ECAT PROGRAMMERS MANUAL

Revision 1.0 – 22 May, 1998

Keytek
One Lowell Research Center
Lowell, MA 01852-4345
Phone 978-275-0800
Fax 978-275-0850

www.keytek.com

PRODUCT SAFETY

DO NOT attempt to operate the ECAT and a computer without the FC-11 interface. Surge events can cause transient differences in potential between the ECAT and computer, causing common-mode current to flow down a copper, RS232 link. Such currents may cause disruption of computer operation or damage to computer equipment.

NEVER add modules the ECAT system, remove modules from the system, or swap modules within a system while the ECAT is powered. Doing so can result in damage to the ECAT system.

READ AND FOLLOW the instructions in the ECAT manual for safe connection between the ECAT and Equipment Under Test.

Always assume that – during testing -- all ECAT outputs are hazardous, or may become hazardous at any time.

Always assume that the maximum voltage and energy of a surge module may be delivered in each pulse; use wiring and probes rated for twice the maximum surge voltage.

NOTICE:

The user should be aware that some external commands can reconfigure the unit or render it out of calibration, and he should not experiment beyond the programming processes described in this document.

Introduction

This manual presents a detailed description of how to program ECAT controllers to perform surge testing, using an external computer.

This manual briefly describes the ECAT hardware, then describes in details the ECAT instruction set, the use of the instructions, and their interactions. It gives examples on using these instructions to perform simple to complex ECAT surge routines.

It is assumed that the reader of this manual is familiar with programming a computer, with the use of a programming language to access a computer serial port, and with the software requirements of initializing and using a hardware serial port. The manual does not give programming examples or code for the computer; rather, it details the instructions that the ECAT Controller expects to see transmitted (and the responses that the ECAT Controller generates).

The ECAT Controller acts as the final arbitrator of all commands sent to it, determining what are valid instructions and what are not. The Controller will block any instructions which attempt to program actions which are impossible. For this reason, you can safely program the ECAT and the Controller will prevent damaging conditions.

However, note that it is the nature of the ECAT to generate dangerous – lethal – voltages and currents. It is quite possible for an error in your program -- a typographical error or a programming error -- to cause a higher than expected surge voltage to be generated, or to deliver the surge to a port other than that which you intended.

Read and follow the instructions in the ECAT manual for safe connection between the ECAT and Equipment Under Test.

Always assume that – during testing -- all ECAT outputs are hazardous, or may become hazardous at any time.

Always assume that the maximum voltage and energy of a surge module may be delivered in each pulse; use wiring and probes rated for twice the maximum surge voltage.

The ECAT Hardware

While not required to start programming an ECAT, a brief description of the ECAT hardware is useful, giving some basics which may prevent confusion.

Minimum Hardware Configuration

The following is the minimum set of components required to allow programming an ECAT from an external computer.

- Keytek ECAT system
- At least one Keytek ECAT surge module
- KeyTek FC-11 Fiber-Com interface
- Personal Computer with serial port; Serial port must be set to:
    - 2400 baud
    - no parity
    - 8-bits
    - 1-stop bit

The ECAT Chassis

An ECAT system consists of a controller mounted atop one or more ECAT instrument bays. ECAT modules – surge modules, couplers, EFT modules, etc. – are installed into the instrument bays. ECAT modules may be half-wide modules -- using half of an available bay – or full-wide modules which occupy an entire bay.

Each ECAT instrument bay is assigned two location numbers to support two, half-wide modules; one number for the right-side of the bay, and one for the left-side of the bay. The numbering starts at the top-right of a stack of bays, and increases going down the stack.

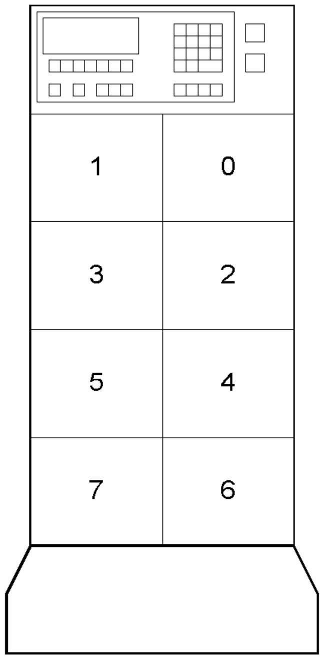

Figure 1 -- ECAT Bay Assignments

An ECAT module is assigned a number corresponding to the bay it occupies. A full-wide module assumes the number of the right-side bay.

A maximum of eight bays can be supported in an ECAT system, giving module numbers from zero to fifteen.

All bays are interlocked for safety. All bays must have a module installed, or the system will not allow the AC mains to be switched to the output, and will not allow the application of high-voltage. In addition, all ECAT modules have protective barriers – hinged front panels – to prevent access to hazardous voltages. All barriers must be in place -- closed – for the ECAT controller to allow the AC mains to be switched to the output, or the application of high-voltage.

Coupler modules take an AC mains as input, and provide an AC mains with surge as output. The purpose of the coupler is to couple the surge onto the test mains output, and decouple the surge from the test mains input, preventing the surge from getting onto the building mains and causing interference or disruption of other equipment. For safety, the AC mains for the coupler is positively-interlocked, requiring user intervention to activate. The AC mains for the coupler cannot be turned on by computer; first, the AC mains must be enabled, either from the VFP or computer, then the operator must toggle the blue, EUT power button.

The ECAT Controller

The ECAT Controller contains the power supply for the ECAT chassis, a high-voltage supply shared by all modules, a peak detect board for measurements, and a microprocessor. The microprocessor includes firmware, which holds lists of all possible modules, all features of the modules, and all options which may be installed in the modules. The microprocessor checks all commands sent to a module to verify that they are possible and reasonable; any illegal commands are rejected and an error message is returned.

The microprocessor controls timing and sequencing within the Surge modules and coupler modules, and controls communications with an external computer.

The ECAT Communications Hardware

The ECAT communicates with a computer using the optional Fiber-Com interface -- FC-11. This device provides optical isolation between ECAT and computer, assuring the safety of an operator and protection of the computer equipment.

DO NOT attempt to operate the ECAT and a computer without the FC-11 interface. Surge events can cause transient differences in potential between the ECAT and computer, causing common-mode current to flow down a copper, RS232 link. Such currents may cause disruption of computer operation or damage to computer equipment.

The FC-11 draws power from the serial port of the computer. In rare instances, if the computer uses a low-voltage RS-232 interface, an auxiliary supply may be required (Keytek part number 02-701-695-00 {USA} or 02-050-415-00 {European}).

The ECAT requires that the computer serial port be set to 2400 baud, no parity, 8-bits, 1-stop bit.

ECAT Surge Modules

An ECAT surge module consists of polarity relays, an energy storage capacitor, pulse-shaping network, surge trigger relay, and an optional VI monitor board. Also included is a non-volatile memory device which holds calibration and configuration information required by the ECAT controller.

During the surge event, the charged energy-storage capacitor is discharged through the pulse-forming network, generating a precise, high-voltage surge.

The timing and sequence of all surges events are set by the ECAT controller. A rough sequence of events is this:

- Unlock safety dump relay in the Surge module to allow charging.
- Program the polarity of the surge using polarity relays in the Surge module.
- Set the level of the high-voltage supply within the Controller.
- The Controller calculates the time required to charge the capacitor, and waits this time while giving the user a count down.

- When the capacitor is charged, the Controller disconnects the high-voltage supply from the energy storage capacitor – preparing for the surge – and notifies the user that the system is ready.
- When the user presses the surge switch, the Controller calculates timing (if required for line synchronization), sets surge module relays and coupler relays to deliver the surge to the proper lines (as set by coupling mode), then fires the surge trigger relay to generate the surge. (The user must press a switch to fire the surge, or the computer must send a surge-fire command. If a command is not issued within a given time, the Controller resets and safes the system -- discharges the energy storage capacitor and sets the high-voltage supply to zero.)
- After the surge event the Controller returns all relays to a safe state, isolating the surge module from the surge output.

Some surge modules provide a number of different waveforms, varying in amplitude, impedance, or waveshape.

For proper operation, the user must:

- Select a waveform.
- Select voltage (and polarity).
- Select the pulse destination (coupling).
- Provide a start-charging command (button press or software command).
- Provide a fire command (button press or software command) to fire the surge.

ECAT Couplers

Mains Coupler modules take an AC mains as input, and provide an AC mains with surge as output. The purpose of the coupler is to couple the surge onto the test mains output, and decouple the surge from the test mains input, preventing the surge from getting onto the building mains and causing interference or disruption of other equipment.

Data Couplers, i.e. the E571, take an analog or digital signal line as input, and output the signal line with surge. The Data Coupler isolated the surged line from the input line to prevent damage to other equipment.

All Couplers functions are set by the ECAT Controller, as with the Surge modules. Couplers have minimum information stored in non-volatile memory: simply a list of installed options.

Mains Couplers receive a differential voltage from the surge module and are programmed by setting one or more mains phases to receive the surge high, and one line to receive the surge low. The rules for coupling modes are as follows:

- At least one line must be set high.
- PE may never be set high.
- One line must be set low.
- Only one line may be set low.
- L1 may never be set low.

Three-phase Coupler, Coupling Modes

L1	L2	L3	N	PE
High	Low			
High		Low		
High			Low	
High				Low
	High	Low		
	High		Low	
	High			Low
	Low	High		
		High	Low	
		High		Low
	Low		High	
		Low	High	
			High	Low
High	High	Low		
High	High		Low	
High	High			Low
High	Low	High		
High		High	Low	
High		High		Low
High	Low		High	
High		Low	High	
High			High	Low
	High	High	Low	
	High	High		Low
	High	Low	High	
	High		High	Low
	Low	High	High	
		High	High	Low
High	High	High	Low	
High	High	High		Low
High	High	Low	High	
High	High		High	Low
High	Low	High	High	
High		High	High	Low
	High	High	High	Low
High	High	High	High	Low

One-phase Coupler, coupling modes

L1	N	PE
High		Low
	High	Low
High	Low	
High	High	Low

Figure 2 -- Mains Coupler Coupling Modes

Programming Basics

The ECAT Communications Protocol

The ECAT Controller communicates over an RS-232 serial link, using plain ASCII text, in a format similar to that used in IEEE-488 instruction strings.

In the IEEE-488 format, instructions are in a tree-like structure, starting with ":" as the root. Thus all surge control instructions begin with ":SRG" and continue with further arguments. The ECAT Controller also accepts the standard IEEE-488 control commands of: ***IDN?. *OPC?**, ***RST**, and ***TRG**.

Instructions to the ECAT should be sent one instruction at a time, separated by the newline key. The ECAT does not support stringing together many commands on one line, separated with semicolons.

ECAT instruction are not case sensitive; they may be sent in lower case, upper case, or a mixture of both. In this document, instructions are shown with required characters in upper case, optional characters in lower case. Thus the command:

:BAY:NAme? <nbr>

indicates that BAY must be spelled out: name may be spelled out, or the first two letter, NA, will suffice as an abbreviation for the command;

:BAY:NAme? and **:BAY:NA?**

are equivalent.

All commands sent to the ECAT are echoed back to the computer; many commands – but not all – will also generate a response; check the list of commands in the appendix when there are questions as to whether a response will be generated. All responses from the ECAT will be enclosed in square brackets – "[" and "]".

Responses from an ECAT may vary slightly between different versions of the Controller firmware. With returned numbers this is little problem. With strings, it may present a problem if you attempt a direct comparison between a returned string and an argument in your program. To avoid stumbling over such problems, you should:

- Trim white space before and after the returned strings.
- Force capitalization of the returned string before doing a comparison.

When sending commands, where a response may or may not be returned, it is useful to check the serial port receive queue for the number of characters received, and if characters are in the queue, to read the characters and check for the string, "ERR", within the returned characters. If there has not been an error, the characters can be discarded.

Several commands and queries to the ECAT Controller will return a delayed response. Typically this will result in the receipt of the "[" character, a noticeable delay, and the receipt of the remainder of the string. When writing code to interpret ECAT return strings, the code should look for a complete bracket set, "[" and "]", as delimiters to the return. That is, your code should not attempt to parse the return until both brackets are in the return string.

In the event of an un-parsable command, the ECAT Controller typically responds with the message;

[(ERR)-COMMAND]

If a character received by the ECAT is outside the normal ASCII range of printing character (0 to 127 decimal) the following error message may be returned:

[(ERR)-CHAR]

If an instruction asks for data which does not exist, e.g. a delay time for a waveform which does not exist, or attempts to set a value outside of the allowed range, or attempts to program a function for which no hardware is installed to support, e.g. set a mains coupling mode without a coupler installed, the error message is typically:

[(ERR)-VALUE]

After any instruction is sent to the ECAT, the receiving queue of the serial port should be checked to see if an error occurred. If an error did occur, it should be assumed that the command failed, and the instruction should be re-transmitted. In all errors, the ECAT returns a string which contains “ERR”.

To get started with programming the ECAT, any terminal emulator may be used, such as TERMINAL.EXE under Windows 3.1 or Hypertrm.exe under Windows 95. Note however that terminal emulators may handle echoes differently, with some returning the echoed string and others suppressing the string.

The ECAT Controller is hardwired to a communications protocol of; 2400 baud, no parity, 8-bits, 1-stop bit.

ECAT Commands vs Queries

The instruction set of the ECAT consists of queries, which always return information, and commands, which cause actions within the ECAT and may or may not return information to the computer.

Many queries also have a command form. Thus the query to find the name of a module in a bay is:

:BAY:NAme? <nbr>

and this query has a command form which sets the name of a module within a bay;

:BAY:NAme <bay nbr> <bay name>.

The command form of this query is used for initial programming of a module – it has no practical use when attempting to program a surge. Using the command form changes the name of the module within the Configuration Table, making it difficult for you to identify the module in the future.

All command forms of instructions used for factory programming of modules have been deleted from the appendix of ECAT commands.

The ECAT Configuration Table

When the ECAT is first turned on, a hardware initialization routine is executed. During this routine the controller polls the ECAT instrument bays to determine what modules are installed and what options are installed in each module. The module information is stored within each module, in a non-volatile memory device. During initialization, the data is copied into the Controller, into a structure called the Configuration Table.

When programming the ECAT, only the copy of the module information stored within the Configuration Table is available to the computer. Any changes written into the Configuration Table will be lost when the ECAT is turned off.

Since the data is copied only once, at power up, it is important that modules never be added to the system, removed from the system, or swapped within a system while the ECAT is powered. Doing so can result in damage to the ECAT system.

The Configuration Table is used by the controller to determine what operations are possible with each module. The Configuration Table also holds information which can be useful in programming and reporting.

Identifying an ECAT

The ECAT controller will respond to an *IDN? query by returning the following string:

[KeyTek Instrument,ECAT,yymmddd,XXYY]

where the last set of characters indicate the firmware revision.

Retrieving Module Information

The ECAT Configuration Table holds useful information such as:

- The module name,
- The module serial number,
- The number of waveforms supported, with a description of each, and,
- The minimum charge time required for each waveform.

The module name is useful as a cross-check that the module you are attempting to program actually exists in your system, and that you are attempting to program the proper module. It can also be used to locate a module within an ECAT system, if the module might be moved within the ECAT rack, and found in different instrument bays.

The command to read a module name is:

:BAY:NAme? < bay>

where < bay> is replaced with the number of the bay of interest, from 0 to 15.

If a module exists in that bay, the Controller will respond with the name of the module, e.g.

[E501]

or, If no module exists in the bay, the Controller will respond with,

[E000]

The module serial number can be used in reports, or used as a tracking aid. The command to read a modules serial number is:

:BAY:SErial? < bay>

where < bay> is replaced with the number of the bay of interest, from 0 to 15.

If a module exists in that bay, the Controller will respond with the serial number of the module, e.g.

[9705102]

or, If no module exists in the bay, the Controller will respond with,

[0]

The number of waveforms within a module can be read with the Waveform query:

:BAY:WAveform? <bay> 0

which returns the number of waveforms. A full set of information on each waveform can then be read using the Waveform query and passing the wave number:

:BAY:WAveform? <bay> <wave nbr>

For example, with an E502A in bay zero, the following conversation might take place (responses from the ECAT are in square brackets – see the appendix for an explanation of each field):

```
:BAY:NAME? 0
    [E502A]
:BAY:SERIAL? 0
    [9706123]
:BAY:WAVEFORM? 0 0
    [3]
:BAY:WAVEFORM? 0 1
    [3 1 0 0 0  6600  0 0 18 0 0 , 6kv,  0.5/700 Exponential]
:BAY:WAVEFORM? 0 2
    [3 1 0 0 0  6600 0 4400 18 0 18 , 5kv, 100/700 Exponential]
:BAY:WAVEFORM? 0 3
    [3 1 0 0 0 5500 0 0 18 0 0 , 5kv, 100/700 Exponential]
```

Example 1 -- Querying Waveform Information

The minimum charge time for any waveform can be returned using the Delay query. This information is also available in the information packet returned via the Waveform query, but the Delay query returns the information in a format which does not require extended parsing.

:BAY:DELay? <bay> <wave>

returns a number;

[18]

Polling an ECAT System

The ECAT controller polls the entire ECAT chassis at power up. This is a useful procedure for any program, to allow for modules which may be moved within an ECAT, and to allow the user of the program to select any of the modules available.

Polling may be simply done by using one of more of the queries discussed above, and repeating them for every bay. The returned information can then be placed in an array and presented to the user on request.

Checking ECAT Status

Since the state of the ECAT system can be changed at anytime by an operator, software to operate the ECAT should check that an operator has not changed anything which might invalidate a test.

Before performing any operation the software should check the state of the interlocks. The query;

:SYStem:ILock?

will return 0 if the interlocks are OK: 1 if an interlock is open. If an interlock is open the query:

:SYStem:IText?

will return a description of the interlock fault. This description might be useful to the operator, to clear the fault.

If the state of the EUT power is important, then before performing any operation the software should check the state of the EUT power. The query;

:EUT?

will return 0, 1, or 2. Two, indicates that the EUT power is enabled (in software) and on; one, indicates that the EUT power is enabled and off; zero, indicates that the EUT power is disabled and off.

When the ECAT is performing a surge, the *OPC? instruction returns the current status of the ECAT within the surge sequence. The command:

***OPC?**

will return a number between 0 and 3; 0 indicates that the unit is idle and a surge can be programmed; 1 indicated that the ECAT is charging for a surge; 2 indicates that the surge is ready to be fired; any other return indicated that the ECAT is not ready for a surge (see the appendix for full details on the *OPC? query).

Enabling the ECAT, EUT AC Mains

Turning on the EUT AC Mains, providing AC power to equipment under test, is a two-step process. The EUT mains must be enabled via a software command, then switched on manually via the blue button on the front of the ECAT. The command to enable the EUT mains is:

:EUT 1

Turning off the EUT AC Mains by the computer simply requires sending the :EUT command with argument of zero:

:EUT 0

The operator can always enable the EUT mains from the VFP, surge menu, then switch on the blue button on the front of the ECAT – eliminating the need to program this step.

Programming Surges

Programming a surge is a straight-forward process; it requires setting each of the surge parameters, very similar to programming a surge from the ECAT VFP.

The basic procedure can be described this way:

- Choose the surge module to use.
- Choose the waveform of the surge module to use.
- Choose a destination (coupling) for the surge.
- Set a surge voltage.
- Start the charge process.
- Wait until the charge is complete.
- Send a trigger command.

The order in which commands are issued is important; e.g. if you chose a waveform before choosing a surge module, you might cause an error if the waveform did not exist in the default (current) surge module -- or get unexpected results if the waveform does not exist in the surge module you select after setting the waveform (the latter condition does not generate an error).

It is also important that all of the steps be performed, at least once. If you do not choose a surge module or a coupler, if you simply start a charge cycle, the ECAT Controller will use the <u>current</u> surge module and <u>current</u> coupler, i.e. the last surge module and coupler used. If no surge module and coupler have yet been used, the ECAT Controller uses the <u>default</u> Surge module and coupler, i.e. the first surge module and the first coupler found when polling the system at power-up. Neither of these may be what you intend to do, so every time the program is run, a surge module, waveform, etc. should be selected. If it is possible for the controlling software to be paused and restarted, and if an operator may change the settings of the ECAT via the VFP while the software is paused, then it would be reasonable to set all of the parameters before each surge.

The steps to programming a surge are described in detail on the following pages.

Choose the surge module

You select a surge module using the :SRG:NETWORK command, and passing a bay number;

:SRG:NEtwork <bay>

where the bay address is between 0 and 15. If the bay selected is not a surge module, the controller returns ERR—VALUE as a response.

If you know the bay in which your surge network resides – and if the location of the network is unlikely to change – then you can 'hardwire' a bay number into your code. In this case it would be reasonable to check the identity of the module you are about to program with the :BAY:NAME? query, and compare the return value against the name of the module you want to use.

However, it is also reasonably simple to poll the ECAT system, identify each module, find the module you require, and use the bay number of the module in all further communications.

The current network may be found by using the :SRG:NETWORK? query.

Choose the waveform of the surge module

Many surge modules have more than one waveform available. You select a surge waveform to use with the :SRG:WAVEFORM command, passing an argument between 1 and 5:

:SRG:WAveform <wave>

The waveform must exist, or the ECAT Controller returns an ERR—VALUE response. If you do not select a waveform, the ECAT Controller will default to waveform one.

You can retrieve the number of waveforms available in a surge module, as well as a brief description of the waveforms by using the :BAY:WAVEFORM? query; see, Checking ECAT Status, on page 14.

The current waveform may be found by using the :SRG:WAVEFORM? query.

Choose a destination (coupling)

You select a surge destination using the :SRG:OUTPUT and the :SRG:COUPLING commands. Use the :SRG:OUTPUT to select whether to send the surge to the surge module front panel or to a coupler; if you select a coupler, use the :SRG:COUPLING command to select the coupling mode [to the mains].

The :SRG:OUTPUT command requires a single argument, which is the number of a bay with a coupler, or is 255 to route the surge to the module front panel.

:SRG:OUTput	<bay>	specifies a coupler, when <bay> is between 0 and 15
:SRG:OUTput	**255**	specifies the module front panel.

If you do not select an output, the destination defaults to the last used coupling mode, or to the front panel if no prior selection has been made.

The current output may be found by using the :SRG:OUTPUT? query.

Couplers receive a differential voltage from the surge module and the couplers deliver a differential voltage to their outputs. That is, surge modules and couplers provide a surge-high and a surge-low signal, where the full surge voltage is delivered from surge-high to surge-low

A Coupler is programmed by selecting one or more mains phases to receive the surge high, and one mains phase to receive the surge low. The rules for coupling modes are as follows:

- At least one line must be set high.
- PE may never be set high.
- One line must be set low.
- Only one line may be set low.
- L1 may never be set low.

The coupling mode is selected using the :SRG:COUPLING command, passing an integer argument for the surge-high coupling mode, and for the surge-low coupling mode:

:SRG:COupling <high> <low>

where:

<high>	::= High Coupling selection	(1 to 16)
<low>	::= Low Coupling selection	(2 to 16)

The coupling selection arguments are formed by assigning a number to each of the phases, as in the list below, and summing these numbers to form the argument:

1	L1
2	L2
4	L3
8	N
16	PE

Thus, to couple to L1 and L2 with respect to PE, the <high> argument would be 3 (1 plus 2), and the low argument would be 16. To couple to all lines with respect to PE, the <high> argument would be 15 and the low argument would be 16. Some examples of coupling arguments are given below:

L1 with respect to L2	:SRG:COUPLING 1 2
L1 with respect to PE	:SRG:COUPLING 1 16
L1 & N with respect to PE	:SRG:COUPLING 9 16
L1,L2,L3 with respect to PE	:SRG:COUPLING 7 16
L1,L2,L3,N with respect to PE	:SRG:COUPLING 15 16

The current coupling may be found by using the :SRG:COUPLING? query.

If the output is to a mains coupler, there is an additional option of synchronizing the surge with a phase of the AC mains. This is done by specifying the synchronization mode, via the :LINESYNC:MODE command, and the phase angle via the :LINESYNC:ANGLE command.

The :LINESYNC:MODE command accepts a single argument which specifies which line to use for synchronization:

:LInesync:MOde <mode>

where <mode> is one of the following:

0	Random – no synch
1	L1
2	L2
3	L3

If the selected mode is L1, L2, or L3, the :LINESYNC:ANGLE command is used to define the phase angle, from 0 to 360 degrees:

:LInesync:ANgle <angle>

The mode and angle can be read using the :LINESYNC:MODE? and :LINESYNC:ANGLE? Queries.

Set a surge voltage

The surge voltage is set with the :SRG:VOLTAGE command, passing the voltage as a positive or negative integer:

:SRG:VOltage <volts>

The programmed voltage must be within the allowed range for the module, waveform, and coupling mode, or the ECAT Controller returns an ERR—VALUE response. If the voltage is not a number, the Controller returns an ERR-COMMAND response.

You can retrieve the maximum voltage allowed for a of waveform using the :BAY:WAVEFORM? query; see, Checking ECAT Status, on page 14. See the appendix for a definition of the returned values of this command.

The current waveform may be found by using the :SRG:VOLTAGE? query.

Start the charge process

When all surge parameters have been set, the surge is started by telling the ECAT controller to begin a charge, using the :SRG:CHARGE command:

:SRG:CHarge

This command typically returns “[0]”, however, it typically has a noticeable delay between the “[“ and the remainder of the string.

Before sending a :SRG:CHARGE command, you should consider checking the status of the ECAT interlocks and of the EUT power – if you use EUT power. See, Checking ECAT Status, page 14.

Wait until the charge is complete

The charge cycle in an ECAT can take a considerable time, from 10 seconds to 150 seconds depending on module and waveform. Since the ECAT requires a trigger command to fire the surge, and the trigger command cannot be issued until charging is complete, the software must periodically check the ECAT for status, and issue a trigger command when charging is complete.

The *OPC? query is used to check charging status. This query will return:

0	when the ECAT system is idle, ready for the next charge,
1	when the ECAT system is charging,
2	when charging is complete, and a surge may be triggered.

Any other response indicates that the ECAT is not ready to start a surge sequence.

There are two methods of performing a poll.

The simplest is to start polling immediately after a charge has been started, and either exit the polling loop if the *OPC? response returns to zero, or trigger the surge if the response changes to two. A polling interval of one second is sufficient; one-half second is sufficient.

The alternative is the ask the ECAT how long a charge is likely to take; then, after starting a charge (checking that the charge truly did begin by looking for the *OPC? response to change from zero to one), wait this time interval before starting a fast polling at the one-half second to one second interval.

The time required for charging can be read from the ECAT Controller using the :SRG:DELAY? query.

Note: the time returned by the :SRG:DELAY? query is the time required for the present surge charge. It requires that a surge module and waveform be selected before reading. This value can be changed by the programmer using the :SRG:DELAY command – however, for remote programming there is little reason to do so. The time returned by the :SRG:DELAY? query is not necessarily *the same as the time returned by the :BAY:WAVEFORM? query -- :BAY:WAVEFORM? always returns the minimum charge time, and is not affected by changes made using the :SRG:DELAY command.*

Send a trigger command

When the *OPC? query returns a value of 2, the surge can be triggered by sending the *TRG command:

```
*TRG 1
```

The *TRG command must be issued within 5 seconds of the change in *OPC? status, or the ECAT will time-out and return the system to an idle state.

An ECAT controller with firmware of 5.xx (all controllers after 1995) will respond to the trigger command in one of two ways. If the trigger was unsuccessful, the response will be:

```
[5]
```

If successful, the response will be a string in the format;

```
[ 0    +nnnn     -nnnn    +nnnn     -nnnn]
```

where the numbers indicate, from left to right,

0	a place-holder,
+nnnn	the peak, positive voltage recorded,
-nnnn	the peak, negative voltage recorded,
+nnnn	the peak, positive current recorded,
-nnnn	the peak, negative current recorded.

These numbers are read from the ECAT Controller peak detector circuits, and are only valid numbers if the ECAT system has a VI board installed in the Surge module or coupler, and the measurement system has been configured before the surge event: see Making Automatic Surge Measurements, page 1424.

There may be a considerable delay between issuing the *TRG command and receiving a response from the ECAT controller (on the order of one-quarter second to one second). If the response is to be collected, the software must plan on pausing long enough to receive the entire response.

Before attempting to trigger a surge, you should consider checking the ECAT status, to insure that the ECAT is still ready for a surge. For example, if an ECAT interlock is open, the surge will not take place (also, if an interlock is open it is possible that an operator is working in the EUT, and aborting the test would be the wisest course). See, Checking ECAT Status, page 14.

Troubleshooting Surge Programming Problems

As the above should show, Surge programming an ECAT is straight-forward and simple. However, you may still have problems as you write and debug code. Some common questions and answers follow:

Question:	Am I communicating with the ECAT?
Answer:	If you are communicating with the ECAT, the ECAT VFP display will change from a menu, to the single line, "REMOTE." The ECAT Controller will ignore strings of less that three characters, so check for response by using valid command strings. By pressing the menu button the ECAT will return to local mode, and the Surge menu screen should show the values you have programmed. If you do not see the "REMOTE" legend, check the hardware and serial port settings.
Question:	is the surge happening?
Question:	is the surge going to the programmed destination?
Answer:	You need a safe method of monitoring the surge event. If you need exact numbers, if you must verify the amplitude of the delivered pulse, you should consider purchasing a voltage probe capable of measuring the surge voltage, such as Keytek PK1000 voltage probes, or Tektronix P6015A voltage probes. If a simple indication of presence is sufficient, a gas-tube arrestor might be used, to display a flash with each surge [take appropriate precautions that the arrestor is rated for the surge energy, and shield the arrestor so that if it shatters during a pulse, the operator is not injured by fragments].

In the event that you are communicating with the ECAT Controller and then seem to lose the ability to communicate, try issuing the *RST command to reset the ECAT.

Two problems that you might see involve timing and command responses.

The ECAT microprocessor may be busy when you attempt to talk to it. As the serial port for the Controller is queued, this is usually not a problem, but by sending commands at the maximum rate, you may be able to overrun the queue. If communications between the ECAT and computer seem to have intermittent problems, try adding a small delay between transmissions. The delay need not be large: 0.1 second to a maximum of 0.5 seconds should suffice.

The responses your software sees from the ECAT may be changed by the serial-port driver you use. As example, the driver may suppress the ECAT echo of your command. You should use your driver to send and receive several basic commands, to verify the responses returned.

There may be variations in responses seen between different versions of ECAT Controller firmware, and differences in response for different commands. The ECAT Controller firmware was optimized for Keytek software, and a consistent, unchanging set of commands and responses was not the primary objective of the code. Your software should allow for the possibility of a response (such as an error message), even on commands which are not listed as giving a response.

Making Automatic Surge Measurements

If the ECAT system has a VI board installed in the Surge module or coupler, the VI board can be used with the ECAT Controller peak detector circuitry to return the peak voltage and current seen during the surge.

Note: unlike other ECAT commands, the Measurement commands do not have defaults. All of the selection commands must be sent to the ECAT to insure a valid return during surges.

Please consult your ECAT manual for instructions on connecting to the VI monitors.

The measurement system is configured in two steps: since there may be more than one VI board in an ECAT system, the first step is to select the VI board to use, by sending the :MEASURE:BAY command with the bay number of the module containing the VI board;

:MEasure:BAy <bay>

Then, the voltage channel and current channel to use on the selected board, are set with the :MEASURE:IMON and :MEASURE:VMON commands:

:MEasure:IMon <Imonitor>

:MEasure:VMon <Vmonitor>

The <Imonitor> argument is an integer ranging from one to five. The number of monitors available varies with module (it can be read using the :BAY:OPTIONS? query).

As with surge modules and couplers, the voltage monitors are differential. To use the voltage monitors one channel must be set as the high-channel, and one channel set as the low-channel. The selection argument <Vmonitor> is an integer argument, formed by encoding the identity of the high monitor in the upper nibble of a byte, an the identity of the low monitor in the lower nibble of a byte. This can be simply coded as:

16 * <High channel> + <Low channel>

The voltage monitors on an ECAT chassis are labeled A, B, C, etc., and to program the voltage monitor, channel A is assigned a value of one, channel B a value of two, channel C, three, etc.

The current monitors on an ECAT are internal to the modules. For Couplers the definition is fairly obvious, with current monitor A assigned to L1 current, monitor B to L2 (or N), etc. For the definition of the current monitors within a surge monitor, you may have to refer to the ECAT manual.

The values recorded by the ECAT peak detectors are returned immediately after a surge triggered by the *TRG command. If the surge was triggered successful, the response will be a string in the format;

[0 +nnnn -nnnn +nnnn -nnnn]

where the numbers indicate, from left to right,

0	a place-holder,
+nnnn	the peak, positive voltage recorded,
-nnnn	the peak, negative voltage recorded,
+nnnn	the peak, positive current recorded,
-nnnn	the peak, negative current recorded.

The peak voltage and current values may also be read at any time by using the :MEASURE:IPEAK? and :MEASURE:VPEAK? commands – see the appendix for details on these commands.

Appendix – ECAT Command Summary

*IDN?

query

The *IDN query identifies the instrument by returning a string containing the company name, instrument model, serial number, and software revision. The following illustrates the format of the return string:

"KeyTek Instrument,ECAT,<yymmddd>,<XXYY>"

Where:

<yymmddd>	::= The instrument's serial number
<XXYY>	::= The software revision of the instrument

Query Syntax:	*IDN?
Returned Format:	[KeyTek Instrument,ECAT,yymmddd,XXYY]

*OPC?

query

The *OPC query returns the current status of the surge sequence. Since the length of time involved in a surge sequence extends into tens of seconds, the *OPC status is used to monitor the state of the sequences. The table below lists all of the return values and their associated surge sequence.

Query Syntax: *OPC?

Returned Format: [<value>]

where:
    <value>   ::= 0 to 3   (integer)

Return Value	Surge Sequence	
0	IDLE	Ready for next operation
1	RAMP	High voltage charging is in process
2	READY	Unit is ready to execute a surge
3	DELAY	Unit is performing inter-surge cooldown

Table 1. *OPC return values

*RST

command

The *RST command resets the instrument and places it into a known state. Upon resetting, all currently running operations (i.e. surge, burst, PQF, etc) will terminate and all relay drivers in all modules will be turned off. Please note that any relay drivers that are required to always be on (i.e. EP7x contactor control) will NOT be turned off.

Command Syntax:	*RST
Example:	*RST
Returns:	[ ]

*TRG

command

The *TRG command will initiate a specified sequence. This command varies from the standard, GPIB trigger command in that it has a parameter associated with it. This allows the *TRG command to specifically act upon only one of the ECAT's many modes of operation. The table below lists the parameter values and their associated trigger functions.

The *TRG command will return the values of the peak detector voltage and current when triggering a surge event..

Command Syntax: *TRG <n>

Example: *TRG 1

Returns: [0 +nnnn -nnnn +nnnn -nnnn]

(Response may be delayed)

where the arguments indicate, from left to right,

0	a place-holder,
+nnnn	the peak, positive voltage recorded,
-nnnn	the peak, negative voltage recorded,
+nnnn	the peak, positive current recorded,
-nnnn	the peak, negative current recorded.

OR

[<n>]

in the event of a failure to trigger, where <n> is typically 5.

Parameter	Trigger
1	Start a SURGE sequence
2	Start an EFT sequence

Table 2. Trigger parameters

ABort

command

The ABORT command will abort all current test processes and return the instrument to an idle state.. Please note that only SURGE and EFT affected by this command.

Command Syntax:	ABort
Example:	ABort
Returns:	[ ]

COupler

command / query

The COUPLER command selects the coupler which will be used as the EUT mains source. The required parameter is the module address of a valid EUT source coupler.

The query function returns the module address of the current EUT source coupler.

Command Syntax:	:COupler <bay>
	where: <bay> ::= Module Address (0 to 15)
Example:	:COupler 4
Returns:	[ ]
Query Syntax:	:COupler?
Returned Format:	[<bay>]
	where: <bay> ::= Module Address (0 to 15)

EUt

command / query

The EUT command enables or disables the EUT control button on the ECAT controller. This command will also terminate the EUT power if the EUT power is currently on at the time of disabling. This command will not physically turn on the EUT power - it must be done through the EUT control button itself.

The query function returns the current EUT control state as well as the actual state of the EUT power itself.

Command Syntax:	:EUt { 0 | 1 }
	where: 0 ::= DISABLE the EUT control 1 ::= ENABLE the EUT control
Example:	:EUt 1
Returns:	[ ]
Query Syntax:	:EUt?
Returned Format:	[<n>]
	where n::= 0 ::= EUT control is DISABLED and EUT power is currently OFF 1 ::= EUT control is ENABLED and EUT power is currently OFF 2 ::= EUT control is ENABLED and EUT power is currently ON

LIM

query

The AC:LIM query returns the current settings for the real-time AC measurement system. This command is only valid whenever a coupler with the AC measurement option is present in the ECAT.

Query Syntax: :AC:LIM?

Returned Format: [<peak> , <rmax> , <rmin>]

where:
<peak> ::= Peak current limit
<rmax> ::= Maximum RMS current limit
<rmin> ::= Minimum RMS current limit

PLIM

command

The AC:PLIM command sets the limit for the AC peak measurement. The value is measured in amperes. If this limit is exceeded, the EUT power is automatically shutdown to protect the device under test. This command is only valid when a coupler with the AC measurement option is present in the ECAT.

Command Syntax: :AC:PLIM <amps>

where:

<amps>	::= Maximum peak limit in amps	(0 to 300 – dependent on coupler)

Example: :AC:PLIM 15

Returns: [ ]

RMAX

command

The AC:RMAX command sets the limit for the maximum RMS limit of the AC peak measurement. The value is measured in amperes. If this limit is exceeded, the EUT power is automatically shutdown to protect the device under test. This command is only valid when a coupler with the AC measurement option is present in the ECAT system.

Command Syntax: :AC:RMAX <amps>

where:
<amps> ::= Maximum RMS limit in amps (0 to 300 – dependent on coupler)

Example: :AC:RMAX 15

Returns: [ ]

RMIN

command

The AC:RMIN command sets the limit for the minimum AC RMS peak measurement. The value is measured in amperes. If this limit is exceeded, the EUT power is automatically shutdown to protect the device under test. This command is only valid when a coupler with the AC measurement option is present in the ECAT system.

Command Syntax: :AC:RMIN <amps>

where:

<amps>	::= Minimum RMS limit in amps	(0 to 300 – dependent on coupler)

Example: :AC:RMIN 15

Returns: [ ]

STatus

query

The AC:STATUS query returns the peak measurements for all three phase lines (LI, L2, L3), the state of the EUT power, and a status byte which identifies any AC measurement errors or interlock errors. Note that the trailer value of 1234 has no bearing whatsoever on any of the data. This command is only valid when a coupler with the AC measurement option is present in the ECAT.

Query Syntax: :AC:STatus?

Returned Format: [<status> <peak1> <rms1> <peak2> <eut> <rms2> <peak3> <rms3> 1234]

where:

<status>	::= Error status byte	(0 to 255)
<peak1>	::= L1 phase peak reading	(0 to 65535, -1 on failure)
<rms1>	::= L1 phase RMS peak reading	(0 to 65535, -1 on failure)
<peak2>	::= L2 phase peak reading	(0 to 65535, -1 on failure)
<eut>	::= State of EUT power	(0 | 1 | 2)
<rms2>	::= L2 phase RMS peak reading	(0 to 65535, -1 on failure)
<peak3>	::= L3 phase peak reading	(0 to 65535, -1 on failure)
<rms3>	::= L3 phase RMS peak reading	(0 to 65535, -1 on failure)
<peak>	::= Peak current limit	

DELay

query

This command will return the MINIMUM charge/wait delay for a specific waveform in a specific surge network. The surge network, waveform, EHV capability, and current output setup (front panel or coupler) is used in calculating the minimum delay. The value returned is in seconds.

Query Syntax: :BAY:DELay? <bay> <wavenum>

where:

<bay>	::= Module Address	(0 to 15)
<wavenum>	::= Waveform Number	(1 to 5)

Returned Format: [<delay>]

where:

<delay>	::= Minimum Delay	(integer)

DUmp

query

This query returns a hexadecimal representation of a module's option information, excluding the calibration data. This is a quick and easy method to gather a module's information with minimal communication. The following shows the returned value for an E4554 with a 5-channel instrumentation board, 5 voltage monitors, 3 current monitors, AC and DC options, and a serial number of 9412777:

```
Returned Format:
    000001500000000000745343535340000000530000C0010000A0A9<NL>
    [typepri  ][ typesec ][ modname              ][I][options   ][serial     ]

where:
    typepri     ::= Primary Module Type        (8 characters)
    typesec     ::= Secondary Module Type      (8 characters)
    modname     ::= Module Name                (16 characters)
    I           ::= Module Identifier          (2 characters)
    options     ::= Module Options             (8 characters)
    serial      ::= Module Serial Number       (8 characters)
```

Query Syntax: :BAY:DUmp? <bay>

where:
<bay> ::= Module Address (0 to 15)

Returned Format: [<data>]

where:
<data> ::= Module Data (text, see above)

MEasure

query

This query reads the Module Measurement parameter from memory. The measurement parameter is an encoded value used to denote the module's measurement capabilities; that is, how many voltage and current monitors the module possesses. This parameter has a maximum value of 255 (0xFF), which yields space for 15 voltage and 15 current monitors. This value is an encoded byte using the upper 4 bits for the voltage monitors and the lower 4 bits for the current monitors. The table below shows the byte format.

The query function will return the current module measurement residing in memory and not from the actual hardware.

Query Syntax: :BAY:MEasure? <bay>

where:

<bay>	::= Module Address	(0 - 15)

Returned Format: [<vibyte>]

where:

<vibyte>	::= Encoded Voltage/Current Monitors	(integer)

One (1) Byte		
Data Bits	7 6 5 4	3 2 1 0
Monitors	Voltage	Current

Table 3. Voltage/Current Encoding Scheme

NAme

query

This query reads the Module Name parameter from memory. The name parameter is a textual identification used for display purposes only (i.e. "E501", "E4554", etc). This parameter has a maximum limit of seven (7) characters.

The query function will return the current module name residing in memory and not from the actual hardware.

Query Syntax: :BAY:NAme? <bay>

where:

<bay>	::= Module Address	(0 - 15)

Returned Format: [<name>]

where:

<name>	::= Module Name	(text)

OPtion

query

This query reads the Module Option parameter from memory. The option parameter is an OR'ed value denoting which options are installed in a module (i.e. 8KV EFT, AC RMS, etc). A value of zero (0) denotes no installed options. This parameter has a maximum value of 4,294,967,295 (0xFFFFFFFF).

The query function will return the current module options residing in memory and not from the actual hardware.

Query Syntax: :BAY:OPtion? <bay>

where:

<bay>	::= Module Address	(0 - 15)

Returned Format: [<options>]

where:

<options>	::= Module Options	(integer)

SErial

query

This query reads the Module Serial Number parameter from memory. The serial number parameter is used to store the module's KeyTek serial number. This parameter has a maximum value of 4,294,967,295 (0xFFFFFFFF). The KeyTek serial number is created by combining a datecode with a 3-digit trailer (i.e. 9412777). The formatting method used in the KeyTek serial number is depicted below:

The query function will return the current module serial number residing in memory and not from the actual hardware.

```
<yy><mm><xxx>
where:
     <yy>    ::= Last two Digits of Year   (i.e. 94)
     <mm>    ::= Numerical Month           (i.e. 12)
     <xxx>   ::= 3-digit trailer portion   (i.e. 777)
```

Query Syntax: :BAY:SErial?   <bay>

```
where:
     <bay>      ::= Module Address          (0 - 15)
```

Returned Format: [<serial>]

```
where:
     <serial>   ::= Module Serial Number    (integer)
```

TYpe

query

This query reads the Module Type parameter from memory. The type parameter is an OR'ed field used to distinguish between different module types (i.e. Surge, PQF, EFT, etc). Both fields have a maximum value of 4,294,967,295 (0xFFFFFFFF). There are two type fields - Primary and Secondary. The Primary field is to be used for the main type field and will yield 32 types. The Secondary field is for future expansion and will yield another 32 types. Since the Secondary field is not currently being used it MUST always be set to zero (0).

The query function will return the current module type residing in memory and not from the actual hardware.

Query Syntax: :BAY:TYpe?   <bay>

where:

<bay>	::= Module Address	(0 to 15)

Returned Format: [<primary > <secondary>]

where:

<primary>	::= Primary Module Type	(integer)
<secondary>	::= Secondary Module Type	(integer)

VAlid

query

This query returns a code indicating the existence of a module and/or the validity of the checksums in its serial EEPROM. This command will return either a ZERO (0) if the module is valid or an error code. Table 4 lists the possible error codes returned.

Query Syntax: :BAY:VAlid? <bay>

where:

```
     <bay>   ::= Module Address       (0 to 15)
```

Returned Format: [<chkcode>]

where:

```
     <chkcode>  ::= validity code     (integer)
```

Code	Description
0	Module installed and ALL checksums verify
-1	No module exists in designated slot
-2	Module installed, MAIN checksum failed
-3	Module installed, CALIBRATION checksum failed

Table 4. Error Codes for Valid Command

WAveform

query

This query returns the number of waveforms in a surge network and all LIMIT information relating to a specific waveform. This information can be used for checking surge network parameters in external programs (i.e. SurgeWare). Setting <wavenum> to ZERO (0) will cause the number of waveforms to be returned; while setting <wavenum> to a valid waveform number will return all needed limit information pertaining to that waveform. The information needed for each waveform is the name of the waveform; the ability to output through a front panel connector; the ability to couple to each of the different couplers; the maximum voltage allowable for each of the different couplers; and the minimum charge/wait delay for each of the different couplers. Note that the limits for a standard coupler also pertain to the front panel output.

Query Syntax: :BAY:WAveform? <bay> <wavenum>

where:

<bay>	::= Module Address	(0 to 15)
<wavenum>	::= Waveform Number	(0 to 5)

Returned Format: [<waves>]
or
[<fpf> <scpl> <hcpl> <dcpl> <svlt> <hvlt> <dvlt> <sdly> <hdly> <ddly>,<name>]

where:

<waves>	::= Number of waveforms in surge network	(integer)
<fpf>	::= Front Panel Output available?	(0 | 1)
<scpl>	::= Couples to STANDARD coupler?	(0 | 1)
<hcpl>	::= Couples to HV coupler?	(0 | 1)
<dcpl>	::= Couples to DATA coupler?	(0 | 1)
<svlt>	::= MAXIMUM voltage for STANDARD coupler	(integer)
<hvlt>	::= MAXIMUM voltage for HV coupler	(integer)
<dvlt>	::= MAXIMUM voltage for DATA coupler	(integer)
<sdly>	::= MINIMUM delay for STANDARD coupler	(integer)
<hdly>	::= MINIMUM delay for HV coupler	(integer)
<ddly>	::= MINIMUM delay for DATA coupler	(integer)
<name>	::= Waveform name	(text)

Note: The fields <svlt> and <dvlt> also pertain to the front panel output

CLamp

query / command

This command sets the internal clamp of an E571 Data-line coupler to one of three voltage limits. Query returns the current setting.

Command Syntax :DAta:CLamp <mode>

where:

mode	::= 0	(20-volt clamp)
mode	::= 1	(220-volt clamp)
mode	::= 2	(external clamp)

Returns: [ ]

Query Syntax: :DAta:CLamp?

Returned Format: [<mode>]

where mode:=:

0	::= 20-volt clamp
1	::= 220-volt clamp
2	::= external clamp

CMode

query / command

This command sets the coupling mode of an E571 Data-line coupler to ASYMMETRIC or to SYMMETRIC. Query returns the current setting.

Command Syntax :DAta:CMode <mode>

```
where:
     mode            ::= 0    (ASYMMETRIC)
     mode            ::= 1    (SYMMETRIC)
```

Returns: [ ]

Query Syntax: :DAta:CMode?

Returned Format: [<mode>]

```
where mode:=
     0    ::= ASYMMETRIC
     1    ::= SYMMETRIC
```

OUtput

query / command

This command sets the data output of an E571 Data-line coupler to either ON or OFF. Query returns the current setting.

Command Syntax :DAta:OUtput <mode>

where:

mode	::= 0	(OFF)
mode	::= 1	(ON)

Returns: [ ]

Query Syntax: :DAta:OUtput?

Returned Format: [<mode>]

where mode:=

0	::= OFF
1	::= ON

ANgle

query / command

This command sets the angle for phase-based Surge and EFT operations. Query returns the current setting.

Command Syntax	:LInesync:ANgle <deg>

```
where:
      <deg>              ::= line sync angle, in degrees   (0 to 360)
```

Returns:	[ ]
Query Syntax:	:LInesync:ANgle?
Returned Format:	[<deg>]

```
where:
      <deg>              ::= line sync angle, in degrees   (0 to 360)
```

MOde

query / command

This command sets the mode for phase-based Surge and EFT operations. Query returns the current setting.

Command Syntax :LInesync:MOde <mode>

where:

mode	::= 0	(RANDOM – no sync)
mode	::= 1	(L1)
mode	::= 2	(L2)
mode	::= 3	(L3)

Returns: [ ]

Query Syntax: :LInesync:MOde?

Returned Format: [<mode>]

where:

0	::= RANDOM – no sync
1	::= L1
2	::= L2
3	::= L3

BAy

query / command

This command sets the module to use for peak detection. Query returns the current setting.

Command Syntax :MEasure:BAy <bay>

where:

<bay>	::= Module Address	(0 to 15)

Returns: [ ]

Query Syntax: :MEasure:BAy?

Returned Format: [<bay>]

where:

<bay>	::= Module Address	(0 to 15)

IMon

query / command

This command sets the current monitor to use for peak detection. Query returns the current setting.

Command Syntax	:MEasure:IMon <monitor>

where:

<monitor>	::= Monitor selection	(1 to 5)

Returns: [ ]

Query Syntax: :MEasure:IMon?

Returned Format: [<bay>]

where:

<monitor>	::= Monitor selection	(1 to 5)

IPeak?

query

This query returns the last read, peak current measurements, positive and negative.

Query Syntax:	:MEasure:IPeak?
Returned Format:	[<peak positive> , <peak negative>]

VMon

query / command

This command sets the voltage monitor to use for peak detection. Query returns the current setting.

There are five voltage monitors (maximum) available in a module. One monitor is assigned as the high monitor and one is assigned as the low monitor; voltage readings are then made differentially as <high> minus <low>.

Command Syntax :MEasure:VMon <monitor>

where:

<monitor>	::= Monitor selection	(0 to 255)

The monitor selection argument is an integer argument, formed by encoding the identity of the high monitor in the upper nibble of a byte, an the identity of the low monitor in the lower nibble of a byte. This can be simply coded as:

16 * <High channel> + <Low channel>

where the two channel arguments may be between 1 and 5.

Returns: [ ]

Query Syntax: :MEasure:VMon?

Returned Format: [<HI> , <LO>]

where:

<HI>	::= High Voltage Monitor selection	(1 to 5)
<LO>	::= Low Voltage Monitor selection	(1 to 5)

VPeak?

query

This query returns the last read, peak voltage measurements, positive and negative.

Query Syntax:	:MEasure:VPeak?
Returned Format:	[<peak positive> , <peak negative>]

CHarge

query / command

This command starts the charge process. *OPC is used to track the charge status.

Command Syntax	:SRG:CHarge
Returns:	[0] (Response may be delayed)

COupling

query / command

This command sets the coupling mode to use for Surge couplers. Query returns the current setting.

Command Syntax :SRG:COupling <high> <low>

where:

<high >	::= High Coupling selection	(1 to 16)
<low>	::= Low Coupling selection	(2 to 16)

The coupling selection arguments are formed by assigning a number to each of the phases, as in the list below, and summing these numbers to form the argument:

1	L1
2	L2
4	L3
8	N
16	PE

Thus, to couple to L1 and L2 with respect to PE, the <high> argument would be 3 (1 plus 2), and the low argument would be 16. To couple to all lines with respect to PE, the <high> argument would be 15 and the low argument would be 16.

Returns: [ ]

Query Syntax: :SRG:COupling?

Returned Format: [<HI> , <LO>]

where:

<HI>	::= High Coupling selection	(1 to 16)
<LO>	::= Low Coupling selection	(2 to 16)

DElay

query / command

This command sets the inter-surge delay. The query returns the current setting.

Command Syntax :SRG:DElay   <time>

where:

```
<time>     ::= Inter-surge delay     (integer, seconds)
```

Returns: [ ]

Query Syntax: :SRG:DElay?

Returned Format: [<time>]

where:

```
<time>     ::= Inter-surge delay     (integer, seconds)
```

NEtwork

query / command

This command sets the surge module for all surge operations. Query returns the current setting.

Command Syntax :SRG:NEtwork <bay>

where:

<bay >	::= Module address (0 to 15)

Returns: [ ]

Query Syntax: :SRG:NEtwork?

Returned Format: [<bay>]

where:

<bay >	::= Module address (0 to 15)

OUTput

query / command

This command sets the output -- destination – of the Surge, which can be a coupler, or the surge module front panel. Query returns the current setting.

Command Syntax :SRG:OUTput <dest>

where:

<dest >	::= Module address	(0 to 15)

or,

<dest >	::= front panel	(255)

Returns: [ ]

Query Syntax: :SRG:OUTput?

Returned Format: [<dest>]

where:

(0 to 15)	::= Module address

or,

(255)	::= front panel

VOltage

query / command

This command sets the surge voltage. Query returns the current setting. The maximum surge voltage is module dependent.

Command Syntax :SRG:VOltage <volts>

where:

```
      <volts >            ::= surge voltage     (integer)
```

Returns: [ ]

Query Syntax: :SRG:VOltage?

Returned Format: [<volts>]

where:

```
      <volts >            ::= surge voltage     (integer)
```

WAveform

query / command

This command sets the surge waveform. Query returns the current setting.

Command Syntax :SRG:WAveform <wave>

where:

<wave >	::= surge waveform (1 to 5)

Returns: [ ]

Query Syntax: :SRG:WAveform?

Returned Format: [<wave>]

where:

<wave >	::= surge waveform (1 to 5)

ILock?

query

This query returns an argument indicating the interlock status.

Query Syntax: :SYStem:ILock?

Returned Format: [<state>]

where:

0	::= No interlock fault
1	::= Interlock fault

IText?

query

This query returns a text description of the open interlock (if any).

Query Syntax: :SYStem:IText?

Returned Format: [<text>]

where:
<text> ::= Textual description of the interlock fault