<NL>
The parser returns to the root due to the message terminator.

STAT:OPER?;PRES<NL>
The parser moves one level in from STAT. The next command is expected at the level defined by the colon in front of OPER?. Thus you can combine the following message units STAT:OPER? and STAT:PRES;

STAT:OPER:COND?;ENAB 16<NL>
After the OPER:COND? message unit, the parser moves in one level from OPER, allowing the abbreviated notation for STAT:OPER:ENAB.

6.7 PROGRAM MESSAGE SYNTAX SUMMARY

- Common commands begin with an asterisk (*).
- Queries end with a question mark (?).
- Program messages consist of a root keyword and, in some cases, one or more message units separated by a colon (:) followed by a message terminator. Several message units of a program message may be separated by a semicolon (;) without repeating the root keyword.
- If a program message has more than one message unit, then a colon (:) must precede the next keyword in order to set the parser back to the root (otherwise the next keyword will be taken as a subunit of the previous message unit).

  e.g., the command **`meas:volt?;curr?`** will read output voltage and output current since both **`volt?`** and **`curr?`** are interpreted as subunits of the **`meas`** command.

- Several commands may be sent as one message; a line feed terminates the message. Commands sent together are separated by a semicolon (;). The first command in a message starts at the root, therefor a colon (:) at the beginning is not mandatory.

  e.g., the command **`meas:volt?;:curr?`** will read output voltage and programmed current since the colon preceding **curr?** indicates that **`curr?`** is not part of the **`meas`** command and starts at the root.

- UPPER case letters in mnemonics are mandatory (short form). Lower case letters may either be omitted, or must be specified completely (long form)
  e.g., **INSTrument** (long form) has the same effect as **INST** (short form).

- Commands/queries may be given in upper/lower case (long form)
  e.g., **SoUrCe** is allowed.

- Text shown between brackets [ ] is optional.
  e.g., **:[SOUR:]VOLT:[LEV:]TRIG[:AMPL]?** has the same effect as **:VOLT:TRIG?**

6.7.1 EXCEPTIONS TO THE RULES

The volt:lim:high and curr:lim:high commands do not follow the SCPI rules for syntax parsing. Always send the entire string to insure the commands execute correctly. For example: **VOLT:LIM:HIGH? MIN;:volt:lim:high? max** and **VOLT:LIM:HIGH?MIN;lim:high? max** will correctly return the min and max voltage limit levels of the power supply. Sending **VOLT:LIM:HIGH? MIN;high? max** will not work and will result in an error.

6.8 PROGRAMMING EXAMPLES

Figure 6-3 is a programming example that can be used over any interface to measure voltage and current. Table 6-2 describes the VISA resource string corresponding to the Interface chosen for communication.

TABLE 6-2. VISA RESOURCE STRING CORRESPONDING TO INTERFACE

INTERFACE	VISA RESOURCE STRING	COMMENT
GPIB	GPIB::xx::INSTR	The GPIB address replaces xx.
SERIAL	ASRLy::INSTR	The com port number replaces y.
LAN-SCPI-RAW	TCIP::192.168.0.100::5025::SOCKET	This is the fastest LAN interface, similar to the serial port with automatic XON XOFF protocol support.
LAN-VXI-11	TCIP::192.168.0.100::INSTR	This LAN interface requires a more complex handshake for data and is inherently slower than a socket interface. It is similar to the GPIB interface where you tell the device when to take data and when it is acceptable to receive data.

```
/************************************************************************/
/* Sample Program For KEPCO power supply, using National Instruments */
/* any interface                          */
/************************************************************************/
#include "visa.h"// visa controls and functions
#include <stdio.h>
#include "decl.h"

ViChar rd_str[80]; // Input buffer
ViChar dat_str[80]; // Output buffer

/* select one of the following strings depending upon port to be used */

ViChar id_str{}="GPIB::06::INSTR";       // change 06 to the GPIB address of unit.

/* LAN addresses - changed the 192.168.0.100 to IP address found via the unit front panel inter-
face */

ViChar id_str()="TCIP::192.168.0.100::INSTR";      // VXI-11 instrument- via LAN port 1024
ViChar id_str()="TCIP::i92.168.0.100::5025::SOCKET";   // SCPI-RAW port (5025) LAN instrument

/* change 0 to actual communications port in use */

ViChar id_str()="ASRL0::INSTR";          // serial port

/* ViOpen uses the resource string to find the device and returns a session id to the user */

ViSession Kepco_Session;

main() {

if (viOpen(Kepco_Session,id_str,VI_NULL,VI_NULL,VI_NULL) == 0){
    strcpy(dat_str,"VOLT 5;CURR 1");     // Define a set command
    strcat(dat_str,"\r\n");              // Append delimiter - needed for Serial port and
                                         // Socket operation
    viWrt(Kepco_Session,dat_str);        // Send string to power supply

    strcpy(dat_str,"MEAS:VOLT?;CURR?");     // Define a measure command
    strcat(dat_str,"\r\n");              // Append delimiter
    ViWrt(Kepco_Session,dat_str);        // Send string to power supply
    strset(rd_str,'\0');                 // Clear input buffer
    ViRd(Kepco_Session,rd_str,64,i);     // Read result of measure
    printf("received : %s\n",rd_str);    // Print voltage and current

    viClose(Kepco_Session);              // close the instrument
}
```

FIGURE 6-3. TYPICAL EXAMPLE OF KLR POWER SUPPLY PROGRAM USING SCPI COMMANDS

SECTION 7 - IEEE 488.2 (GPIB) INTERFACE

7.1 DIGITAL REMOTE MODE PROGRAMMING USING IEEE 488.2 (GPIB) INTERFACE

KLR Power Supplies may be programmed over the IEEE 488 standard communication bus (General Purpose Interface Bus, GPIB) control bus using SCPI (Standard Commands for Programmable Instruments). SCPI provides a common language conforming to IEEE 488.2 for instruments used in an automatic test system (see Section 6). Refer to Table 7-1 for GPIB port input/output signal allocations. All power supply functions available from the front panel can be programmed via remote commands, as well as some that are not available from the front panel: Save/Recall (see PAR. 1.2.5), List (see PAR. 1.2.6) and Relay Control (see PARs. B.10 and B.11).

TABLE 7-1. IEEE 488 PORT CONNECTOR (J4) PIN ASSIGNMENTS

PIN	SIGNAL NAME	FUNCTION
1	DI01	I/O Line
2	DI02	I/O Line
3	DI03	I/O Line
4	DI04	I/O Line
5	EOI	End or Identify
6	DAV	Data Valid
7	NRFD	Not Ready for Data
8	NDAC	Not Data Accepted
9	IFC	Interface Clear
10	SRQ	Service Request
11	ATN	Attention
12	SHIELD	Shield
13	DI05	I/O Line
14	DI06	I/O Line
15	DI07	I/O Line
16	DI08	I/O Line
17	REN	Remote Enable
18	GND	Ground (signal common)
19	GND	Ground (signal common)
20	GND	Ground (signal common)
21	GND	Ground (signal common)
22	GND	Ground (signal common)
23	GND	Ground (signal common)
24	LOGIC GND	Logic Ground

7.2 IEEE 488 (GPIB) BUS PROTOCOL

Table 7-2 defines the interface capabilities of the KLR power supply (Talker/Listener) relative to the IEEE 488 (GPIB) bus (reference document *ANSI/IEEE Std 488: IEEE Standard Digital Interface for Programmable Instrumentation*) communicating with a Host Computer—Controller (Talker/Listener). Tables 7-3 and 7-4 define the messages sent to the KLR, or received by the KLR, via the IEEE 488 bus in IEEE 488 command mode and IEEE 488 data mode, respectively. These messages are enabled during the “handshake” cycle, with the KLR power supply operating as either a Talker or a Listener.

TABLE 7-2. IEEE 488 (GPIB) BUS INTERFACE FUNCTIONS

FUNCTION	SUBSET SYMBOL	COMMENTS
Source Handshake	SH1	Complete Capability (Interface can receive multiline messages)
Acceptor Handshake	AH1	Complete Capability (Interface can receive multiline messages)
Talker	T6	Basic talker, serial poll, unaddress if MLA (My Listen Address) (one-byte address)
Listener	L4	Basic listener, unaddress if MTA (My Talk Address) (one-byte address).
Service Request	SR1	Complete Capability. The interface sets the SRQ line true if there is an enabled service request condition.
Remote/Local	RL1	Complete capability. Interface selects either local or remote information. In local mode the KLR executes front panel commands, but can be set to remote mode via IEEE 488 bus. When in Remote mode all front panel keys are disabled except LOCAL. LOCAL key can be disabled using keypad lockout command (see Appendix B, PAR. B.95) so that only the controller or a power on condition can restore Local mode.
Parallel Poll	PP0	No Capability
Device Clear	DC1	Complete Capability. KLR accepts DCL (Device Clear) and SDC (Selected Device Clear).
Device Trigger	DT1	Respond to *TRG and <GET> trigger functions.
Controller	C0	No Capability

TABLE 7-3. IEEE 488 (GPIB) BUS COMMAND MODE MESSAGES

MNEMONIC	MESSAGE DESCRIPTION	COMMENTS
DCL	Device Clear	Received
GET	Group Execute Trigger	Received
GTL	Go To Local	Received
IFC	Interface Clear	Received
LLO	Local Lockout	Received
MLA	My Listen Address	Received
MTA	My Talk Address	Received
OTA	Other Talk Address	Received (Not Used)
RFD	Ready for Data	Received or Sent
SDC	Selected Device Clear	Received
SPD	Serial Poll Disable	Received
SPE	Serial Poll Enable	Received
SRQ	Service Request	Sent
UNL	Unlisten	Received
UNT	Untalk	Received

TABLE 7-4. IEEE 488 (GPIB) BUS DATA MODE MESSAGES

MNEMONIC	MESSAGE DESCRIPTION	COMMENTS
DAB	Data Byte	Received or Sent
END	End	Received or Sent
EOS	End of String	Received or Sent
RQS	Request Service	Sent
STB	Status Byte	Sent

7.2.1 CHANGING THE GPIB ADDRESS

Use the **SYST:COMM:GPIB:ADDR?** query to read the current GPIB address. Use the **SYST:COMM:GPIB:ADDR** command to change it.

SECTION 8 - RS 232C INTERFACE [STANDARD MODELS ONLY]

8.1 DIGITAL REMOTE MODE PROGRAMMING USING RS 232 [STANDARD MODELS ONLY]

KLR standard models may be programmed over the RS 232 control bus using SCPI (Standard Commands for Programmable Instruments) (see Section 6). Refer to the KLR User Manual for RS232 input/output signal allocations. All power supply functions available from the front panel can be programmed via remote commands, as well as some that are not available from the front panel: Save/Recall (see PAR. 1.2.5), List (see PAR. 1.2.6) and Relay Control (see PARs. B.10 and B.11).

8.2 RS232-C OPERATION [STANDARD MODELS ONLY]

The KLR Power Supply may be operated via an RS232-C terminal, or from a PC using a terminal emulation program. Refer to KLR User Manual for RS 232 connections and default port settings.

All RS 232 parameters may be changed using SCPI commands (see Appendix B) as follows:

- To enable RS 232, refer to PAR's B.86 and B.87
- For baud rate, refer to PAR's B.82 and B.83
- For prompt, refer to PAR's B.90 and B.91;
- For echo, refer to PAR's B.84 and B.85;
- For XON/XOFF, refer to PAR's B.88 and B.89.

8.2.1 RS 232 IMPLEMENTATION [STANDARD MODELS ONLY]

The following paragraphs are provided to help the user understand how the RS 232 serial interface is implemented in the KLR Power Supply.

The serial interface behaves like the GPIB interface in that the command is parsed after receiving the appropriate control character, in this case either a Line Feed or Carriage Return.

Upon initial application of source power to the KLR input, or after restoration of source power following an interruption, the power supply will return the identification string described in Appendix A, PAR. A.6, to the host computer, indicating that the KLR is initialized and ready to accept programming commands.

Only five control characters (characters between 00_H and $1F_H$) are acknowledged by the power supply:

- Cancel (CAN, 18_H, Keyboard command = CTRL+X)
- Carriage Return (CR, $0D_H$, Keyboard command = Enter)
- Line Feed (LF, $0A_H$, Keyboard command = Enter)
- Back Space (BS, 08_H, Keyboard command = Backspace)
- Escape (ESC, $01B_H$, Keyboard command = ESC)

BS removes the last character in the input buffer queue, with the exception of CR or LF/NL characters. Either the CR or LF (NL) acts as the line terminator, initiating parsing of the ASCII data sent to the KLR Power Supply by the command originator. When the line is parsed, the KLR sends a) the line terminator sequence CR LF to the command originator if Prompt is enabled,or b) the LF character if Echo is enabled and Prompt is off, or c) XON if both Echo and Prompt are off.

The ESC character is used for synchronization, causing the KLR Power Supply to reset (clear) its input buffer and return a CR LF sequence. The CAN character operates similarly, except that both input and output buffers of the KLR Power Supply are reset (cleared). Note that ESC and CAN characters must be followed by LF for the port to recognize these characters.

Six characters (#, $, !, @, &, %) are not permitted in RS 232 programming strings. These characters are reserved for LAN communications, and their use in other serial programming protocols will likely result in interrupted communications.

8.2.2 DATA TRANSFER PROTOCOLS [STANDARD MODELS ONLY]

RS 232 programming can be executed using either manual command/data entry, i.e. computer keyboard commands, or electronic control (e.g. Labview or LabWindows programming). The KLR Power Supply offers data transfer protocols tailored for each method.

Since the RS 232 protocol does not use a parity bit, Echo mode is the default method used to ensure reliable communications between the command originator (computer) and the KLR Power Supply. When the KLR Power Supply is in the RS 232 echo mode it returns all data received from the host computer. Echo mode is highly recommended when using the manual (keyboard) entry method.

The KLR Power Supply provides two additional options that allow handshake-style communication: the prompt method and the XON XOFF method. In the standard implementation of echo mode the controller must verify that each character sent is echoed back by the KLR. However, as can be seen in Figure 8-1, there are times when the KLR does not echo back the character sent from the controller, requiring that the controller resend the character. By using the handshake options (prompt and XON XOFF) the host computer can ensure that the serial data interrupts occurring after parsing of the incoming message do not result in lost data.

Figure 8-1 illustrates the default echo mode, the prompt method and the XON XOFF methods described in the following paragraphs.

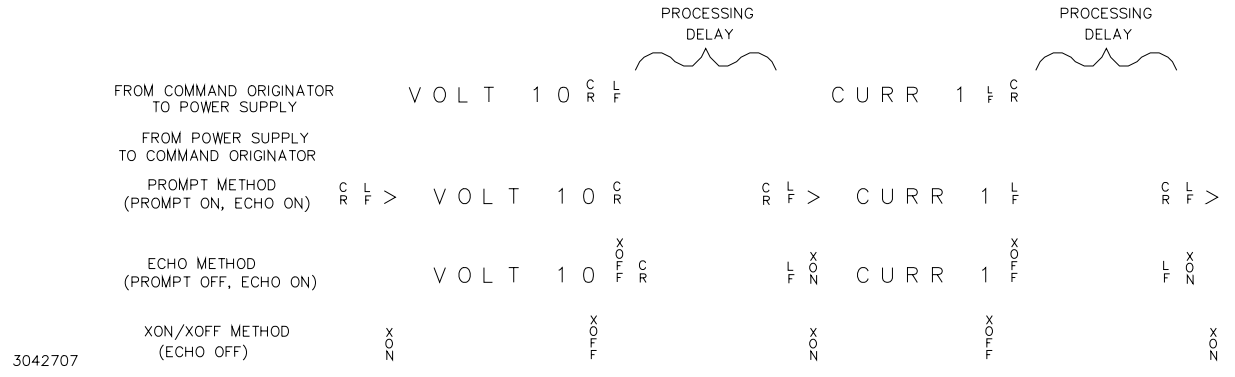


FIGURE 8-1. RS 232 IMPLEMENTATION [STANDARD MODELS ONLY]

While both echo and prompt modes are very useful when communicating with the instrument using manual (keyboard) entry, these functions will create problems if implemented with electronic control. For this reason the Kepco drivers for KLR will automatically disable these two modes as part of initial instrument communications. If the input mode is subsequently changed back to manual entry, the Echo and Prompt modes are not automatically restored; it is the user's responsibility to re-enable these features as necessary. By contrast, XON XOFF is compatible with both manual and electronic input methods, and Kepco strongly recommends that XON XOFF be enabled for all communications methods.

8.2.2.1 ECHO MODE [STANDARD MODELS ONLY]

Echo mode is the default method of ensuring data is transferred without errors. Each byte (character) is echoed back to the sender where it is verified as the same character that was just sent. If the character is incorrect or missing, the sender sends the character again until the correct character is verified as having been received.

All non-control characters are sent via the serial port of the command originator. The control character BS is echoed as BS Space BS. Only the first control character is returned in response to either a CR LF or LF CR character sequence (see Figure 8-1).

8.2.2.2 PROMPT METHOD [STANDARD MODELS ONLY]

The command originator sends a message line (command) to the KLR Power Supply and waits until the prompt sequence CR LF > is received. The KLR sends the prompt sequence CR LF > to the command originator indicating the power supply is ready to receive the next command so that data will not be lost. The prompt method is similar to the echo method described above, except that the command originator does not have to compare each character and repeat any characters dropped. The operation of the KLR is identical for echo mode and prompt mode; implementation of prompt mode is at the command originator.

8.2.2.3 XON XOFF METHOD [STANDARD MODELS ONLY]

The XON XOFF method allows the KLR Power Supply to control when the command originator is allowed to send data. The command originator can only send data after the XON (transmission on) character (011_H) has been received; the command originator stops sending data after receiving the XOFF (transmission off) character (013_H), and waits until the XON character is received before sending additional data.

Control characters, either CR or LF, are returned as XOFF CR if echo mode is on, and as XOFF if echo mode is off. XOFF stops data from the command originator and the KLR returns the normal sequence of CR LF (if echo mode is enabled).

NOTE: Kepco strongly recommends the XON XOFF method for data transfer via RS 232 protocol for all Kepco products. If this method is not selected, it is the user's responsibility to ensure completion of any response by the power supply prior to issuance of subsequent commands.

8.2.2.4 ISOLATING RS 232 COMMUNICATIONS PROBLEMS [STANDARD MODELS ONLY]

A Loop Back test can be run from the KLR front panel to aid in isolating RS 232 communications problems.

1. With the power supply in local mode, if the status display shows **SET**, tap either the **CURRENT** or **VOLTAGE** controls to take the unit out of setpoint mode (status display goes from **SET** to blank)

2. Use a thin tool (e.g., a paper clip), to press the **FUNCTION** switch repeatedly until the status display shows **UTIL**. Press the **DC OUTPUT** switch to enter the Utilities menu. Rotate either the **VOLTAGE** or **CURRENT** control until the status display shows **LBT** (Loop Back Test). At this point the VOLTS display will show **----**.

3. With the power supply's RS 232 port open (no connections), press the **DC OUTPUT** switch once to run the test. The **VOLTS** display should show FAIL; if it reads PASS, the power supply is defective and requires repair.

4. Install the loop-back test connector (P/N 195-0112) at the RS 232 port, located on the rear panel of the power supply. (If the loop back test connector is not available, install a jumper from pin 2 to pin 3 of the RS 232 port connector.) Press the **DC OUTPUT** switch once to rerun the test. The **VOLTS** display should now read **PASS**; if it reads **FAIL**, the power supply is defective and requires repair.

5. To test the integrity of the cable assembly connecting the power supply RS 232 port to the computer, remove the loop back test connector or jumper and connect the cable in its place. Install a jumper wire from pin 2 to pin 3 at the opposite end of the cable and repeat the test of (step 4) above. If the **VOLTS** display reads **FAIL**, the cable is either the improper type (not null modem) or defective. If the **VOLTS** display reads **PASS**, the cable is correct. Remove the jumper and reconnect the cable to the computer.

6. To exit the utilities menu, press the **FUNCTION** switch repeatedly until the status display is blank.

   If each of the above steps is completed successfully, the problem lies in the computer hardware and/or software. Refer to the Product Support area of the Kepco website for additional information regarding RS 232 Communications problems: www.kepcopower.com/support.

SECTION 9 - LAN INTERFACE [E-Series MODELS ONLY]

9.1 INTRODUCTION

KLR Power Supplies may be programmed via the LAN interface using the ports illustrated in Figure 2-1. These ports are described in the following paragraphs.

9.2 USING PORT 80 (WEB INTERFACE)

The web interface via port 80 supports up to eight connections to the unit. For details as to how to access, operate and configure the unit using the web interface, refer to the KLR User Manual.

9.3 USING PORT 5024 (TELNET)

To access the unit via Telnet port 5024 the proper command line is TELENET IP ADDRESS PORT. e.g.:

```
TELNET 192.168.0.100 5024
```

NOTE: Port 5025 (SCPI RAW) can also be accessed with the Telnet utility, but the data sent is not echoed back to the user and there is no prompt string.

The SCPI-TELNET port transfers ASCII SCPI data using the Control M or Control J characters as line terminators. Table 9-1 lists the Control characters applicable to this port.

TABLE 9-1. TELNET PORT 5024 AND SCPI RAW PORT 5025 CONTROL CHARACTERS

HEX Name	Key	Function
1SOH	Ctrl A	Terminate connection
2STX	Ctrl B	Execute Trigger, respond with <TRIGGER>
3ETX	Ctrl C	Execute Device clear to unit respond with <DEVICE CLEAR>
	Ctrl J or Ctrl M	Line Terminator
5 ENQ	Ctrl E	Unit will respond with the user description of the device.
12DC2	Ctrl R	Lock port
14DC4	Ctrl T	Unlock Port

On receipt of Lock Port, unit verifies other ports (5025 and 1024) are unused and grants the requesting port exclusive access to control KLR. If port lock is unsuccessful, unit responds with <DENIED LOCK>. If successful unit responds with <GRANTED LOCK>.

9.4 USING PORT 5025 (SCPI-RAW)

The SCPI-Raw port provides faster access than the VXI-11 port (1024) as it has little overhead. This port is accessed using the VISA resource string, e.g.:

```
TCIPO::192.168.1.100::5025:SOCKET
```

This port can also be accessed via the telnet utility but the data sent is not echoed back to the user and there is no prompt string.

The SCPI-RAW port transfers ASCII SCPI data using the Control M or Control J characters as line terminators. Table 9-1 lists the Control characters applicable to this port

On receipt of Lock Port, unit verifies other ports (5024 and 1024) are unused and grants the requesting port exclusive access to control KLR. If port lock is unsuccessful, unit responds with <DENIED LOCK>. If successful unit responds with <GRANTED LOCK>.

9.5 USING PORT 1024 (VXI-11)

VXI-11 uses one standard port (1024) and two assigned by the instrument when connections are opened. The 1024 port is open at all times to accept connection requests. The maximum number of open connections is two. The VXI-11 port requires the use of a lock which prevents other VXI-11 connections from gaining access to the instrument. This port is accessed using the VISA resource string ending in INST, e.g.:

```
TCIP0::192.168.1.100::INST
```

On receipt of Lock Port, unit verifies other ports (5024 and 5025) are unused and grants the requesting port exclusive access to control KLR. If port lock is unsuccessful, unit responds with <DENIED LOCK> If successful unit responds with <GRANTED LOCK>.

9.6 USING PORT 5044 (*TRG COMMAND)

This class C interface supports both multicast trigger events and TCP/IP trigger events. Port 5044 does not support arbitrary names; it supports only domain 1 with a name of LAN0. If this is sent to the KLR when it is armed and waiting for trigger, the unit will trigger (change the output) within 2.5 milliseconds of the receipt of this request.

9.7 SUNRPC PORT 111

The SUNRPC port is used for discovery when sent as a UDP broadcast message. The only command supported is the GETPORT. The ports that can be requested are detailed in the VXI specification and repeated in the LXI specifications.

The SUNRPC port can be also used with the TCP/IP protocol. Again, the only command supported is the GETPORT.

APPENDIX A - IEEE 488.2 COMMAND/QUERY DEFINITIONS

A.1 INTRODUCTION

This appendix defines the IEEE 488.2 commands and queries used with the KLR Power Supply These commands and queries are preceded by an asterisk (*) and are defined and explained in PAR. A.2 through A.17, arranged in alphabetical order. Table A-1 provides a quick reference of all IEEE 488.2 commands and queries supported in the KLR Power Supply.

TABLE A-1. IEEE 488.2 COMMAND/QUERY INDEX

COMMAND	PAR.	COMMAND	PAR.
*CLS	A.2	*RST	A.13
*ESE, ?	A.3, A.4	*SAV	A.13
*ESR?	A.5	*SRE, ?	A.13, A.14
*IDN?	A.6	*STB?	A.15
*OPC, ?	A.7, A.8	*TRG	A.16
*OPT?	A.13	*TST?	A.17
*RCL	A.13	*WAI	A.18

A.2 *CLS — CLEAR STATUS COMMAND *CLS

Syntax: *CLS

Description: **Clears status data.** Clears the following registers without affecting the corresponding Enable Registers: Standard Event Status Register (ESR), Operation Status Event Register, Questionable Status Event Register, and Status Byte Register (STB). Also clears the Error Queue. Related commands: *OPC, *OPC?. (See example, Figure A-1.)

A.3 *ESE — STANDARD EVENT STATUS ENABLE COMMAND *ESE

Syntax: *ESE <integer> where <integer> = positive whole number: 0 to 255 per Table A-2.
Default Value: 0

Description: **This command programs the standard Event Status Enable register bits.** The contents function as a mask to determine which events of the Event Status Register (ESR) are allowed to set the ESB (Event Summary Bit) of the Status Byte Register. Enables the Standard events to be summarized in the Status Byte register (1 = set = enable function, 0 = reset = disable function). All of the enabled events of the standard Event Status Enable register are logically ORed to cause ESB (bit 5) of the Status Byte Register to be set (1 = set = enable, 0 = reset = disable). (See example, Figure A-1.)

TABLE A-2. STANDARD EVENT STATUS ENABLE REGISTER AND STANDARD EVENT STATUS REGISTER BITS

CONDITION	PON	NU	CME	EXE	DDE	QUE	NU	OPC
BIT	7	6	5	4	3	2	1	0
VALUE	128	64	32	16	8	4	2	1

PON	Power On
NU	(Not Used)
CME	Command Error
EXE	Execution Error
DDE	Device Dependent Error
QUE	Query Error
OPC	Operation Complete

A.4 *ESE? — STANDARD EVENT STATUS ENABLE QUERY *ESE?

Syntax**:** *ESE? Return value: Integer> value per Table A-2.

Description: **Returns the mask stored in the Standard Event Status Enable Register.** Contents of Standard Event Status Enable register (*ESE) determine which bits of Standard Event Status register (*ESR) are enabled, allowing them to be summarized in the Status Byte register (*STB). All of the enabled events of the Standard Event Status Enable Register are logically ORed to cause ESB (bit 5) of the Status Byte Register to be set (1 = set = enable function, 0 = reset = disable function). (See example, Figure A-1.)

A.5 *ESR? — EVENT STATUS REGISTER QUERY *ESR?

Syntax**:** *ESR?
Return value: <integer> (Value = contents of Event Status register as defined in Table A-2.)

Description: **Causes the power supply to return the contents of the Standard Event Status register. After it has been read, the register is cleared.** The Standard Event Status register bit configuration is defined in Table A-2 (1 = set, 0 = reset). The error bits listed in Table A-2 are also related to error codes produced during parsing of messages and to errors in the power supply (see PAR. B.92 and Table B-5)

- Any 1xx type error sets the Command error bit (5).
- Any 2xx type error sets the Execution error bit (4).
- Any 3xx type error sets the Device error bit (3). The Device error bit will be set when Current Error or Voltage Error is detected and the corresponding Status Questionable Enable bit is set.
- Any 4xx type error sets the Query error bit (2).

Related Commands: *CLS, *ESE, *OPC. (See example, Figure A-1.)

A.6 *IDN? — IDENTIFICATION QUERY *IDN?

Syntax**:** *IDN?
Return value: Character string

Description: **Identifies the instrument.** This query requests identification. The power supply returns a string which contains the manufacturer name, model, option(s) installed, serial number and firmware version.

The character string for standard models contains the following fields:
<Manufacturer>,<Model>-1200,<Last Calibration Date>,<Serial Number>,<Firmware revision>

The character string for two standard models in series contains the following fields:
<Manufacturer>,<Model _S> Series <Manufacturing Date master unit>,<Serial Number of master unit>,<Firmware revision>

The character string for two standard models in parallel contains the following fields:
<Manufacturer>,<Model_P> Parallel <Manufacturing Date master unit>,<Serial Number of master unit>,<Firmware revision>

The character string for standard models with R option (RODC) contains the following fields:
<Manufacturer>,<Model> RODC,<Last Calibration Date>,<Serial Number>,<Firmware revision>

The character string for E-Series models contains the following fields:
<Manufacturer>,<Model> LAN,<Last Calibration Date>,<Serial Number>,<Firmware revision>-<LAN Firmware revision>

The character string for E-Series models with RODC contains the following fields:
<Manufacturer>,<Model> LAN RODC,<Last Calibration Date>,<Serial Number>,<Firmware revision>-<LAN Firmware revision>

where:

- <Manufacturer> = KEPCO
- <Model> = KLR V-C (V is E_{oMAX}, C is I_{oMAX}
  <Model_S> = KLR (2V-C (2V is E_{oMAX} x 2, C is I_{oMAX}
  <Model_P> = KLR V-2C (V is E_{oMAX}, 2C is I_{oMAX}x 2
- LAN = E-Series model (ethernet-enabled with LAN port conforming to LXI protocols
- RODC = Rapid Output Discharge Circuit option installed
- <Manufacturing Data> = DDMMYYYY-Axxxxxx (DD=day, MM=month, and YYYY=year of manufacture, Axxxxxx=unit serial number)
- <Firmware revision> = Vn.m (original factory firmware) or vn.m (updated or secondary firmware) (n.m revision, e.g, V1.01 or v4.00)
- <LAN Firmware revision> [E-Series models only] = Vn.m (original factory LAN firmware) or vn.m (updated or secondary LAN firmware) (n.m revision, e.g, V1.01 or v4.00)

(See example, Figure A-1.)

A.7 *OPC — OPERATION COMPLETE COMMAND *OPC

Syntax**:** *OPC

Description: **Causes power supply to set status bit 0 (Operation Complete) when pending operations are complete.** This command sets Standard Event Status Register bit 0 (see Table A-2) to “1” when all previous commands have been executed and changes in output level have been completed. This command does not prevent processing of subsequent commands, but bit 0 will not be set until all pending operations are completed. (1 = set = enable function, 0 = reset = disable function). (See example, Figure A-1.) As an example, the controller sends command(s), then sends *OPC. If controller then sends *ESR?, the power supply responds with either a “0” (if the power supply is busy executing the programmed commands), or a “1” (if the previously programmed commands are complete). (See example, Figure A-1.)

A.8 *OPC? — OPERATION COMPLETE QUERY *OPC?

Syntax**:** *OPC?
Return value: <1> (ASCII) placed in output queue when power supply has completed operation.

Description: **Indicates when pending operations have been completed.**When all pending operations are complete (all previous commands have been executed and changes in output level have been completed) a “1” is placed in the Output Queue. Subsequent commands are inhibited until the pending operations are completed. *OPC? is intended to be used at the end of a command line so that the application program can monitor the bus for data until it receives the “1” from the power supply Output Queue. (See example, Figure A-1.)

A.9 *OPT? — OPERATION COMPLETE QUERY

*OPT?

Syntax**:** *OPT?
Return value: option string which is a comma separated list
250 LST,.01 TMIN,655.36 TMAX,3 RL,40 MEM,4 DSM,CCAL
where:
xxx LST indicates maximum number of points supported by list
yyy TMIN indicates minimum list dwell time in seconds
zzzzzz TMAX indicates maximum list dwell time in seconds
3 RL indicates keyboard lockout supported
aa MEM indicates number (aa) of save and recall memory locations
b DSM indicates number (b) of characters in front panel display
CCAL indicates calibration can be performed without manually adjusted pots.

Description: **Lists configuration parameters in format compatible with LabView and C routines.** Configuration parameters include List settings (number of points, minimum dwell time, maximum dwell time), whether keyboard lockout is supported, number of save and recall locations, and whether calibration is performed without mechanical pots. For example 250 LST,.01 TMIN,655.36 TMAX,3 RL,40 MEM,4 DSM,CCAL means the unit supports up to 250 points in a list, minimum list dwell time is 0.01 seconds, maximum list dwell time is 655.36 seconds, keyboard lockout is supported, 40 memory locations available for Save and Recall, the front panel has a four-character display, and calibration does not require manually adjusted pots.

A.10 *RCL — RECALL COMMAND

*RCL

Syntax**:** *RCL<integer> (1 to 40)

Description: **Restores power supply to previously defined levels of output voltage, output current, overvoltage protection, overcurrent protection, output state (on or off) and relay state.** Executing *RCL recalls the settings previously saved by *SAV from one of 40 memory locations, changing the voltage, current, overvoltage protection, overcurrent protection, and output on/off state accordingly. If RELAY mode was set to MANUAL when *SAV was executed, relay state is also restored; relay state is not saved upon power off. The following parameters are affected by *RCL: VOLT, CURR, VOLT:PROT, CURR:PROT, and OUTP.

A.11 *RST — RESET COMMAND

*RST

Syntax**:** *RST

Description: **Resets power supply to the power on default state.** The power supply is programmed to the power on values of the following parameters: CURR[:LEV][:IMM] = MIN (minimum), VOLT[:LEV][:IMM] = 0, OUTP[:STAT] = 0 (OFF). In addition, *RST sets INIT:CONT to OFF, sets VOLT:MODE to FIXED, stops all LIST or TRAN operations. (See example, Figure A-1.)

A.12 *SAV — SAVE COMMAND

*SAV

Syntax**:** *SAV<integer> (1 to 40)

Description: **Saves the present state of output voltage, output current, overvoltage protection, overcurrent protection, and output state on/off and (if RELAY mode is set to MANUAL) relay state to the specified memory location.** This command stores the present state of the power supply to one of 40 memory locations in non-volatile Memory. If RELAY mode is set to MANUAL from the front panel UTIL menu, relay state, on or off, is also saved. The following parameters are stored by *SAV: VOLT, CURR, OUTP, and VOLT:PROT, CURR:PROT. The stored values can be restored by the *RCL command.NOTE: If a Limit Model setting is changed, previously stored settings may be outside the range established by the new limit model. In this case, when *RCL is executed, the value is cleared to the default minimum (0V, minimum A) and an “out of range” error message is generated.

NOTE: The following example assumes KLR 75-32 with limit model set for 36V, 32A and output operating in voltage stabilization mode.

*CLS	Clears data from all status registers.
*ESE 60	Enables bits 5 (command error), 4 (execution error), 3 (device dependent error) and 2 (query error) (see Table A-1) to set the event status summary bit when an STB command is executed.
*ESE?	Returns 60 (value of mask) verifying that bits 5,4,3 and 2 are enabled.
*ES	Invalid command; this will set error bit 5 (command error).
*ESR?	Returns 32 (bit 5 set), indicating command error has occurred since the last time the register was read.
*IDN?	Returns: “KEPCO, KLR 75-32, mm-dd-yyyy, Axxxxxx, Vx.xx" where mm-dd-yyyy indicates date of last calibration, Axxxxxx indicates unit serial number and Vx.xx indicates firmware revision.
*OPC	Directs status bit 0 to be set once pending operations are completed.
OUTP ON	Enables output. NOTE: *RST and power-up condition is output off.
VOLT 35;CURR 30	Sets output voltage to 35V, current limit to 30A.
*ESR?	Returns 129, indicating 128 (PON, bit 7 = 1) + 1 (OPC, bit 1 = 1).
*ESR?	Returns 0 (event status register cleared by prior reading).
VOLT 41.5;CURR 21.5	Sets output voltage to 41.5V, current limit to 21.5A.
*OPC	Returns 1 once command operations are completed.
*RST	Resets power supply to output off (zero volts, minimum current).
OUTP ON	Enables output.
*SRE 40	Enables bits 5 (event status byte summary) and 3 (questionable status summary) (see Table A-3) to set the request for service bit.
*SRE?	Returns 40 (value of mask) indicating bits 5 and 3 are enabled.
*STB?	Returns the value of the status byte register, including bit 6 (master status summary), without clearing the register; for example, a response of 96 would indicate 64 (MSS, bit 6 =1) + 32 (ESB, bit 5 = 1) have been set. A return of 00 indicates no bits have been set.
VOLT 25	Sets output voltage to 25V; since output current has not been set, current limit defaults to the model minimum of 0.4A.
*TST?	Power supply executes self test of digital controls and responds with 0 if test is completed successfully, 1 if test fails.

FIGURE A-1. GPIB COMMANDS

A.13 *SRE — SERVICE REQUEST ENABLE COMMAND

***SRE**

Syntax**:** *SRE <integer> where <integer> = value from 0 - 255 per Table A-3, except bit 6 cannot be programmed.

Description: **Sets the condition of the Service Request Enable register.** The Service Request Enable register determines which events of the Status Byte Register are summed into the MSS (Master Status Summary) and RQS (Request for Service) bits. RQS is the service request bit that is cleared by a serial poll, while MSS is not cleared when read. A “1” (1 = set = enable, 0 = reset = disable) in any Service Request Enable register bit position enables the corresponding Status Byte bit to set the RQS and MSS bits. All the enabled Service Request Enable register bits then are logically ORed to cause Bit 6 of the Status Byte Register (MSS/RQS) to be set. Related Commands: *SRE?, *STB?. (See example, Figure A-1.)

TABLE A-3. SERVICE REQUEST ENABLE AND STATUS BYTE REGISTER BITS

CONDITION	OPER	MSS RQS	ESB	MAV	QUES	ERR QUE	LIST RUN	NU
BIT	7	6	5	4	3	2	1	0
VALUE	128	64	32	16	8	4	2	1

OPER	Operation Status Summary
MSS	Master Status Summary
RQS	Request for Service
ESB	Event Status Byte summary
MAV	Message available
QUES	QUEStionable Status Summary
ERR QUE	1 or more errors occurred (see PAR. B.92)
LIST RUN	1 when list is running.
NU	(Not Used)

A.14 *SRE? — SERVICE REQUEST ENABLE QUERY *SRE?

Syntax**:** *SRE? Response: <integer> = value from 0 - 255 per Table A-3.

Description: **Reads the Service Enable Register.** Used to determine which events of the Status Byte Register are programmed to cause the power supply to generate a service request (1 = set = function enabled, 0 = reset = function disabled). Related Commands: *SRE, *STB? (See example, Figure A-1.)

A.15 *STB? — STATUS BYTE REGISTER QUERY *STB?

Syntax**:** *STB? Response: <integer> value from 0 to 255 per Table A-3.

Description: **Reads Status Byte Register without clearing it.** This Query reads the Status Byte Register (bit 6 = MSS) without clearing it (1 = set = function enabled, 0 = reset = function disabled). The register is cleared only when subsequent action clears all set bits. MSS is set when the power supply has one or more reasons for requesting service. (A serial poll also reads the Status Byte Register, except that bit 6 = RQS, not MSS; and RQS will be reset.) Related Commands: *SRE, *SRE?. (See example, Figure A-1.)

A.16 *TRG — TRIGGER COMMAND *TRG

Syntax**:** *TRG

Description: **Triggers the power supply to be commanded to preprogrammed values of output current and voltage.** When the trigger is armed (checked by examining WTG bit in Status Operational Condition register) and TRIG:SOUR BUS is sent, *TRG generates a trigger signal. The trigger will change the output of the power supply to the output voltage and current levels specified by VOLT:TRIG and CURR:TRIG commands and clear the WTG bit in the Status Operation Condition register. If INIT:CONT ON has been issued, the trigger subsystem is immediately rearmed for subsequent triggers, and the WTG bit is again set to 1. *TRG or GET are both addressed commands (only devices selected as listeners will execute the command). Related Commands: ABOR, INIT, TRIG, CURR:TRIG, VOLT:TRIG. (See example, Figure B-1.)

A.17 *TST? — SELF TEST QUERY *TST?

Syntax**:** *TST? Returned value: 0 or 1 (0 = pass test, 1 = fail test)

Description: **Power Supply test.**This query causes the power supply to do a self test and provide the controller with pass/fail results. A cyclic redundancy check (CRC) is performed on non-volatile RAM. A “1” is returned if there is an error.

A.18 *WAI — WAIT COMMAND *WAI

Syntax**:** *WAI

Description: **Causes the power supply to wait until all previously issued commands and queries are complete before executing subsequent commands or queries.** This command can be used to guarantee sequential execution of commands and queries. When all pending operations are complete (all previous commands have been executed, changes in output level have been completed), the WAI command is completed and execution of subsequent commands can continue. NOTE: *WAI is a no-operation for KLR.

APPENDIX B - SCPI COMMAND/QUERY DEFINITIONS

B.1 INTRODUCTION

This appendix defines the SCPI subsystem commands and queries used with the KLR Power Supply. Subsystem commands are defined in PAR. B.3 through B.106, arranged Alphabetically in groups as they appear in the tree diagram, Figure 6-2. Table B-1 provides a quick reference of all SCPI subsystem commands and queries used in the KLR power supply.

TABLE B-1. SCPI SUBSYSTEM COMMAND/QUERY INDEX

COMMAND	PAR.	COMMAND	PAR.
ABOR	B.3	STAT:OPER:COND?	B.58
CAL	B.4	STAT:OPER:ENAB, ?	B.59, B.60
DISP:TEXT?	B.5	STAT:OPER[:EVEN]?	B.61
INIT[:IMM]	B.6	STAT:PRES	B.62
INIT:CONT, ?	B.7, B.8	STAT:QUES[:EVEN]?	B.63
[SOUR:]LIST:CLE	B.9	STAT:QUES:COND?	B.64
[SOUR:]LIST:CONT, ?	B.10, B.11	STAT:QUES:ENAB, ?	B.65, B.66
[SOUR:]LIST:CONT:POIN?	B.12	SYST:COMM:GPIB:ADDR, ?	B.67, B.68
[SOUR:]LIST:COUN, ?	B.13, B.14	SYST:COMM:LAN:AUTO, ?	B.69, B.70
[SOUR:]LIST:COUN:SKIP, ?	B.15, B.16	SYST:COMM:LAN:DHCP, ?	B.71, B.72
[SOUR:]LIST:CURR, ?	B.17, B.18	SYST:COMM:LAN:DNS, ?	B.73, B.74
[SOUR:]LIST:CURR:POIN?	B.19	SYST:COMM:LAN:GATE, ?	B.75, B.76
[SOUR:]LIST:DIR, ?	B.20, B.21	SYST:COMM:LAN:IP, ?	B.77, B.78
[SOUR:]LIST:DWEL, ?	B.22, B.23	SYST:COMM:LAN:MAC?	B.79
[SOUR:]LIST:DWEL:POIN?	B.24	SYST:COMM:LAN:MASK, ?	B.80, B.81
[SOUR:]LIST:QUER, ?	B.25, B.26	SYST:COMM:SER:BAUD, ?	B.82, B.83
[SOUR:]LIST:VOLT, ?	B.27, B.28	SYST:COMM:SER:ECHO, ?	B.84, B.85
[SOUR:]LIST:VOLT:POIN?	B.29	SYST:COMM:SER:ENAB, ?	B.86, B.87
MEAS:CURR?	B.30	SYST:COMM:SER:PACE, ?	B.88, B.89
MEAS:VOLT?	B.31	SYST:COMM:SER:PROM, ?	B.90, B.91
MEM:LOC, ?	B.32, B.33	SYST:ERR?	B.92
OUTP[:STAT], ?	B.34, B.35	SYST:ERR:CODE?	B.93
[SOUR:]CURR[:LEV][:IMM][:AMPL], ?	B.36, B.37	SYST:ERR:CODE:ALL?	B.94
[SOUR:]CURR:LIM:HIGH, ?	B.38, B.39	SYST:KLOCK, ?	B.95, B.96
[SOUR:]CURR:MODE, ?	B.40, B.41	SYST:LANG, ?	B.97, B.98
[SOUR:]CURR:PROT[:LEV], ?	B.42, B.43	SYST:PASS:CEN	B.99
[SOUR:]CURR[:LEV]:TRIG[:AMPL], ?	B.44, B.45	SYST:PASS:CDIS	B.100
[SOUR:]FUNC:MODE, ?	B.46, B.47	SYST:PASS:NEW	B.101
[SOUR:]VOLT[:LEV][:IMM][:AMPL], ?	B.48, B.49	SYST:PASS:STAT?	B.102
[SOUR:]VOLT:LIM:HIGH, ?	B.50, B.51	SYST:SEC	B.103
[SOUR:]VOLT:MODE, ?	B.52, B.53	SYST:SET, ?	B.104, B.105
[SOUR:]VOLT:PROT:LEV, ?	B.54, B.55	SYST:VERS?	B.106
[SOUR:]VOLT[:LEV]:TRIG[:AMPL]?	B.56, B.57	TRIG:SOUR, ?	B.107, B.108

B.2 NUMERICAL VALUES

The SCPI data parser on the KLR supports a maximum of 8 digits after the decimal point and a maximum integer of 4 x 10^8. Any values greater than these are not processed by the device and no error is generated. The largest string that can be received or transmitted by the KLR is 253 characters.

All numerical data is returned in scientific notation, digits with decimal point and Exponent, e.g., 2.71E1 for 27.1 after calibration constants have been applied. Thus. for example, VOLT 14;VOLT? may return 1.39997E1 which indicates that the unit has been calibrated to provide 13.9997V for a programmed value of 14V, within the calculation accuracy of the KLR. Error "-120" results from syntactical errors, e.g., the exponent exceeds 8, a letter is identified, etc. Error "-222" is produced if the value exceeds the range of acceptable values for the parameter.

B.3 ABORt COMMAND — ABOR

Syntax: Short Form: ABOR Long Form: ABORt

Description: **Cancels previously armed trigger, resets WTG.** if the trigger system is armed (INIT:CONT set to OFF and INIT:IMM sent), sending ABORt disarms the trigger system so subsequent trigger commands have no effect. If INIT:CONT ON has been programmed, this command has no effect. Related Commands: INIT, *TRG. (See example, Figure B-1.)

B.4 CALibrate COMMANDS AND QUERIES — CAL

CAL commands and queries are used to perform calibration of the unit via the control interface. These commands must be issued in a specific sequence in order to properly calibrate the unit. To use these commands, refer to Kepco's website (www.kepcopower.com/drivers) and download the LabWindows/CVI Version 5 driver for KLR. This file provides remote calibration capability and uses the following supported commands and queries:

CAL:CEXT command
CAL:CGA command
CAL:CURR:LEV command
CAL:CURR[:DATA] command
CAL:DATA command
CAL:DPOT command
CAL:SAVE command
CAL:STAT command and query
CAL:VEXT command
CAL:VGA command
CAL:VOLT:LEV command
CAL:VOLT[:DATA] command
CAL:ZERO command

B.5 DISPlay[:WINDow]:TEXT[:DATA]? QUERY — DISP:TEXT?

Syntax: Short Form: DISP[:WIND]:TEXT[:DATA]? Long Form: DISPlay[:WINDow]:TEXT[:DATA]?
Return Value: Character string displayed on front panel Status display.

Description: **Returns the text displayed on front panel Status display.** Returns the character string displayed on Status display.

B.6 INITiate[:IMMediate] COMMAND — INIT[:IMM]

Syntax: Short Form: INIT[:IMM] Long Form: INITiate[:IMMediate]

Description: **Arms a single trigger event.** If INIT:CONT is OFF (disabled), then INIT[:IMM] arms the trigger system for a single trigger event and sets the WTG bit in the Questionable Status register to 1. If INIT:CONT is ON (enabled), then the trigger system is continuously armed. Sending INIT[:IMM] after enabling INT:CONT is redundant and causes the power supply to return an E213 "INIT ignored" error message. Sending an active trigger command (*TRG) completes the sequence, causing the power supply output to be programmed to the previously stored values of VOLT:TRIG and CURR:TRIG. After a trigger command has been received, subsequent trigger commands have no effect unless preceded by INIT[:IMM] or INIT:CONT ON. Related Commands: <GET>, *RST, *TRG (See example, Figure B-1).

NOTE: If the selected trigger source is external (TRIG:SOUR EXT) and INIT:CONT is enabled, the first trigger command must be preceded by INIT[:IMM]; subsequent trigger commands will be processed normally.

NOTE: The following example assumes KLR 75-32 with limit model set for 75V, 16A and output operating in voltage stabilization mode. See Figure B-3 for directions on programming limit model settings.

OUTP ON	Enables output.
OUTP?	Returns 1 (output enabled).
STAT:PRES	Disables all status event reporting.
STAT:OPER:ENAB 32	Enables only bit 5 (WTG, Waiting for Trigger)
STAT:OPER:ENAB?	Returns 32, WTG bit enabled.
VOLT 75;CURR 5E-1	Programs output voltage to 75V, current limit to 0.5A.
INIT:CONT ON	Enables continuous triggering.
INIT:CONT?	Returns 1 (continuous triggering enabled).
STAT:QUES?	Returns 32, indicating WTG bit is set (trigger armed).
VOLT:TRIG 31.5;CURR:TRIG8E-1	Programs output voltage for 31.5V, current limit for 0.8A upon receipt of trigger stimulus.
*TRG	Output voltage is set to 31.5V with 0.8A current limit.
STAT:QUES?	Returns 32 again, indicating WTG bit was set again (continuous triggering).
VOLT 32.1;CURR 5	Programs output voltage to 32.1V, current limit to 5A.
MEAS:VOLT?	Returns actual output voltage (if output voltage is 32.15V, power supply returns 3.215E1).
MEAS:CURR?	Returns actual output current.
FUNC:MODE?	Returns operating mode of power supply followed by programmed operating mode, e.g., VOLT,VOLT.
CURR:TRIG?	Returns 5E0 (current value established by CURR:TRIG).
VOLT:TRIG?	Returns 3.15E1 (voltage value established by VOLT:TRIG).
ABOR	Changes pending trigger levels to immediate values (32.1V, 5A).
VOLT 37.7;CURR 2.5	Programs output voltage to 37.7V, current limit to 2.5A.
*TRG	Output returns to trigger values established by ABOR (32.1V, 5A).
INIT:CONT 0	Disables continuous triggering.
INIT:CONT?	Returns 0 (continuous triggering disabled).
INIT:IMM	Arms single trigger event.
STAT:QUES?	Returns 32 indicating WTG bit is set (trigger armed).
*TRG	Output returns to trigger values established by ABOR (32.1V, 5A).
STAT:QUES?	Returns 0, indicating trigger is not rearmed (single trigger event).
VOLT 37.7, CURR 2.5	Programs output voltage to 37.7V, current limit to 2.5A.
*TRG	Output remains in current state as init:cont is set to 1.
INIT	Trigger is armed for a single event
*TRG	Output returns to previous trigger values (32.1V, 5A) without rearming trigger.
OUTP OFF	Output disabled.
OUTP?	Returns 0 (output disabled).
MEAS:VOLT?	Returns 0 (measured output voltage).
VOLT?	Returns 3.21E1 (last programmed output voltage).
CURR?	Returns 5E0 (last programmed output current).
CURR? MAX	Returns 1.6E1 (maximum allowable current for present limit model).
CURR? MIN	Returns 4E-1 (minimum allowable programming current for model).

FIGURE B-1. PROGRAMMING THE OUTPUT

B.7 INITiate:CONTinuous COMMAND

INIT:CONT

Syntax: Short Form: INIT:CONT {ON | OFF} or {1 | 0} (1 = on, 0 = off)
Long Form: INITiate:CONTinuous {ON | OFF} or {1 | 0} (1 = on, 0 = off)
Default setting is OFF.

Description: **INIT:CONT ON enables continuous triggers.; INIT:CONT OFF disables continuous triggers.** If INIT:CONT is OFF, then INIT[:IMM] arms the trigger system for a single trigger. If INIT:CONT is ON, then the trigger system is continuously armed. INIT:CONT ON sets the WTG bit in the Questionable Status register to 1 after each trigger event. Executing *RST command sets INIT:CONT to OFF. (See example, Figure B-1.)

B.8 INITiate:CONTinuous QUERY — INIT:CONT?

Syntax**:** Short Form: INIT:CONT? Long Form: INITiate:CONTinuous?
Return Value: 1 or 0 (1 = on, 0 = off)

Description: **Determines whether continuous triggers are enabled or disabled.** Power supply returns value of INIT:CONT flag: “1” = continuous triggers are enabled (INIT:CONT ON); “0” = continuous triggers disabled (INIT:CONT OFF). (See example, Figure B-1.)

B.9 [SOURce:]LIST:CLEar COMMAND — LIST:CLE

Syntax: Short Form: LIST:CLE Long Form: LIST:CLEar>

Description: **Clears all list entries by setting all pointers to 0.** Also sets LIST:DIR to UP. Related Commands: All LIST commands (See example, Figure B-2.)

B.10 [SOURce:]LIST:CONTrol COMMAND — LIST:CONT

Syntax: Short Form: LIST:CONT {1 | 0} (1 = on, 0 = off)
Long Form: LIST:CONTrol>{1 | 0} (1 = on, 0 = off)

Description: **If relay control set to LIST, allows user to energize (1 or on) or de-energize (0 or off) internal relay using LIST commands.** A LIST:CONT value (0 or 1) must be entered, regardless of relay control setting selected. Related Commands: LIST:CONT?.

B.11 [SOURce:]LIST:CONTrol? QUERY — LIST:CONT?

Syntax: Short Form: LIST:CONT? Long Form: LIST:CONTrol?
return value = 0 (off, de-energized) or 1 (on, energized)

Description: **Identifies whether internal relay is energized (1) or de-energized (0).** Related Commands: LIST:CONT.

B.12 [SOURce:]LIST:CONTrol:POINts? QUERY — LIST:CONT:POIN?

Syntax: Short Form: LIST:CONT:POIN? Long Form: LIST:CONTrol:POINts?
Return Value: <int_value>

Description: **Identifies number of control points in the list.** If this command returns 0, no control points were provided: the list will not execute and error, -226,“Lists not same length” will be returned.

B.13 [SOURce:]LIST:COUNt COMMAND — LIST:COUN

Syntax: Short Form: LIST:COUN <int_value 0 to 65535>Long Form: LIST:COUNt> <int_value 0 to 65535>

Description: **Establishes how many times the list is executed.** Allows user to establish how many times the list is executed. The order (beginning to end or end to beginning) is determined by LIST:DIR. For LIST:COUN 0, the unit will execute the sequence indefinitely until a VOLT:MODE FIXED command is received. Related Commands: LIST:DIR. (See example, Figure B-2.)

B.14 [SOURce:]LIST:COUNt? QUERY — LIST:COUN?

Syntax: Short Form: LIST:COUNT? Long Form: LIST:COUNt?
Return Value: <int_value>

Description: **Identifies how many times the list will be executed.** Returns value set by LIST:COUN command. (See example, Figure B-2.)

B.15 [SOURce:]LIST:COUNt:SKIP COMMAND — LIST:COUN:SKIP

Syntax: Short Form: LIST:COUN:SKIP nn Long Form: LIST:COUNt:SKIP nn
nn = 0 to 250

Description: **Allows beginning steps of list-generated waveform to be run once, then ignored.**

When a list is to be repeated using LIST:COUNT, this command allows the user to skip the first nn steps once the full set has been executed. After the first iteration (which executes all steps), the first nn steps are skipped. The LIST:COUN:SKIP command allows the user to precondition a list-generated waveform by setting unique conditions at the beginning that are not repeated for the rest of the repetitions. LIST:CLEar sets nn to 0. Only works in LIST:DIR UP mode, if LIST:DIR DOWN is issued, this command has no effect. Related Commands: LIST:COUN, LIST:COUN:SKIP?, LIST:DIR, LIST:CLE. (See example, Figure B-2.)

B.16 [SOURce:]LIST:COUNt:SKIP? QUERY

LIST:COUN:SKIP?

Syntax: Short Form: LIST:COUN:SKIP? Long Form: LIST:COUNt:SKIP?
Return Value: <int_value>

Description: **Identifies how many steps will skipped the first time the list is executed.** Returns value set by LIST:COUN:SKIP command. (See example, Figure B-2.)

B.17 [SOURce:]LIST:CURRent COMMAND

LIST:CURR

Syntax: Short Form: LIST:CURR <exp_value>, <exp_value>, . . . (to max of 250 data points)
Long Form: LIST:CURRent <exp_value>, <exp_value>, . . . (to max of 250 data points)
<exp_value> = digits with decimal point and Exponent, e.g., 2.71E1 for 27.1

Description: **Adds the current value (in Amps) to list.** This command sequentially adds LIST:CURRent values to the main channel List Data Table. Starting location is indicated by LIST:CURR:POIN? The maximum number of entries is 250. Since the input buffer of the KLR has a limit of 253 characters, multiple commands are necessary to complete the full 250 entries of the list. If LIST:CURR is entered for only the first location, that current level will apply to all points when either VOLT:MODE LIST or CURR:MODE LIST is executed.

Related Commands: LIST:CURR:POIN?. (See example, Figure B-2.)

B.18 [SOURce:]LIST:CURRent? QUERY

LIST:CURR?

Syntax: Short Form: LIST:CURR? Long Form: LIST:CURRent?
Return Value: <value1>, <value2>, . . . to <value16>

Description: **Identifies the parameters (main channel) entered for the list.** Starting at location established by LIST:QUERy, returns comma-separated list of up to 16 values indicating the main channel parameters entered. i.e., the contents of main channel List Data Table. Related Commands: LIST: CURR, LIST:QUERy. If LIST:VOLT has any entries, an error message: -221,"Settings conflict" is posted in the error queue. (See example, Figure B-2.)

B.19 [SOURce:]LIST:CURRent:POINts? QUERY

LIST:CURR:POIN?

Syntax: Short Form: LIST:CURR:POIN? Long Form: LIST:CURRent:POINts?
Return Value: <value> (0 to 250)

Description: **Identifies the total number of points in a list and the next location to be filled by LIST:CURR command.** The LIST:CURR pointer is initially at 0 via LIST:CLE. For each data point entered by a LIST:CURR command the list pointer is incremented If LIST:CURR:POIN? returns 5, the LIST:CURR pointer is at 5 indicating there are 6 data points comprising the list. If LIST:VOLT has any entries, an error message: -221,"Settings conflict" is posted in the error queue.Related Commands: LIST:CURR.

B.20 [SOURce:]LIST:DIRection COMMAND

LIST:DIR

Syntax: Short Form: LIST:DIR {UP | DOWN} Long Form: LIST:DIRection {UP | DOWN}

Description: **Allows the list to be executed from beginning to end (UP) or from end to the beginning (DOWN).** *RST or LIST:CLEar sets the list to the UP direction (beginning to end)**.** Related Commands: LIST:DWEL?.

B.21 [SOURce:]LIST:DIRection? QUERY

LIST:DIR?

Syntax: Short Form: LIST:DIR? Long Form: LIST:DIRection?
Return Value: 1 or 0 (1 = up, 0 = down)

Description: **Identifies the direction for executing the list established by LIST:DIR.** Related Commands: LIST: DIR. (See example, Figure B-2.)

B.22 [SOURce:]LIST:DWELI COMMAND

LIST:DWEL

Syntax: Short Form: LIST:DWEL <value> (0.010 to 655.35),<value>,<value>, . . . to maximum of 250 values
Long Form: LIST:DWELI <value> (0.010 to 655.35),<value>,<value>, . . . to maximum of 250 values

Description: **Determines how long the main channel parameters will be active.** Sets time value (from 0.010 to 655.35) in seconds for List:Dwell locations of the List Data Table. The main channel (either voltage or current) is determined by the load. If LIST:DWEL is entered for only location 0, that time duration will apply to all steps when either VOLT:MODE LIST or CURR:MODE LIST is executed. Related Commands: VOLT:MODE, LIST:CURR, LIST:VOLT, LIST:DWEL?. (See example, Figure B-2.)

B.23 [SOURce:]LIST:DWELI? QUERY

LIST:DWEL?

Syntax: Short Form: LIST:DWEL? Long Form: LIST:DWELI?
Return Value: <value>

Description: **Identifies the dwell times entered for the list.** Starting at location established by LIST:QUERy, returns comma-separated list of up to 16 values indicating the dwell time parameters entered. i.e., the contents of LIST:DWEL locations of the List Data Table. Related Commands: LIST: DWEL, LIST:QUERy.

B.24 [SOURce:]LIST:DWELI:POINts? QUERY

LIST:DWEL:POIN?

Syntax: Short Form: LIST:DWEL:POIN? Long Form: LIST:DWELI:POINts?
Return Value: <value> (0 to 250)

Description: **Identifies the number of locations for which time values have been entered and the next location to be filled by a LIST:DWEL command.** If LIST:DWEL:POIN? returns 6, dwell times have been entered for locations 0 through 5 and location 6 is the next to be filled by a LIST:DWEL command. LIST:DWEL, LIST:DWEL:POIN.

B.25 [SOURce:]LIST:QUERy COMMAND

LIST:QUER

Syntax: Short Form: LIST:QUER <int_value> Long Form: LIST:QUERy <int_value>
int_value = 0 to 250

Description: **Determines first location to be queried by LIST:VOLT?, LIST:CURR?, LIST:DWEL?, LIST:CONT? queries.** Related Commands: LIST:QUER?. (See example, Figure B-2.) Sending LIST:QUER 0 suppresses the 16-entry response to LIST:VOLT?, LIST:CURR?, LIST:DWEL? and LIST:CONT? and causes the entire list to be returned. The list may be up to 2000 bytes (250 points x 10 characters).

B.26 [SOURce:]LIST:QUERy? QUERY

LIST:QUER?

Syntax: Short Form: LIST:QUER? Long Form: LIST:QUERy?
Return Value: <int_value>

Description: **Identifies first location to be queried by LIST:VOLT?, LIST:CURR?, LIST:DWEL?, LIST:CONT? queries.** Related Commands: LIST:QUER. (See example, Figure B-2.)

B.27 [SOURce:]LIST:VOLTage COMMAND

LIST:VOLT

Syntax: Short Form: LIST:VOLT[:LEV] <exp_value>, <exp_value>, . . . (to max of 250 data points)
Long Form: LIST:VOLTage[:LEVel] <exp_value>, <exp_value>, . . . (to max of 250 data points)
<exp_value> = digits with decimal point and Exponent, e.g., 2.71E1 for 27.1

Description: **Adds the voltage value (in Volts) to list.** This command sequentially adds LIST:VOLTage values to the main channel List Data Table locations. LIST:CLE sets starting location to 0. Next location indicated by LIST:VOLT:POIN? The maximum number of entries is 250. Since the input buffer of the KLR has a limit of 253 characters, multiple commands are necessary to complete the full 250 entries of the list. If LIST:VOLT is entered for only the first location, that voltage level will apply to all steps when either VOLT:MODE LIST or CURR:MODE LIST is executed. Related Commands: LIST:VOLT:POIN?, LIST:CLE, *RST. (See example, Figure B-2.)

B.28 [SOURce:]LIST:VOLTage? QUERY

LIST:VOLT?

Syntax: Short Form: LIST:VOLT? Long Form: LIST:VOLTage?
Return Value: <value1>, <value2>, . . . to <value16>

Description: **Identifies the parameters (main channel) entered for the list.** Starting at location established by LIST:QUER, returns comma-separated list of up to 16 values indicating the main channel parameters entered. i.e., the contents of the main channel List Data Table. Related Commands: LIST: VOLT, LIST:QUER. If LIST:CURR has any entries, an error message: -221,"Settings conflict" is posted in the error queue. (See example, Figure B-2.)

B.29 [SOURce:]LIST:VOLTage:POINts? QUERY LIST:VOLT:POIN?

Syntax: Short Form: LIST:VOLT:POIN? Long Form: LIST:VOLTage:POINts?
Return Value: <value> (0 to 250)

Description: **Identifies the total number of points in a list and the next location to be filled by LIST:VOLT command.** The LIST:VOLT pointer is initially at 0 via *RST or LIST:CLE. For each data point entered by a LIST:VOLT command the list pointer is incremented If LIST:VOLT:POIN? returns 5, the LIST:VOLT pointer is at 5 indicating there are 5 data points comprising the list (locations 0 though 4) and location 5 is the next to be filled. If LIST:CURR has any entries, an error message: -221,"Settings conflict" is posted in the error queue. Related Commands: LIST:VOLT. (See example, Figure B-2.)

B.30 MEASure[:SCALar]:CURRent[:DC]? QUERY MEAS:CURR?

Syntax: Short Form: MEAS[:SCAL]:CURR[:DC]? Long Form: MEASure[:SCALar]:CURRent[:DC]?
Return Value: <num_value> (digits with decimal point and Exponent)

Description: **Measures actual current.** This query returns the actual value of output current (measured at the output terminals) as determined by the programmed value of voltage and current and load conditions. NOTE: Current measurement error may be as high as 0.5% for open circuit conditions. (See example, Figure B-1.)

B.31 MEASure[:SCALar]:VOLTage[:DC]? QUERY MEAS:VOLT?

Syntax: Short Form: MEAS[:SCAL]:VOLT[:DC]? Long Form: MEASure[:SCALar]:VOLTage[:DC]?
Return Value: <num_value> (digits with decimal point and Exponent)

Description: **Measures actual voltage.** This query returns the actual value of output voltage (measured at the output terminals) as determined by the programmed value of voltage and current and load conditions. (See example, Figure B-1.)

B.32 MEMory:LOCation COMMAND MEM:LOC

Syntax: Short Form: MEM:LOC <int_value> Long Form: MEMory:LOCation <int_value>
int_value = 1 to 40 (memory cell location)

Description: **Stores power supply parameters in memory cell specified.** Stores voltage, current, overvoltage, overcurrent values and output on/off state as 0 (off) or 1 (on). Related Commands: MEM:LOC?.

B.33 MEMory:LOCation? QUERY MEM:LOC?

Syntax: Short Form: MEM:LOC? <int_value> Long Form: MEMory:LOCation? <int_value>
int_value: 1 to 40 (cell location)
Return value: <val1>,<val2>,<val3>,<val4>,boolean
val1 = current; val2 = voltage; val3 = overcurrent; val4 = overvoltage,
boolean = 0 (output off) or 1 (output on)

Description: **Used to determine contents of 40 memory locations without applying values to KLR output.** Returns the contents of the memory location specified by int_value without affecting KLR output. Memory locations 1 to 40 are programmed to store settings using *SAV or MEM:LOC. The settings can also be recalled and applied to the output by *RCL. If a memory cell has not been programmed, this command returns 0,CURR MIN value,VOLT:PROT MAX value,CURR:PROT MAX value,0 (0 Volts, minimum Amps, overvoltage protection value, overcurrent protection value and output off). Related Commands: MEM:LOC, *SAV, *RCL.

NOTE: The following example assumes KLR 75-32 with limit model set for 36V, 32A and output operating in voltage stabilization mode. See Figure B-3 for directions on programming limit model settings.

LIST:CLE	Clears main channel (LIST:CURR or LIST:VOLT) and LIST:DWEL data tables and initializes the LIST processor to add entries.
LIST:DWEL 2	Sets the time duration for Location 0 to be 2 seconds. NOTE: If specific dwell times are not entered for the rest of the locations in the list before running the list, this dwell time will be assigned to all entries.
LIST:VOLT 28,32,18	Starting at Location 0, fills the first three locations with the voltages shown.
LIST:VOLT:POIN?	Returns 3, indicating that 3 data points have been entered; also indicates that the next data location to be filled is Location 3.
LIST:QUER?	Returns 0 indicating list starting point (default location set by LIST:CLE).
LIST:VOLT?	Returns 2.8E1,3.2E1,1.8E1 (the contents of the filled locations).
LIST:VOLT 20,22,24,26,28	Adds five points to the list (Locations 3 through 7). The list now has eight points.
LIST:VOLT:POIN?	Returns 8.
LIST:QUER 3	Starts LIST queries from Location 3.
LIST:COUN 10	Specifies that upon receipt of VOLT:MODE LIST, the above sequence will be executed 10 times.
LIST:COUN:SKIP 2	Specifies that the first two steps of the sequence will only be executed during the first cycle through the list; thereafter, the list shall execute Locations 2 through 7 an additional 9 times.
LIST:CURR 3	Programs output current to 3A for all locations. NOTE All active locations must have complete sets of data entered, or list will not run.
LIST: CONT 0	Programs relay control to de-energized state. NOTE: This command is REQUIRED even if the relay control is not set to LIST.
OUTP ON	Enables output.
VOLT 24;CURR 3	Initializes output voltage to 24V, output current to 3A.
VOLT:MODE LIST	Executes the list. For the first cycle the voltage will step to 28V and 32V in 2-second dwell increments, then begin a cycle of steps from 18V to 28V in 2V steps and 2-second dwell increments repeated ten times, at the end of which time the output will remain constant at 28V and 3A.
VOLT:MODE FIX	Halts execution of the list and maintains the outputs at their immediate settings.

FIGURE B-2. USING LIST COMMANDS AND QUERIES

B.34 OUTPut[:STATe] COMMAND — OUTP

Syntax: Short Form: OUTP[:STAT] {ON | OFF} or {1 | 0} (1 = on, 0 = off)
Long Form: OUTPut[:STATe] {ON | OFF} or {1 | 0} (1 = on, 0 = off)

Description: **Enables or disables the power supply output (see KLR User manual for disabling if analog programming used).** Upon power up the output is enabled (OUTP ON). When OUTP OFF is executed, the digitally programmed values of voltage and current are saved, then voltage and current are programmed to 0; analog programming is not affected. When OUTP ON is executed, the power supply output is restored to the previously saved programmed values. The saved values of voltage and current can be viewed by VOLT? and CURR? queries. This command also clears fault errors for faults not set to PROTECT mode.

Related Commands: OUTP?. (See example, Figure B-1.)

B.35 OUTPut[:STATe]? QUERY OUTP?

Syntax: Short Form: OUTP[:STAT]? Long Form: OUTPut[:STATe]?
Return Value: <int_value> (1 or 0) (1 = on, 0 = off)

Description: **Indicates whether power supply output is enabled or disabled.** Returns 0 if output disabled, returns 1 if output enabled. Related Commands: OUTP. (See example, Figure B-1.)

B.36 [SOURce:]CURRent[:LEVel][:IMMediate][:AMPLitude] COMMAND CURR

Syntax: Short Form: [SOUR:]CURR[:LEV][:IMM][:AMPL] <exp_value>
Long Form: [SOURce:]CURRent[:LEVel][:IMMediate][:AMPLitude] <exp_value>
<exp_value> = digits with decimal point and Exponent, e.g., 2.71E+1 for 27.1

Description: **Sets programmed current level at power supply output.** This command programs the output current regulation point to a specific value; actual output current will depend on load conditions. If the programmed value exceeds the rated model maximum for the model being programmed, error message -222,"Data out of range" is posted in output queue. If the programmed value exceeds either the Limit Model limit (CURR:LIM:HIGH) or 80% of current protection (CURR:PROT), a value corresponding to CURR:LIM:HIGH or 80% of CURR:PROT, respectively, is programmed, and error message -301,"Value too large" is posted in output queue. If a value programmed is lower than the model minimum current, a value corresponding to the model minimum will be programmed; no error message is generated. Related Commands: CURR:LIM:HIGH. (See example, Figure B-1.)

B.37 [SOURce:]CURRent[:LEVEL][:IMMediate][:AMPLitude]? QUERY CURR?

Syntax: Short Form: [SOUR:]CURR[:LEV][:IMM][:AMPL]? MIN, MAX
Long Form: [SOURce:]CURRent[:LEVel][:IMMediate][:AMPLitude]? MIN, MAX
Return Value:<exp_value> = digits with decimal point and Exponent, e.g., 2.71E+1 for 27.1

Description: **Returns either the programmed value, maximum value, or minimum value of current.** The CURR? query returns the programmed value of current. Actual output current will depend on load conditions. The CURR? MAX query returns the maximum current allowed for a particular model. CURR? Returns programmed current value. CURR? MAX returns maximum current allowed for power supply (the lower of CURR:LIM:HIGH and 80% of CURR:PROT). CURR? MIN returns minimum current allowed for the power supply model. Related Commands: CURR. (See example, Figure B-1.)

B.38 [SOURce:]CURRent:LIMit:HIGH COMMAND CURR:LIM:HIGH

Syntax: Short Form: [SOUR:]CURR:LIM:HIGH <exp_value> [MAX]
Long Form: [SOURce:]CURRent:LIMit:HIGH <exp_value> [MAX]
<exp_value> = digits with decimal point and Exponent, e.g., 2.71E+1 for 27.1

Description: **Sets Limit Model maximum programmable limit for power supply output current.** CURR:LIM:HIGH <exp_value> sets maximum current that can be programmed. CURR:LIM:HIGH MAX sets maximum programmable limit of the unit to the rated model maximum (e.g., 32A for KLR 75-32). This command is password protected, requiring SYST:PASS:CEN prior to execution.

After executing CURR:LIM:HIGH, the Output is set to OFF and overcurrent protection (CURR:PROT) is recalculated to be the greater of 20% above the CURR:LIM:HIGH setting or 72% of rated model maximum. Previously stored trigger levels are reset to zero volts and minimum current. Stored settings (PAR. A.13) or programmed sequences using the LIST commands may be outside the range established by the new limit model. In this case, prior to execution, the value is cleared to the default minimum (minimum current) and error message -222,"Data out of range" is generated.

If the value is out of the acceptable current range for the power supply model, error message -222,"Data out of range" is posted in the output queue. If the user tries to set an output current value larger than the CURR:LIM:HIGH setting, a value corresponding to the CURR:LIM:HIGH will be programmed and error message -301,"Value bigger than limit" is posted in the output queue. Once the limit is established, the unit will not allow values higher than the programmed limit. Related Commands: CURR, CURR:PROT, SYST:PASS:CEN. (See example, Figure B-3.)

NOTE: The following example assumes KLR 75-32 with limit model set for 75V, 16A and output operating in voltage stabilization mode.

SYST:PASS:CEN 7533	Enables access to the protected commands. NOTE: Password shown is factory default value for KLR 75-32; values will vary for other models (see Table B-4) and can be altered by user.
VOLT:LIM:HIGH?	Returns 7.5E1 (75V, limit model voltage rating).
VOLT:LIM:HIGH 70	Programs limit model voltage to 70V.
VOLT:LIM:HIGH?	Returns 7.0E1 (70V, new limit model voltage rating).
CURR:LIM:HIGH?	Returns 1.6E1 (16A, limit model current rating).
CURR:LIM:HIGH 30	Programs limit model current to 30A.
CURR:LIM:HIGH?	Returns 3E1 (30A, limit model current rating).
VOLT:LIM:HIGH 33	Programs limit model voltage to 30A. Limit model settings are now programmed to 33V and 30A.

FIGURE B-3. PROGRAMMING LIMIT MODELS

B.39 [SOURce:]CURRent:LIMit:HIGH? QUERY — CURR:LIM:HIGH?

Syntax: Short Form: [SOUR:]CURR:LIM:HIGH? (MIN | MAX)
Long Form: [SOURce:]CURRent:LIMit:HIGH? (MIN | MAX)
Return Value:<exp_value> = digits with decimal point and Exponent, e.g., 2.71E+1 for 27.1

Description: **Returns value representing Limit Model current limit.** CURR:LIM:HIGH? returns the current limit programmed by CURR:LIM:HIGH. CURR:LIM:HIGH? MAX returns the rated maximum programmable current for that model (e.g., 32 (Amperes) for KLR 75-32. CURR:LIM:HIGH? MIN returns the minimum allowable current that can be programmed.
Related Commands: CURR:LIM:HIGH, CURR. (See example, Figure B-3.)

B.40 [SOURce:]CURRent:MODE COMMAND — CURR:MODE

Syntax: Short Form: [SOUR:]CURR:MODE (FIX | LIST | TRAN,n,A]
Long Form: [SOURce:]CURRent:MODE (FIXed | LIST | TRANsient,n,A]
n = <value> = time between 0.04 to 655.36 in seconds
A = <value> = Current level in Amperes

Description: **Allows the user to execute or stop a list, or to execute a transient.** The default mode is FIX: the power supply executes commands normally, and LIST commands can be issued to establish the parameters and sequence of a list.

Two milliseconds after CURR:MODE LIST is issued, a list is executed (See LIST commands and Figure B-2). While the list is being executed, LIST commands are not accepted and will produce a command error. The 4- character front panel display shows `dLST` when a list is executing.

Issuing CURR:MODE FIX while the list is running will stop the execution of the list and return power supply to settings in effect prior to running the list. If the list runs to completion, the settings of the last step of the list will be in effect.

CURR:MODE TRAN,<n>,<A> [R option (RODC installed) models only.] Two milliseconds after issuing this command the KLR output current changes to the specified current level <A> for the period of time <n> specified. At the end of time <n>, the current is returned to the previously established value. The status display shows `dLST` during the execution of the transient. If any of the variables are incorrect, the transient is not generated, and an appropriate error message is generated (see Table B-5). If CURR:MODE TRAN command is issued to a non-R option unit, error message -113, “undefined header” is posted to the error queue. If current value is less than model minimum, the minimum value is entered and no error is generated.

Related Commands: LIST commands. (See example, Figure B-2.)

B.41 [SOURce:]CURRent:MODE? QUERY — CURR:MODE?

Syntax: Short Form: [SOUR:]CURR:MODE? Long Form: [SOURce:]CURRent:MODE?
Return value: FIXED or LIST or TRAN

Description: **Identifies active voltage mode.** Returns LIST while list is being executed. TRAN (transient) is not supported at this time, contact Kepco for more information. Returns FIXED while in fixed (default) mode of operation. Related Commands: LIST commands. (See example, Figure B-2.)

B.42 [SOURce:]CURRent:PROTection[:LEVel] COMMAND

CURR:PROT

Syntax: Short Form: [SOUR:]CURR:PROT[:LEV] <exp_value>
Long Form: [SOURce:]CURRent:PROTection[:LEVel] <exp_value>
<exp_value> = digits with decimal point and Exponent, e.g., 2.71E+1 for 27.1

Description: **Sets overcurrent protection (OCP) level for power supply.** Restricts valid programmable current to be no greater than 80% of this setting, overriding Limit Model current limit (CURR:LIM:HIGH) when overcurrent protection is less than 80% of than the current limit. Valid range for CURR:PROT setting is 72% to 120% of rated model maximum current (e.g., 23.98 to 38.6A for KLR 75-32).

After executing CURR:PROT, the Output is set to OFF and previously stored trigger levels are reset to zero volts and minimum current.

If the programmed OCP value is out of the acceptable overcurrent range for the power supply model, error message -222,"Data out of range" is posted in the output queue. If the power supply output exceeds the current protection level programmed, then the power supply output is disabled (programmed to 0V, minimum current) and the OCP bit in the Questionable Condition status register is set. The value of CURR:PROT is automatically recalculated if CURR:LIM:HIGH is executed. (See example, Figure B-4.)

NOTE: The following example assumes KLR 75-32 with limit model set for 36V, 32A and output operating in current stabilization mode. See Figure B-3 for directions on programming limit model settings.

VOLT 32.1;CURR 4	Programs output voltage for 32.1V, output current for 4A.
OUTP ON	Enables output. NOTE: Load determines output operating mode; if load resistance is less than 32.1V/4A = 8.025 Ohms, output will operate as current stabilizer.
MEAS:CURR?	Returns measured output current, approximately 4E0 assuming current stabilizer operation.
CURR?	Returns 4E0 (4A, programmed output current).
CURR 3.3E-1	Programs output current to 0.33A. NOTE: For this model, 0.4A is the minimum load current, therefore although 0.33 value is programmed; load current will not be less than 0.4A. No explicit indication is given regarding this change.
CURR?	Returns 4E-1 (0.4A).
CURR? MAX	Returns 3.333E1 (33.33A, the limit model current setting).
CURR:PROT .5	For this model, minimum overcurrent protection is 0.72 x 32 = 23.04A, therefore power supply returns error message -222 (Data out of range, ESR bit 4 =1) and the limit remains unchanged.
CURR:PROT 25	Programs overcurrent protection limit to 25A.
CURR:PROT?	Returns 2.5E1 (25A).
CURR 26	Returns error message -301 (Value bigger than limit, ESR bit 3=1) and current setting remains unchanged at 0.4A.
CURR:PROT?MAX	Returns 4E1 (maximum allowable overcurrent protection limit for model is 40A, 20% above maximum current (32A) for KLR 75-32, regardless of limit model setting).

FIGURE B-4. PROGRAMMING AS A CURRENT STABILIZER

B.43 [SOURce:]CURRent:PROTection[:LEVel]? QUERY CURR:PROT?

Syntax: Short Form: [SOUR:]CURR:PROT[:LEV]? {MIN | MAX}
Long Form: [SOURce:]CURRent:PROTection[:LEVel]? {MIN | MAX}
Return Value: <exp_value> = digits with decimal point and Exponent, e.g., 2.71E+1 for 27.1

Description: **Returns value representing current protection level.**CURR:PROT? returns value set by CURR:PROT. CURR:PROT?MAX returns maximum current protection value. This value is determined at the factory and cannot be changed by the user. CURR:PROT?MIN returns the minimum current protection value. (See example, Figure B-4.)

B.44 [SOURce:]CURRent:[:LEVel]TRIGgered[:AMPLitude] COMMAND CURR:TRIG

Syntax: Short Form: [SOUR:]CURR[:LEV]:TRIG[:AMPL] <exp_value> MAX
Long Form: [SOURce:]CURRent[:LEVel]:TRIGgered[:AMPLitude] <exp_value> MAX
<exp_value> = digits with decimal point and Exponent, e.g., 2.71E+1 for 27.1

Description: **Programs current value to be transferred to output by *TRG commands.** Actual output current will depend on load conditions. CURR:TRIG MAX programs output current value to be transferred by *TRG to be the lower of CURR:LIM:HIGH or 80% of CURR:PROT. If the value exceeds the maximum for the model being programmed, error message -222,"Data out of range" is posted in output queue. If value exceeds CURR:LIM:HIGH value, a value corresponding to the current limit will be programmed. This value is automatically checked after execution of CURR:LIM:HIGH and CURR:PROT and is set to minimum current if there is a conflict. NOTE: A voltage trigger level must be entered at least once. Related Commands: CURR. (See example, Figure B-1.)

B.45 [SOURce:]CURRent:[:LEVel]TRIGgered[:AMPLitude]? QUERY CURR:TRIG?

Syntax: Short Form: [SOUR:]CURR[:LEV]:TRIG[:AMPL]? MAX
Long Form: [SOURce:]CURRent[:LEVel]:TRIGgered[:AMPLitude]? MAX
Return Value: <exp_value> = digits with decimal point and Exponent, e.g., 2.71E+1 for 27.1

Description: **Returns value representing current value to be programmed by *TRG command established by CURR:TRIG command).** CURR:TRIG? MAX returns maximum permissible value that can be programmed by CURR:TRIG (the lower of CURR:LIM:HIGH and 80% of CURR:PROT) (See example, Figure B-1.)

B.46 [SOURce:]FUNCtion:MODE COMMAND FUNC:MODE

Syntax: Short Form: FUNC:MODE {VOLT | CURR} Long Form: FUNCtion:MODE {VOLT | CURR}

Description: **Establishes the expected operating mode of the power supply.** VOLT = Constant Voltage mode (CV). CURR = Constant Current mode (CC). Default operating mode is CV. NOTE: Actual operating mode is determined by programmed output limits and load impedance.

B.47 [SOURce:]FUNCtion:MODE? QUERY FUNC:MODE?

Syntax: Short Form: FUNC:MODE? Long Form: [SOURce:]FUNCtion:MODE?
Return Value: String1,String2 where string 1 is the actual operating mode: VOLT = Constant Voltage mode (CV).CURR = Constant Current mode (CC) and string 2 is the mode that was established using FUNC:MODE.

Description: **Identifies the operating mode and commanded operating of the power supply.**

B.48 [SOURce:]VOLTage[:LEVel][:IMMediate][:AMPLitude] COMMAND VOLT

Syntax: Short Form: [SOUR:]VOLT[:LEV][:IMM][:AMPL] <exp_value>
Long Form: [SOURce:]VOLTage[:LEVel][:IMMediate][:AMPLitude] <exp_value>
<exp_value> = digits with decimal point and Exponent, e.g., 2.71E+1 for 27.1

Description: **Sets programmed voltage level at power supply output.** This command programs output voltage regulation point to a specific value; actual output voltage will depend on load conditions. If the programmed value exceeds the rated model maximum for the model being programmed, error message -222,"Data out of range" is posted in output queue. If the programmed value exceeds either the Limit Model voltage limit (VOLT:LIM:HIGH) or 80% of voltage protection limit (VOLT:PROT), a value corresponding to VOLT:LIM:HIGH or VOLT:PROT, respectively, is programmed, and error message -301,"Value too large" is posted in output queue. Related Commands: VOLT:LIM:HIGH. (See example, Figure B-1.)

B.49 [SOURce:]VOLTage[:LEVel][:IMMediate][:AMPLitude]? QUERY VOLT?

Syntax: Short Form: [SOUR:]VOLT[:LEV][:IMM][:AMPL]? {MIN | MAX}
Long Form: [SOURce:]VOLTage[:LEVel][:IMMediate][:AMPLitude]? {MIN | MAX}

Description: **Identifies programmed voltage, maximum allowable voltage, or minimum voltage (always 0).** The VOLT? query returns the programmed value of voltage. Actual output voltage will depend on load conditions. The VOLT? MAX query returns the maximum voltage allowed for a particular model (the lower of VOLT:LIM:HIGH and 80% of VOLT:PROT). VOLT? MINReturns minimum voltage allowed for power supply (always 0). Related Commands: VOLT. (See example, Figure B-5.)

B.50 [SOURce:]VOLTage:LIMit:HIGH COMMAND VOLT:LIM:HIGH

Syntax: Short Form: [SOUR:]VOLT:LIM:HIGH <exp_value> [MAX]
Long Form: [SOURce:]VOLTage:LIMit:HIGH <exp_value> [MAX]
<exp_value> = digits with decimal point and Exponent, e.g., 2.71E+1 for 27.1

Description: **Sets Limit Model maximum programmable limit for power supply output voltage.** VOLT:LIM:HIGH <exp_value> sets maximum voltage that can be programmed. VOLT:LIM:HIGH MAX sets maximum programmable limit of the unit to the rated maximum (e.g., 75V for KLR 75-32). This command is password protected, requiring SYST:PASS:CEN prior to execution.

After executing VOLT:LIM:HIGH, the Output is set to OFF and overvoltage protection (VOLT:PROT) is recalculated to be 20% above the VOLT:LIM:HIGH setting. Previously stored trigger levels are reset to zero volts and minimum current. Stored settings (PAR. A.13) or programmed sequences using the LIST commands may be outside the range established by the new limit model. In this case, prior to execution, the value is cleared to the default minimum (zero volts) and -222, “Data out of range” error message is generated.

If the value is out of the acceptable voltage range for the power supply model, error message -222,”Data out of range” is posted in the output queue. If the user tries to set an output voltage value larger than the VOLT:LIM:HIGH setting, a value corresponding to the VOLT:LIM:HIGH will be programmed and error message -301,”Value bigger than limit” is posted in the output queue. Once the limit is established, the unit will not allow values higher than the programmed limit. Related Commands: VOLT, VOLT:PROT, SYST:PASS:CEN. (See example, Figure B-3.)

B.51 [SOURce:]VOLTage:LIMit:HIGH? QUERY VOLT:LIM:HIGH?

Syntax: Short Form: [SOUR:]VOLT:LIM:HIGH? [MIN | MAX]
Long Form: [SOURce:]VOLTage:LIMit:HIGH? [MIN | MAX]
Return Value: <exp_value> = digits with decimal point and Exponent, e.g., 2.71E+1 for 27.1

Description: **Returns value representing Limit Model voltage limit.** VOLT:LIM:HIGH? returns the voltage limit programmed by VOLT:LIM:HIGH. VOLT:LIM:HIGH? MAX returns the rated maximum programmable voltage for that model (e.g., 75 (Volts) for KLR 75-32. VOLT:LIM:HIGH? MIN returns the minimum allowable voltage that can be programmed. Related Commands: VOLT:LIM:HIGH, VOLT. (See example, Figure B-5.)

B.52 [SOURce:]VOLTage:MODE COMMAND — VOLT:MODE

Syntax: Short Form: [SOUR:]VOLT:MODE (FIX | LIST | TRAN,<n>,<V
Long Form: [SOURce:]VOLTage:MODE (FIXed | LIST | TRANsient,n.V
n = <value> = time between 0.04 to 655.36 in seconds
V = <value> = Voltage level in Volts

Description: **Allows the user to execute or stop a list, or to execute a transient.** The default mode is FIX: the power supply executes commands normally, and LIST commands can be issued to establish the parameters and sequence of a list.

Two milliseconds after VOLT:MODE LIST is issued, a list is executed (See LIST commands and Figure B-2). While the list is being executed, LIST commands are not accepted and will produce a command error.

Issuing VOLT:MODE FIX while the list is running will stop the execution of the list and return power supply to settings in effect prior to running the list. If the list runs to completion, the settings of the last step of the list will be in effect.

VOLT:MODE TRAN,<n>,<V> [R option (RODC installed) models only.] Two milliseconds after issuing this command the KLR output voltage changes to the specified voltage level <V> for the period of time <n> specified. At the end of time <n>, the voltage is returned to the previously established value. The status display shows `dLST` during the execution of the transient. If any of the variables are incorrect, the transient is not generated, and an appropriate error message is generated (see Table B-5). If VOLT:MODE TRAN command is issued to a non-R option unit, error message -113, “undefined header” is posted to the error queue.
Related Commands: LIST commands. (See example, Figure B-2.)

B.53 [SOURce:]VOLTage:MODE? QUERY — VOLT:MODE?

Syntax: Short Form: [SOUR:]VOLT:MODE? Long Form: [SOURce:]VOLTage:MODE?
Return value**:** FIXED or LIST or TRAN

Description: **Identifies active voltage mode.** Returns LIST while list is being executed. TRAN (transient) is not supported at this time, contact Kepco for more information. Returns FIXED while in fixed (default) mode of operation. Related Commands: LIST commands. (See example, Figure B-2.)

B.54 [SOURce:]VOLTage:PROTection[:LEVel] COMMAND — VOLT:PROT

Syntax: Short Form: [SOUR:]VOLT:PROT[:LEV] <exp_value>
Long Form: [SOURce:]VOLTage:PROTection[:LEVel] <exp_value>
<exp_value> = digits with decimal point and Exponent, e.g., 2.71E+1 for 27.1

Description: **Sets overvoltage protection (OVP) level for power supply.** Restricts valid programmable voltage to be no greater than 80% of this setting, overriding Limit Model voltage limit (VOLT:LIM:HIGH) when overvoltage protection is less than 80% of the voltage limit. Valid range is 20% to 120% of rated model maximum voltage (e.g., 15V to 90V for KLR 75-32)

After executing VOLT:PROT, the Output is set to OFF and previously stored trigger levels are reset to zero volts and minimum current.

If the programmed OVP value is out of the acceptable overvoltage range for the power supply model, error message -222,”Data out of range” is posted in the output queue. If the power supply output exceeds the voltage protection level programmed, then the power supply output is disabled (programmed to 0V, minimum current) and the OVP bit in the Questionable Condition status register is set. The value of VOLT:PROT is automatically recalculated if VOLT:LIM:HIGH is executed. (See example, Figure B-5.)

B.55 [SOURce:]VOLTage:PROTection[:LEVel]? QUERY — VOLT:PROT?

Syntax: Short Form: [SOUR:]VOLT:PROT[:LEV] MIN, MAX
Long Form: [SOURce:]VOLTage:PROTection[:LEVel] MIN, MAX
Return Value: <exp_value> = digits with decimal point and Exponent, e.g., 2.71E+1 for 27.1

Description: **Identifies overvoltage protection setting, maximum allowable overvoltage protection, or minimum overvoltage protection.** VOLT:PROT? returns value set by VOLT:PROT. VOLT:PROT? MAX returns maximum voltage protection value; this value is determined at the factory and cannot be changed by the user. VOLT:PROT? MIN returns the minimum voltage protection value (0.2 x model voltage, e.g., 1.5V for KLR 75-32). (See example, Figure B-5.)

B.56 [SOURce:]VOLTage:[LEVel:]TRIGgered[:AMPLitude] COMMAND VOLT:TRIG

Syntax: Short Form: [SOUR:]VOLT:[LEV:]TRIG[:AMPL] <exp_value> MAX
Long Form: [SOURce:]VOLTage:[LEVel:]TRIGgered[:AMPLitude] <exp_value> MAX
<exp_value> = digits with decimal point and Exponent, e.g., 2.71E+1 for 27.1

Description: **Programs voltage value to be transferred to output by *TRG commands.** Actual output voltage will depend on load conditions. VOLT:TRIG MAX programs output voltage value to be transferred by *TRG to be the lower of VOLT:LIM:HIGH or 80% of VOLT:PROT. If the value exceeds the maximum for the model being programmed, error message -222,"Data out of range" is posted in output queue. If value exceeds VOLT:LIM:HIGH value, a value corresponding to the voltage limit will be programmed. This value is automatically checked after execution of VOLT:LIM:HIGH and VOLT:PROT, and is set to zero volts if there is a conflict. NOTE: A current trigger must be entered at least once. (See example, Figure B-1.)

B.57 [SOURce:]VOLTage:[:LEVel]TRIGgered[:AMPLitude]? QUERY VOLT:TRIG?

Syntax: Short Form: [SOUR]:VOLT[:LEV]:TRIG[:AMPL]? MAX
Long Form: [SOURce]:VOLTage[:LEVel]:TRIGgered[:AMPLitude]? MAX
Return Value: <exp_value> = digits with decimal point and Exponent, e.g., 2.71E+1 for 27.1

Description: **Returns value representing voltage value to be programmed by *TRG command established by VOLT:TRIG command).** VOLT:TRIG? MAX returns maximum permissible value that can be programmed by VOLT:TRIG (the lower of VOLT:LIM:HIGH and 80% of VOLT:PROT) (See example, Figure B-1.)

NOTE: The following example assumes KLR 75-32 with limit model set for 36V, 32A and output operating in voltage stabilization mode. See Figure B-3 for directions on programming limit model settings.

VOLT 21.8;CURR .5	Programs output voltage for 21.8V, output current for 0.5A.
OUTP ON	Enables output. NOTE: Load determines output operating mode; if load resistance is greater than 21.8/0.5, output will operate as voltage stabilizer.
MEAS:VOLT?	Returns measured output voltage, approximately 2.18E1 (21.8V) assuming voltage stabilizer operation.
VOLT?	Returns 2.18E1 ((21.8V, programmed output voltage).
VOLT 2.15	Programs output voltage to 2.15V.
VOLT?	Returns 2.15E0 (2.15V, programmed output voltage).
VOLT? MAX	Returns 3.6E1 (36V, the limit model voltage setting).
VOLT? MIN	Returns 0 (minimum allowable program voltage).
VOLT:PROT 6.5	For this model, minimum overvoltage protection is 0.2 x 75 = 15V, therefore power supply returns error message -222 (Data out of range, ESR bit 4 =1) and the limit remains unchanged.
VOLT:PROT 25	Sets overvoltage protection limit to 25V.
VOLT:PROT?	Returns 2.5E1 (25V).
VOLT 26	Returns error message -301 (Value bigger than limit, ESR bit 3=1) and voltage setting remains unchanged at 2.15V.
VOLT:PROT?MAX	Returns 9E1. Maximum allowable overvoltage protection limit for model is 90V, 20% above maximum voltage (75V for KLR 75-32).

FIGURE B-5. PROGRAMMING AS VOLTAGE STABILIZER

B.58 STATus:OPERation:CONDition? QUERY

STAT:OPER:COND?

Syntax**:** Short Form: STAT:OPER:COND? Long Form: STATus:OPERation:CONDition?
Return Value: 0 to 1313 (1 + 32 + 256 + 1024).

Description: **Returns the value of the Operation Condition Register (see Table B-2).** The Operation Condition Register contains unlatched real-time information about the operating conditions of the power supply. Bit set to 1 = function enabled (active, true); bit reset to 0 = function disabled (inactive, false). (See example, Figure B-6.)

TABLE B-2. OPERATION CONDITION REGISTER, OPERATION ENABLE REGISTER, AND OPERATION EVENT REGISTER BITS

CONDITION	NU	PR	NU	CC	NU	CV	NU	WTG	NU	CAL
BIT	15	14	11-13	10	9	8	7 - 6	5	4 - 1	0
VALUE	32,768	16,384	2048 - 8192	1024	512	256	128 - 64	32	16 - 2	1

CAL - POWER SUPPLY IN CALIBRATION MODE
CC - POWER SUPPLY IN CONSTANT CURRENT MODE
CV - POWER SUPPLY IN CONSTANT VOLTAGE MODE
NU - NOT USED
PR - PROGRAM RUNNING
WTG - WAIT FOR TRIGGER (SET BY TRIG SUBSYSTEM)

B.59 STATus:OPERation:ENABle COMMAND

STAT:OPER:ENAB

Syntax**:** Short Form:STAT:OPER:ENAB <int_value> 0 to 1313 (1 + 32 + 256 + 1024)
Long Form:STATus:OPERation:ENABle <int_value> 0 to 1313 (1 + 32 + 256 + 1024)

Description: **Sets Operation Enable Register.** The Operation Enable Register is a mask for enabling specific bits in the Operation Event Register which will cause the operation summary bit (bit 7) of the Status Byte register to be set. Bit set to 1 = function enabled (active, true); bit reset to 0 = function disabled (inactive, false). The operation summary bit is the logical OR of all the enabled bits in the Operation Event register. (See example, Figure B-6.)

B.60 STATus:OPERation:ENABle? QUERY

STAT:OPER:ENAB?

Syntax**:** Short Form: STAT:OPER:ENAB? Long Form: STATus:OPERation:ENABle?
Return Value: <int_value> 0 to 1313 (1 + 32 + 256 + 1024).

Description: **Reads Operation Enable Register (see Table B-2).** Returns value of Operation Enable Register bits. Bit set to 1 = function enabled (active, true); bit reset to 0 = function disabled (inactive, false). (See example, Figure B-6.)

B.61 STATus:OPERation[:EVENt]? QUERY

STAT:OPER?

Syntax**:** Short Form: STAT:OPER[:EVEN]? Long Form: STATus:OPERation[:EVENt]?
Return Value: <int_value> 0 to 1313 (1 + 32 + 256 + 1024).

Description: **Indicates changes in conditions monitored by Operational Event Register.** Returns the value of the Operation Event register. The Operation Event register is a read-only register which holds (latches) all events that occur. Reading the Operation Event register clears it. (See example, Figure B-6.)

B.62 STATus:PRESet COMMAND

STAT:PRES

Syntax**:** Short Form: STAT:PRES Long Form: STATus:PRESet

Description: **Disables reporting of all status events.** This command sets all bits of the Operation Condition (Table B-2) and Questionable Condition Enable Registers to 0, preventing all status events from being reported. (See example, Figure B-6.)

NOTE: The following example assumes KLR 75-32 with limit model set for 75V, 16A and output operating in voltage stabilization mode. See Figure B-3 for directions on programming limit model settings.

volt:prot:max	Sets OVP limit to maximum.
outp on	Power supply output is on.
volt 20;curr 1	Power supply output is programmed to 20V, 1A.
syst:err?	Returns 0, “No error” message.
stat:oper?	Returns 1280, indicating that power supply has entered both CV and CC modes during start-up.
stat:oper:enab 1312	Mask enabled for CV, CC, and WTG bits.
stat:oper:enab?	Returns 1312 (1024 + 256 + 32), indicating CV, CC, and WTG bits are set.
init:cont on	Continuous triggers enabled.
stat:oper:cond?	Power supply returns 288 (256 + 32), indicating power supply is in constant voltage mode and Wait For Trigger is set.
stat:oper?	Returns 32, CV mode bit cleared by prior query, but continuous triggering results in WTG bit always being set.
volt 30	Output voltage programmed to 30V; current remains at 1A.
*stb?	Return 128, Operation Status Summary bit is set.
stat:ques?	Returns 16, loss of source power detected. (This event was set at prior power-down of power supply and is retained for retrieval at next power-up; reading the event register clears the bit.)
stat:ques:enab 3	Mask enabled for OVP and OCP bits (1 + 2).
stat:ques:enab?	Returns 3 (1 + 2) indicating OVP and OCP bits are enabled.
volt:prot 25	Overvoltage protection limit set to 25V with output programmed to 30V, creating an OVP fault condition.
*stb?	Returns 140, Operation, Questionable and Error Status bits were set.
syst:err?	Returns -305 “Voltage Protection Fault”
stat:ques?	Returns 1, overvoltage protection error detected.
stat:ques?	Returns 0, Reading prior register cleared register contents.
stat:ques:cond?	Returns 1, power supply is in overvoltage protection.
outp?	Returns 0, output set to off when overvoltage error detected.
stat:pres	Operation enable and Questionable enable registers reset.
stat:ques:enab?	Returns 0, Questionable Condition Enable register reset prevents any questionable events from being reported.
stat:oper:enab?	Returns 0, Operation Condition Enable register reset prevents any operational events form being reported.

FIGURE B-6. USING STATUS COMMANDS AND QUERIES

B.63 STATus:QUEStionable[:EVENt]? QUERY

STAT:QUES?

Syntax**:** Short Form: STAT:QUES[EVEN]? Long Form: STATus:QUEStionable[EVENT]?
Return Value: <int_value> actual register value

Description: **Indicates questionable events that occurred since previous STAT:QUES? query.** Returns the value of the Questionable Event register (see Table B-3). The Questionable Event register is a read-only register which holds (latches) all events. Reading the Questionable Event register clears it. (See example, Figure B-6.)

NOTE: Removing source power from the unit (e.g., setting POWER ON/OFF circuit breaker to OFF) causes the unit to generate and store the PWR bit. Therefore the first query of the Questionable Event Register after the unit is turned on will always show a PWR fault - this is normal.

TABLE B-3. QUESTIONABLE EVENT REGISTER, QUESTIONABLE CONDITION REGISTER AND QUESTIONABLE CONDITION ENABLE REGISTER BITS

CONDITION	NU	M/S	FAN	PWR	OTP	OLF	OCP	OVP
BIT	15 - 7	6	5	4	3	2	1	0
VALUE	32,768 - 128	64	32	16	8	4	2	1

M/S - MASTER/SLAVE FAILURE
FAN - INTERNAL FAN FAILURE
PWR - LOSS OF SOURCE POWER
OTP - OVERTEMPERATURE
OLF - OUTPUT LEAD FAULT
OCP - OVERCURRENT
OVP - OVERVOLTAGE
NU - NOT USED

B.64 STATus:QUEStionable:CONDition? QUERY STAT:QUES:COND?

Syntax**:** Short Form: STAT:QUES:COND? Long Form: STATus:QUEStionable:CONDition?
Return Value: <int_value> actual register value

Description: **Returns the value of the Questionable Condition Register (see Table B-3).** The Questionable Condition Register contains unlatched real-time information about questionable conditions of the power supply. Bit set to 1 = condition (active, true); bit reset to 0 = condition (inactive, false). (See example, Figure B-6.)

B.65 STATus:QUEStionable:ENABle COMMAND STAT:QUES:ENAB

Syntax**:** Short Form: STAT:QUES:ENAB <int_value>
Long Form: STATus:QUESionable:ENABle <int_value>

Function: **Programs Questionable Condition Enable Register.**

Description: **Programs Questionable Condition Enable Register (see Table B-3).**The Questionable Condition Enable Register determines which conditions are allowed to set the Questionable Condition Register; it is a mask for enabling specific bits in the Questionable Event register that can cause the questionable summary bit (bit 3) of the Status Byte register to be set. The questionable summary bit is the logical OR of all the enabled bits in the Questionable Event register. Bit set to 1 = function enabled (active, true); bit reset to 0 = function disabled (inactive, false). (See example, Figure B-6.)

B.66 STATus:QUEStionable:ENABle? QUERY STAT:QUES:ENAB?

Syntax**:** Short Form: STAT:QUES:ENAB? Long Form: STATus:QUEStionable:ENABle?
Return Value: <int_value> actual register value

Description: **Reads Questionable Condition Enable Register (see Table B-3).** Power supply returns value of Questionable Condition Enable Register, indicating which conditions are being monitored. Bit set to 1 = function enabled (active, true); bit reset to 0 = function disabled (inactive, false). Related Commands: STAT:QUES?. (See example, Figure B-6.)

B.67 SYSTem:COMMunication:GPIB:ADDRess COMMAND SYST:COMM:GPIB:ADDR

Syntax**:** Short Form: SYST:COMM:GPIB:ADDR<int_value> 1 to 30
Long Form: SYSTem:COMMunication:GPIB:ADDRess<int_value> 01 to 30

Description: Sets selected power supply GPIB address.

B.68 SYSTem:COMMunication:GPIB:ADDRess? QUERY SYST:COMM:GPIB:ADDR?

Syntax**:** Short Form: SYST:COMM:GPIB:ADDR?
Long Form: SYSTem:COMMunication:GPIB:ADDRess?
Return Value: <int_value> (GPIB address)

Description: **Returns GPIB address.**

B.69 SYSTem:COMMunication:LAN:AUTO COMMAND SYST:COMM:LAN:AUTO

[E-SERIES MODELS ONLY]

Syntax**:** Short Form: SYST:COMM:LAN:AUTO {ON | OFF}
Long Form: SYSTem:COMMunication:LAN:AUTO {ON | OFF}

Description: **Specifies the AUTO MODE.** When set to OFF, the static addresses set by the user from the front panel or using SYST:COMM:LAN commands are used for LAN communication. When set to ON, the unit obtains a valid IP address automatically.

B.70 SYSTem:COMMunication:LAN:AUTO? QUERY SYST:COMM:LAN:AUTO?

[E-SERIES MODELS ONLY]

Syntax: Short Form: SYST:COMM:LAN:AUTO? Long Form: SYSTem:COMMunication:LAN:AUTO?
Return Value: {ON | OFF}

Description: Indicates whether AUTO IP is set to ON or OFF.

B.71 SYSTem:COMMunication:LAN:DHCP COMMAND SYST:COMM:LAN:DHCP

[E-SERIES MODELS ONLY]

Syntax**:** Short Form: SYST:COMM:LAN:DHCP {ON | OFF}
Long Form: SYSTem:COMMunication:LAN:DHCP {ON | OFF}

Description: **Specifies the DHCP MODE.** When set to OFF, the static addresses set by the user from the front panel or using SYST:COMM:LAN commands are used for LAN communication. When set to ON, the unit queries the DHCP server for a valid IP address.

B.72 SYSTem:COMMunication:LAN:DHCP? QUERY SYST:COMM:LAN:DHCP?

[E-SERIES MODELS ONLY]

Syntax: Short Form: SYST:COMM:LAN:DHCP? Long Form: SYSTem:COMMunication:LAN:DHCP?
Return Value: {ON | OFF}

Description: **Indicates whether DHCP is set to ON or OFF.**

B.73 SYSTem:COMMunication:LAN:DNS COMMAND SYST:COMM:LAN:DNS

[E-SERIES MODELS ONLY]

Syntax**:** Short Form: SYST:COMM:LAN:DNS {w,x,y,z}
Long Form: SYSTem:COMMunication:LAN:DNS {w,x,y,z}
where w, x, y, and z are numbers from 0 to 255

Description: **Specifies the DNS address when DHCP MODE (see PAR. B.69) is set to OFF.** (Note that DNS address must be comma-separated.)

B.74 SYSTem:COMMunication:LAN:DNS? QUERY SYST:COMM:LAN:DNS?

[E-SERIES MODELS ONLY]

Syntax: Short Form: SYST:COMM:LAN:DNS? Long Form: SYSTem:COMMunication:LAN:DNS?
Return Value: <int_value> {w,x,y,z}
where w, x, y, and z are numbers from 0 to 255

Description: **Indicates the static DNS address (comma separated).** For example, DNS address of 4.4.4.1 returns 4,4,4,1.

B.75 SYSTem:COMMunication:LAN:GATE COMMAND SYST:COMM:LAN:GATE

[E-SERIES MODELS ONLY]

Syntax**:** Short Form: SYST:COMM:LAN:GATE {w,x,y,z}
Long Form: SYSTem:COMMunication:LAN:GATE {w,x,y,z}
where w, x, y, and z are numbers from 0 to 255

Description: **Specifies the GATE address when DHCP MODE (see PAR. B.69) is set to OFF.** (Note that GATE address must be comma-separated.)

B.76 SYSTem:COMMunication:LAN:GATE? QUERY SYST:COMM:LAN:GATE?

[E-SERIES MODELS ONLY]

Syntax: Short Form: SYST:COMM:LAN:GATE? Long Form: SYSTem:COMMunication:LAN:GATE?
Return Value: <int_value> {w,x,y,z}
where w, x, y, and z are numbers from 0 to 255

Description: **Indicates the static GATE address (comma separated).** For example, GATE address of 0.0.0.0 returns 0,0,0,0.

B.77 SYSTem:COMMunication:LAN:IP COMMAND SYST:COMM:LAN:IP

[E-SERIES MODELS ONLY]

Syntax**:** Short Form: SYST:COMM:LAN:IP {w,x,y,z}
Long Form: SYSTem:COMMunication:LAN:IP {w,x,y,z}
where w, x, y, and z are numbers from 0 to 255

Description: **Specifies the IP address when DHCP (see PAR. B.69) is set to OFF.** (Note that IP address must be comma-separated.)

B.78 SYSTem:COMMunication:LAN:IP? QUERY SYST:COMM:LAN:IP?

[E-SERIES MODELS ONLY]

Syntax: Short Form: SYST:COMM:LAN:IP? Long Form: SYSTem:COMMunication:LAN:IP?
Return Value: <int_value> {w,x,y,z}
where w, x, y, and z are numbers from 0 to 255

Description: **Indicates the static IP address (comma separated).**

B.79 SYSTem:COMMunication:LAN:MAC? QUERY SYST:COMM:LAN:MAC?

[E-SERIES MODELS ONLY]

Syntax: Short Form: SYST:COMM:LAN:MAC? Long Form: SYSTem:COMMunication:LAN:MAC?
Return Value: xx,xx,xx,xx where xx = hex character pairs

Description: **Return the MAC address of the unit.** For example, MAC address of 01-23-45-67-89-AB returns 01,23,45,67,89,AB.

B.80 SYSTem:COMMunication:LAN:MASK COMMAND SYST:COMM:LAN:MASK

[E-SERIES MODELS ONLY]

Syntax**:** Short Form: SYST:COMM:LAN:MASK {w,x,y,z}
Long Form: SYSTem:COMMunication:LAN:MASK {w,x,y,z}
where w, x, y, and z are numbers from 0 to 255

Description: **Specifies the MASK address when DHCP (see PAR. B.69) is set to OFF.** (Note that MASK address must be comma-separated.)

B.81 SYSTem:COMMunication:LAN:MASK? QUERY SYST:COMM:LAN:MASK?

[E-SERIES MODELS ONLY]

Syntax: Short Form: SYST:COMM:LAN:MASK? Long Form: SYSTem:COMMunication:LAN:MASK?
Return Value: <int_value> {w,x,y,z}
where w, x, y, and z are numbers from 0 to 255

Description: **Indicates the static MASK address (comma separated).** For example, mask of 255.255.255.0 returns 255,255,255,0.

B.82 SYSTem:COMMunication:SERial:BAUD COMMAND SYST:COMM:SER:BAUD

[STANDARD MODELS ONLY]

Syntax**:** Short Form: SYST:COMM:SER:BAUD {38400 | 19200 | 9600 | 4800 | 2400}
Long Form: SYSTem:COMMunication:SERial:BAUD {38400 | 19200 | 9600 | 4800 | 2400}

Description: **Sets the unit to operate at the specified baud rate.** Factory default is 38400.

B.83 SYSTem:COMMunication:SERial:BAUD? QUERY SYST:COMM:SER:BAUD?

[STANDARD MODELS ONLY]

Syntax: Short Form: SYST:COMM:SER:BAUD?
Long Form: SYSTem:COMMunication:SERial:BAUD?
Return Value: <int_value> {38400 | 19200 | 9600 | 4800 | 2400}

Description: **Indicates baud rate.**

B.84 SYSTem:COMMunication:SERial:ECHO COMMAND SYST:COMM:SER:ECHO

[STANDARD MODELS ONLY]

Syntax**:** Short Form: SYST:COMM:SER:ECHO {1 | 0} (1 = on, enabled, 0 = off, disabled)
Long Form: SYSTem:COMMunication:SERial:ECHO {1 | 0} (1 = on, enabled, 0 = off, disabled)

Description: **Enables (1 = ON) or disables (0 = OFF) echo mode** (see PAR. 8.2.2.1). When echo mode is ON causes all subsequent characters to be echoed back. When echo mode is OFF, turns off the character echo after the nest line terminator character. The *RST command has no effect on echo status

B.85 SYSTem:COMMunication:SERial:ECHO? QUERY SYST:COMM:SER:ECHO?

[STANDARD MODELS ONLY]

Syntax**:** Short Form: SYST:COMM:SER:ECHO?
Long Form: SYSTem:COMMunication:SERial:ECHO?
Return Value: <int_value> (01_H = on, enabled or 00_H = off, disabled)

Description: **Indicates whether echo mode is on (01_H) or off (00_H).** (See PAR. 8.2.2.1.)

B.86 SYSTem:COMMunication:SERial:ENABle COMMAND SYST:COMM:SER:ENAB

[STANDARD MODELS ONLY]

Syntax**:** Short Form: SYST:COMM:SER:ENAB {1 | 0} (1 = on, enabled, 0 = off, disabled)
Long Form: SYSTem:COMMunication:SERial:ENAB {1 | 0} (1 = on, enabled, 0 = off, disabled)

Description: **Enables (1 = ON) or disables (0 = OFF) transmission of data via the RS 232 port** (see PAR. 8.2.2) Once RS 232 transmission has been disabled, it can be enabled again by either turning the unit off, then on, or by sending SYST:COMM:SER:ENAB 1 via the GPIB port.

B.87 SYSTem:COMMunication:SERial:ENABle? QUERY SYST:COMM:SER:ENAB?

[STANDARD MODELS ONLY]

Syntax**:** Short Form: SYST:COMM:SER:ENAB?
Long Form: SYSTem:COMMunication:SERial:ENAB?
Return Value: <int_value> (01_H = on, enabled or 00_H = off, disabled)

Description: **Indicates whether RS 232 transmission mode is enabled (01_H = ON) or disabled (00_H = OFF).** (See PAR. 8.2.2.)

B.88 SYSTem:COMMunication:SERial:PACE COMMAND SYST:COMM:SER:PACE

[STANDARD MODELS ONLY]

Syntax**:** Short Form: SYST:COMM:SER:PACE {NONE | XON}
Long Form: SYSTem:COMMunication:SERial:PACE {NONE | XON}

Description: **Enables (XON) or disables (NONE) data flow control via the serial interface** (see PAR. 8.2.2.3)

B.89 SYSTem:COMMunication:SERial:PACE? QUERY SYST:COMM:SER:PACE?

[STANDARD MODELS ONLY]

Syntax**:** Short Form: SYST:COMM:SER:PACE?
Long Form: SYSTem:COMMunication:SERial:PACE?
Return Value: <int_value> (01_H = on, XON enabled or 00_H = off, XON disabled or NONE)

Description: **Identifies whether data flow control via the serial interface is enabled (01_H = XON) or disabled (00_H = NONE)** (see PAR. 8.2.2.3).

B.90 SYSTem:COMMunication:SERial:PROMpt COMMAND SYST:COMM:SER:PROM

[STANDARD MODELS ONLY]

Syntax**:** Short Form: SYST:COMM:SER:PROM {1 | 0} (1 = on, enabled, 0 = off, disabled)
Long Form: SYSTem:COMMunication:SERial:PROMpt {1 | 0} (1 = on, enabled, 0 = off, disabled)

Description: **Enables (1 = ON) or disables (0 = OFF) prompt** (see PAR. 8.2.2.2). When prompt is ON, causes the unit to return > character after the command is parsed.

B.91 SYSTem:COMMunication:SERial:PROMpt? QUERY SYST:COMM:SER:PROM?

[STANDARD MODELS ONLY]

Syntax**:** Short Form: SYST:COMM:SER:PROM?
Long Form: SYSTem:COMMunication:SERial:PROMpt?
Return Value: <int_value> (01_H = on, enabled or 00_H = off, disabled)

Description: **Indicates whether prompt is enabled (01_H = ON) or disabled (00_H = OFF)** (see PAR. 8.2.2.2).

B.92 SYSTem:ERRor[:NEXT]? QUERY SYST:ERR?

Syntax**:** Short Form: SYST:ERR[:NEXT]? Long Form: SYSTem:ERRor[:NEXT]?
Return Value: <int_value,string>

Description: **Posts error messages to the output queue.** Returns the next error number followed by its corresponding error message string from the instrument error queue. The error queue is a FIFO (first in first out) buffer that stores errors as they occur. As it is read, each error is removed from the queue and the next error message is made available. When all errors have been read, the query returns 0,"No error". If more than 31 errors are accumulated, it will overflow. The oldest errors stay in the queue but the most recent errors are discarded. The last error in the queue will be -350,"Too many errors." Error messages are defined in Table B-5.

NOTE: While the parser follows strict SCPI protocol for outgoing data, it is designed to be more forgiving of incoming data. As a consequence, all IEEE 488.2 commands and queries (see Table A-1) are evaluated in multiple tables, potentially resulting in multiple, consecutive occurrences of the error code "-113, undefined header" in the error queue."

B.93 SYSTem:ERRor:CODE[:NEXT]? QUERY SYST:ERR:CODE?

Syntax**:** Short Form: SYST:ERR:CODE[:NEXT]?] Long Form: SYSTem:ERRor:CODE[:NEXT]?
Return Value: nnn (nnn = 3 character error code)

Description: Returns the three character error code without the ASCII definition string. The error codes are defined in table B-5 (See example, Figure B-1.)

B.94 SYSTem:ERRor:CODE:ALL? QUERY SYST:ERR:CODE:ALL?

Syntax**:** Short Form: SYST:ERR:CODE:ALL?] Long Form: SYSTem:ERRor:CODE:ALL?
Return Value: comma-separated list of all error codes

Description: Returns a comma-separated list of all error codes. A maximum of 15 codes will be returned; if the queue is empty, the power supply returns 0.

B.95 SYSTem:KLOCk COMMAND SYST:KLOC

Syntax**:** Short Form: SYST:KLOC {1 | 0} (1 = on, locked, 0 = off, unlocked)
Long Form: SYSTem:KLOCk {1 | 0} (1 = on, locked, 0 = off, unlocked)

Description: **Locks (ON or 1) or unlocks (OFF or 0) the front panel controls.** When set to OFF, local lockout is disabled and the power supply enters Remote mode as soon as a command or query is received. Once in Remote mode (with local lockout disabled) the front panel controls are disabled with the exception that local mode can be restored by pressing the VOLTAGE and CURRENT adjustment knobs at the same time.

After sending a SYST:KLOC ON (local lockout enabled) command, all front panel controls (including pressing VOLTAGE and CURRENT adjustment knobs at the same time to return to local mode) are disabled. The power supply is now in the "local lockout" state and the status display reads **`dRwl`**. Local lockout can also be entered by sending the GPIB <LLO> code (hex 11) to the power supply. The power supply remains in "local lockout" until a SYST:KLOC OFF command is received or the power supply is turned off, ANALOG I/O DIP switch position 5 is set to OFF, and then the power supply is turned on again. Related Commands: SYST:KLOCK?. (See example, Figure B-7.)

B.96 SYSTem:KLOCk? QUERY

SYST:KLOC?

Syntax: Short Form: SYST:KLOC? Long Form: SYSTem:KLOCk?
Return Value: 0 or 1 (1 = on, locked, 0 = off, unlocked)

Description: **Identifies whether keypad is locked or unlocked.** 0 = keypad unlocked, local operation possible by pressing LOCAL key. 1 = keypad locked, LOCAL key disabled, only remote operation possible. Related Commands: SYST:KLOCK. (See example, Figure B-7.)

B.97 SYSTem:LANGuage COMMAND

SYST:LANG

Syntax: Short Form: SYST:LANG {COMP | SCPI} Long Form: SYSTem:LANGuage {COMP | SCPI}

Description:**Determines whether unit responds to older command formats.** SYST:LANG SCPI is the factory default setting. SYST:LANG COMP allows Kepco power supplies to respond to older formats such as LIST:TIME for compatible operation with software written for earlier firmware, however this is not applicable to KLR.

NOTE: The following examples are generic to all models, assuming factory default settings.

SYST:VERS?	Returns xxxx.v (SCPI version number, i.e. 2003.0).
SYST:SET?	Returns DC0, STR, KL0, LF0, L250, M40, SCP
SYST:KLOC ON	Locks out local (front panel) controls; only remote control is possible.
SYST:KLOC?	Returns 1, indicating local lockout enabled.
SYST:KLOC OFF	Disables local lockout; see PAR. B.95 for instructions on reverting to local control.
SYST:PASS:CEN xxxx	Enables access to protected commands assuming password (xxxx) matches.
SYST:PASS:STAT?	Returns 1 indicating protected access is enabled.
SYST:PASS:CDIS xxxx	Disables access to protected commands assuming password matches.
SYST:PASS:STAT?	Returns 0 indicating protected access is disabled.
SYST:PASS:NEW xxxx,yyyy	Resets password to new value (yyyy) assuming present value (xxxx) matches.
SYST:ERR?	Returns error messages from instrument error queue sequentially using the format -xxx,"_____" where -xxx is the error code and "_____" is the ASCII definition string (see Table B-5 and PAR. B.92). Returns 0,"No error" when queue is empty.
SYST:ERR:CODE?	Returns error messages from instrument error queue sequentially using-xxx error code only (no ASCII definition string).
SYST:ERR:CODE:ALL?	Returns comma-separated list of all stored error messages (maximum of 15).

FIGURE B-7. USING SYSTEM COMMANDS AND QUERIES

B.98 SYSTem:LANGuage? QUERY

SYST:LANG?

Syntax: Short Form: SYST:LANG? Long Form: SYSTem:LANGuage?
Return Value: <string> SCPI or COMP

Description: **Identifies whether unit responds to older command formats.** SCPI means the unit will not respond to older formats such as LIST:TIME. COMP means the unit will respond to older formats such as LIST:TIME for compatible operation with software written for KLR power supplies up to Revision 1

B.99 SYSTem:PASSword:CENable COMMAND SYST:PASS:CEN

Syntax: Short Form: SYST:PASS:CEN <val> Long Form: SYSTem:PASSword:CENable <val>

Description: Sets the password enable state if the value matches the current password. This command allows other commands such as the CAL commands to operate. Table B-4 shows the factory default calibrationpasswords.

TABLE B-4. FACTORY DEFAULT CALIBRATION PASSWORDS

MODEL (** Standard or E-Series)	PASSWORD
KLR 20-120-**	2012
KLR 40-60-**	3660
KLR 75-32-**	7533
KLR 150-16-**	1516
KLR 300-8-**	3008

B.100 SYSTem:PASSword:CDISable COMMAND SYST:PASS:CDIS

Syntax: Short Form: SYST:PASS:CDIS <val> Long Form: SYSTem:PASSword:CDISable <val>

Description: Clears the password enable state if the value matches the current password.

B.101 SYSTem:PASSword:NEW COMMAND SYST:PASS:NEW

Syntax: Short Form: SYST:PASS:NEW <old password>,<new password>
Long Form: SYSTem:PASSword:NEW <old password>,<new password>

Description: **Establishes new password.** The old (current) password is checked, then replaced by the new password.

B.102 SYSTem:PASSword[:CENable]:STATe? QUERY SYST:PASS:STAT?

Syntax: Short Form: SYST:PASS[:CEN]:STAT? Long Form: SYSTem:PASSword[:CENable]:STATe?
Return Value: <int_value> 0 or 1 (1 = enabled, 0 = disabled)

Description: Returns a 1 if the password state is enabled or a 0 if it is disabled.

B.103 SYSTem:SECurity:IMMediate COMMAND SYST:SEC:IMM

Syntax: Short Form: SYST:SEC:IMM Long Form: SYSTem:SECurity:IMMediate

Description: Initializes all NVRAM variable to factory defaults. This includes clearing all *SAV/*RCL locations, returning all relay functions to factory default settings, and setting Quick Boot feature to OFF. This command does not affect limit model and corresponding protection limits, last setting recall register or present operating conditions. NOTE: System password must be enabled prior to issuing this command (see PAR. B.99).

B.104 SYSTem:SET COMMAND SYST:SET

Syntax: Short Form: SYSTem:SET {CM0 | DC0 | DC1 | LF0 | LF1 | KL0 | KL1 | L250 | M40 | SCP}
Long Form: SYSTem:SET {CM0 | DC0 | DC1 | LF0 | LF1 | KL0 | KL1 | L250 | M40 | SCP}

Description: **Establishes Device Clear, Line Feed, and Reset functions. Sending SYST:SEC:IMM sets LF1, DC0, and RO0 (as if CM0 was sent).** Sending SYST:SET CM1 sets the KLR to operate in compatible mode and have all GPIB functions compatible with software version 2.9 and lower units. Sending SYST:SET CM0 sets the unit to be fully SCPI 1997 compliant.

- CM0 Establishes DC0, LF0, conditions described below (SCPI 1997 Standard compliance).
- DC0 Device Clear functions per IEEE 488.2 Standard. (No effect on the device (power supply), only clears internal status registers.)
- LF0 Line Feed not provided upon empty buffer condition (no effect when access is via LAN interface).
- DC1 Device Clear functions identical to *RST (Output set to 0V, voltage mode and output set to OFF except if RO1 (see below) is set.)
- LF1 Line Feed provided if buffer is empty and a read is performed. (no effect when access is via LAN interface).

KL0 Enables front panel controls.
KL1 Disables front panel controls.
L250 Number (250) of locations programmable using LIST commands.
M40 Number (40) of locations programmable using *SAV and *RCL commands.
SCP SCPI Programming Language.

B.105 SYSTem:SET? QUERY SYST:SET?

Syntax: Short Form: SYST:SET? Long Form: SYSTem:SET?
Return Value: <comma separated string>

Description: Returns comma-separated string indicating command functions in effect. See PAR. B.104 for command function definitions. (See Figure B-7 for example.)

B.106 SYSTem:VERSion? QUERY SYST:VERS?

Syntax: Short Form: SYST:VERS? Long Form: SYSTem:VERSion?
Return Value: <int_value>.<int_value> (YYYY.V)

Description: **Identifies SCPI Version implemented.** Returns SCPI Version number:
YYYY = year, V = Revision number for specified year. (See example, Figure B-7.)

B.107 TRIGger[:SEQuence]:SOURce COMMAND TRIG:SOUR

Syntax: Short Form: TRIG[:SEQ]:SOUR {EXT | BUS | IMM}
Long Form: TRIGger[:SEQuence]:SOURce {EXT | BUS | IMM}

Description: **Selects the active trigger source.** When TRIG:SOUR BUS is set, an event trigger is created by sending either *TRG or GPIB <GET> commands. When TRIG:SOUR EXT is set, the trigger is created by connecting the external trigger line (J2, pin 4 or pin 3 of the mating connector for J2) to analog signal ground (J2, pin 1 or J2 mating connector, pin 1). The trigger must be armed by sending either INIT:IMM or INIT:CONT enable (ON) commands prior to sending the selected trigger command. (see PAR. A.16). Receipt of the active trigger causes the power supply to program to the previously stored values of VOLT:TRIG and CURR:TRIG (actual output voltage/current is dependent upon the load). Sending TRIG:SOUR IMM causes the next VOLT:TRIG or CURR:TRIG to immediately program the output, regardless of trigger arming. *RST sets TRIG:SOUR to IMMediate. Related Commands: <GET>, *TRG, INIT:IMM, INIT:CONT.

B.108 TRIGger[:SEQuence]:SOURce? QUERY TRIG:SOUR?

Syntax: Short Form: TRIG[:SEQ]:SOUR? Long Form: TRIGger[:SEQuence]:SOURce?
Return Value: Return Value: <string> EXT or BUS or IMM

Description: **Returns string indicating trigger source.** EXT indicates trigger source is J2 pin 14. BUS indicates source is either *TRG or GPIB <GET> command IMM indicates source is VOLT:TRIG or CURR:TRIG command. See PAR. B.107 for details.

TABLE B-5. ERROR MESSAGES

ERROR MESSAGE	ESR ERROR BIT SET (SEE PAR. A.5)	EXPLANATION
0,“No error”	None	No error
-100,“Command error”	Command Error bit 5	Command and data understood, but more information included which is not recognized.
-102,”Syntax error”	Command Error bit 5	First 4 characters recognized, subsequent characters not recognized.
-103,”Invalid separator”	Command Error bit 5	For example, VOLT.PROT received instead of VOLT:PROT.
-108,”Parameter Not Allowed Error”	Command Error bit 5	Volt12 sequence, channel number is invalid.
-109,”Missing parameter”	Command Error bit 5	For example, VOLT instead of VOLT 21.
-111,”Header separator error”	Command Error bit 5	Missing space between volt and value or ; missing.
-113,”Undefined header”	Command Error bit 5	First 4 characters could not be identified as legal command.For example, command VLT instead of VOLT.
-120,”Numeric data error”	Command Error bit 5	Expected number but other characters were detected.
-121,”Invalid character in number”	Command Error bit 5	Volt 1,500 (comma not allowed).

TABLE B-5. ERROR MESSAGES (CONTINUED)

ERROR MESSAGE	ESR ERROR BIT SET (SEE PAR. A.5)	EXPLANATION
-123,"Exponent too large"	Command Error bit 5	Exponent E+3 or greater is invalid.
-141,"Invalid character data"	Command Error bit 5	For example OUTP OFD or OUTP STOP instead of OUTP OFF.
-150,"String data error"	Command Error bit 5	Invalid characters were detected in numeric entry.For example E.1 instead of E+1 or 4d3 instead of 4.3.
-203,"Command protected"	Execution Error bit 4	Password must be CENabled.
-207,"Location is empty"	Execution Error bit 4	Recalled location has no data stored.
-211,"Trigger Ignored"	Execution Error bit 4	Trigger ignored because output is off.
-213,"INIT ignored"	Execution Error bit 4	Request for measurement ignored because another measurement already in progress.
-221,"Settings conflict"	Execution Error bit 4	E.g. , Invalid password from syst:pass:cen command. or Calibration state not enabled but CALibrate command received.
-222,"Current, Voltage or Data out of range"	Execution Error bit 4	Value (current or voltage) exceeds power supply rating or (data) exceeds acceptable command parameters.
-223,"Data format error"	Execution Error bit 4	Multiple decimal points, Multiple exponents, etc.
-224,"Illegal parameter value"	Execution Error bit 4	For example, OUTP 2 instead of OUTP 1.
-226,"Lists not same length"	Execution Error bit 4	During a LIST command, number of DWEL list entries was not equal to 1 and did not match number of LIST:VOLT or LIST:CURR entries.
-280,"Program Error"	Execution Error bit 4	LIST execution error, e.g., program already running.
-301,"Value bigger than limit"	Device Error bit 3	E.g., requesting a voltage or current that exceeds the limit.
-302, "Mode changed to Voltage" -302, "Mode changed to Current"	Device Error bit 3	Power supply operating in current mode changed to voltage mode. Power supply operating in voltage mode changed to current mode.
-303, "Output Lead Fault"	Device Error bit 3	Excessive voltage drop detected between power and sense lead.
-304, "Current Protection Fault"	Device Error bit 3	Current exceeded programmed protection value.
-305, "Voltage Protection Fault"	Device Error bit 3	Voltage exceeded programmed protection value.
-306, "Over Temperature Shutdown"	Device Error bit 3	Internal temperature exceeded thermal protection limit.
-307, "Power Loss Fault"	Device Error bit 3	Source power loss occurred.
-308, "Fan Fault"	Device Error bit 3	Fan fault detected.
-310,"System Failure"	Device Error bit 3	Program checksum incorrect.
-311,"Memory Error"	Device Error bit 3	E.g., power-up NV RAM error, or CALibrate:STORe error.
-313,"Calibration Memory Lost"	Device Error bit 3	No valid calibration found.
-314,"Save/recall memory error"	Device Error bit 3	Using cell other than 1 - 40 for SAV and RCL.
-315,"Configuration Memory Error"	Device Error bit 3	Configuration Memory Error.
-330,"Self Test Failed"	Device Error bit 3	Self test failed.
-340,"Calibration Failure"	Device Error bit 3	Unable to perform auto calibration.
-341,"Non Volatile Mem. CRC error"	Device Error bit 3	Power supply constants may be corrupted; recalibration may be necessary.
-350,"Queue Overflow"	Device Error bit 3	More than 15 errors are in queue. Error queue was full, error events have been lost.
-410,"Query interrupted"	Query Error bit 2	New command sent before data from previous query read. Previous query data lost.
-430,"Query Deadlocked"	Query Error bit 2	Over 255 characters received in single input string.