Model 2001 Quick Reference SCPI Guide

Roger Chaplin

June 4, 1993

Introduction

The Model 2001 DMM is the most flexible and capable instrument ever produced by Keithley. The user has direct access to many control settings in the Model 2001 which are hidden from his reach in older instruments. This increased capability does, however, produce increased complexity. To help accommodate this increased complexity, the Model 2001 incorporates the SCPI command set.

Because the Model 2001 presents many more control points to the user, it generally requires more commands than its older counterparts to perform comparable tasks. Also, since the command set in Model 2001 is much larger than that of an older instrument, finding the right commands to do the job is more difficult. Finally, SCPI command names are longer than command names in our older instruments, sometimes by a great deal.

This paper is a quick introduction to SCPI in the Model 2001. Its goal is threefold:

1. Describe the syntax of SCPI commands, and their documentation in the User Manual. Included are descriptions of short forms and default nodes, which can help you reduce the amount of data you have to send to the instrument, thereby improving the performance of your test system.

2. Provide some small simple programs which cause the Model 2001 to perform some simple, commonly-used tasks. These are:

* Change function and range
* Do one-shot triggering
* Do continuous triggering
* Generate SRQ on buffer full
* Store readings in the buffer and then read them out
* Take readings using the scanner

3. Describe the SCPI commands used in the example programs sufficiently that you can extend or otherwise modify the programs to suit your needs.

SCPI Command Syntax

Tree Structure

SCPI commands are organized in a tree structure, similar to disk directories in most modern computer operating systems. Each level in the hierarchy is called a subsystem. For example, here is the Model 2001 SENSe1 subsystem (which controls the acquisition of readings - the SENSe2 subsystem is the rear panel digital input):

```
    [SENSe[1]]
        :DATA?
```

```
:FUNCtion   "<name>"
:VOLTage
    :DC
        :RANGe
            [:UPPer]    <n>
             :AUTO  <Boolean> | ONCE
        :REFerence  <n>
            :STATe  <Boolean>
            :ACQuire
```

The command summary table is the usual way of documenting SCPI commands, but it does not show the complete command names. The complete command name is formed by joining together the components of the name. For example, the complete name of the :STATe command is SENSe1:VOLTage:DC:REFerence:STATe. (The square brackets are not part of the command names. Rather, they indicate optional parts of the command names - more on that later.)

This hierarchical approach permits the same command names to be re-used many times. For example, many subsystems contain a :STATe command, but each one is unique because the complete command name is unique. This is analogous to having a file named, for example, INDEX.TXT in each directory of your computer's disk. Although all the files have the same name, they are unique because they are each in a different directory.

Long and Short Form Command Names

The command names shown above are written with mixed capitalization.

Every SCPI command name has a short form, and most also have a long form. The notation used in documentation is to show the short form in upper case, with the remainder of the name which creates the long form shown in lower case. There are no intermediate forms of the command name - you must send the exact short or long form. However, you do not have to use the mixed capitalization - the Model 2001 accepts commands in any combination of upper and lower case. For example, all of the following are valid forms of the SENSe1:VOLTage:DC:REFerence command:

```
sense1:voltage:dc:reference:state
sens1:volt:dc:ref:stat
SENS1:volt:DC:rEfErEnCe:STat
```

The varieties are almost endless. Wherever command names are used in descriptive text in this paper, the SCPI mixed case notation is used. All of the example programs use the short forms, and send them in lower case.

Query Commands

With few exceptions, every SCPI command has a corresponding query. The command is used to set a control point in the instrument; the query is used to determine the present setting of the control point. The query is simply the command name with a `?' attached.

Some commands are actions rather than control point settings. These commands have no query form. For example, the

SENSe1:VOLTage:DC:REFerence:ACQuire command is an action, not a setting, and so has no query form.

There are also some queries which have no corresponding command. For example, SENSe1:DATA? causes the instrument to return its latest reading. There is no corresponding command, because it would be silly to try to `jam' a reading value into the instrument.

Default Nodes

SCPI utilizes the concept of default nodes. Consider the example command tree shown above in Tree Structure. What's within square brackets is not necessary to send to the instrument, but the instrument will accept it if you send it. Consider the :UPPer command, which sets the measurement range. To set the Model 2001 to measure 15 volts DC, any of the following commands would work. Note: these are shown all in long form - short forms could also be used.

```
SENSe1:VOLTage:DC:RANGe:UPPer 15
SENSe:VOLTage:DC:RANGe:UPPer 15
SENSe1:VOLTage:DC:RANGe 15
VOLTage:DC:RANGe:UPPer 15
VOLTage:DC:RANGe 15
```

There are, of course, many more combinations.

Command Syntax

Notice in the preceding examples that there is no colon character `:' at the beginning of the commands. A leading colon instructs the Model 2001 to interpret the command starting at the root (highest level) of the command tree. Since the Model 2001 also starts at the root each time you send it a new command, the leading colon is not required (although the instrument will accept it if you send it). You can send multiple commands to the Model 2001 in a single message. You separate the commands with a semi-colon character `;'. When the Model 2001 encounters a command following a semi-colon, it attempts to interpret the command starting at the level of the previous command, unless you precede the second command with a colon. For example, either of the following command strings programs the Model 2001 to the 20 volt range for DC voltage measurements, and to use 5 volts as a rel value (the REFerence commands):

```
volt:dc:rang 20;ref 5;ref:stat on
volt:dc:rang 20;:volt:dc:ref 5;:volt:dc:ref:stat on
```

The two command strings are treated identically by the Model 2001. In the first string when the instrument encounters `;ref 5' it notices the following: it is not the first command in the string; there is no leading colon on the command; the previous command was at the VOLTage:DC level. Therefore it interprets the command as though it were also at the VOLTage:DC level.

Example Programs

All examples presume QuickBASIC version 4.5 or higher and a CEC IEEE 488 interface card with CEC driver version 2.11 or higher, with the Model 2001 at address 16 on the IEEE 488 bus.

* Change function and range

Unlike our older DMMs, the Model 2001 has independent "controls" for each of its measurement functions. This means, for example, that autorange can be turned ON for DC volts while leaving it OFF for AC volts. Therefore, each function has its own set of SCPI commands for setting the "controls" specific to that function.

Another difference is in the parameter to the range command. In our older instruments, a single number was used to denote each range. For example, range 1 on a Model 194 is 320 mV full-scale, while range 1 on a Model 182 is 3 mV full-scale. The parameter of the SCPI RANGe command is given as `the maximum value I wish to measure'. The instrument interprets this parameter and goes to the appropriate range. When you query the range (RANGe?) the instrument sends back the full-scale value the range it is presently on.

The following example program illustrates changing function and range. It sets the range for several functions, then takes readings on each of those functions.

```
---------------------------------begin program--------------------------------
'Program funcrng.bas
'Example 2001 program
'Demonstrates changing function and range, taking readings on
```

```
'various functions
'For QuickBASIC 4.5 and CEC PC488 interface card

'Roger Chaplin Tue May 25 09:46:42 EDT 1993
'Keithley Instruments, Inc.

'Edit the following line according to where the QuickBASIC
'libraries live on your machine
'$INCLUDE: 'c:\qb45\ieeeqb.bi'

'Initialize the CEC interface as address 21
CALL initialize(21, 0)

' Reset the SENSe1 subsystem settings, along with the trigger
' model -- each READ? will cause one trigger
CALL SEND(16, "*rst", status%)

' set range for each function we will measure
CALL SEND(16, "volt:dc:rang .1", status%)
CALL SEND(16, "volt:ac:rang 20", status%)
CALL SEND(16, "res:rang 10e3", status%)

' switch to DC volts and take reading
CALL SEND(16, "func 'volt:dc';:read?", status%)
reading$ = SPACE$(80)
CALL ENTER(reading$, length%, 16, status%)
PRINT reading$
```

```
' switch to AC volts and take reading
CALL SEND(16, "func 'volt:ac';:read?", status%)
reading$ = SPACE$(80)
CALL ENTER(reading$, length%, 16, status%)
PRINT reading$

' switch to 2-wire ohms and take reading
CALL SEND(16, "func 'res';:read?", status%)
reading$ = SPACE$(80)
CALL ENTER(reading$, length%, 16, status%)
PRINT reading$
----------------------------------end program---------------------------------
```

Be aware that the Model 2001 rounds the range parameter to an integer before choosing the appropriate range. Sending VOLTage:DC:RANGe 20.45 will set the Model 2001 to its 20 volt range. This is probably not what you want.

* Do one-shot triggering

Our older DMMs generally have two types of triggering: one-shot and continuous. In one-shot, each activation of the selected trigger source causes one reading. In continuous, the DMM is idle until the trigger source is activated, at which time it begins taking readings at a specified rate. Typical trigger sources are: IEEE 488 talk; IEEE 488 Group Execute Trigger (GET); `X' command; External (rear panel BNC).

The trigger controls in the Model 2001 are vastly different from those in our older DMMs. Arming the instrument to respond to triggers is

implicit in the non-SCPI DMMs. Simply sending a command to a non-SCPI DMM to change any of the trigger controls causes the instrument to arm itself for triggers. The SCPI trigger model implemented in the Model 2001 gives you explicit control over the trigger source (the TRIGger subsystem), a two-level control for arming the instrument for triggers (the ARM:LAYer1 and ARM:LAYer2 subsystems), plus a way for completely disabling triggers (the IDLE layer of the trigger model along with the INITiate subsystem). Changing any of the settings in the TRIGger subsystem does not automatically arm the Model 2001 for triggers.

The following example program sets up the Model 2001 to take one reading each time it receives an external trigger pulse.

```
---------------------------------begin program--------------------------------
'Program sgltrig.bas
'Example 2001 program
'Demonstrates one-shot external triggering
'For QuickBASIC 4.5 and CEC PC488 interface card

'Roger Chaplin Tue May 25 09:45:54 EDT 1993
'Keithley Instruments, Inc.

'Edit the following line according to where the QuickBASIC
'libraries live on your machine
'$INCLUDE: 'c:\qb45\ieeeqb.bi'

'Initialize the CEC interface as address 21
CALL initialize(21, 0)
```

```
' Reset controls in INIT, ARM:LAY1, ARM:LAY2 and TRIG subsystems
' and put trigger model in IDLE state
CALL SEND(16, "*rst", status%)

CALL SEND(16, "trig:sour ext;coun inf", status%)

' start everything off and running
CALL SEND(16, "init", status%)
----------------------------------end program---------------------------------
```

After the Model 2001 receives the INITiate command, it stops in the TRIGger layer of the trigger model, waiting for a pulse on the external trigger jack. Each time a pulse arrives on the external trigger jack, the Model 2001 takes one reading. Because TRIGger:COUNt has been set to INFinity, the trigger model never exits from the TRIGger layer. You can send the ABORt command to put the trigger model in the IDLE state, disabling triggers until another INITiate command is sent.

* Do continuous triggering

See the introductory text under `Do one-shot triggering.' The following example program sets up the Model 2001 to take readings as fast as it can once it receives an external trigger. The actual reading rate will depend upon other factors, such as A/D integration time, auto-zero mode, autorange on/off, etc.

```
---------------------------------begin program--------------------------------
'Program contrig1.bas
```

```
'Example 2001 program
'Demonstrates continuous triggering as fast as the meter can
'For QuickBASIC 4.5 and CEC PC488 interface card

'Roger Chaplin Tue May 25 09:45:42 EDT 1993
'Keithley Instruments, Inc.

'Edit the following line according to where the QuickBASIC
'libraries live on your machine
'$INCLUDE: 'c:\qb45\ieeeqb.bi'

'Initialize the CEC interface as address 21
CALL initialize(21, 0)

' Reset controls in INIT, ARM:LAY1, ARM:LAY2 and TRIG subsystems
' and put trigger model in IDLE state
CALL SEND(16, "*rst", status%)

' *RST sets TRIG:SOUR to IMM
CALL SEND(16, "arm:lay2:sour ext", status%)
CALL SEND(16, "trig:coun inf", status%)

' start everything off and running
CALL SEND(16, "init", status%)
----------------------------------end program---------------------------------
```

After the Model 2001 receives the INITiate command, it stops in the ARM:LAYer2 layer of the trigger model, waiting for a pulse on the

external trigger jack. After the external trigger signal occurs, the 2001 moves to the TRIGger layer. Since TRIGger:SOURce is set to IMMediate, a reading is triggered immediately, with a subsequent reading triggered as soon as the previous one is finished.

The following example program sets up the Model 2001 to take readings continuously after an external trigger is received. The trigger rate set to one reading every 50 milliseconds.

```
---------------------------------begin program--------------------------------
'Program contr1g2.bas
'Example 2001 program
'Demonstrates continuous triggering at a specified rate
'For QuickBASIC 4.5 and CEC PC488 interface card

'Roger Chaplin Tue May 25 09:46:31 EDT 1993
'Keithley Instruments, Inc.

'Edit the following line according to where the QuickBASIC
'libraries live on your machine
'$INCLUDE: 'c:\qb45\ieeeqb.bi'

'Initialize the CEC interface as address 21
CALL initialize(21, 0)

' Reset controls in INIT, ARM:LAY1, ARM:LAY2 and TRIG subsystems
' and put trigger model in IDLE state
CALL SEND(16, "*rst", status%)
```

```
CALL SEND(16, "arm:lay2:sour ext", status%)
CALL SEND(16, "trig:coun inf;sour tim;tim .05", status%)

' start everything off and running
CALL SEND(16, "init", status%)
----------------------------------end program---------------------------------
```

After the Model 2001 receives the INITiate command, it stops in the ARM:LAYer2 layer of the trigger model, waiting for a pulse on the external trigger jack. After the external trigger signal occurs, the 2001 moves to the TRIGger layer. Since TRIGger:SOURce is set to TIMer, a reading is triggered immediately, with a subsequent reading triggered every 50 milliseconds. Because TRIGger:COUNt has been set to infinity, the trigger model never exits from the TRIGger layer.

* Generate SRQ on buffer full

In most cases when your program must wait until the Model 2001 has completed an operation, it is most efficient to program the 2001 to assert the IEEE 488 SRQ line when it is finished, rather than repeatedly serial polling the instrument. An IEEE 488 controller will typically address the instrument to talk, then unaddress it, each time it performs a serial poll. Each time the Model 2001 is addressed and unaddressed, it must devote some of its internal computer resources to the IEEE 488 bus, which `steals' time away from actually taking readings. Repeatedly serial polling the Model 2001 will generally reduce its overall reading throughput. Therefore, use the srq%() function call.

The Model 2001 provides a status bit for nearly every operation it can perform. It can be programmed to assert the IEEE 488 SRQ line whenever any of these status bits becomes TRUE or FALSE. The IEEE 488 controller (your computer) can examine the state of the SRQ line without performing a serial poll, thereby detecting when the 2001 has completed its task without interrupting it in the process.

The following example program segment sets up the Model 2001 to assert SRQ when the reading buffer has completely filled, then arms the reading buffer, initiates readings, and waits for the Model 2001 to indicate that the buffer is full. This is not a complete program. Not shown are the commands to configure the trigger model and the reading buffer - a complete example, showing the use of SRQ and programming the Model 2001 reading buffer and trigger model, is given in the next section. The example shown here can be modified for any event in the Model 2001 status reporting system.

```
---------------------------------begin program--------------------------------
' Reset STATus subsystem (not affected by *RST)
CALL SEND(16, "stat:pres;*cls", status%)

CALL SEND(16, "stat:meas:enab 512", status%)    'enable BFL
CALL SEND(16, "*sre 1", status%)        'enable MSB
CALL SEND(16, "trac:feed:cont next", status%)

' start everything off and running
CALL SEND(16, "init", status%)
```

```
WaitSRQ:
IF (NOT(srq%()) THEN GOTO WaitSRQ
CALL SPOLL(16, poll%, status%)
IF (poll% AND 64)=0 THEN GOTO WaitSRQ
----------------------------------end program---------------------------------
```

Notice that after the program has detected an asserted SRQ line, it serial polls the Model 2001 to determine if it is the device requesting service. This is necessary for two reasons:

1. Serial polling the Model 2001 causes it to quit asserting the SRQ line.
2. In test systems which have more than one IEEE 488 instrument programmed to assert SRQ, your program must determine which instrument is actually requesting service.

Once an event register has caused a service request, it cannot cause another service request until you clear it by reading it (in this case using STATus:MEASurement[:EVENt]?) or by sending the *CLS command.

* Store readings in the buffer and then read them out

The reading buffer in the Model 2001 is very flexible and capable. This also makes its controls a little more complex. The reading buffer has basically four controls, found in the TRACe subsystem. There are commands to control:

1. The size of the buffer (in readings).

```
    TRACe:POINts    <NRf>
```

2. Whether or not `extra' data is stored with each reading (e.g. channel number, time stamp). Storing the extra data reduces the maximum size of the buffer.

TRACe:EGRoup	FULL	include extra data
TRACe:EGRoup	COMPact	don't include extra data

3. Where the is data coming from (before or after the CALCulate1 math post-processing).

TRACe:FEED	SENSe1	store unprocessed readings
TRACe:FEED	CALCulate1	store math processed readings

4. What event turns the `spigot' into the buffer on and off.

TRACe:FEED:CONTrol	NEVer	immediately stop storing readings
TRACe:FEED:CONTrol	NEXT	start now, stop when buffer is full
TRACe:FEED:CONTrol	ALWays	start now, never stop
TRACe:FEED:CONTrol	PRETrigger	start now, stop when pretrigger is satisfied

The following example program sets up the Model 2001 to take 20 readings as fast as it can into the buffer, then reads the data back after the buffer has filled. The readings will be stored with timestamp, etc., but the program reads back only the reading values and timestamp. This program uses the techniques described in `Do continuous triggering' and `Generate SRQ on buffer full'.

```
---------------------------------begin program--------------------------------
'Program buffer.bas
'Example 2001 program
```

```
'Demonstrates the reading buffer
'For QuickBASIC 4.5 and CEC PC488 interface card

'Roger Chaplin Tue May 25 09:46:54 EDT 1993
'Keithley Instruments, Inc.

'Edit the following line according to where the QuickBASIC
'libraries live on your machine
'$INCLUDE: 'c:\qb45\ieeeqb.bi'

'Initialize the CEC interface as address 21
CALL initialize(21, 0)

' Reset controls in INIT, ARM:LAY1, ARM:LAY2 and TRIG subsystems
' and put trigger model in IDLE state
CALL SEND(16, "*rst", status%)

' Reset STATus subsystem (not affected by *RST)
CALL SEND(16, "stat:pres;*cls", status%)

CALL SEND(16, "stat:meas:enab 512", status%)    'enable BFL
CALL SEND(16, "*sre 1", status%)        'enable MSB
CALL SEND(16, "trig:coun 20", status%)

' TRACe subsystem is not affected by *RST!!!
CALL SEND(16, "trac:poin 20;egr full", status%)
CALL SEND(16, "trac:feed sens1;feed:cont next", status%)
' start everything off and running
```

```
CALL SEND(16, "init", status%)

' initialize reading$ while the 2001 is busy taking readings
reading$ = SPACE$(4000)

WaitSRQ:
IF (NOT(srq%) THEN GOTO WaitSRQ
CALL SPOLL(16, poll%, status%)
IF (poll% AND 64)=0 THEN GOTO WaitSRQ

CALL SEND(16, "form:elem read,time", status%)
CALL SEND(16, "trac:data?", status%)
CALL ENTER(reading$, length%, 16, status%)
PRINT reading$
----------------------------------end program---------------------------------
```

* Take readings using the scanner

The Model 2001's optional 10-channel scanner is a simple multiplexor. Only one channel can be closed at a time. If you close a channel while another one is already closed, the first one opens automatically, with break-before-make operation guaranteed.

You can use the Model 2001's scanner two ways. One is to simply issue a command to close a particular channel before sending other commands to take readings. The other way is to program the scan list, and let the meter take care of closing the a channel before taking a reading. There is a fundamental difference in the Model 2001's behavior in these two approaches. In the scan list, a measurement function is bound to each

channel, so that when the next channel in the list closes, the meter switches to the associated function. Simply sending a `channel close' command, however, does NOT change the measurement function. While using the scan list, the meter responds to a trigger by first going to the next channel in the list, switching to the function bound to that channel, before taking a reading.

The following example program simply measures DC volts channel 1, AC volts on channel 2 and two-wire resistance on channel 3, using the ROUTe:CLOSe command.

```
---------------------------------begin program--------------------------------
'Program chanrdgs.bas
'Example 2001 program
'Demonstrates taking readings on different scanner channels
'For QuickBASIC 4.5 and CEC PC488 interface card

'Roger Chaplin Tue May 25 09:46:19 EDT 1993
'Keithley Instruments, Inc.

'Edit the following line according to where the QuickBASIC
'libraries live on your machine
'$INCLUDE: 'c:\qb45\ieeeqb.bi'

'Initialize the CEC interface as address 21
CALL initialize(21, 0)

' Reset controls in INIT, ARM:LAY1, ARM:LAY2 and TRIG subsystems
```

```
' and put trigger model in IDLE state, set function to VOLT:DC
CALL SEND(16, "*rst", status%)

' close channel 1, take DC volts reading
CALL SEND(16, "rout:clos (@1);:read?", status%)
reading$ = SPACE$(80)
CALL ENTER(reading$, length%, 16, status%)
PRINT reading$

' close channel 2, take AC volts reading
CALL SEND(16, "func 'volt:ac'", status%)
CALL SEND(16, "rout:clos (@2);:read?", status%)
reading$ = SPACE$(80)
CALL ENTER(reading$, length%, 16, status%)
PRINT reading$

' close channel 3, take ohms reading
CALL SEND(16, "func 'res'", status%)
CALL SEND(16, "rout:clos (@3);:read?", status%)
reading$ = SPACE$(80)
CALL ENTER(reading$, length%, 16, status%)
PRINT reading$
----------------------------------end program---------------------------------
```

The following example program sets the Model 2001 up use the scan list to measure DC volts on channel 1, AC volts on channel 2, and two-wire resistance on channel 3. The meter takes 10 sets of readings, with each set spaced 15 seconds apart, and each of the three readings in each group taken as quickly as possible. The Model 2001 stores the readings

in the buffer, and asserts SRQ when the buffer is full. The program waits for the SRQ, then reads the readings from the buffer.

```
---------------------------------begin program--------------------------------
'Program scanlist.bas
'Example 2001 program
'Demonstrates using the scan list
'For QuickBASIC 4.5 and CEC PC488 interface card

'Roger Chaplin Tue May 25 09:46:07 EDT 1993
'Keithley Instruments, Inc.

'Edit the following line according to where the QuickBASIC
'libraries live on your machine
'$INCLUDE: 'c:\qb45\ieeeqb.bi'

'Initialize the CEC interface as address 21
CALL initialize(21, 0)

' Reset controls in INIT, ARM:LAY1, ARM:LAY2 and TRIG subsystems
' and put trigger model in IDLE state
CALL SEND(16, "*rst", status%)

' Reset STATus subsystem (not affected by *RST)
CALL SEND(16, "stat:pres;*cls", status%)

CALL SEND(16, "stat:meas:enab 512", status%)    'enable BFL
CALL SEND(16, "*sre 1", status%)        'enable MSB
```

```
' *RST sets TRIG:SOUR to IMM
CALL SEND(16, "trig:coun 3", status%)
CALL SEND(16, "arm:lay2:sour tim;tim 15", status%)
CALL SEND(16, "arm:lay2:coun 10", status%)

' TRACe subsystem is not affected by *RST!!!
CALL SEND(16, "trac:poin 30;egr full", status%)
CALL SEND(16, "trac:feed sens1;feed:cont next", status%)
' now the buffer is armed

CALL SEND(16, "rout:scan (@1:3)", status%)
CALL SEND(16, "rout:scan:func (@1), 'volt:dc'", status%)
CALL SEND(16, "rout:scan:func (@2), 'volt:ac'", status%)
CALL SEND(16, "rout:scan:func (@3), 'res'", status%)
CALL SEND(16, "rout:lsel int", status%)

' start everything off and running
CALL SEND(16, "init", status%)

' initialize reading$ while the 2001 is busy taking readings
reading$ = SPACE$(2500)

WaitSRQ:
IF (NOT(srq%()) THEN GOTO WaitSRQ
CALL SPOLL(16, poll%, status%)
IF (poll% AND 64)=0 THEN GOTO WaitSRQ

CALL SEND(16, "form:elem read,time,chan", status%)
```

```
CALL SEND(16, "trac:data?", status%)

CALL ENTER(reading$, length%, 16, status%)

PRINT reading$

----------------------------------end program---------------------------------
```