QuickBASIC 4.5 and Microsoft BASIC 7.1
Quick C2.5
Turbo Pascal 6.0

Model 2001 Multimeter and Model 7001 / 7002 Switch System

Support Software

Contains Programming Information

KEITHLEY

IEEE-488 Interfaces Supported

Capital Equipment Corporation, v2.14 or later
National Instruments NI-488 rev C.10 or later
National Instruments NI-488.2 for PC/II/IIA or AT-GPIB rev 1.5
IOtech Driver 488, v2.6 or later

2001-906-01 Rev. A

QuickBASIC 4.5 and Microsoft BASIC 7.1
pages 1 through 84

Quick C2.5
pages 85 through 178

Turbo Pascal 6.0
pages 179 through 262

QuickBASIC 4.5 and Microsoft BASIC 7.1

Table of Contents

Section 1
Using the Library Routines

1.1 Installation

1.1.1 National Instruments GPIB.COM

You must have Rev C.10 or later of the National Instruments NI-488 Software or Rev 1.0 or later of the National Instruments NI-488.2 Software to use the Model 2001/7001 Support Software. Some older revisions may work, but they are not guaranteed to do so. Use the following settings to set up your National Instruments Card (AT-GPIB card listed here; use all that are applicable) when you are configuring GPIB.COM with IBCONF.EXE.

Primary GPIB Address..... 0
Secondary GPIB Address ..... NONE
Timeout setting T10s
EOS byte..... 0AH
Terminate Read on EOS..... no
Set EOI with EOS on Write..... yes
Type of compare on EOS 7-bit
Set EOI w/last byte of Write ..... yes
System Controller ..... yes
Assert REN when SC ..... yes
Enable Auto Serial Polling ..... no
Timing 350nsec
Enable 488.2 Protocols ..... yes
CIC Protocol ..... no
Disable Device Unaddressing..... no

1.1.2 Capital Equipment Corporation

You must have Rev 2.14 or later of the Capitial Equipment Corp. software to use the Model 2001/7001 Support Software. All older revisions will not work without the removal of SETATNMODE() from SetupIEEE%() in CECQBX.BAS for BASIC 7.1 or in CEC.BAS for QuickBASIC 4.5.

1.1.3 IOTech Driver 488

You must have Rev 2.6 or later of the IOTech Driver 488 software to use the Model 2001/7001 Support Software. Some older revisions may work, but they are not guaranteed to do so.

1.1.4 Microsoft BASIC 7.1

Type A:INSTALL or B:INSTALL to install the Microsoft BASIC 7.1 Model 2001/7001 support software. The installation program will prompt you for certain information to copy files and build the make files MAKECEC.BAT, MAKEIOT.BAT, MAKENAT1.BAT, and MAKENAT2.BAT. These files will build the *.OBJ, *.LIB, *.QLB, and *.EXE files needed to use the routines.

Installation Notes:

1. BC.EXE, LINK.EXE, and LIB.EXE must be installed in the BASIC 7.1 Compiler Directory.
2. NMAKE.EXE must be installed in the NMAKE Directory.
3. BCL71EFR.LIB, DTFMTER.LIB, QBXQLB.LIB, QBX.LIB and IEEE488.LIB (if installing CEC) must reside in the Library Files Directory.
4. FORMAT.BI, MOUSE.BI, MENU.BI, and MOUSE.BI must reside in the Include Files Directory.
5. The Model 2001/7001 Support Software Directory must exist, and you will be prompted to make it if it does not.

1.1.5 QuickBASIC 4.5

Type A:INSTALL or B:INSTALL to install the QuickBASIC 4.5 Model 2001/7001 support software. The installation program will prompt you for certain information to copy files and build the make files MAKECEC.BAT, MAKEIOT.BAT, MAKENAT1.BAT, and MAKENAT2.BAT. These files will build the *.OBJ, *.LIB, and *.QLB files needed to use the routines.

Installation Notes:

1. BC.EXE, LINK.EXE, and LIB.EXE must be installed in the QuickBASIC 4.5 Compiler Directory.
2. BQLB45.LIB and IEEEQB.LIB (if installing CEC) must reside in the Library Files Directory.
3. The Model 2001/7001 Support Software directory must exist, and you will be prompted to make it if it does not.

1.2 General Instructions

1.2.1 Microsoft BASIC 7.1

To use a Quick Library in BASIC 7.1, start by issuing at the DOS prompt:

QBX /Lcecqbx.qlb	to use CEC
QBX /Liotqbx.qlb	to use IOTech
QBX /Lnatqbx1.qlb	to use NI-488
QBX /Lnatqbx2.qlb	to use NI-488.2

In your program, use **'$INCLUDE:'20017001.BI'** which contains the Model 2001/7001 library function definitions and CONSTants OR use **'$INCLUDE:'2001DEMO.BI'** to use the BASIC 7.1 User Interface routines (WINDOW.BAS, MOUSE.BAS, MENU.BAS, GENERAL.BAS, UIASM.OBJ, and QBX.LIB), which are included in the libraries and quick-libraries. A sample setup follows:

' To use the BASIC 7.1 Windowing and Menu capabilities use:

```
'$INCLUDE:'2001DEMO.BI'
MenuInit                        ' Initialize Menu system (also initializes mouse with MouseInit)
WindowInit                      ' Initializes Windowing system
MouseShow                       ' Reveal mouse cursor
SetupErr%=SetupIEEE%(0,0)       ' Setup IEEE interface
SetupErr%=SetupIEEE%(2001,16)   ' Setup Model 2001 at IEEE address 16
SetupErr%=SetupIEEE%(7001,7)    ' Setup Model 7001 at IEEE address 7
'
```

```
' User Program Follows
'
' OR: (No Windowing or pop-up error messages)
' (SEE QuickBASIC 4.5 Example)
'
```

Note: When using the National Instruments NI-488 rev C.10 or C.11 interface with BASIC 7.1, the following ibxxx commands are unavailable:

ibrdi, ibrdia, ibwrti, ibwrtia, ilrdi, ilrdia, ilwrti, and ilwrtia.

Also, note that the ibfind and ibdev subroutines are now functions like ilfind% and ildev%. These limitations result from the BASIC 7.1 to Quick C link to support far strings in BASIC 7.1 which are not supported by National Instruments in their older NI-488 drivers (rev C.11 or older). To use the entire line of ibxxx commands, National Instruments suggests upgrading to the latest NI-488 driver (rev C.14 as of the time of this printing) and using the NI-488.2/NI-488 rev C.12 or newer Model 2001/7001 Support Software.

1.2.2 QuickBASIC 4.5

To use a Quick Library in QuickBASIC 4.5, start by issuing at the DOS prompt:

QB /Lcec.qlb	to use CEC
QB /Liotech.qlb	to use IOTech
QB /Lnat488_1.qlb	to use NI-488
QB /Lnat488_2.qlb	to use NI-488.2

In your program, use **'$INCLUDE:'20017001.BI'**, which contains the Model 2001/7001 library function definitions and CONSTants. A sample setup follows:

```
'$INCLUDE:'20017001.BI'
CONST TRUE = -1
CONST FALSE = 0
SetupErr%=SetupIEEE%(0,0)          ' Setup IEEE interface
SetupErr%=SetupIEEE%(2001,16)      ' Setup Model 2001 at IEEE address 16
SetupErr%=SetupIEEE%(7001,7)       ' Setup Model 7001 at IEEE address 7
'
' User Program Follows
'
```

1.2.3 Documentation Notes

1. All functions in this document use QuickBASIC 4.5 type cast notation (% — integer, $ — String, etc.).
2. Query forms of a function have the format of *function-nameQ$*, which returns a query string containing the results of the specified Model 2001/7001 parameters. Automatic values are returned as 0 for OFF and 1 for ON.
3. If a function query returns more than one parameter, the responses will be separated by a semi-colon (;) within the return string. Use the ParseQuery$ routine to return the specified response string (parameter number 1, 2, 3, etc.) from the query return string.

4. Any Model 2001/7001 command parameters out of range will not be sent to the instrument, leaving the previously set or default values in tact. Also, the global variable, *OutOfRange%*, will be set to TRUE (-1).
5. Most Model 2001/7001 parameters that are double precision values will accept the constants MAXIMUM, MINIMUM, and DEFAULT. INF will be accepted for Trigger Model Count1# and Count2# parameters.
6. All example programs assume that QuickBASIC 4.5 or BASIC 7.1 setup commands listed above were used before issuing any commands.
7. All string parameters that have their valid parameters listed with mixed case, like IMMediate, accept either the short form (IMM) or the long form (IMMEDIATE) in any combination of case. This is comparable to the short and long form notation used for SCPI commands.
8. See Appendix A for a description of the Global Variables used in the Model 2001/7001 support software.
9. See Appendix B for a description the defined CONSTants used in the Model 2001/7001 support software.
10. See Appendix C for a list of all QuickBASIC 4.5 and BASIC 7.1 file names used by the Model 2001/7001 support software.

Section 2
Model 2001 / 7001 IEEE-488 Interface Routines

These functions and subroutines control the Models 2001/7001 with low-level IEEE-488 bus commands specific to each IEEE-488 interface manufacturer. If any IEEE-488 timeout errors occur, an error message will be displayed. For BASIC 7.1 the WindowInit command must be issued at start up to see the messages. Also, the TimeOutError% global variable (see Global Variables in Appendix A) will be set TRUE.

NOTE

These functions and subroutines also support the Model 7002 Switch System. Just send Model 7001 commands to a Model 7002.

2.1 FUNCTION SetupIEEE%(Device%, Address%)

Description

Sets up the specific IEEE-488 interface to correctly handle data transfers between the Models 2001/7001 and the host PC computer. Also, initializes either the 2001 or 7001 at the specified addresses. CEC controllers will be at IEEE address 21. IOTech controllers are usually at IEEE address 21. National Instruments controllers are usually at IEEE address 0.

Parameters

Device%

0	Initialize Interface Only
2001	Change the IEEE address of the Model 2001 only
7001	Change the IEEE address of the Model 7001 only

Address%

0-30	if Device% is Model 2001 or 7001
ignored	if Device% is 0

Returns

TRUE (-1) if a TimeOutError occurred or a parameter is OutOfRange.

Global Variables Affected (see Global Variables Appendix A):

KI2001%, KI7001%, TimeOutError%, OutOfRange%, IeeeInterface%
brd0%, Nat2001Addr%, Nat7001Addr% (Nat'l. Instruments only)
IeeeIn%, IeeeOut% (IOTech Driver 488 only)

Example

Before using any of the Model 2001 routines, the following commands must be issued:

```
'$INCLUDE:'20017001.BI'
SetupErr%=SetupIEEE%(0,0)          ' Setup IEEE interface
SetupErr%=SetupIEEE%(2001,16)      ' Setup Model 2001 at IEEE address 16
SetupErr%=SetupIEEE%(7001,7)       ' Setup Model 7001 at IEEE address 7
```

2.2 SUB SendSDC2001
SUB SendSDC7001

Description

Sends the IEEE bus command SDC (Selected Device Clear) to the Model 2001/7001.

Global Variables Affected (see Global Variables in Appendix A)

TimeOutError%

2.3 FUNCTION Q2001$(Cmd$)
FUNCTION Q7001$(Cmd$)

Description

Gets a query response from the Model 2001/7001. Cmd$ must be a valid Model 2001/7001 query or else the instrument will TimeOut. Multiple queries are allowed, and responses can be separated using the ParseQuery$ function. If Cmd$="", then the function will still try to read data from the instrument. This feature is good for reading large amounts of data from the Model 2001/7001 since the most that can be read at one time is 2048 bytes.

Parameters

Cmd$

" " try to read the Model 2001/7001
<> " " send query and try to read Model 2001/7001

Returns

Query response from the Model 2001/7001 if Cmd$ was valid.
"Error!" if Cmd$ not valid (i.e., a timeout error occurred).

Global Variables Affected

TimeOutError%, Resp$

Example

```
' BASIC 7.1 or QB45 setup commands called before here
A$ = Q2001$("VOLT:DC:RANGE?")                    ' return DC Voltage Range
```

2.4 SUB Send2001(Cmd$)
SUB Send7001(Cmd$)

Description

Sends IEEE-488.2 and SCPI command strings to the Model 2001/7001.

Parameters

Cmd$ Valid Model 2001/7001 488.2 or SCPI command or query.

Returns

Nothing

Must check the Model 2001/7001 EAV bit in the serial poll register to see if a command was accepted, or look at the front panel of the instrument for an error message.

Global Variables Affected (see Global Variables in Appendix A)

TimeOutError%, OutOfRange%

Example

```
' BASIC 7.1 or QB45 setup commands called before here
Send7001 "ROUTE:CLOSE (@1!1:1!40)"          'Close channels 1-40 on card 1 of Model 7001
```

2.5 SUB SendGET2001
SUB SendGET7001

Description

Sends the IEEE bus command GET to the Model 2001/7001.

Global Variables Affected (see Global Variables in Appendix A)

TimeOutError%

Example

```
' BASIC 7.1 or QB45 setup commands called before here
SendGET2001                          ' Trigger the Model 2001 to take reading
Reading$ = Q2001$("DATA?")           ' Get Model 2001 reading
```

2.6 FUNCTION Wait2001RQS%
FUNCTION Wait7001RQS%

Description

Waits for the Model 2001/7001 to generate a Request for Service. The routine serial polls the Model 2001/7001 to verify that the instrument is indeed generating a Request for Service. The wait can be aborted by pressing the Esc key (QuickBASIC 4.5 and BASIC 7.1) or the right mouse button (BASIC 7.1 only).

Returns

FALSE (0) - if aborted.
TRUE (-1) - if a Model 2001/7001 Request for Service.

Example

```
' BASIC 7.1 or QB45 setup commands called before here
Poll% = Poll2001%                ' Clear and pending SRQ's
SRE2001 MAV                      ' Set up to SRQ on MAV
Send2001 "FETCH?"                ' Fetch a Model 2001 reading
DO                               ' Set up an uninterruptable wait
LOOP UNTIL Wait2001RQS%
Reading$ = Q2001$("")            ' Get reading
```

Section 3
Model 2001 Buffer Routines

These routines are used to acquire readings in the Model 2001's data buffer. Up to 30,092 readings can be stored in the Model 2001 with the MEM2 option and the compact format.

All routines will set the Global Variable TimeOutError% if an IEEE-488 Timeout error occurs. Functions that have bounded parameters (maximum and/or minimum parameters) will set the Global Variable OutOfRange% if a parameter is not within the limits specified (see Global Variables in Appendix A).

3.1 FUNCTION BufferSize2001%

Description

Finds the actual number of data points in the Model 2001 buffer. This function should be used since TRACe:POINts? may not return the correct number of data points if the data buffer acquisition was aborted.

Returns

The actual number of data points in the Model 2001 buffer (anywhere from 2 to 30,092) depending on the memory configuration of the Model 2001 being used.

Example

```
' BASIC 7.1 or QB45 setup commands called before here NumPoints% = BufferSize2001%
' Read Back Readings in double precision DIM BufferData#(1 to NumPoints%)
NumPoints% = Read2001BufferD%(BufferData#(), 0, BUFRDGS)
```

3.2 FUNCTION Take2001BufferReadings% (Func%, BurstMode%, NumDataPoints1%, Compact%)

Description

Sets up and acquires up to 30,092 readings in the Model 2001's data buffer. (Use with Read2001BufferS% or Read2001BufferD% to read data). The data acquisition may be aborted by pressing Esc (QuickBASIC or BASIC 7.1) or by pressing the right mouse button (BASIC 7.1 Only).

Parameters

Func%, (also, see Function Constants, Appendix B)

0 — Current Function
1 — DCV
2 — ACV

3 — DCA
4 — ACA
5 — 2-wire Resistance
6 — 4-wire Resistance
7 — Frequency
8 — Temperature

BurstMode%

If TRUE (non-zero), uses the fast Burst Mode of Reading acquisition available from the Model 2001. Burst-Mode is only applicable if Func% is 1 to 5.

NumDataPoints1%

2 to TRACE:POINTS? MAX (depends on the setting of Compact% and the memory option installed in the Model 2001).

Compact%

TRUE (non-zero) — use the COMPACT buffer format (readings only).
FALSE (zero) — use the FULL buffer format (reading, time stamp, channel number, status, and units).

NOTE

The COMPACT format allows five times as many readings as does FULL. If BurstMode% is set, only COMPACT format is valid.

Returns

The actual number of data points read from the Model 2001. If zero, then either a TimeOutError occurred or an OutOfRange error occurred for Func% or NumDataPoints% < 2.

Example

```
' BASIC 7.1 or QB45 setup commands called before here
' Take 2000 DC Volt Burst Mode Readings:
NumPoints% = Take2001BufferReadings%(DCV, TRUE, 2000, TRUE)
' Read Back data in Single Precision Format:
DIM BufferData!(1 to NumPoints%)
NumPoints%=Read2001BufferS%(BufferData!(), 0, BUFRDGS)
```

3.3 FUNCTION Read2001BufferS% (ArrayName!(), DMA%, Fmt%) FUNCTION Read2001BufferD% (ArrayName#(), DMA%, Fmt%)

Description

Retrieves all of the readings (up to 29,908 IEEE754 single or double precision readings) from the Model 2001's data buffer and stores them into a single or double precision array.

Parameters

ArrayName!, ArrayName#

Single or double precision array dimensioned large enough to handle the number of data points returned by BufferSize2001% or Take2001BufferReadings%.

DMA%

0 Don't use DMA
1 Use DMA (IOTech or National Instruments)
1-7 Use DMA channel configured on CEC IEEE-488 Interface card

Fmt%

1 Return Readings (FULL or COMPact Format)
2 Return TimeStamp (Full Format only)
3 Return Channel (Full Format only)
4 Return Status (Full Format only)
5 Return Units (Full Format only)

Also, see Read2001BufferS% and Read2001BufferD% Constants in Appendix B.

Returns

The actual number of data points transferred to the array. If zero, then either a TimeOutError occurred, an OutOfRange error occurred, or ArrayName!() or ArrayName#() is dimensioned too small.

Example

```
' BASIC 7.1 or QB45 setup commands called before here
' Take 5000 AC Volt Full Format Mode Readings:
NumPoints% = Take2001BufferReadings%(ACV,FALSE,5000,FALSE)
' Read Back Readings and TimeStamp in double precision
DIM BufferData#(1 to NumPoints%), Times#(1 to NumPoints%)
NumPoints%=Read2001BufferD%(BufferData#(), 0, BUFRDGS)
NumPoints%=Read2001BufferD%(Times#(), 0, BUFTIMESTAMP)
DataViewD BufferData#(), NumPoints%
```

Section 4
General Routines

These routines provide data display, graphing, and data manipulating functions that make the handling of the returned data from the Models 2001 and 7001 easier to handle.

NOTE

These functions and subroutines also support the Model 7002 Switch System. Just send Model 7001 commands to a Model 7002.

All functions that have bounded parameters (maximum and/or minimum parameters) will set the Global Variable OutOfRange% if a parameter is not within the limits specified (see Global Variables in Appendix A).

4.1 FUNCTION ParseQuery$(Quer$, QuerNum%)

Description

Returns the query specified by QuerNum inside Quer$. Queries are separated either by commas or semi-colons. Also, leading and trailing spaces are removed.

Parameters

Quer$	A string returned by any of the Model 2001/7001 query functions.
QuerNum%	The number of the query to retrieve

Returns

"Error!" if Quer$ = "",
"Invalid Query Number." if QuerNum% <= 0,
QuerNum% query in Quer$, or last query if QuerNum% is too large

Example

```
' BASIC 7.1 or QB45 setup commands called before here
A$ = "100; FFF; oii, 4423.223; 4,000"
B$ = ParseQuery$(A$, 1)        ' B$ = "100"
B$ = ParseQuery$(A$, 2)        ' B$ = "FFF"
B$ = ParseQuery$(A$, 3)        ' B$ = "oii"
B$ = ParseQuery$(A$, 4)        ' B$ = "4423.223"
B$ = ParseQuery$(A$, 5)        ' B$ = "4"
B$ = ParseQuery$(A$, 6)        ' B$ = "000"
B$ = ParseQuery$(A$, 7)        ' B$ = "000"
B$ = ParseQuery$(A$, 0)        ' B$ = "Invalid Query Number."
```

4.2 FUNCTION Str1$(DoubleNumber#)

Description

Returns a string representation of a double precision number with an E in the exponent instead on a D, if DoubleNumber# is large or small enough to be displayed in exponential format.

Parameters

DoubleNumber# Any valid double precision number

Returns

Double-precision mantissa, E, then exponent, or;
Double-precision number

Example

```
' BASIC 7.1 or QB45 setup commands called before here
A$ = Str1$(9.9d34) ' A$ = "9.9E+34"
```

4.3 FUNCTION IOTECH$(Address%)

Description

Returns a two-digit number string with a leading 0 for use with IOTech's IEEE Addressable commands.

Parameters

Address% 0 to 30 Returns: "00","01",..,"09","10","11",..,"30"

Example

```
' BASIC 7.1 or QB45 setup commands called before here
PRINT #1, "OUTPUT "+IOTECH$(9)
```

4.4 SUB XYGraphS(XArray!(), YArray!(), YStart%, YStop%, XTitle$, YTitle$, Title$, UseCGA2%, MaxMinScale%) SUB XYGraphD(XArray#(), YArray#(), YStart%, YStop%, XTitle$, YTitle$, Title$, UseCGA2%, MaxMinScale%)

Description

Produces a simple X and Y auto-scaled graph using single (XYGraphS) or double (XYGraphD) precision data. These routines will plot YArray versus XArray if the size of the two arrays are equal. If they are not the same, only the YArray is plotted versus its corresponding data point number. Use YStart and YStop to zoom in on a particular area of the graph. Both the X and Y axes are scalable to the maximum and minimum of the arrays within the specified YStart and YStop interval.

Parameters

XArray!()	Single-precision X-axis data array
XArray#()	Double-precision X-axis data array
YArray!()	Single-precision Y-axis data array
YArray#()	Double-precision Y-axis data array
YStart%	First Y data point to plot
YStop%	Last Y data point to plot
XTitle$	X-axis Title
YTitle$	Y-axis Title
Title$	Graph Title
UseCGA2%	If non-zero, use CGA 640x200 mode so that GRAPHICS.COM can do a screen dump.
MaxMinScale%	If non-zero, scale to the minimum and maximum values of the Y-axis data. This has an effect only if all of the data points are the same sign.

Example

```
' BASIC 7.1 or QB45 setup commands called before here
NumPoints% = Take2001BufferReadings%(ACV,FALSE,5000,FALSE)
DIM BufferData#(1 to NumPoints%), Times#(1 to NumPoints%)
NumPoints%=Read2001BufferS%(BufferData#(), 0, BUFRDGS)
NumPoints%=Read2001BufferD%(Times#(), 0, BUFTIMESTAMP)
' Plot AC Voltage vs. Time scaled to AC Data
XYGraphD BufferData#(), Times#(), 1, NumPoints%, "Time (sec)", "AC Voltage (Vrms)", "AC Voltage vs. Time", FALSE, TRUE
```

4.5 SUB DataViewS(DataArray!(), NumDataPoints%) SUB DataViewD(DataArray#(), NumDataPoints%)

Description

Views a single or double precision array, DataArray, NumDataPoints long using PageUp, PageDn, Home, End, and the Arrow keys. Pressing Esc aborts the data display (QuickBASIC 4.5 and BASIC 7.1) as does clicking the right mouse button (BASIC 7.1 only).

Parameters

DataArray!()	Single precision data array to display
DataArray#()	Double precision data array to display
NumDataPoints%	Number of data points to display. If zero or more than the number of points actually in the array, display whole array.

Returns

NumDataPoints% with the maximum number of data points in the array if NumDataPoints% was originally a variable and was 0 or larger than the number of points in the array.

Example

```
' BASIC 7.1 or QB45 setup commands called before here
```

```
NumPoints% = BufferSize2001%
' Read Back Readings in double precision
DIM BufferData#(1 to NumPoints%)
NumPoints% = Read2001BufferD%(BufferData#(), 0, BUFRDGS)
' View all Data Buffer Readings
DataViewD BufferData#(), NumPoints%
```

4.6 FUNCTION AutoGraphicsMode% (NumYPixels%, NumTextRows%, UseCGA2%)

Description

Automatically detects the best available 80 Column Text graphics mode. Graphics Driver Order of Precedence:

SCREEN MODE	NumXPixels	NumYPixels	NumTextCols	NumTextRows
11 (VGA)	640	480	80	60
10 (EGA)	640	350	80	43
9 (EGA)	640	350	80	43
8 (EGA)	640	200	80	25
2 (CGA)	640	200	80	25

Parameters

UseCGA2%

TRUE (non-zero) — forces Screen Mode 2.
FALSE — Automatically detect highest resolution
NumYPixels% — must be a variable name
NumTextRows% — must be a variable name

Returns

Screen Mode used.
NumYPixels in NumYPixels% if NumYPixels% is a variable.
NumTextRows in NumTextRows% if NumTextRows% is a variable.

Example

```
' BASIC 7.1 or QB45 setup commands called before here
' Given: a PC with a VGA monitor
Mode% = AutoGraphicsMode%(NumYPixels%, NumTextRows%, FALSE)
' Mode% = 11, NumYPixels% = 480, NumTextRows% = 60
```

Section 5
General Model 2001 / 7001 Routines

The following routines perform some extra functions that are not in the Model 2001/7001 and manipulate the Model 2001/7001's front panel.

NOTE

These functions and subroutines also support the Model 7002 Switch System. Just send Model 7001 commands to a Model 7002.

All of these routines will set the Global Variable TimeOutError% if an IEEE-488 Timeout Error occurs. Functions that have bounded parameters (maximum and/or minimum parameters) will set the Global Variable OutOfRange% if a parameter is not within the limits specified (see Global Variables in Appendix A).

5.1 SUB Set2001Function(Func%)

Description

Puts the Model 2001 into the specified measurement function.

Parameters

Func% (1-8) see Function Constants, Appendix B.

Example

```
' BASIC 7.1 or QB45 setup commands called before here
' All three of the following put the Model 2001 in AC Volts:
Set2001Function ACV
ACV2001
Send2001 "FUNC 'VOLT:AC'"
```

5.2 FUNCTION Get2001Units$(Func%)

Description

Gets the proper Model 2001 units for the function specified.

Parameters

Func% 1-8, see Function Constants, Appendix B.

Returns

"Error!" if a TimeOutError occurred, or
Func% = 1 (DC Volts): "VDC "
Func% = 2 (AC Volts):
<5-character string> + <0-4 character string>, where:
<5-character string> = " dB ", " dBm ", or "VAC "
<0-4 character string> = "", "Avg", "Peak", "+Pk", "- Pk", or "RMS"
Func% = 3 (DC Current): "ADC" or "ADC ICkt"
Func% = 4 (AC Current): "AAC Avg" or "AAC RMS"
Func% = 5 (2-wire Resistance): "Ω2W" or "Ω2W Ocmp"
Func% = 6 (4-wire Resistance): "Ω4W" or "Ω4W Ocmp"
Func% = 7 (Frequency): "Hz"
Func% = 8 (Temperature): "°F", "°C", or " K"

Example

```
' BASIC 7.1 or QB45 setup commands called before here
Send2001 "*RST"
' AC Voltage with the detector set to RMS will
' return "VAC RMS"
A$ = Get2001Units$(ACV)
```

5.3 FUNCTION Get2001SaveRecallSize%

Description

Finds the number of Model 2001 Save/Recall (*SAV, *RCL) locations available for storing Model 2001 configurations.

Returns

Model 2001 Option	SaveRecallSize
2001	1
2001/MEM1	5
2001/MEM2	10

Example

```
' BASIC 7.1 or QB45 setup commands called before here
' Model 2001 with MEM1 option
' RecNum% = number of setup location to recall
IF RecNum% < Get2001SaveRecallSize% AND RecNum% >= 0 THEN
   Send2001 "*RCL" + STR$(RecNum%)
ELSE
   PRINT "Recall Number is too large!"
ENDIF
```

5.4 FUNCTION Get2001FuncHeader$(Func%)

Description

Returns the proper SENSe subsystem SCPI Header string for the function specified.

Parameters

Func% (1-8) see Function Constants, Appendix B.

Returns

Func%	Returns
1	VOLT:DC:
2	VOLT:AC:
3	CURR:DC:
4	CURR:AC:
5	RES:
6	FRES:
7	FREQ:
8	TEMP:

Example

```
' BASIC 7.1 or QB45 setup commands called before here
A$ = Get2001FuncHeader$(DCV) ' A$ = "VOLT:DC:"
```

5.5 FUNCTION Get2001Func%

Description

Finds the present Model 2001 Function

Returns

The Model 2001 function number from 1 to 8 as specified by the Function Constants, Appendix B.

Example

```
' BASIC 7.1 or QB45 setup commands called before here
DCV2001
Func% = Get2001Func%                    ' Put Model 2001 in DC Volts mode
' Func% = 1, which is the CONSTant DCV
```

5.6 FUNCTION Check2001Val$(CheckVal#, MinVal#, MaxVal#, CheckMinInf%)

Description

Checks a given double precision value (CheckVal#) against the given minimum allowed value (MinVal#) and the maximum allowed value (MaxVal#). It is used internally by the Model 2001/7001 routines to check parameter ranges.

Parameters

CheckVal# Any double precision value or one of the following constants:

INF = 9.9D+37
MINIMUM = 9.8D+37
MAXIMUM = 9.7D+37
DEFAULT = 9.6D+37

CheckMinInf%

0 — Don't check for MINIMUM, MAXIMUM, DEFAULT, or INF
1 — Check for MINIMUM, MAXIMUM, or DEFAULT
2 — Check for MINIMUM, MAXIMUM, DEFAULT, or INF

Associated constants:

CHECKNONE = 0
CHECKMINMAX = 1
CHECKINF = 2

Returns

A null string and sets OutOfRange global variable to TRUE (-1) if CheckVal# is out of range
An ASCII string representation of CheckVal# if in range
"MAX", "MIN", "DEF", or "INF" if CheckVal# is equivalent to one of the constants above.

Example

```
' BASIC 7.1 or QB45 setup commands called before here
' Checks for a value to be between 100# an 1000# inclusive
Value$ = Check2001Val$(MINIMUM, 100#, 1000#, CHECKMINMAX)
' Value$ = "MIN"
```

5.7 SUB Hit2001Key(HitKey%)

Description

Presses the specified Model 2001 front panel key using SYSTEM:KEY.

Parameters

HitKey% 1-23, 26-31 (OutOfRange% set and does nothing otherwise)

Defined keys are as follows:

UPKEY = 1	TEMPKEY = 2	LEFTKEY = 3
MENUKEY = 4	ACIKEY = 5	STOREKEY = 6
LOCALKEY = 7	PREVIOUSKEY = 8	AUTOKEY = 9
RIGHTKEY = 10	EXITKEY = 11	R2KEY = 12
RECALLKEY = 13	CHANKEY = 14	DCVKEY = 15
NEXTKEY = 16	DOWNKEY = 17	ENTERKEY = 18
R4KEY = 19	FILTERKEY = 20	SCANKEY = 21
ACVKEY = 22	RELKEY = 23	FREQKEY = 26
MATHKEY = 27	CONFIGKEY = 28	DCIKEY = 29
TRIGKEY = 30	INFOKEY = 31	

Example

```
' BASIC 7.1 or QB45 setup commands called before here
Hit2001Key AUTOKEY              ' Hit Auto Range key on Model 2001
```

5.8 SUB Disp2001(line1$, line2$)
SUB Disp7001(line1$, line2$)

Description

Immediately displays line1$ on the first line and line2$ on the second line of the Model 2001/7001's display.

Parameters

Line1$ maximum of 20 characters
Line2$ maximum of 32 characters

Returns

Line1$ truncated to 20 characters if line1$ is a variable name.
Line2$ truncated to 32 characters if line2$ is a variable name.

Example

```
' BASCI 7.1 or QB45 setup commands called before here
Line1$ = "2001/7001 Support Software"
Disp2001 Line1$,"(c) 1992 Keithley Instruments"
' Line1$ now = "2001/7001 Support So"
NoDisp2001              ' turn off user display
```

5.9 SUB NoDisp2001
SUB NoDisp7001

Description

Turns off the user's displayed messages on the Model 2001/7001.

Section 6
Model 2001 SENSe[1] Subsystem Commands

All of these routines will set the Global Variable TimeOutError% if an IEEE-488 Timeout Error occurs. Functions that have bounded parameters (maximum and/or minimum parameters) will set the Global Variable OutOfRange% if a parameter is not within the limits specified (see Global Variables, Appendix A).

See Model 2001 Minimum and Maximum Sense Constants in Appendix B for CONSTants for use with the Model 2001 SENSe[1] Subsystem commands.

6.1 Model 2001 Function Change Subroutines

Theses subroutines change the present function of the Model 2001.

6.1.1 SUB DCV2001

Description

Puts Model 2001 into DC Volts mode.

Example

```
' BASIC 7.1 or QB45 setup commands called before here
' All three of the following put the Model 2001 in DC Volts:
DCV2001
Set2001Function DCV
Send2001 "FUNC 'VOLT:DC'"
```

6.1.2 SUB ACV2001

Description

Puts Model 2001 into AC Volts mode.

6.1.3 SUB DCA2001

Description

Puts Model 2001 into DC Current mode.

6.1.4 SUB ACA2001

Description

Puts Model 2001 into AC Current mode.

6.1.5 SUB RES2.2001

Description

Puts Model 2001 into 2-wire Resistance mode.

6.1.6 SUB RES4.2001

Description

Puts Model 2001 into 4-wire Resistance mode.

6.1.7 SUB FREQ2001

Description

Puts Model 2001 into Frequency mode.

6.1.8 SUB TEMP2001

Description

Puts Model 2001 into Temperature mode.

6.2 Return Model 2001 Reading Functions

Theses subroutines return the latest reading on the specified function of the Model 2001. All routines use the SCPI command, MEASure:(FunctionName)?, except Get2001Rdg$ which uses the SCPI command, "FETCh?".

6.2.1 FUNCTION Get2001Rdg$

Description

FETCHes a Model 2001 reading in the present Mode and FORMat.

Returns

"Error!" if a TimeOutError occurred, or Reading String.

Example

```
' BASIC 7.1 or QB45 setup commands called before here
' The following two statements are equivalent:
A$ = Get2001Rdg$
B$ = Q2001$("FETCH?")
```

6.2.2 FUNCTION GetDCV2001$

Description

Gets a Model 2001 DC Volts Reading

Returns

"Error!" if a TimeOutError occurred, or DC Volts Reading String.

Example

```
' BASIC 7.1 or QB45 setup commands called before here
' The following two statements are equivalent:
A$ = GetDCV2001$
B$ = Q2001$("MEASURE:VOLT:DC?")
```

6.2.3 FUNCTION GetACV2001$

Description

Gets a Model 2001 AC Volts Reading

Returns

"Error!" if a TimeOutError occurred, or AC Volts Reading String.

6.2.4 FUNCTION GetDCA2001$

Description

Gets a Model 2001 DC Current Reading

Returns

"Error!" if a TimeOutError occurred, or DC Current Reading String.

6.2.5 FUNCTION GetACA2001$

Description

Gets a Model 2001 AC Current Reading

Returns

“Error!” if a TimeOutError occurred, or AC Current Reading String.

6.2.6 FUNCTION Get2RES2001$

Description

Gets a Model 2001 2-wire Resistance Reading

Returns

“Error!” if a TimeOutError occurred, or 2-wire Resistance Reading String.

6.2.7 FUNCTION Get4RES2001$

Description

Gets a Model 2001 4-wire Resistance Reading

Returns

“Error!” if a TimeOutError occurred, or 4-wire Resistance Reading String.

6.2.8 FUNCTION GetFREQ2001$

Description

Gets a Model 2001 Frequency Reading

Returns

“Error!” if a TimeOutError occurred, or Frequency Reading String.

6.2.9 FUNCTION GetTEMP2001$

Description

Gets a Model 2001 Temperature Reading

Returns

“Error!” if a TimeOutError occurred, or Temperature Reading String.

6.3 DC Voltage Functions

These functions setup and return the settings of the configurable options of the Model 2001's DC Voltage measurement function.

6.3.1 SUB Set2001DCV(Range#, Time#, Digits#)

Description

Sets the Model 2001's DC Voltage Range, Aperture Time, and Number of Digits.

Parameters

Range#

0 to +1100
MAXIMUM, MINIMUM, or DEFAULT

Time# (Aperture Time)

166.667e-6s to 166.666667e-3s (60Hz); 2e-4 to .2 (50Hz)
MAXIMUM, MINIMUM, or DEFAULT

Digits#

3.5 to 8.4999 (7.5-8.4999=>7.5d, 3.5-4.4999=>3.5d)
MAXIMUM, MINIMUM, or DEFAULT

Example

```
' BASIC 7.1 or QB45 setup commands called before here
' Set DC Voltage MAXIMUM range, 1 Power Line Cycle
' Integration, and 6-1/2 digits
Set2001DCV MAXIMUM, 1/60, 7
```

6.3.2 FUNCTION Set2001DCVQ$

Description

Queries the Model 2001 for its DC Voltage Range, Aperture Time, and Number of Digits settings.

Returns

"Error!" if a TimeOutError occurred, or
Query 1: Range 0 to +1100
Query 2: Time: 166.667e-6 to .2
Query 3: Digits 4 to 8 (4=3.5d, 8=7.5d, etc.)

Use ParseQuery$ to separate return string into components.

Example

```
' BASIC 7.1 or QB45 setup commands called before here
A$ = Set2001DCVQ$                ' Get 2001 DC Voltage Settings
Range$ = ParseQuery$(A$, 1)      ' Extract Range setting
AperTime$ = ParseQuery$(A$, 2)   ' Extract Aperture Time
Digits$ = ParseQuery$(A$, 3)     ' Extract Number of Digits
```

6.3.3 SUB Auto2001DCV(AutoRange%, AutoTime%, AutoDigits%)

Description

Sets the Model 2001's DC Voltage Auto Range, Auto Time, and Auto Digits settings.

Parameters

AutoRange%, *AutoTime%*, *AutoDigits%*

-1 = Don't set, 0 = Off, +1 = On, +2 = Once

The following constants can also be used:

NO = -1, TOFF = 0, TON = +1, ONCE = +2

Example

```
' BASIC 7.1 or QB45 setup commands called before here
' Turn 2001's DC Voltage Auto Range ON, Time OFF,
' and Digits ONCE:
Auto2001DCV TON, TOFF, ONCE
```

6.3.4 FUNCTION Auto2001DCVQ$

Description

Queries the Model 2001 for its DC Voltage Auto Range, Auto Time, and Auto Digits settings.

Returns

"Error!" if a TimeOutError occurred, or
Query 1: Auto Range: 1 or 0 (1=ON, 0=OFF)
Query 2: Auto Time: 1 or 0 (1=ON, 0=OFF)
Query 3: Auto Digits: 1 or 0 (1=ON, 0=OFF)

Use ParseQuery$ to separate return string into components.

Example

```
' BASIC 7.1 or QB45 setup commands called before here
A$ = Auto2001DCVQ$               ' Get 2001 DC Voltage Auto Settings
AutoRange$ = ParseQuery$(A$, 1)  ' Extract Auto Range setting
```

```
AutoTime$ = ParseQuery$(A$, 2)          ' Extract Auto Time
AutoDigits$ = ParseQuery$(A$, 3)        ' Extract Auto Digits
```

6.4 AC Voltage Functions

These functions setup and return the settings of the configurable options of the Model 2001's AC Voltage measurement function.

6.4.1 SUB Set2001ACV(Range#, Time#, Digits#)

Description

Sets the Model 2001's AC Voltage Range, Aperture Time, and Number of Digits.

Parameters

Range#

0 to +775 (+1100 if Peak, +Peak or -Peak On) MAXIMUM, MINIMUM, or DEFAULT

Time# (Aperture Time)

166.667e-6s to 166.666667e-3s (60Hz); 2e-4 to .2 (50Hz) MAXIMUM, MINIMUM, or DEFAULT

Digits#

3.5 to 8.4999 (7.5-8.4999=>7.5d, 3.5-4.4999=>3.5d) MAXIMUM, MINIMUM, or DEFAULT

Example

```
' BASIC 7.1 or QB45 setup commands called before here
' Set AC Voltage MINIMUM range, 10 Power Line Cycle
' Integration, and 6½ digits
Set2001ACV MINIMUM, 1/6, 6.5
```

6.4.2 FUNCTION Set2001ACVQ$

Description

Queries the Model 2001 for its AC Voltage Range, Aperture Time, and Number of Digits settings.

Returns

"Error!" if a TimeOutError occurred, or
Query 1: Range: 0 to +1100
Query 2: Time: 166.667e-6 to .2
Query 3: Digits: 4 to 8 (4=3.5d, 8=7.5d, etc.)

Use ParseQuery$ to separate return string into components.

Example

```
' BASIC 7.1 or QB45 setup commands called before here
A$ = Set2001ACVQ$                    ' Get 2001 AC Voltage Settings
Range# = VAL(ParseQuery$(A$, 1))     ' Extract Range setting
AperTime# = VAL(ParseQuery$(A$, 2))  ' Extract Aperture Time
Digits# = VAL(ParseQuery$(A$, 3))    ' Extract Number of Digits
```

6.4.3 SUB Auto2001ACV(AutoRange%, AutoTime%, AutoDigits%)

Description

Sets the Model 2001's AC Voltage Auto Range, Auto Time, and Auto Digits settings.

Parameters

AutoRange%, *AutoTime%*, *AutoDigits%*

-1 = Don't set, 0 = Off, +1 = On, +2 = Once

The following constants can also be used:

NO = -1, TOFF = 0, TON = +1, ONCE = +2

Example

```
' BASIC 7.1 or QB45 setup commands called before here
' Turn Model 2001's AC Voltage Auto Range ON, Time OFF,
' and Digits unaffected:
Auto2001ACV TON, TOFF, NO
```

6.4.4 FUNCTION Auto2001ACVQ$

Description

Queries the Model 2001 for its AC Voltage Auto Range, Auto Time, and Auto Digits settings.

Returns

"Error!" if a TimeOutError occurred, or
Query 1: Auto Range: 1 or 0 (1=ON, 0=OFF)
Query 2: Auto Time: 1 or 0 (1=ON, 0=OFF)
Query 3: Auto Digits: 1 or 0 (1=ON, 0=OFF)

Use ParseQuery$ to separate return string into components.

Example

```
' BASIC 7.1 or QB45 setup commands called before here
A$ = Auto2001ACVQ$                   ' Get 2001 AC Voltage Auto Settings
AutoRange$ = ParseQuery$(A$, 1)      ' Extract Auto Range setting
```

```
AutoTime$ = ParseQuery$(A$, 2)          ' Extract Auto Time
AutoDigits$ = ParseQuery$(A$, 3)        ' Extract Auto Digits
```

6.5 DC Current Functions

These functions setup and return the settings of the configurable options of the 2001's DC Current measurement function.

6.5.1 SUB Set2001DCA(Range#, Time#, Digits#)

Description

Sets the Model 2001's DC Current Range, Aperture Time, and Number of Digits.

Parameters

Range#

0 to +2.1, ignored if In Circuit Mode is enabled. MAXIMUM, MINIMUM, or DEFAULT

Time# (Aperture Time)

166.667e-6s to 166.666667e-3s (60Hz); 2e-4 to .2 (50Hz) MAXIMUM, MINIMUM, or DEFAULT

Digits#

3.5 to 8.4999 (7.5-8.4999=>7.5d, 3.5-4.4999=>3.5d) MAXIMUM, MINIMUM, or DEFAULT

Example

```
' BASIC 7.1 or QB45 setup commands called before here
' Set DC Current to .2A range, 10ms Integration, & 6-1/2d
Set2001DCA .2, .01, 7.1
```

6.5.2 FUNCTION Set2001DCAQ$

Description

Queries the Model 2001 for its DC Current Range, Aperture Time, and Number of Digits settings.

Returns

"Error!" if a TimeOutError occurred, or
Query 1: Range: 0 to +2.1
Query 2: Time: 166.667e-6 to .2
Query 3: Digits: 4 to 8 (4=3.5d, 8=7.5d, etc.)

Use ParseQuery$ to separate return string into components.

Example

```
' BASIC 7.1 or QB45 setup commands called before here
A$ = Set2001DCAQ$                ' Get 2001 DC Current Settings
Range$ = ParseQuery$(A$, 1)      ' Extract Range setting
AperTime$ = ParseQuery$(A$, 2)   ' Extract Aperture Time
Digits$ = ParseQuery$(A$, 3)     ' Extract Number of Digits
```

6.5.3 SUB Auto2001DCA(AutoRange%, AutoTime%, AutoDigits%)

Description

Sets the Model 2001's DC Current Auto Range, Auto Time, and Auto Digits settings.

Parameters

AutoRange%, *AutoTime%*, *AutoDigits%*

-1 = Don't set, 0 = Off, +1 = On, +2 = Once

The following constants can also be used:

NO = -1, TOFF = 0, TON = +1, ONCE = +2

Example

```
 BASIC 7.1 or QB45 setup commands called before here
' Turn DC Current Auto Range ON, Time OFF, and Digits ONCE:
Auto2001DCA 1, 0, 2
```

6.5.4 FUNCTION Auto2001DCAQ$

Description

Queries the Model 2001 for its DC Current Auto Range, Auto Time, and Auto Digits settings.

Returns

"Error!" if a TimeOutError occurred, or
Query 1: Auto Range: 1 or 0 (1=ON, 0=OFF)
Query 2: Auto Time: 1 or 0 (1=ON, 0=OFF)
Query 3: Auto Digits: 1 or 0 (1=ON, 0=OFF)

Use ParseQuery$ to separate return string into components.

Example

```
' BASIC 7.1 or QB45 setup commands called before here
A$ = Auto2001DCAQ$                ' Get 2001 DC Current Auto Settings
AutoRange$ = ParseQuery$(A$, 1)   ' Extract Auto Range setting
AutoTime$ = ParseQuery$(A$, 2)    ' Extract Auto Time
AutoDigits$ = ParseQuery$(A$, 3)  ' Extract Auto Digits
```

6.6 AC Current Functions

These functions setup and return the settings of the configurable options of the Model 2001's AC Current measurement function.

6.6.1 SUB Set2001ACA(Range#, Time#, Digits#)

Description

Sets the Model 2001's AC Current Range, Aperture Time, and Number of Digits.

Parameters

Range#

0 to +2.1 MAXIMUM, MINIMUM, or DEFAULT

Time# (Aperture Time)

166.667e-6s to 166.666667e-3s (60Hz); 2e-4 to .2 (50Hz) MAXIMUM, MINIMUM, or DEFAULT

Digits#

3.5 to 8.4999 (7.5-8.4999=>7.5d, 3.5-4.4999=>3.5d) MAXIMUM, MINIMUM, or DEFAULT

Example

```
' BASIC 7.1 or QB45 setup commands called before here
' Set AC Current to 2mA range, .1 msec
' Integration, and 7-1/2 digits
Set2001ACA 2e-3, 1e-4, 7.5
```

6.6.2 FUNCTION Set2001ACAQ$

Description

Queries the Model 2001 for its AC Current Range, Aperture Time, and Number of Digits settings.

Returns

"Error!" if a TimeOutError occurred, or
Query 1: Range: 0 to +2.1
Query 2: Time: 166.667e-6 to .2
Query 3: Digits: 4 to 8 (4=3.5d, 8=7.5d, etc.)

Use ParseQuery$ to separate return string into components.

Example

```
' BASIC 7.1 or QB45 setup commands called before here
A$ = Set2001ACAQ$                    ' Get 2001 AC Current Settings
```

```
Range$ = ParseQuery$(A$, 1)          ' Extract Range setting
AperTime$ = ParseQuery$(A$, 2)       ' Extract Aperture Time
Digits$ = ParseQuery$(A$, 3)         ' Extract Number of Digits
```

6.6.3 SUB Auto2001ACA(AutoRange%, AutoTime%, AutoDigits%)

Description

Sets the Model 2001's AC Current Auto Range, Auto Time, and Auto Digits settings.

Parameters

AutoRange%, *AutoTime%*, *AutoDigits%*

-1 = Don't set, 0 = Off, +1 = On, +2 = Once

The following constants can also be used:

NO = -1, TOFF = 0, TON = +1, ONCE = +2

Example

```
' BASIC 7.1 or QB45 setup commands called before here
' Turn 2001's AC Current Auto Range ON, Time OFF,
' and Digits unaffected:
Auto2001ACA TON, 0, -1
```

6.6.4 FUNCTION Auto2001ACAQ$

Description

Queries the Model 2001 for its AC Current Auto Range, Auto Time, and Auto Digits settings.

Returns

"Error!" if a TimeOutError occurred, or
Query 1: Auto Range: 1 or 0 (1=ON, 0=OFF)
Query 2: Auto Time: 1 or 0 (1=ON, 0=OFF)
Query 3: Auto Digits: 1 or 0 (1=ON, 0=OFF)

Use ParseQuery$ to separate return string into components.

Example

```
' Basic 7.1 or QB45 setup commands called before here
A$ = Auto2001ACAQ$                   ' Get 2001 AC Current Auto Settings
AutoRange$ = ParseQuery$(A$, 1)      ' Extract Auto Range setting
AutoTime$ = ParseQuery$(A$, 2)       ' Extract Auto Time
AutoDigits$ = ParseQuery$(A$, 3)     ' Extract Auto Digits
```

6.7 Two-Wire Resistance Functions

These functions setup and return the settings of the configurable options of the Model 2001's 2-wire Resistance measurement function.

6.7.1 SUB Set2001R2(Range#, Time#, Digits#)

Description

Sets the Model 2001's 2-wire Resistance Range, Aperture Time, and Number of Digits.

Parameters

Range#

0 to +1.05e9 or 2.1e5 if SENSe1:RESistance:OCOMpensated ON is set.
MAXIMUM, MINIMUM, or DEFAULT

Time# (Aperture Time)

166.667e-6s to 166.666667e-3s (60Hz); 2e-4 to .2 (50Hz) MAXIMUM, MINIMUM, or DEFAULT

Digits#

3.5 to 8.4999 (7.5-8.4999≥7.5d, 3.5-4.4999≥3.5d) MAXIMUM, MINIMUM, or DEFAULT

Example

```
' BASIC 7.1 or QB45 setup commands called before here
' Set 2-wire Resistance to 200kΩ range, 100 msec
' Integration, and 5-1/2 digits
Set2001R2 1e5, .1, 6.49
```

6.7.2 FUNCTION Set2001R2Q$

Description

Queries the Model 2001 for its 2-wire Resistance Range, Aperture Time, and Number of Digits settings.

Returns

"Error!" if a TimeOutError occurred, or
Query 1: Range: 0 to +1.05e9
Query 2: Time: 166.667e-6 to .2
Query 3: Digits: 4 to 8 (4=3.5d, 8=7.5d, etc.)

Use ParseQuery$ to separate return string into components.

Example

```
' BASIC 7.1 or QB45 setup commands called before here
A$ = Set2001R2Q$                    ' Get 2001 2-wire Resistance settings
```

```
Range$ = ParseQuery$(A$, 1)          ' Extract Range setting
AperTime$ = ParseQuery$(A$, 2)       ' Extract Aperture Time
Digits$ = ParseQuery$(A$, 3)         ' Extract Number of Digits
```

6.7.3 SUB Auto2001R2(AutoRange%, AutoTime%, AutoDigits%)

Description

Sets the Model 2001's 2-wire Resistance Auto Range, Auto Time, and Auto Digits settings.

Parameters

AutoRange%, *AutoTime%*, *AutoDigits%*

-1 = Don't set, 0 = Off, +1 = On, +2 = Once

The following constants can also be used:

NO = -1, TOFF = 0, TON = +1, ONCE = +2

Example

```
' BASIC 7.1 or QB45 setup commands called before here
' Turn 2001's 2-wire Resistance Auto Range ON, Time OFF,
' and Digits ONCE:
Auto2001R2 TON, TOFF, ONCE
```

6.7.4 FUNCTION Auto2001R2Q$

Description

Queries the Model 2001 for its 2-wire Resistance Auto Range, Auto Time, and Auto Digits settings.

Returns

"Error!" if a TimeOutError occurred, or
Query 1: Auto Range: 1 or 0 (1=ON, 0=OFF)
Query 2: Auto Time: 1 or 0 (1=ON, 0=OFF)
Query 3: Auto Digits: 1 or 0 (1=ON, 0=OFF)

Use ParseQuery$ to separate return string into components.

Example

```
' BASIC 7.1 or QB45 setup commands called before here
A$ = Auto2001R2Q$                    ' Get 2001 2-wire Resistance Auto Settings
AutoRange$ = ParseQuery$(A$, 1)      ' Extract Auto Range setting
AutoTime$ = ParseQuery$(A$, 2)       ' Extract Auto Time
AutoDigits$ = ParseQuery$(A$, 3)     ' Extract Auto Digits
```

6.8 Four-Wire Resistance Functions

These functions setup and return the settings of the configurable options of the Model 2001's 4-wire Resistance measurement function.

6.8.1 SUB Set2001R4(Range#, Time#, Digits#)

Description

Sets the Model 2001's 4-wire Resistance Range, Aperture Time, and Number of Digits.

Parameters

Range#

0 to 2.1e5
MAXIMUM, MINIMUM, or DEFAULT

Time# (Aperture Time)

166.667e-6s to 166.666667e-3s (60Hz); 2e-4 to .2 (50Hz) MAXIMUM, MINIMUM, or DEFAULT

Digits#

3.5 to 8.4999 (7.5-8.4999≥7.5d, 3.5-4.4999≥3.5d) MAXIMUM, MINIMUM, or DEFAULT

Example

```
' BASIC 7.1 or QB45 setup commands called before here
' Set 4-wire Resistance to 20kΩ range, 100 msec
' Integration, and 5-1/2 digits
Set2001R4 15000#, 100d-3, 5.649
```

6.8.2 FUNCTION Set2001R4Q$

Description

Queries the Model 2001 for its 4-wire Resistance Range, Aperture Time, and Number of Digits settings.

Returns

"Error!" if a TimeOutError occurred, or
Query 1: Range: 0 to +2.1e5
Query 2: Time: 166.667e-6 to .2
Query 3: Digits: 4 to 8 (4=3.5d, 8=7.5d, etc.)

Use ParseQuery$ to separate return string into components.

Example

```
' BASIC 7.1 or QB45 setup commands called before here
```

```
A$ = Set2001R4Q$                ' Get 2001 4-wire Resistance settings
Range$ = ParseQuery$(A$, 1)     ' Extract Range setting
AperTime$ = ParseQuery$(A$, 2)  ' Extract Aperture Time
Digits$ = ParseQuery$(A$, 3)    ' Extract Number of Digits
```

6.8.3 SUB Auto2001R4(AutoRange%, AutoTime%, AutoDigits%)

Description

Sets the Model 2001's 4-wire Resistance Auto Range, Auto Time, and Auto Digits settings.

Parameters

AutoRange%, *AutoTime%*, *AutoDigits%*

-1 = Don't set, 0 = Off, +1 = On, +2 = Once

The following constants can also be used:

NO = -1, TOFF = 0, TON = +1, ONCE = +2

Example

```
' BASIC 7.1 or QB45 setup commands called before here
' Turn 2001's 4-wire Resistance Auto Range ON, Time OFF,
' and Digits ONCE:
Auto2001R4 TON, TOFF, ONCE
```

6.8.4 FUNCTION Auto2001R4Q$

Description

Queries the Model 2001 for its 4-wire Resistance Auto Range, Auto Time, and Auto Digits settings.

Returns

"Error!" if a TimeOutError occurred, or
Query 1: Auto Range: 1 or 0 (1=ON, 0=OFF)
Query 2: Auto Time: 1 or 0 (1=ON, 0=OFF)
Query 3: Auto Digits: 1 or 0 (1=ON, 0=OFF)

Use ParseQuery$ to separate return string into components.

Example

```
' BASIC 7.1 or QB45 setup commands called before here
A$ = Auto2001R4Q$                ' Get 2001 4-wire Resistance Auto Settings
AutoRange$ = ParseQuery$(A$, 1)  ' Extract Auto Range setting
AutoTime$ = ParseQuery$(A$, 2)   ' Extract Auto Time
AutoDigits$ = ParseQuery$(A$, 3) ' Extract Auto Digits
```

6.9 Frequency Functions

These functions setup and return the settings of the configurable options of the Model 2001's Frequency measurement function.

6.9.1 SUB Set2001F(Digits#, Source%)

Description

Sets the Model 2001's Frequency Number of Digits and Source settings.

Parameters

Digits#

3.5 to 5.4999 (4.5-5.4999≥4.5d, 3.5-4.4999≥3.5d) MAXIMUM, MINIMUM, or DEFAULT

Source%

0 — Current
1 — Voltage

Example

```
' BASIC 7.1 or QB45 setup commands called before here
' Set Frequency to 5 digits, Current triggered
Set2001F 5.49, 0
```

6.9.2 FUNCTION Set2001FQ$

Description

Queries the Model 2001 for its Frequency Number of Digits and Measurement Source settings.

Returns

"Error!" if a TimeOutError occurred, or
Query 1: Digits: 4 to 5 (4=3.5d, 5=4.5d)
Query 2: Source: VOLT or CURR

Use ParseQuery$ to separate return string into components.

Example

```
' BASIC 7.1 or QB45 setup commands called before here
A$ = Set2001FQ$                  ' Get Model 2001 Frequency settings
Digits$ = ParseQuery$(A$, 1)     ' Extract Number of Digits
Source$ = ParseQuery$(A$, 2)     ' Extract Frequency Source
```

6.10 Temperature Functions

These functions setup and return the settings of the configurable options of the Model 2001's Temperature measurement function.

6.10.1 SUB Set2001T(Time#, Digits#)

Description

Sets the Model 2001's Temperature Aperture Time and Number of Digits.

Parameters

Time# (Aperture time)

166.667e-6s to 166.666667e-3s (60Hz); 2e-4 to .2 (50Hz)
MAXIMUM, MINIMUM, or DEFAULT

Digits#

3.5 to 7.4999 (6.5-7.4999≥6.5d, 3.5-4.4999≥3.5d)
MAXIMUM, MINIMUM, or DEFAULT

Example

```
' BASIC 7.1 or QB45 setup commands called before here
' Set Temperature to 100msec Integration, and 5½ digits
Set2001T .1, 6.49
```

6.10.2 FUNCTION Set2001TQ$

Description

Queries the Model 2001 for its Temperature Aperture Time and Number of Digits settings.

Returns

"Error!" if a TimeOutError occurred, or
Query 1: Time: 166.667e-6 to .2
Query 2: Digits: 4 to 7 (4=3.5d, 7=6.5d, etc.)

Use ParseQuery$ to separate return string into components.

Example

```
' BASIC 7.1 or QB45 setup commands called before here
A$ = Set2001TQ$                    ' Get Model 2001 Temperature settings
AperTime$ = ParseQuery$(A$, 1)     ' Extract Aperture Time
Digits$ = ParseQuery$(A$, 2)       ' Extract Number of Digits
```

6.10.3 SUB Auto2001T(AutoTime%, AutoDigits%)

Description

Set the Model 2001's Temperature Auto Time and Auto Digits settings.

Parameters

AutoTime%, *AutoDigits%*

-1 = Don't set, 0 = Off, +1 = On, +2 = Once
The following constants can also be used:
NO = -1, TOFF = 0, TON = +1, ONCE = +2

Example

```
' BASIC 7.1 or QB45 setup commands called before here
' Turn Model 2001's Temperature Auto Time OFF, and Digits ONCE:
Auto2001T TOFF, ONCE
```

6.10.4 FUNCTION Auto2001TQ$

Description

Queries the Model 2001 for its Temperature Auto Time and Auto Digits settings.

Returns

"Error!" if a TimeOutError occurred, or
Query 1: Auto Time: 1 or 0 (1=ON, 0=OFF)
Query 2: Auto Digits: 1 or 0 (1=ON, 0=OFF)

Use ParseQuery$ to separate return string into components.

Example

```
' Basic 7.1 or QB45 setup commands called before here
A$ = Auto2001TQ$                    ' Get Model 2001 Temperature Auto Settings
AutoTime$ = ParseQuery$(A$, 1)      ' Extract Auto Time
AutoDigits$ = ParseQuery$(A$, 2)    ' Extract Auto Digits
```

6.10.5 SUB Set2001RTD(Mode%, RType%, Alpha#, Beta#, Delta#, RZero#)

Description

Configure and use RTDs to make temperature measurements.

Parameters

Mode% 2 or 4 (2-wire or 4-wire RTD)
Type% 0 = PT385, 1 = PT3916, 2 = USER
Alpha# 0.00 to 0.01 (ignored if not USER)
Beta# 0.00 to 1.00 (ignored if not USER)
Delta# 0.00 to 5.00 (ignored if not USER)
RZero# 0.0 to 1000.0 (ignored if not USER)

Example

```
' BASIC 7.1 or QB45 setup commands called before here
' Set RTD Mode to 4-wire RTD, User, Alpha=.005, Beta= .5
' Delta=2.4, and RZero=500
Set2001RTD 4, 2, .005, .5, 2.4, 500
```

6.10.6 FUNCTION Set2001RTDQ$

Description

Queries the Model 2001 for the Temperature transducer Type, the RTD Type, Alpha, Beta, Delta, and RZero settings.

Returns

"Error!" if a TimeOutError occurred, or
Query 1: Mode: RTD or FRTD or TC
Query 2: RType: USER, PT385, or PT3916
Query 3: Alpha: 0.00 to 0.01
Query 4: Beta: 0.00 to 1.00
Query 5: Delta: 0.00 to 5.00
Query 6: RZero: 0 to 1000

Use ParseQuery$ to separate return string into components.

Example

```
' BASIC 7.1 or QB45 setup commands called before here
A$ = Set2001RTDQ$                       ' Get 2001 RTD settings
Mode$ = ParseQuery$(A$, 1)              ' Extract Temp. Device
RType$ = ParseQuery$(A$, 2)             ' Extract RTD Type
Alpha# = VAL(ParseQuery$(A$, 3))        ' Extract Alpha setting
Beta# = VAL(ParseQuery$(A$, 4))         ' Extract Beta setting
Delta# = VAL(ParseQuery$(A$, 5))        ' Extract Delta setting
RZero# = VAL(ParseQuery$(A$, 6))        ' Extract RZero setting
```

6.10.7 SUB Set2001TC(TType$)

Description

Sets the thermocouple type and uses TC's for temperature measurement.

Parameters

TType$ "J", "K", "T", "E", "R", "S", or "B"

Example

```
' BASIC 7.1 or QB45 setup commands called before here
' Use Type J thermocouples
Set2001TC "J"
```

6.10.8 FUNCTION Set2001TCQ$

Description

Queries the Model 2001 for the Thermocouple Type.

Returns

"Error!" if a TimeOutError occurred, or "J", "K", "T", "E", "R", "S", or "B"

Example

```
' BASIC 7.1 or QB45 setup commands called before here
A$ = Set2001TCQ$ ' Get Model 2001 Thermocouple Type
```

Section 7
Model 2001 / 7001 Status Commands

These routines control the SCPI Status Model of the Model 2001/7001. Constants (see Status Model Constants in Appendix B) are defined for all of the registers and their bits to make programming the Model 2001/7001 status model simpler. Please refer to the IEEE-488 Reference Section of the Model 2001 or 7001 manual for more details on the specific bit patterns of these registers and bit patterns.

NOTE

These functions and subroutines also support the Model 7002 Switch System. Just send Model 7001 commands to a Model 7002.

All of these routines will set the Global Variable TimeOutError% if an IEEE-488 Timeout Error occurs. Functions that have bounded parameters (maximum and/or minimum parameters) will set the Global Variable OutOfRange% if a parameter is not within the limits specified (see Global Variables, Appendix A).

7.1 SUB Stat2001(Event2001%, PTF%, NTF%, SEN%) SUB Stat7001(Event7001%, PTF%, NTF%, SEN%)

Description

Sets the Model 2001/7001's Operation, Trigger, Arm, Sequence, Questionable, or Measurement Events' Positive Transition Filter, Negative Transition Filter, and Status Enable Registers.

Parameters

Event2001%, *Event7001%*

1 = Operation Event
2 = Trigger Event
3 = Arm Event
4 = Sequence Event
5 = Questionable Event
6 = Measurement Event (Model 2001 Only)

PTF% (Positive Transition Filter)	0 to 32767
NTF% (Negative Transition Filter)	0 to 32767
SEN% (Status Enable Register)	0 to 32767

Example

```
' BASIC7.1 or QB45 setup commands called before here
' Cause the TRIG and/or ARM bits to be set in the 2001's
' Measurement Event Register when the Model 2001 enters or exits
' Triggering or Arming
Stat2001 MEASUREMENT, TRIG + ARM, TRIG + ARM, TRIG + ARM
```

7.2 FUNCTION Stat2001Q$(Event2001%)
FUNCTION Stat7001Q$(Event7001%)

Description

Queries the Model 2001/7001's Operation, Trigger, Arm, Sequence, Questionable, or Measurement Events' Status, Condition, Positive Transition Filter, Negative Transition Filter, and Status Enable Registers.

Parameters

Event2001%, *Event7001%*

1 = Operation Event
2 = Trigger Event
3 = Arm Event
4 = Sequence Event
5 = Questionable Event
6 = Measurement Event (Model 2001 Only)

Returns

"Error!" if a TimeOutError occurred, or
Query 1: Status Register (0 to 32767)
Query 2: Condition Register (0 to 32767)
Query 3: Positive Transition Filter (0 to 32767)
Query 4: Negative Transition Filter (0 to 32767)
Query 5: Status Enable Register (0 to 32767)

Use ParseQuery$ to separate return string into components.

Example

```
' BASIC 7.1 or QB45 setup commands called before here
A$ = Stat7001Q$(OPERATION)          ' Read Model 7001's Operation Event Registers
StatReg$ = ParseQuery$(A$, 1)       ' Extract Status Register
CondReg$ = ParseQuery$(A$, 2)       ' Extract Condition Register
PTranReg$ = ParseQuery$(A$, 3)      ' Extract +Transition Reg.
NTranReg$ = ParseQuery$(A$, 4)      ' Extract -Transition Reg.
SEnableReg$ = ParseQuery$(A$, 5)    ' Status Enable Reg.
```

7.3 SUB SRE2001(mask%)
SUB SRE7001(mask%)

Description

Enables the Model 2001/7001 to generate a Service Request when the indicated bit(s) of the Status Byte Register are set. If a bit is already set when this command is given, no Service Request is generated.

Parameters

mask% 0 to 255

Example

```
' BASIC 7.1 or QB45 setup commands called before here
Poll% = Poll2001%                    ' Clear any pending SRQ's
SRE2001 MAV                          ' Set up to SRQ on MAV
Send2001 "FETCH?"                    ' fetch a 2001 reading
DO                                   ' Set up an uninterruptable wait
LOOP UNTIL Wait2001RQS%
Reading$ = Q2001$("")                ' Get reading
```

7.4 SUB ESE2001(mask%)
SUB ESE7001(mask%)

Description

Sets which bits of the Model 2001/7001's Standard Event Status Register cause the Event Summary Bit (ESB) of the Status Byte Register to be set.

Parameters

mask% 0 to 255

Example

```
' BASIC 7.1 or QB45 setup commands called before here
' Setup an SRQ on OPeration Complete
ESE2001 OPC
SRE2001 ESB
Send2001 "VOLTage:DC:RANGe 200;*OPC"
RQS2001% = Wait2001RQS%              ' Wait until range change complete
```

7.5 FUNCTION OPC2001$(UnInterruptable%)
FUNCTION OPC7001$(UnInterruptable%)

Description

Performs a Model 2001/7001 *OPC? command which returns a "1" when the present operation is complete. However, the "1" may not be output by the 2001/7001 for a long time, which would cause an IEEE timeout on reading data immediately after sending the *OPC?. Thus, this routine waits for a Model 2001/7001 SRQ on MAV after sending the query. The Model 2001/7001 may wait forever to send the "1" out if :INITiate:CONTinuous ON was explicitly set as in the case of the Factory Defaults. Thus, the :ABORt command would have to be issued before calling these routines.

Parameters

UnInterruptable%

FALSE (0) - the command can be aborted by pressing the Esc key or the right mouse button (BASIC 7.1 only).
TRUE (non-zero) - the command cannot be aborted.

Returns

"Error!" if a TimeOutError occurred, or
"1" if operation was completed
"Cancel" if aborted

Example

```
' BASIC 7.1 or QB45 setup commands called before here
' Guarantee that *OPC? will not wait forever.
IF Q2001$("INIT:CONT?") = "1" THEN Send2001 "ABORT"
Send2001 "VOLTage:DC:RANGe 200"
OPC$=OPC2001$(TRUE)                    ' Wait until range change complete
```

7.6 SUB Clear2001
SUB Clear7001

Description

Sends the SCPI command *CLS (Clear Status Model) to the 2001/7001.

Section 8
Model 2001 / 7001 Scanning Commands

These Routines control the SCPI Trigger Model and Scanning functions of the Model 2001/7001. Please refer to the IEEE-488 Reference Section of the Model 2001 or 7001 manual for more details. All of these routines will set the Global Variable TimeOutError% if an IEEE-488 Timeout Error occurs. Functions that have bounded parameters (maximum and/or minimum parameters) will set the Global Variable OutOfRange% if a parameter is not within the limits specified (see Global Variables in Appendix A).

NOTE

These functions and subroutines also support the Model 7002 Switch System. Just send Model 7001 commands to a Model 7002.

See Model 2001/7001 Scanning Minimum and Maximum Constants in Appendix B for CONSTants for use with the Model 2001/7001 Scanning Commands.

8.1 SUB Close2001(Channel%)

Description

Closes a single channel on the Model 2001 scanner card.

Parameters

Channel% 1 to 10

Example

```
' BASIC 7.1 or QB45 setup commands called before here
' Take Readings on Channel 3
Close2001 3
```

8.2 SUB Close7001(ChanList$)

Description

Closes the specified channels on the Model 7001 scanner card.

Parameters

ChanList$

Any valid SCPI channel-list like (@1!1, 1!2, 1!4:1!10, 2!1!2) with valid channels depending on the scanner card selected. The Model 7001 EAV bit in the serial poll register must be checked to see if the command was accepted, or watch the front panel of the instrument for an error message.

Example

```
' BASIC 7.1 or QB45 setup commands called before here
' Close channels 3 to 6 on card 1
Close7001 "(@1!3:1!6)"
```

8.3 FUNCTION Close2001Q$
FUNCTION Close7001Q$

Description

Queries the Model 2001/7001 for a list of closed channels. The Model 2001 can have only one channel closed at a time on its internal scanner, whereas the Model 7001 many have none or all channels closed.

Returns

"Error!" if a TimeOutError occurred, or a list of closed channels.

Example

```
' BASIC 7.1 or QB45 setup commands called before here
Close2001 2                    ' Close Channel 2
A$ = Close2001Q$               ' A$ = "(@2)"
```

8.4 SUB Open2001(Channel%)

Description

Opens one or all channel(s) on the Model 2001 scanner card.

Parameters

Channel%

0 — Open all channels
1 to 10

Example

```
' BASIC 7.1 or QB45 setup commands called before here
Open2001 1      ' Open channel 1
Open2001 0      ' Open all Model 2001 channels
```

8.5 SUB Open7001(ChanList$)

Description

Opens the specified channels on the 7001 scanner card.

Parameters

ChanList$

Any valid SCPI channel-list with valid channels depending on the scanner card selected. The Model 7001 EAV bit in the serial poll register must be checked to see if the command was accepted, or watch the front panel of the instrument for an error message.

Example

```
' BASIC 7.1 or QB45 setup commands called before here
Open7001 "(@1!3:1!6)"        ' Open channels 3 to 6 on card 1
Open7001 "ALL"               ' Open all 7001 channels
```

8.6 SUB Scan2001(ChanList$)
SUB Scan7001(ChanList$)

Description

Defines the Model 2001's Internal ScanList or the Model 7001's scanlist.

Parameters

ChanList$

Any valid SCPI channel-list like (@1, 2, 4:10) with channels ranging from 1 to 10 for the Model 2001, or like (@1!1:1!40, 2!1:2!40) for the Model 7001. The Model 2001/7001 EAV bit in the serial poll register must be checked to see if the command was accepted, or watch the front panel of the instrument for an error message.

Example

```
' BASIC 7.1 or QB45 setup commands called before here
Scan2001 "(@1:10)"                ' Scan all Model 2001 channels
Scan7001 "(@1!1:1!40, 2!1:2!40)"  ' Scan all Model 7001 channels
```

8.7 SUB Arm2001(Count1#, Source1$, Count2#, Delay2#, Source2$)
SUB Arm7001(Count1#, Source1$, Count2#, Delay2#, Source2$)

Description

Sets up the Model 2001/7001's Arm Layer 1 (Arm Layer) and Arm Layer 2 (Scan Layer)

Parameters

Count1# and *Count2#*

1 to 99999, 9.9e37
MAXIMUM, MINIMUM, DEFAULT, or INF

Source1$* and *Source2$

HOLD — Hold
MANual — Manual
IMMediate — Immediate
TIMEr — Timer (Source2$ only)
BUS — IEEE-488 Bus (GET or *TRG)
TLINk — Trigger Link
EXTernal — External

Delay2#

0 to 999999.999 seconds
MAXIMUM, MINIMUM, or DEFAULT

Example

```
' BASIC 7.1 or QB45 setup commands called before here
' Setup the Model 7001 to do three sets of five scans. Each scan starts
' immediately at 1 hour intervals.
Arm7001 3, "IMM", 5, MINIMUM, "TIMER"
Timers7001 3600, 1.5
```

8.8 FUNCTION Arm2001Q$
FUNCTION Arm7001Q$

Description

Queries the Model 2001/7001 for its Arm Layer 1 (Arm Layer) and Arm Layer 2 (Scan Layer) settings.

Returns

"Error!" if a TimeOutError occurred, or
Query 1: Arm Layer 1 Count (1 to 99999, 9.9e+37)
Query 2: Arm Layer 1 Source (see Arm2001/7001, short form)
Query 3: Arm Layer 2 Count (1 to 99999, 9.9e+37)
Query 4: Arm Layer 2 Source (see Arm2001/7001, short form)
Query 5: Arm Layer 2 Delay (0 to 999999.999)

Use ParseQuery$ to separate return string into components.

Example

```
' BASIC 7.1 or QB45 setup commands called before here
A$ = Arm7001Q$                      ' Read 7001's Arm Layers 1&2 Setup
Count1# = VAL(ParseQuery$(A$, 1))   ' Get Arm Layer 1 Count
Source1$ = ParseQuery$(A$, 2)       ' Get Arm Layer 1 Source
```

```
Count2# = VAL(ParseQuery$(A$, 3))     ' Get Arm Layer 2 Count
Delay2# = VAL(ParseQuery$(A$, 4))     ' Get Arm Layer 2 Delay
Source2$ = ParseQuery$(A$, 5)         ' Get Arm Layer 2 Source
```

8.9 SUB Trig2001(Count1#, Delay1#, Source1$)
SUB Trig7001(Count1#, Delay1#, Source1$)

Description

Sets up the Model 2001/7001's Trigger Sequence (Trigger Layer, i.e., Model 2001 Measure Layer and Model 7001 Channel Layer).

Parameters

Count1#

1 to 99999, 9.9e37
MAXIMUM, MINIMUM, DEFAULT or INF

Delay1#

0 to 999999.999 seconds
MAXIMUM, MINIMUM, or DEFAULT

Source1$

HOLD — Hold
MANual — Manual
IMMediate — Immediate
BUS — IEEE-488 Bus (GET or *TRG)
TLINK — Trigger Link
EXTernal — External

Example

```
' BASIC 7.1 or QB45 setup commands called before here
' Setup the 7001 to scan 40 channels with no delay at
' 1.5 second intervals.
Trig7001 40, MINIMUM,"TIM"
Timers7001 3600, 1.5
```

8.10 FUNCTION Trig2001Q$
FUNCTION Trig7001Q$

Description

Queries the Model 2001/7001 for its Trigger Sequence (Trigger Layer) settings.

Returns

"Error!" if a TimeOutError occurred, or
Query 1: Trigger Sequence 1 Count (1 to 99999, 9.9e+37)
Query 2: Trigger Sequence 1 Source (see Trig2001/7001, short form)
Query 3: Trigger Sequence 1 Delay (0 to 999999.999)
Use ParseQuery$ to separate return string into components.

Example

```
' BASIC 7.1 or QB45 setup commands called before here
A$ = Trig7001Q$                          ' Read 7001's Trigger Setup
Count1# = VAL(ParseQuery$(A$, 1))        ' Get Trigger Count
Delay1# = VAL(ParseQuery$(A$, 2))        ' Get Trigger Delay
Source1$ = ParseQuery$(A$, 3)            ' Get Trigger Source
```

8.11 SUB Timers2001(ArmTimer2#, TrigTimer1#) SUB Timers7001(ArmTimer2#, TrigTimer1#)

Description

Sets the Model 2001/7001's Trigger Model timers in Arm Layer 2 (Scan Layer) and Trigger Sequence 1 (Trigger Layer).

Parameters

ArmTimer2#, TrigTimer1#

.001 to 999999.999
MAXIMUM, MINIMUM, or DEFAULT

Example

```
' BASIC 7.1 or QB45 setup commands called before here
' Scan at one hour intervals, trigger at 1.5 seconds intervals.
Timers7001 3600, 1.5
```

8.12 FUNCTION Timers2001Q$ FUNCTION Timers7001Q$

Description

Queries the Model 2001/7001 for its Arm Layer 2 (Scan Layer) and Trigger Sequence 1 (Trigger Layer) timer settings.

Returns

"Error!" if a TimeOutError occurred, or
Query 1: Arm Layer 2 Timer (see Timers2001/7001)

Query 2: Trigger Sequence Timer (see Timers2001/7001)

Use ParseQuery$ to separate return string into components.

Example

```
' BASIC 7.1 or QB45 setup commands called before here
A$ = Timers7001Q$                       ' Read Arm Layer 1 and Trigger Timers
ArmTimer# = VAL(ParseQuery$(A$, 1))     ' Get Arm Layer 1 Timer
TrigTimer# = VAL(ParseQuery$(A$, 2))    ' Get Trigger Timer
```

8.13 SUB ArmTcon2001(Dir1$, ILine1%, Oline1%, Dir2$, ILine2%, Oline2%) SUB ArmTcon7001(Dir1$, ILine1%, Oline1%, Dir2$, ILine2%, Oline2%)

Description

Sets the Model 2001/7001's Arm Layer 1 (Arm Layer) and Arm Layer 2 (Scan Layer) Trigger configurations. Note that OLine can not be the same as ILine. If they are, the Oline will be made 1 line number (wrapping around to 1 if necessary) higher than the Iline.

Parameters

Dir1$* and *Dir2$

ACCeptor — Disable Source Bypass
SOURce — Enable Source Bypass

ILine1% and *Iline2%*

1 to 6 — Trigger Link input line number

OLine1% and *Oline2%*

1 to 6 — Trigger Link output line number

Example

```
' BASIC 7.1 or QB45 setup commands called before here
' Don't bypass Arm Layer 1 source and use Trigger Link Lines
' 1 and 2 as input and output. Bypass Arm Layer 2 source
' and use Trigger Link lines 3 and 4 as I/O.
ArmTcon7001 "ACC", 1, 2, "SOURCE", 3, 4
```

8.14 FUNCTION ArmTcon2001Q$ FUNCTION ArmTcon7001Q$

Description

Queries the Model 2001/7001 for its Arm Layer 1 (Arm Layer) and Arm Layer 2 (Scan Layer) trigger configuration settings.

Returns

"Error!" if a TimeOutError occurred, or
Query 1: Arm Layer 1 Direction (ACC or SOUR)
Query 2: Arm Layer 1 Input Line (1-6)
Query 3: Arm Layer 1 Output Line (1-6)
Query 4: Arm Layer 2 Direction (ACC or SOUR)
Query 5: Arm Layer 2 Input Line (1-6)
Query 6: Arm Layer 2 Output Line (1-6)

Use ParseQuery$ to separate return string into components.

Example

```
' BASIC 7.1 or QB45 setup commands called before here
A$ = ArmTcon7001Q$                  'Get Arm Layers 1&2 Trigger Configuration
Dir1$ = (ParseQuery$(A$, 1)         ' Get Arm Layer 1 Direction
In1% = VAL(ParseQuery$(A$, 2))      'Get Arm Layer 1 Input Line
Out1% = VAL(ParseQuery$(A$, 3))     'Get Arm Layer 1 Output Line
Dir2$ = (ParseQuery$(A$, 4)         ' Get Arm Layer 2 Direction
In2% = VAL(ParseQuery$(A$, 5))      'Get Arm Layer 2 Input Line
Out2% = VAL(ParseQuery$(A$, 6))     'Get Arm Layer 2 Output Line
```

8.15 SUB TrigTcon2001(Dir1$, Synch1$, ILine1%, Oline1%) SUB TrigTcon7001(Dir1$, Synch1$, ILine1%, Oline1%)

Description

Sets the Model 2001/7001's Trigger Sequence 1 and Trigger Sequence 2 trigger configurations. Note that OLine1% can not be the same as ILine1% if Synch1$="ASYN". If they are, the Oline1% will be made one line number (wrapping around to 1 if necessary) higher than the Iline1%.

Parameters

Dir1$

ACCeptor — Disable Source Bypass
SOURce — Enable Source Bypass

Synch1$

ASYNchronous — Asynchronous Trigger Link
SSYNchronous — Semi-Synchronous Link

ILine1%

1-6 — Trigger Link input line number (I/O if SSYN)

OLine1%

1-6 — Trigger Link output line number (ignored if SSYN)

Example

```
' BASIC 7.1 or QB45 setup commands called before here
' Don't bypass Trigger Sequence 1 source, use the Semi-
' Synchronous Trigger Link, and use Trigger Link Line 5 as
' both input and output.
TrigTcon7001 "ACC", "SSYN", 5, 6
```

8.16 FUNCTION TrigTcon2001Q$
FUNCTION TrigTcon7001Q$

Description

Queries the Model 2001/7001 for its Trigger Sequence 1 (Trigger Layer, i.e., Model 2001 Measure Layer and Model 7001 Channel Layer) trigger configuration settings.

Returns

"Error!" if a TimeOutError occurred, or
Query 1: Trigger Sequence 1 Direction (ACC or SOUR)
Query 2: Trigger Sequence 1 Trigger Link Mode (ASYN or SSYN)
Query 3: Trigger Sequence 1 Input Line (1-6)
Query 4: Trigger Sequence 1 Output Line (1-6, 0 if SSYN)

Use ParseQuery$ to separate return string into components.

Example

```
' BASIC 7.1 or QB45 setup commands called before here
A$ = TrigTcon7001Q$                 ' Get Trigger Layer setup from 7001
Dir1$ = (ParseQuery$(A$, 1))        ' Get Trigger Layer Direction
Syn1$ = (ParseQuery$(A$, 2))        ' Get Trigger Layer Protocol
In1% = VAL(ParseQuery$(A$, 3))      ' Get Trigger Layer Input Line
Out1% = VAL(ParseQuery$(A$, 4))     ' Get Trigger Layer Output Line
```

Section 9
Model 2001 Calculate Commands

These Routines control the Model 2001's Calculate subsystem capabilities, which include limit testing, mX+B, percent, and data buffer statistics.

All of these routines will set the Global Variable TimeOutError% if an IEEE-488 Timeout Error occurs. Functions that have bounded parameters (maximum and/or minimum parameters) will set the Global Variable OutOfRange% if a parameter is not within the limits specified (see Global Variables in Appendix A).

See Model 2001 Minimum and Maximum Calculate Constants in Appendix B for CONSTants for use with the Model 2001 Calculate commands.

9.1 SUB Set2001Calc1MXB(MMFactor#, MBFactor#)

Description

Sets the Model 2001's CALCulate1 subsystem's Mx+B coefficients and enables the Mx+B mode of operation.

Parameters

MMFactor#

-9.999999999e20 to +9.999999999e20

MBFactor#

-9.999999999e30 to +9.999999999e30

Example

```
' BASIC 7.1 or QB45 setup commands called before here
' Set and use mX+b with slope of 10 and intercept of 5
Set2001Calc1MXB 10, 5
```

9.2 SUB Set2001Calc1PERC(Percent#)

Description

Sets the Model 2001's CALCulate1 Subsystem's Percentage target and enables the percentage mode of operation.

Parameters

Percent# -9.999999999e35 to +9.999999999e35

Example

```
' BASIC 7.1 or QB45 setup commands called before here
' Set 100% of full scale at 15.24
Set2001Calc1PERC 15.24
```

9.3 SUB Calc1.2001(State%)

Description

Sets the state of the Model 2001's CALCulate1 Subsystem.

Parameters

State%

0, TOFF = Turn CALC1 Off
+1, TON = Turn CALC1 On

Example

```
' BASIC 7.1 or QB45 setup commands called before here
Calc1.2001 TOFF                ' Turn off CALC1 System
```

9.4 FUNCTION Set2001Calc1Q$

Description

Queries the Model 2001 for its CALCulate1 Subsystem's State, MMFactor, MBFactor, and Percent settings.

Returns

"Error!" if a TimeOutError occurred, or
Query 1: CALCulate1's State (1 or 0)
Query 2: CALCulate1's MMFactor (see Set2001Calc1MXB)
Query 3: CALCulate1's MBFactor (see Set2001Calc1MXB)
Query 4: CALCulate1's Percent (see Set2001Calc1PERC)

Use ParseQuery$ to separate return string into components.

Example:

```
' BASIC 7.1 or QB45 setup commands called before here
A$ = Set2001Calc1Q$                    ' Get 2001 CALC1 settings
Calc1State$ = ParseQuery$(A$, 1)       ' Get CALC1's State
```

```
MMFactor# = VAL(ParseQuery$(A$, 2))          'Get slope of mX+b
MBFactor# = VAL(ParseQuery$(A$, 3))          ' Get intercept of mX+b
Percent# = VAL(ParseQuery$(A$, 4))           ' Get 100% value
```

9.5 FUNCTION Calc1.2001Q$

Description

Queries the Model 2001 for the result of the latest CALCulate1 calculation.

Returns

"Error!" if a TimeOutError occurred, or Model 2001's present CALCulate1 calculation.

Example

```
' BASIC 7.1 or QB45 setup commands called before here
Set2001Calc1MXB 10, 5
' Reading from SENSe:DATA? is 10.4
PRINT Calc1.2001Q$                    ' Outputs 109
```

9.6 SUB Set2001Calc2(Format$)

Description

Sets up the Model 2001 Data Buffer's Format and activates the CALCulate2 subsystem.

Parameters

Format$

NONE
MEAN
SDEViation (Standard Deviation)
MAXimum
MINimum
PKPK (Peak to Peak)

Example

```
' BASIC 7.1 or QB45 setup commands called before here
Set2001Calc2 "MAXIMUM"          ' calculate maximum of data buffer
Calc2.2001 TON
' Data buffer contains 1.122, 10.211, 10.2222
' A$ = Calc2.2001Q$
' A$ = "10.2222"
```

9.7 SUB Calc2.2001(State%)

Description

Sets the state of the Model 2001's CALCulate2 Subsystem.

Parameters

State%

0, TOFF = Turn CALC2 Off
+1, TON = Turn CALC2 On

Example

```
' BASIC 7.1 or QB45 setup commands called before here
Calc2.2001 0                    ' Turn off CALC2 System
```

9.8 FUNCTION Set2001Calc2Q$

Description

Queries the Model 2001 for its CALCulate2 subsystem's Format and State.

Returns

"Error!" if a TimeOutError occurred, or
Query 1: CALCulate 2 Format (see Set2001Calc2 above)
Query 2: CALCulate 2 State (0 or 1)

Use ParseQuery$ to separate return string into components.

Example

```
' BASIC 7.1 or QB45 setup commands called before here
A$ = Set2001Calc2Q$             ' Get 2001 CALC2 settings
Format$ = ParseQuery$(A$, 1)    ' Get CALC2's Format
Calc2State$ = ParseQuery$(A$, 2)    ' Get CALC2's State
```

9.9 FUNCTION Calc2.2001Q$

Description

Queries the Model 2001 for the result of the present CALCulate2 subsystem's calculation.

Returns

"Error!" if a TimeOutError occurred, or the result of the present CALC2 calculation.

Example

```
' BASIC 7.1 or QB45 setup commands called before here
A$ = Calc2.2001Q$                    ' Get 2001 CALC2's last result
```

9.10 SUB Set2001Calc3(Upper1#, Lower1#, Upper2#, Lower2#)

Description

Sets the Model 2001's CALCulate3 subsystems Limit Testing Hi/Lo Limits.

Parameters

Upper1#, Upper2#, Lower1#, Lower2#

-9.999999999e35 to +9.999999999e35
MAXIMUM, MINIMUM, or DEFAULT

Example

```
' BASIC 7.1 or QB45 setup commands called before here
' Setup Limit Testing Ranges
' Limit 1 Range: -100.56 to -10.5 or 10.5 to 100.56
' Limit 2 Range: <-100.56 or >100.56
' Pass Range: -10.5 to 10.5
Set2001Calc3 10.5, -10.5, 100.56, -100.56
```

9.11 FUNCTION Set2001Calc3Q$

Description

Queries the Model 2001 for its CALCulate3 subsystems Limit Testing Hi/Lo Limits.

Returns

"Error!" if a TimeOutError occurred, or
Query 1: limit testing Upper Limit 1
Query 2: limit testing Lower Limit 1
Query 3: limit testing Upper Limit 2
Query 4: limit testing Lower Limit 2

Use ParseQuery$ to separate return string into components.

Example

```
' BASIC 7.1 or QB45 setup commands called before here
A$ = Set2001Calc3Q$                  ' Get Model 2001 CALC3 settings
Upper1# = VAL(ParseQuery$(A$, 1))    ' Get Upper Limit 1
Lower1# = VAL(ParseQuery$(A$, 2))    ' Get Lower Limit 1
Upper2# = VAL(ParseQuery$(A$, 3))    ' Get Upper Limit 2
Lower2# = VAL(ParseQuery$(A$, 4))    ' Get Lower Limit 2
```

9.12 SUB Calc3.2001(State%)

Description

Sets the state of the Model 2001's CALCulate3 Subsystem.

Parameters

State%

0, TOFF = Turn CALC3 Off
+1, TON = Turn CALC3 On

Example

```
' BASIC 7.1 or QB45 setup commands called before here
Calc3.2001 TON                    ' Turn on CALC3 System
```

9.13 SUB Set2001Calc3Dig(Du1%, Dl1%, Du2%, Dl2%)

Description

Sets the Model 2001's CALCulate3's Limit Testing HI/Lo Digital Output values.

Parameters

Du1%	Value (0-15) to place on digital output to indicate Upper 1 Limit Reached
Dl1%	Value (0-15) to place on digital output to indicate Lower 1 Limit Reached
Du2%	Value (0-15) to place on digital output to indicate Upper 2 Limit Reached
Dl2%	Value (0-15) to place on digital output to indicate Lower 2 Limit Reached

Example

```
' BASIC 7.1 or QB45 setup commands called before here
' Set 1st bit of digital output if Upper Limit 1 Reached
' Set 2nd bit of digital output if Lower Limit 1 Reached
' Set 3rd bit of digital output if Upper Limit 2 Reached
' Set 4th bit of digital output if Lower Limit 2 Reached
Set2001Calc3Dig 1, 2, 4, 8
```

9.14 FUNCTION Set2001Calc3DigQ$

Description

Queries the Model 2001 for its CALCulate3 subsystem's Limit Testing HI/LO Digital Output Values.

Returns

"Error!" if a TimeOutError occurred, or
Query 1: digital output Upper Limit 1 value

Query 2: digital output Lower Limit 1 value
Query 3: digital output Upper Limit 2 value
Query 4: digital output Lower Limit 2 value

Use ParseQuery$ to separate return string into components.

Example

```
' BASIC 7.1 or QB45 setup commands called before here
A$ = Set2001Calc3DigQ$              ' Get Model 2001 CALC3 Digital I/O settings
Du1% = VAL(ParseQuery$(A$, 1))      ' Get UpLim1 DigI/O Value
Dl1% = VAL(ParseQuery$(A$, 2))      ' Get LoLim1 DigI/O Value
Du2% = VAL(ParseQuery$(A$, 3))      ' Get UpLim2 DigI/O Value
Dl2% = VAL(ParseQuery$(A$, 4))      ' Get LoLim2 DigI/O Value
```

9.15 FUNCTION Calc3.2001Q$

Description

Queries the Model 2001 for the result of the present CALCulate3 subsystem's calculation.

Returns

"Error!" if a TimeOutError occurred, or
Query 1: Upper/Lower Limit Range 1 result (0=PASS or 1=FAIL)
Query 2: Upper/Lower Limit Range 2 result (0=PASS or 1=FAIL)

Use ParseQuery$ to separate return string into components.

Example

```
' BASIC 7.1 or QB45 setup commands called before here
A$ = Calc3.2001Q$                   ' Get Model 2001 CALC3 Limit Test Results
Test1$ = ParseQuery$(A$, 1)         ' Get Upper/Lower 1 Test Result
Test2$ = ParseQuery$(A$, 2)         ' Get Upper/Lower 2 Test Result
```

Appendix A
Model 2001/7001 Global Variables

The following is a list of all the global variables used by the Model 2001/7001 Support Software for Microsoft QuickBASIC 4.5 and Professional BASIC 7.1:

KI2001% Global variable that contains the Model 2001's IEEE address.

KI7001% Global variable that contains the Model 7001's IEEE address.

TimeOutError% Set to TRUE (-1) when a timeout error occurs.

brd0% The location of board "GPIB0" for National Instruments.

Nat2001Addr% Integer value of the actual Model 2001 IEEE Address for use with National Instruments board commands.

Nat7001Addr% Integer value of the actual Model 7001 IEEE Address for use with National Instruments board commands.

IeeeIn% Integer value of the file handle used for IOTech Driver 488 input operations.

IeeeOut% Integer value of the file handle used for IOTech Driver 488 output operations.

Resp$ 2048 byte fixed-length string used for the input string on all IEEE-488 read functions.

IeeeInterface% Contains a number from 1 to 4 indicating the IEEE interface as defined by the constants below:

IEEECEC (1)	Capital Equipment Corp.
IEEEIOTECH (2)	IOTech Driver 488
IEEENATIONALOLD (3)	National Instruments NI-488 (C.10 or C.11)
IEEENATIONALNEW (4)	NI-488 (C.12 or greater) or NI-488.2

OutOfRange% Set to TRUE (-1) whenever a parameter given to a Set Model 2001/7001 routine is out of range.

NOTE

These variables also support the Model 7002 Switch System.

Appendix B
Model 2001 / 7001 Constants

The following Model 2001/7001 Support Software CONSTants are defined to make using the support software easier. The use of these constants are defined below and in the appropriate group of routines that use them.

NOTE

These constants also support the Model 7002 Switch System.

B.1 Function Constants

DCV (1)	DC Voltage	ACV (2)	AC Voltage
DCA (3)	DC Current	ACA (4)	AC Current
R2 (5)	2-wire Resistance	R4 (6)	4-wire Resistance
FREQ (7)	Frequency	TEMP (8)	Temperature

B.2 Status Model Constants

For use with Stat2001, Stat2001Q$, Stat7001, Stat7001Q$:

OPERATION (1)	Operation Event
TRIGGER1 (2)	Trigger Event
ARM1 (3)	Arm Event
SEQUENCE (4)	Sequence Event
QUESTIONABLE (5)	Questionable Event
MEASUREMENT (6)	Measurement Event

Constants for use with *STB?, *SRE, *SRE?, and serial poll:

MSB (1)	Measurement Summary Bit (Model 2001 Only)
EAV (4)	Error Available
QSB (8)	Questionable Summary Bit (in Model 7001 only for SCPI)
MAV (16)	Message Available
ESB (32)	Event Summary Bit
RQS1 (64)	Request for Service (Serial Poll)
MSS (64)	Master Summary Status (Status Byte)
OSB (128)	Operation Summary Bit

Constants for use with *ESR?, *ESE, and *ESE?:

OPC (1)	Operation Complete
RQC (2)	Request Control (not used in Model 2001/7001)
QYE (4)	Query Error
DDE (8)	Device Specific Error
EXE (16)	Execution Error
CME (32)	Command Error
URQ (64)	User Request
PON (128)	Power On

Constants for use with the Operation Event Condition Register, Positive Transition Filter, Negative Transition Filter, Status Register, and Status Enable Register:

CAL (1)	Calibrating (Model 2001 only)
SET (2)	Settling
TRIG (32)	Waiting for Trigger
ARM (64)	Waiting for Arm
CALC (512)	Calculating (Model 2001 only)
SCAN (1024)	Scanning (Model 7001 only)
IDLE (1024)	in IDLE layer (Model 2001 only)

Constants for use with the Trigger Event Condition Register, Positive Transition Filter, Negative Transition Filter, Status Register, and Status Enable Register:

SEQ1 (2)	Model 2001/7001 is in the Trigger Layer of Sequence 1.

Constants for use with the Arm Event Condition Register, Positive Transition Filter, Negative Transition Filter, Status Register, and Status Enable Register:

SEQ1 (2)	Model 2001/7001 is in the ARM Layer of Sequence 1.

Constants for use with the Sequence Event Condition Register, Positive Transition Filter, Negative Transition Filter, Status Register, and Status Enable Register:

LAY1 (2)	Model 2001 or 7001 is in the Arm Layer 1 of Sequence 1.
LAY2 (4)	Model 2001 or 7001 is in the Arm Layer 2 of Sequence 1.

Constants for use with the Model 2001 Questionable Event Condition Register, Positive Transition Filter, Negative Transition Filter, Status Register, and Status Enable Register:

QTEMP (16)	Questionable Calibration Process
QCAL (128)	Questionable Calibration Process
WARN (16384)	Command Warning

Constants for use with the Model 2001 Measurement Event Condition Register, Positive Transition Filter, Negative Transition Filter, Status Register, and Status Enable Register:

ROF (1)	Reading OverFlow
LL1 (2)	Low Limit 1
HL1 (4)	High Limit 1
LL2 (8)	Low Limit 2
HL2 (16)	High Limit 2
RAV (32)	Reading AVailable
BAV (128)	Buffer AVailable
BHF (256)	Buffer Half Full
BFL (512)	Buffer FuLl
BPT (2048)	Buffer PreTrigger event occurred

B.3 Automatic Constants

Use these constants with the Model 2001 Auto measurement functions, like AutoDCV, or any Model 2001/7001 function that requires an ON or OFF state.

NO (-1)	Don't set automatic parameter.
TOFF (+0)	Turn off automatic parameter.
TON (+1)	Turn on automatic parameter.
ONCE (+2)	Set auto parameter for the next measurement.

B.4 Model 2001 Minimum and Maximum Sense Constants

MINACA = -2.1#	MAXACA = 2.1#
MINDCA = -2.1#	MAXDCA = 2.1#
MINACV = -775#	MAXACV = 775#
MINDCV = -1100#	MAXDCV = 1100#
MINR2 = 0#	MAXR2 = 1050000000#
MINR4 = 0#	MAXR4 = 21000#
MINFREQ = 0#	MAXFREQ = 1500000#
MINTEMPF = -328#	MAXTEMPF = 3310#
MINTEMPC = -200#	MAXTEMPC = 1821#
MINTEMPK = 73#	MAXTEMPK = 2094#
MINSIMF = 32#	MAXSIMF = 122#
MINSIMC = 0#	MAXSIMC = 50#
MINSIMK = 273#	MAXSIMK = 323#
MINTCO = -.09999#	MAXTCO = .09999#
MINALPHA = 0#	MAXAPLHA = .01#
MINBETA = 0#	MAXBETA = 1#
MINDELTA = 0#	MAXDELTA = 5#
MINRZERO = 0#	MAXRZERO = 1000#
MINDIG = 3.5	MAXDIG = 8.4999
MINNPLC = .01#	MAXNPLC = 10#

MINAVGCNT = 1#	MAXAVGCNT = 100#
MINNTOL = 1#	MAXNTOL = 100#
MINPWIN = .1#	MAXPWIN = 9.9#
MAXTEMPDIG = 7.4999	MAXFREQDIG = 5.4999
MAXFREQV = 1000#	MINFREQV = -1000#
MAXFREQI = 1#	MINFREQI = -1#
MAXFREQPERC = .6#	MINRANGE = 0#

B.5 Model 2001 Minimum and Maximum Calculate Constants

MINCALC = -9.999999D+35	MAXCALC = 9.999999D+35
MINPERC = -9.999999D+35	MAXPERC = 9.999999D+35
MINMMF = -9.999999D+20	MAXMMF = 9.999999D+20
MINMBF = -9.999999D+30	MAXMBF = 9.999999D+30
MINDIGITAL = 0	MAXDIGITAL = 15

B.6 Model 2001/7001 Scanning Minimum and Maximum Constants

MINCOUNT = 1	MAXCOUNT = 99999
MINDELAY = 0#	MAXDELAY = 999999.999#
MINLINE = 1	MAXLINE = 6
MINCHANNEL = 1	MAXCHANNEL = 10
MINTIMER = .001#	MAXTIMER = 999999.999#

B.7 Read2001BufferS% and Read2001BufferD% Constants

BUFRDGS = 1
BUFTIMESTAMP = 2
BUFCHANNELS = 3
BUFSTATUS = 4
BUFUNITS = 5

B.8 Examples

To set the Model 2001 to generate a Service Request on EAV or MAV use the following:

SRE2001 EAV OR MAV

To check for MAV being set in the Status Byte:

poll%=val(Q2001$("*STB?")) IF (poll% AND MAV) THEN PRINT "MAV is set."

Appendix C
Model 2001 / 7001 Support Software File Names and Routines

C.1 Microsoft Basic 7.1 File Names

C.1.1 Header Files

20017001.BI — 2001/7001 Header File w/CONST & Routine declares
2001DEMO.BI — 2001 Demo program include file
GENERAL .BI — Revised BASIC 7.1 include file

C.1.2 Model 2001/7001 Demo Program Files

2001BUFF.BAS — 2001 Demo Buffer handling routines:
BufferCalc, DisplayReadings, GraphReadings, ReadBuffer, TakeBufferRdgs

2001CFG .BAS — 2001 Demo configuration routines:
SetupFunction

2001CFG1.BAS — 2001 Demo configuration routines:
AutoButtons, Reset2001, Set2001AutoZero, Set2001LineSynch, TempRef2001, Units2001

2001DEM1.BAS — 2001 Demo Main program module (secondary):
delay1, InputLine, SetupMenu, WindowBox2

2001DEMO.BAS — 2001 Demo Main program module:
CheckMinMaxVal, ColorDisplay, Comline, ConvertMinMaxDef, GetInfoRecord, MonoDisplay

2001FILE.BAS — 2001 Demo File Handling routines:
GetLSAFileName, Load2001Data, RecallStoreSetup, Save2001Data, ViewFile

2001INFO.BAS — 2001 Demo Help routines:
InfoBuffer, InfoCalc, InfoChange2001Function, InfoChange2001Address, InfoConfig, InfoDeltaRatioConfig, InfoDemo, InfoDisplay1, InfoGraphData, InfoOpenCloseRelay, InfoPick2001ScanFunc, InfoSendGetSCPI, InfoServiceRequest, InfoSetup2001IntScanList, InfoSetupLayer, InfoSetViewStatus, InfoStandarEvent, InfoStoreRecall, InfoTempRef, InfoUnits

2001INFO.IDX — 2001 Demo Help File indices

2001INFO.TXT — 2001 Demo Help File Text

2001MEAS.BAS — 2001 Demo measuring routines:
ButtonFlag1, Display2001Rdg, DisplayCurrent3, DisplayVoltage3, Format2001Reading, LimitTesting, Measure, ReadCalculate, Restore2001Format, Set2001Format, SetupCalculate

2001SCAN.BAS — 2001 Demo Scanning routines:
Change2001ScanFunction, DeltaRatioConfig, OpenClose2001Relay, OpenClose7001Relay, Pick2001ScanFunction, Set2001IntExtChannels, Setup2001ExtScanList, Setup2001IntScanList, Setup7001ScanList, SetupLayer

2001STAT.BAS — 2001 Demo Status Model routines:
ChangeAddress, Check, Checked, ClearStatus, Display1, SendGetSCPI, ServiceRequest, SetViewStatus, StandardEvent, ViewStatusModel

MAKEINFO.BAS — Generates 2001INFO.IDX and 2001INFO.TXT (run from QBX environment):
AddIndex, AddText, Str1

C.1.3 Model 2001/7001 IEEE Interface Independent Files

27001 .BAS — 2001/7001 Routines:

2001/7001 Display and General routines:

Disp2001, Disp7001, KIDisp, KINoDisp, NoDisp2001, NoDisp7001, Check2001Val, KIQ, KISend

2001 Function & Buffer routines:

ACA2001, ACV2001, BufferSize2001, DCA2001, DCV2001, FREQ2001, Get2001Func, Get2001FuncHeader, Get2001Rdg, Get2001SaveRecallSize, Get2001Units, Get2RES2001, Get4RES2001, GetACA2001, GetACV2001, GetDCA2001, GetDCV2001, GetFREQ2001, GetTEMP2001, Hit2001Key, Hit2001Key1, RES2.2001, RES4.2001, Set2001Function, Take2001BufferReadings, TEMP2001

2001 Automatic Configuration routines:

Auto2001, Auto2001ACA, Auto2001ACAQ, Auto2001ACV, Auto2001ACVQ, Auto2001DCA, Auto2001DCAQ, Auto2001DCV, Auto2001DCVQ, Auto2001Q, Auto2001R2, Auto2001R2Q, Auto2001R4, Auto2001R4Q, Auto2001T, Auto2001TQ

2001 CALCulate Subsytem routines:

Calc1.2001, Calc1.2001Q, Calc2.2001, Calc2.2001Q, Calc3.2001, Calc3.2001Q, Set2001Calc1MXB, Set2001Calc1PERC, Set2001Calc1Q, Set2001Calc2, Set2001Calc2Q, Set2001Calc3, Set2001Calc3Dig, Set2001Calc3DigQ, Set2001Calc3Q

2001 configuration routines:

Set2001, Set2001ACA, Set2001ACAQ, Set2001ACV, Set2001ACVQ, Set2001DCA, Set2001DCAQ, Set2001DCV, Set2001DCVQ, Set2001F, Set2001FQ, Set2001Q, Set2001R2, Set2001R2Q, Set2001R4, Set2001R4Q, Set2001RTD, Set2001RTDQ, Set2001T, Set2001TC, Set2001TCQ, Set2001TQ

2001/7001 Status Model routines:

Clear2001, Clear7001, ESE2001, ESE7001, KIESE, KISRE, KIStat, KIStatQ, SRE2001, SRE7001, Stat2001, Stat2001Q, Stat7001, Stat7001Q

2001/7001 Trigger Model routines:

Arm2001, Arm2001Q, Arm7001, Arm7001Q, ArmTcon2001, ArmTcon2001Q, ArmTcon7001, ArmTcon7001Q, Close2001, Close2001Q, Close7001, Close7001Q, KIArm, KIArmQ, KIArmTcon, KIArmTconQ, KICloseQ, KITimers, KITimersQ, KITrig, KITrigQ, KITrigTcon, KITrigTconQ, Open2001, Open7001, Scan2001, Scan7001, Timers2001, Timers2001Q, Timers7001, Timers7001Q, Trig2001, Trig2001Q, Trig7001, Trig7001Q, TrigTcon2001, TrigTcon2001Q, TrigTcon7001, TrigTcon7001Q

GENRLQBX.BAS — 2001/7001 General Routines:

AutoGraphicsMode, IOTECH, ParseQuery, Str1, DataViewS, XYGraphS, DataViewD, XYGraphD

GENERAL .OBJ — Object file from BASIC 7.1 GENERAL.BAS

MAKEFILE. — File for BASIC 7.1 NMAKE program to create various the *.LIB, *.QLB, *.EXE files of the BASIC 7.1 routines

MENU .OBJ — Object file from BASIC 7.1 MENU.BAS

MOUSE .OBJ — Object file from BASIC 7.1 MOUSE.BAS

QBX .LIB — Library file from BASIC 7.1

UIASM .OBJ — Object file from BASIC 7.1

WINDOW .OBJ — Object file from BASIC 7.1 WINDOW.BAS

C.1.4 Capital Equipment Corp. (CEC) IEEE-488 Interface Files

CECQBX .BAS — Basic 7.1 CEC 2001/7001 source

DataInvalid, Gpiberror, OPC2001, OPC7001, Poll2001, Poll7001, Q2001, Q7001, Send2001, Send7001, SendGET2001, SendGET2001, SendSDC2001, SendSDC7001, Wait2001RQS, Wait7001RQS, SetupIEEE, Read2001BufferS, Read2001BufferD

CECQBX .LIB — BASIC 7.1 CEC 2001/7001 driver library
CECQBX .QLB — BASIC 7.1 CEC 2001/7001 quick library
CECQBX .BAT — BASIC 7.1 CEC 2001/7001 QBX environment loader
ERRORDIS.CEC — CEC interface detection disabled if this file exists.
CECB7 .BI — Modified CEC BASIC 7.1 include file
CECB7 .OBJ — CEC driver library
MAKECEC .BAT — Makes all *.OBJ, *.LIB, *.QLB, and *.EXE files for Capital Equipment Corp.
MAKECEC .LNK — Make CEC Demo Program
2001CEC .EXE — 2001 CEC Demo Program

C.1.5 IOTech Driver 488 Interface Files

IOTQBX .BAS — BASIC 7.1 IOTech 2001/7001 driver library source
DataInvalid, Gpiberror, OPC2001, OPC7001, Poll2001, Poll7001, Q2001, Q7001, Send2001, Send7001, SendGET2001, SendGET2001, SendSDC2001, SendSDC7001, Wait2001RQS, Wait7001RQS, SetupIEEE, Read2001BufferS, Read2001BufferD
IOTQBX .LIB — BASIC 7.1 IOTech 2001/7001 driver library
IOTQBX .QLB — BASIC 7.1 IOTech 2001/7001 driver quick library
IOTQBX .BAT — BASIC 7.1 IOTEch 2001/7001 QBX environment loader
MAKEIOT .BAT — Makes all *.OBJ, *.LIB, *.QLB, and *.EXE files for IOTech Driver 488.
MAKEIOT .LNK — Make IOTech Demo Program
2001IOT .EXE — 2001 IOTech Demo Program

C.1.6 National Instruments NI-488 rev C.11 Files

NATQBX1 .BAS — BASIC 7.1 old NI 2001/7001 driver library source
DataInvalid, Gpiberror, OPC2001, OPC7001, Poll2001, Poll7001, Q2001, Q7001, Send2001, Send7001, SendGET2001, SendGET2001, SendSDC2001, SendSDC7001, Wait2001RQS, Wait7001RQS, SetupIEEE, Read2001BufferS, Read2001BufferD
NATQBX1 .LIB — BASIC 7.1 old NI 2001/7001 driver library
NATQBX1 .QLB — BASIC 7.1 old NI 2001/7001 driver quick library
NATQBX1 .BAT — BASIC 7.1 old NI 2001/7001 QBX environment loader
NI4881B7.BI — Old NI header file
NI4881B7.OBJ — Old NI object file
NI4881B7.C — Old NI-488 BASIC 7.1 to Quick C link C file
NI4881B7.H — Old NI-488 BASIC 7.1 to Quick C link header file
NI4881 .LIB — Old NI-488 BASIC 7.1 to Quick C link library
NI4881 .OBJ — Old NI-488 BASIC 7.1 to Quick C link object file
MAKENAT1.BAT — Makes all *.OBJ, *.LIB, *.QLB, and *.EXE files for National Instruments NI-488 (rev C.11 and older)
MAKENAT1.LNK — Make old NI Demo Program
2001NAT1.EXE — 2001 National Instruments (NI-488 rev <= C.11) Demo Program

C.1.7 National Instruments NI-488 rev C.12 (and newer) and NI-488.2 Files

NATQBX2 .BAS — BASIC 7.1 new NI 2001/7001 driver library source

DataInvalid, Gpiberror, OPC2001, OPC7001, Poll2001, Poll7001, Q2001, Q7001, Send2001, Send7001, SendGET2001, SendGET2001, SendSDC2001, SendSDC7001, Wait2001RQS, Wait7001RQS, SetupIEEE, Read2001BufferS, Read2001BufferD

NATQBX2 .LIB — BASIC 7.1 new NI 2001/7001 driver library

NATQBX2 .QLB — BASIC 7.1 new NI 2001/7001 driver quick library

NATQBX2 .BAT — BASIC 7.1 new NI 2001/7001 QBX environment loader

NI4882B7.BI — New NI header file

NI4882B7.OBJ — New NI object file.

MAKENAT2.BAT — Makes all *.OBJ, *.LIB, *.QLB, and *.EXE files for NI-488.2 and NI-488 (rev C.12 and newer)

MAKENAT2.LNK — Make new NI Demo Program

2001NAT2.EXE — 2001 NI-488 rev >= C.12 & NI-488.2 Demo Program

C.2 Microsoft QuickBasic 4.5 File Names

C.2.1 Model 2001/7001 IEEE Interface Independent Files

20017001.BI — 2001/7001 Header File w/CONST & Routine declares

2001 .BAS — 2001 Function & Buffer routines:

ACA2001, ACV2001, BufferSize2001, DCA2001, DCV2001, FREQ2001, Get2001Func, Get2001FuncHeader, Get2001Rdg, Get2001SaveRecallSize, Get2001Units, Get2RES2001, Get4RES2001, GetACA2001, GetACV2001, GetDCA2001, GetDCV2001, GetFREQ2001, GetTEMP2001, Hit2001Key, Hit2001Key1, RES2.2001, RES4.2001, Set2001Function, Take2001BufferReadings, TEMP2001

AUTO2001.BAS — 2001 Automatic Configuration routines:

Auto2001, Auto2001ACA, Auto2001ACAQ, Auto2001ACV, Auto2001ACVQ, Auto2001DCA, Auto2001DCAQ, Auto2001DCV, Auto2001DCVQ, Auto2001Q, Auto2001R2, Auto2001R2Q, Auto2001R4, Auto2001R4Q, Auto2001T, Auto2001TQ

CALC2001.BAS — 2001 CALCulate Subsytem routines:

Calc1.2001, Calc1.2001Q, Calc2.2001, Calc2.2001Q, Calc3.2001, Calc3.2001Q, Set2001Calc1MXB, Set2001Calc1PERC, Set2001Calc1Q, Set2001Calc2, Set2001Calc2Q, Set2001Calc3, Set2001Calc3Dig, Set2001Calc3DigQ, Set2001Calc3Q

SETS2001.BAS — 2001 configuration routines:

Set2001, Set2001ACA, Set2001ACAQ, Set2001ACV, Set2001ACVQ, Set2001DCA, Set2001DCAQ, Set2001DCV, Set2001DCVQ, Set2001F, Set2001FQ, Set2001Q, Set2001R2, Set2001R2Q, Set2001R4, Set2001R4Q, Set2001RTD, Set2001RTDQ, Set2001T, Set2001TC, Set2001TCQ, Set2001TQ

SCPIDISP.BAS — 2001/7001 Display routines:

Disp2001, Disp7001, KIDisp, KINoDisp, NoDisp2001, NoDisp7001

SCPISTAT.BAS — 2001/7001 Status Model routines:

Clear2001, Clear7001, ESE2001, ESE7001, KIESE, KISRE, KIStat, KIStatQ, SRE2001, SRE7001, Stat2001, Stat2001Q, Stat7001, Stat7001Q

SCPITRIG.BAS — 2001/7001 Trigger Model routines:

Arm2001, Arm2001Q, Arm7001, Arm7001Q, ArmTcon2001, ArmTcon2001Q, ArmTcon7001, ArmTcon7001Q, Close2001, Close2001Q, Close7001, Close7001Q, KIArm, KIArmQ, KIArmTcon, KIArmTconQ, KICloseQ, KITimers, KITimersQ, KITrig, KITrigQ, KITrigTcon, KITrigTconQ, Open2001, Open7001, Scan2001, Scan7001, Timers2001, Timers2001Q, Timers7001, Timers7001Q, Trig2001, Trig2001Q, Trig7001, Trig7001Q, TrigTcon2001, TrigTcon2001Q, TrigTcon7001, TrigTcon7001Q

GENERAL1.BAS — 2001/7001 General Routines:

AutoGraphicsMode, Check2001Val, IOTECH, OPC2001, OPC7001, ParseQuery, Poll2001, Poll7001, Q2001, Q7001, Send2001, Send7001, SendGET2001, SendGET7001, SendSDC2001, SendSDC7001, Str1, Wait2001RQS, Wait7001RQS

GENERALS.BAS — 2001/7001 Single Precision General Routines:

DataViewS, XYGraphS

GENERALD.BAS — 2001/7001 Double Precision General Routines:

DataViewD, XYGraphD

NEWER .EXE — Program used to determine if a .BAS file is older than its.OBJ file, returns 2 if an error occurred, 1 if the .BAS is newer than its .OBJ file, 0 if the .BAS file is older

C.2.2 Capital Equipment Corp. (CEC) IEEE-488 Interface Files

CEC .LIB — QuickBASIC 4.5 CEC 2001/7001 library

CEC .QLB — QuickBASIC 4.5 CEC quick library

CEC .BAS — QuickBASIC 4.5 CEC 2001/7001 source DataInvalid, Gpiberror, KIOPC, KIPoll, KIQ, KISend, KISendGET, KISendSDC, KIWaitRQS, ModelNumber, SetupIEEE

CECS .BAS — QuickBASIC 4.5 CEC 2001/7001 source Read2001BufferS

CECD .BAS — QuickBASIC 4.5 CEC 2001/7001 source Read2001BufferD

CECQB .BI — Modified CEC QuickBASIC 4.5 include file

ERRORDIS.CEC — CEC interface detection is disabled if this file exists

MAKECEC .BAT — Makes all *.OBJ, *.LIB, *.QLB, and *.EXE files for Capital Equipment Corp.

C.2.3 IOTech Driver 488 Interface Files

IOTECH .LIB — QuickBASIC 4.5 IOTech 2001/7001 driver library

IOTECH .QLB — QuickBASIC 4.5 IOTech 2001/7001 quick library

IOTECH .BAS — QuickBASIC 4.5 IOTech 2001/7001 source DataInvalid, Gpiberror, KIOPC, KIPoll, KIQ, KISend, KISendGET, KISendSDC, KIWaitRQS, ModelNumber, SetupIEEE

IOTECHS .BAS — QuickBASIC 4.5 IOTech 2001/7001 source Read2001BufferS

IOTECHD .BAS — QuickBASIC 4.5 IOTech 2001/7001 source Read2001BufferD

MAKEIOT .BAT — Makes all *.OBJ, *.LIB, *.QLB, and *.EXE files for IOTech Driver 488.

C.2.4 National Instruments NI-488 rev C.11 Files

NAT488_1.BAS — QuickBASIC 4.5 old NI 2001/7001 source DataInvalid, Gpiberror, KIOPC, KIPoll, KIQ, KISend, KISendGET, KISendSDC, KIWaitRQS, ModelNumber, SetupIEEE

NAT4881S.BAS — QuickBASIC 4.5 old NI 2001/7001 source Read2001BufferS

NAT4881D.BAS — QuickBASIC 4.5 old NI 2001/7001 source Read2001BufferD

NAT488_1.LIB — QuickBASIC 4.5 old NI 2001/7001 driver library

NAT488_1.QLB — QuickBASIC 4.5 old NI 2001/7001 quick library

NI4881QB.BI — Old NI header file

NI4881QB.OBJ — Old NI object file

MAKENAT1.BAT — Makes all *.OBJ, *.LIB, *.QLB, and *.EXE files for National Instruments NI-488 (rev C.11 and older)

C.2.5 National Instruments NI-488 rev C.12 (and newer) and NI-488.2 Files

NAT488_2.BAS — QuickBASIC 4.5 new NI 2001/7001 source DataInvalid, Gpiberror, KIOPC, KIPoll, KIQ, KISend, KISendGET, KISendSDC, KIWaitRQS, ModelNumber, SetupIEEE

NAT4882S.BAS — QuickBASIC 4.5 new NI 2001/7001 source Read2001BufferS

NAT4882D.BAS — QuickBASIC 4.5 new NI 2001/7001 source Read2001BufferD

NAT488_2.LIB — QuickBASIC 4.5 new NI 2001/7001 driver library

NAT488_2.QLB — QuickBASIC 4.5 new NI 2001/7001 quick library

NI4882QB.BI — New NI header file

NI4882QB.OBJ — New NI object file.

MAKENAT2.BAT — Makes all *.OBJ, *.LIB, *.QLB, and *.EXE files for NI-488.2 and NI-488 (rev C.12 and newer)

Quick C2.5

Table of Contents

Section 1
Using the Library Routines

1.1 Installation

1.1.1 National Instruments GPIB.COM

You must have at least Rev C.10 of the National Instruments NI-488 Software or at least Rev 1.0 of the National Instruments NI-488.2 Software to use the 2001/7001 Support Software. Some older revisions may work, but they are not guaranteed by Keithley to do so. Use the following settings to set up your National Instruments Card (AT-GPIB card listed here, use all that are applicable) when you are configuring GPIB.COM with IBCONF.EXE.

Primary GPIB Address..... 0
Secondary GPIB Address ..... NONE
Timeout setting T10s
EOS byte..... 0AH
Terminate Read on EOS ..... no
Set EOI with EOS on Write..... yes
Type of compare on EOS 7-bit
Set EOI w/last byte of Write ..... yes
System Controller ..... yes
Assert REN when SC ..... yes
Enable Auto Serial Polling ..... no
Timing 350nsec
Enable 488.2 Protocols ..... yes
CIC Protocol ..... no
Disable Device Unaddressing..... no

1.1.2 Capital Equipment Corp.

You must have at least Rev 2.14 of the Capital Equipment Corp. Software to use the 2001/7001 Support Software. All older revisions will not work without the removal of setatnmode() from SetupIEEE() in CEC.C.

1.1.3 IOTech Driver 488

You must have at least Rev 2.6 of the IOTech Driver 488 Software to use the 2001/7001 Support Software. Some older revisions may work, but they are not guaranteed by Keithley to do so.

1.1.4 Quick C 2.5

Type A:INSTALL or B:INSTALL to install the Quick C 2.5 2001/7001 support software. The installation program will prompt you for certain information that it needs to copy files and build the make files

MAKECEC.BAT, MAKEIOT.BAT, MAKENAT1.BAT, and MAKENAT2.BAT. These files will build the *.OBJ, *.LIB, and *.QLB needed to use the routines.

Installation Notes:

1. QCL.EXE and LIB.EXE must be installed in the Quick C 2.5 Compiler Directory.
2. NMAKE.EXE must be installed in the NMAKE directory.
3. IEEE488.LIB (if installing CEC) and LLIBC7.LIB and/or LLIBCE.LIB must reside in the Library Files Directory to make programs.
4. STRING.H, STDLIB.H, STDIO.H, CONIO.H, DOS.H, TIMR.H, GRAPH.H and FCNTL.H must reside in the Include Files Directory.
5. The 2001/7001 Support Software directory must exist and you will be prompted to build it if not.
6. The Install Source Code Only option will install only the *.C and *.H files and not check for any of the Quick C 2.5 files listed above. MAKECEC.BAT, MAKEIOT.BAT, MAKENAT1.BAT, and MAKENAT2.BAT will still be created and can be edited later.
7. The file GENERAL1.C is the only module that uses Quick C 2.5 specific routines for data and graphical displays. The other modules were made as ANSI-C as possible, which allows for their use on any platform with the use of the proper CEC, IOTech, and National Instruments include files, object modules, and libraries as supplied by the IEEE-488 interface manufacturers.

1.2 General Instructions

1.2.1 Quick C 2.5

In your program, use #include “20017001.H” which contains the 2001/7001 library function definitions and CONSTants. Also, make sure that you are compiling in the large or huge data model since the libraries use the huge model. Furthermore, at least 4096 bytes of stack space should be used when creating an executable with ILINK. When creating a *.MAK file include your program name and:

- CEC.LIB and IEEE488.LIB for CEC,
- IOTECH.LIB for IOTech Driver 488,
- NAT488_1.LIB for old NI-488, or
- NAT488_2.LIB for new NI-488 and NI-488.2.

If using DataViewS, DataViewD, XYGraphS, or XYGraphD make sure to specify GRAPHICS.LIB if it is not included in LLIBC7.LIB or LLIBCE.LIB. A sample setup follows:

```
#include “20017001.h”
int SetupErr;
char response[2048];
/*
start of user program here
*/
SetupErr = SetupIEEE(0,0);              /* Setup IEEE Interface */
SetupErr = SetupIEEE(2001,16);          /* Setup 2001 at IEEE Address 16 */
SetupErr = SetupIEEE(7001,7);           /* Setup 7001 at IEEE Address 7 */
/*
User program follows
*/
```

1.2.2 Documentation Notes

1. Query forms of a function have the format of **function-nameQ(char * response, ...)** which returns a pointer to response containing a query string of the specified Model 2001/7001 parameters. Automatic values are returned as 0 for OFF and 1 for ON.
2. If a function query returns more than one parameter, the responses will be separated by a semicolon (;) within the return string. Use the ParseQuery routine to return the specified response string (parameter number 1, 2, 3, etc.) from the query return string.
3. Any Model 2001/7001 command parameters out of range will not be sent to the instrument leaving the previously set or default values intact. Most routines that have a return type of int will return a -1 for a parameter out of range.
4. Most Model 2001/7001 parameters that are double precision values will accept the constants MAXIMUM, MINIMUM, and DEFAULT. INF will be accepted for Trigger Model Count1 and Count2 parameters.
5. All example code fragments assume that Quick C 2.5 setup commands listed above were used before issuing any commands.
6. All string parameters that have their valid parameters listed with mixed case, like IMMediate, accept either the short form (IMM) or the long form (IMMEDIATE) in any combination of case. This is comparable to the short and long form notation used for SCPI commands.
7. See Appendix A for a description of the Global Variables used in the Model 2001/7001 support software.
8. See Appendix B for a description of the defined CONSTants used in the Model 2001/7001 support software.
9. See Appendix C for a list of all Quick C 2.5 file names used by the Model 2001/7001 support software.

Section 2
Model 2001 / 7001 IEEE-488 Interface Routines

These functions and subroutines control the Models 2001/7001 with low-level IEEE-488 bus commands specific to each IEEE-488 interface manufacturer.

NOTE

These functions and subroutines also support the Model 7002 Switch System. Just send Model 7001 commands to a Model 7002.

2.1 int SetupIEEE(int Device, int Address);

Description

Sets up the specific IEEE-488 interface to correctly handle data transfers between the Models 2001/7001 and the host PC computer. Also, initializes either the 2001 and 7001 at the specified addresses. CEC controllers will be at IEEE address 21. IOTech controllers are usually at IEEE address 21. National Instruments controllers are usually at IEEE address 0.

Parameters

Device

0	Initialize interface only
2001	Change the IEEE address of the 2001 only
7001	Change the IEEE address of the 7001 only

Address

0-30	if Device is 2001 or 7001
ignored	if Device is 0

Returns

0	if no errors.
-1	if a parameter is out of range.
-2	if an IEEE-488 timeout error occurred.

Global Variables Affected (see Global Variables Appendix A)

KI2001, KI7001, IeeeInterface,
brd0, Nat2001Addr, Nat7001Addr (Nat'l. Instruments only),
ieee (IOTech Driver 488 only)

Example

Before using any of the Model 2001 routines, the following commands must be issued:

```
#include "20017001.H"
char response[2048];
 int SetupErr;
/*
start of user program here
*/
SetupErr = SetupIEEE(0,0);              /* Setup IEEE Interface */
SetupErr = SetupIEEE(2001,16);          /* Setup 2001 at IEEE Address 16 */
SetupErr = SetupIEEE(7001,7);           /* Setup 7001 at IEEE Address 7 */
```

2.2 int SendSDC2001(void);
int SendSDC7001(void);

Description

Sends the IEEE bus command SDC (Selected Device Clear) to the 2001/7001.

Returns

0	if no errors.
-2	if an IEEE-488 timeout error occurred.

2.3 char *Q2001(char *response, char *Cmd);
char *Q7001(char *response, char *Cmd);

Description

Gets a query response from the 2001/7001. Cmd must be a valid 2001/7001 query or else the instrument will TimeOut. Multiple queries are allowed and responses can be separated using the ParseQuery function. If Cmd="" then, the function will still try to read data from the instrument. This is good for reading large amounts of data from the 2001/7001 since the most that can be read at one time is 2048 bytes.

Parameters

Cmd

""	try to read the 2001/7001
<> ""	send query and try to read 2001/7001

Returns

Pointer to response with response = Query response from the 2001/7001 if Cmd was valid.
"Error!" if Cmd not valid (TimeOutError occurred)

Example

```
/* Quick C setup commands called before here */
Q2001( response, "VOLT:DC:RANGE?");          /* return DC Voltage Range */
```

2.4 int Send2001(char *Cmd); int Send7001(char *Cmd);

Description

Sends IEEE-488.2 and SCPI command strings to the 2001/7001.

Parameters

Cmd

Valid 2001/7001 488.2 or SCPI command or query.

Returns

0 if no errors
Must check the 2001/7001 EAV bit in the serial poll register to see if a command was accepted or look at the front panel of the instrument for an error message.
-1 if a parameter is out of range.
-2 if an IEEE-488 timeout error occurred.

Example

```
/* Quick C setup commands called before here */
Send7001( "ROUTE:CLOSE (@1!1:1!40)");          /* Close channels 1-40 on card 1 of 7001 */
```

2.5 int SendGET2001(void); int SendGET7001(void);

Description

Sends the IEEE bus command GET to the 2001/7001.

Returns

0 if no errors
-2 if an IEEE-488 timeout error occurred.

Example

```
/* Quick C setup commands called before here */
SendGET2001();                          /* Trigger the 2001 to take reading */
Q2001( response, "DATA?");              /* Get 2001 reading */
```

2.6 int Wait2001RQS(void);
int Wait7001RQS(void);

Description

Waits for the 2001/7001 to generate a Request for Service. The routine serial polls the 2001/7001 to verify that the instrument is indeed generating a Request for Service. The wait can be aborted by pressing the Esc key.

Returns

FALSE (0)	if aborted.
TRUE (-1)	if a 2001/7001 Request for Service.

Example

```
int Poll;
/* Quick C setup commands called before here */
Poll = Poll2001();                    /* Clear any pending SRQ's */
SRE2001( MAV);                        /* Set up to SRQ on MAV */
Send2001( "FETCH?");                  /* Fetch a 2001 reading */
/* Set up an uninterruptable wait */
while( !Wait2001RQS());
Q2001( response, "");                 /* Get reading */
```

Section 3
Model 2001 Buffer Routines

These routines are used to acquire readings in the Model 2001's data buffer. Up to 30,092 readings can be stored in the Model 2001 with the MEM2 option and the compact format.

3.1 int BufferSize2001(void);

Description

Finds the actual number of data points in the 2001 buffer. This function should be used since TRACe:POINts? may not return the correct number of data points if the data buffer acquisition was aborted.

Returns

The actual number of data points in the 2001 buffer (anywhere from 2 to 30,092) depending on the memory configuration of the Model 2001 being used.

-1 if a parameter is out of range.
-2 if an IEEE-488 timeout error occurred.

Example

```
int NumPoints;
double BufferData[10001];
/* Quick C setup commands called before here */
NumPoints = BufferSize2001();
/* Read Back Readings in double precision */
if( NumPoints <= 10000)
   NumPoints = Read2001Buffer(BufferData, 0, BUFRDGS);
```

3.2 int Take2001BufferReadings(int Func, int BurstMode, int NumData Points1, int Compact);

Description

Sets up and acquires up to 30,092 readings in the 2001's data buffer. (Use with Read2001Buffer to bring read data.) The data acquisition may be aborted by pressing Esc.

Parameters

Func, (also, see Function Constants, Appendix B):

0 — Current Function
1 — DCV

2 — ACV
3 — DCA
4 — ACA
5 — 2-wire Resistance
6 — 4-wire Resistance
7 — Frequency
8 — Temperature

BurstMode

If TRUE (non-zero), uses the fast Burst Mode of Reading acquisition available from the 2001. BurstMode is only applicable if Func is 1 to 5.

NumDataPoints1

2 to TRACE:POINTS? MAX (depends on the setting of Compact and the memory option installed in the 2001).

Compact

TRUE (non-zero) — use the COMPACT buffer format (readings only). Must specify w/Burst Mode = TRUE.
FALSE (zero) — use the FULL buffer format (reading, time stamp, channel number, status, and units).

NOTE

The COMPACT format allows 5 times as many readings as does FULL. If BurstMode is set, only COMPACT format is valid.

Returns

The actual number of data points read from the 2001. If zero, then either an IEEE-488 timeout error occurred or an out of range error occurred for Func or NumDataPoints < 2.

-1 if a parameter is out of range.
-2 if an IEEE-488 timeout error occurred.

Example

```
int NumPoints;
float BufferData[2000];
/* Quick C setup commands called before here */
/* Take 2000 DC Volt Burst Mode Readings: */
NumPoints = Take2001BufferReadings(DCV, TRUE, 2000, TRUE);
/* Read Back data in Single Precision Format: */
NumPoints = Read2001Buffer(BufferData, 0, BUFRDGS);
```

3.3 int Read2001Buffer(float or double *ArrayName, int DMA, int Fmt); int _Read2001Buffer(void *ArrayName, unsigned int ArraySize, unsigned int ElementSize, int DMA, int Fmt);

Description

Retrieves all of the readings (up to 30,092 IEEE754 single or double precision readings) from the 2001's data buffer and stores them into a single or double precision array.

Read2001Buffer actually is #defined as:

_Read2001Buffer(ArrayName, sizeof(ArrayName)/sizeof(ArrayName[0]), sizeof(ArrayName[0]), DMA, Fmt);

Parameters

ArrayName

Single or double precision array dimensioned large enough to handle the number of data points returned by BufferSize2001 or Take2001BufferReadings.

ArraySize

Number of data points ArrayName can hold.

ElementSize

The size of one cell in ArrayName, either sizeof(float) or sizeof(double).

DMA

0 don't use DMA
1 use DMA (IOTech or National Instruments)
1-7 use DMA Channel configured on CEC IEEE-488 Interface Card

Fmt

1 — Return Readings (FULL or COMPact format)
2 — Return TimeStamp (FULL format only)
3 — Return Channel (FULL format only)
4 — Return Status (FULL format only)
5 — Return Units (FULL format only)
Also, see Read2001Buffer Constants in Appendix B.

Returns

The actual number of data points transferred to the array.

-1 if a parameter is out of range or sizeof(ArrayName) is not sizeof(float) or sizeof(double).
-2 if an IEEE-488 timeout error occurred.

Example

int NumPoints;

```
double BufferData[5000], TimeStamp[5000];
/* Quick C setup commands called before here */
/* Take 5000 AC Volt Full Format Mode Readings: */
NumPoints = Take2001BufferReadings(ACV,FALSE,5000,FALSE);
/* Read Back Readings and TimeStamp in double precision */
NumPoints = Read2001Buffer( BufferData, 0, BUFRDGS);
NumPoints = Read2001Buffer( Times, 0, BUFTIMESTAMP);
DataViewD( BufferData, NumPoints);
```

Section 4
General Routines

These routines provide data display, graphing, and data manipulating functions that make the handling of the returned data from the Models 2001 and 7001 easier to handle.

NOTE

These functions and subroutines also support the Model 7002 Switch System. Just send Model 7001 commands to a Model 7002.

4.1 char *ParseQuery(char *response, char *Quer, int QuerNum);

Description

Returns the query specified by QuerNum inside Quer. Queries are separated by either commas or semicolons. Also, leading and trailing spaces are removed.

Parameters

Quer a string returned by any of the 2001/7001 query functions.
QuerNum the number of the query to retrieve

Returns

"Error!" if Quer = "",
"Invalid Query Number." if QuerNum <= 0,
QuerNum query in Quer, or last query if QuerNum too large

Example

```
char A[] = "100; FFF; oii, 4423.223; 4,000";
/* Quick C setup commands called before here */
ParseQuery( response, A, 1); /* response = "100" */
ParseQuery( response, A, 2); /* response = "FFF" */
ParseQuery( response, A, 3); /* response = "oii" */
ParseQuery( response, A, 4); /* response = "4423.223" */
ParseQuery( response, A, 5); /* response = "4" */
ParseQuery( response, A, 6); /* response = "000" */
ParseQuery( response, A, 7); /* response = "000" */
ParseQuery( response, A, 0); /* response = "Invalid Query Number." */
```

4.2 char *IOTECH(char *response, int Address);

Description

Returns a two-digit number string with a leading zero for use with IOTech's IEEE Addressable commands.

Parameters

Address 0 to 30

Returns

Pointer to response with response = "00","01",..,"09","10","11",..,"30" or NULL if Address is out of range.

Example

```
/* Quick C setup commands called before here */
/* send OUTPUT 09 to Driver 488 */
ieeewt( strcat( strins( IOTECH( response, 9), "OUTPUT "), "\n"));
```

4.3 void XYGraphS(float *XArray, float *YArray, unsigned int YStart, unsigned int YStop, char *XTitle, char *YTitle, char *Title, char UseCGA2, char MaxMinScale);

void _XYGraphS(float *XArray, unsigned int NumX, float *YArray, unsigned int NumY, unsigned int StartNum, unsigned int StopNum, char *XTitle, char *YTitle, char *Title, char UseCGA2, char MinMaxScale);

void XYGraphD(double *XArray, double *YArray, unsigned int YStart, unsigned int YStop, char *XTitle, char *YTitle, char *Title, char UseCGA2, char MaxMinScale);

void _XYGraphD(double *XArray, unsigned int NumX, double *YArray, unsigned int NumY, unsigned int StartNum, unsigned int StopNum, char *XTitle, char *YTitle, char *Title, char UseCGA2, char MinMaxScale);

Description

Produces a simple X and Y auto-scaled graph using single (XYGraphS) or double (XYGraphD) precision data. These routines will plot YArray versus XArray if the sizesof the two arrays are equal. If they are not the same, only the YArray is plotted versus its corresponding data point number. Use YStart and YStop to zoom in on a particular area of the graph. Both the X and Y axes are scalable to the maximum and minimum of the arrays within the specified YStart and YStop interval.

XYGraphS is actually #defined as:

_XYGraphS(XArray, sizeof(XArray)/sizeof(float), YArray, sizeof(YArray)/sizeof(float), StartNum, StopNum, XTitle, YTitle, Title, UseCGA2, MinMaxScale)

XYGraphD is actually #defined as:

_XYGraphD(XArray, sizeof(XArray)/sizeof(double), YArray, sizeof(YArray)/sizeof(double), StartNum, StopNum, XTitle, YTitle, Title, UseCGA2, MinMaxScale)

Parameters

XArray	Single or double-precision X-axis data array
NumX	Number of X data points
YArray	Single or double-precision Y-axis data array
NumY	Number of Y data points
YStart	First Y data point to plot
YStop	Last Y data point to plot
XTitle	X-axis Title
YTitle	Y-axis Title
Title	Graph Title
UseCGA2	If non-zero, use CGA 640x200 mode so that GRAPHICS.COM can do a screen dump.
MaxMinScale	If non-zero, scale to the minimum and maximum values of the Y-axis data. This has an effect only if all of the data points are the same sign.

Example

```
double BufferData[5000];
double Times[5000];
/* Quick C setup commands called before here */
NumPoints = Take2001BufferReadings( ACV, FALSE, 5000, FALSE);
NumPoints = Read2001Buffer(BufferData, 0, BUFRDGS);
NumPoints = Read2001Buffer(Times, 0, BUFTIMESTAMP);
/* Plot AC Voltage vs. Time scaled to AC Data */
XYGraphD( BufferData, Times, 1, NumPoints, "Time (sec)", "AC Voltage (Vrms)", "AC Voltage vs. Time",
   FALSE, TRUE);
```

4.4 void DataViewS(float *DataArray, unsigned int NumDataPoints);

void _DataViewS(float *DataArray, unsigned int ArraySize, unsigned int NumDataPoints);

void DataViewD(double *DataArray, unsigned int NumDataPoints);

void _DataViewD(double *DataArray, unsigned int ArraySize, unsigned int NumDataPoints);

Description

Views a single or double precision array, DataArray, NumDataPoints long using PageUp, PageDn, Home, End, and the arrow keys. Pressing Esc aborts the data display.

DataViewS is actually #defined as:

_DataViewS(DataArray, sizeof(DataArray)/sizeof(float), NumDataPoints)

DataViewD is actually #defined as:

_DataViewD(DataArray, sizeof(DataArray)/sizeof(double), NumDataPoints)

Parameters

DataArray	Single precision data array to display
DataArray	Double precision data array to display
NumDataPoints	Number of data points to display. If zero or more than the number of points actually in the array, display whole array.

Returns

NumDataPoints with the maximum number of data points in the array if NumDataPoints was originally a variable and was zero or larger than the number of points in the array.

Example

```
float BufferData[5000];
/* Quick C setup commands called before here */
NumPoints = BufferSize2001();
/* Read Back Readings in double precision */
NumPoints = Read2001Buffer(BufferData, 0, BUFRDGS);
/* View all Data Buffer Readings */
if( NumPoints <= 5000)
   DataViewS( BufferData, NumPoints);
```

Section 5
General Model 2001 / 7001 Routines

The following routines perform some extra functions that are not in the 2001/7001 and manipulate the 2001/7001's front panel.

NOTE

These functions and subroutines also support the Model 7002 Switch System. Just send Model 7001 commands to a Model 7002.

5.1 int Set2001Function(int Func);

Description

Puts the Model 2001 into the specified measurement function.

Parameters

Func (1-8) See Function Constants, Appendix B.

Returns

0 if no errors
-1 if a parameter is out of range.
-2 if an IEEE-488 timeout error occurred.

Example

```
/* Quick C setup commands called before here */
/* All three of the following put the 2001 in AC Volts: */
Set2001Function(ACV);
ACV2001();
Send2001( "FUNC 'VOLT:AC'");
```

5.2 char *Get2001Units(char *response, int Func);

Description

Gets the proper Model 2001 units for the function specified.

Parameters

Func 1-8, See Function Constants, Appendix B.

Returns

Pointer to response with response =
“Error!” if an IEEE-488 timeout error occurred or Func is out of range

or:

Func = 1 (DC Volts): “VDC “
Func = 2 (AC Volts):
<5-character string> + <0-4 character string>, where:
<5-character string> = “ dB “, “ dBm “, or “VAC “
<0-4 character string> = “”, “Avg”, “Peak”, “+Pk”, “-Pk”, or “RMS”
Func = 3 (DC Current): “ADC” or “ADC ICkt”
Func = 4 (AC Current): “AAC Avg” or “AAC RMS”
Func = 5 (2-wire Resistance): “Ω2W” or “Ω2W Ocmp”
Func = 6 (4-wire Resistance): “Ω4W” or “Ω4W Ocmp”
Func = 7 (Frequency): “Hz”
Func = 8 (Temperature): “°F”, “°C”, or “ K”

Example

```
/* Quick C setup commands called before here */
Send2001( “*RST”);
/* AC Voltage with the detector set to RMS will */
/* set response to “VAC RMS” */
Get2001Units( response, ACV);
```

5.3 int Get2001SaveRecallSize(void);

Description

Finds the number of 2001 Save/Recall (*SAV, *RCL) locations available for storing 2001 configurations.

Returns

Model 2001 Option	SaveRecallSize
2001	1
2001/MEM1	5
2001/MEM2	10

-1 if a parameter is out of range.
-2 if an IEEE-488 timeout error occurred.

Example

```
#include <stdio.h>
int recall = 5;
/* Quick C setup commands called before here */
/* Model 2001 with MEM1 option */
```

```
/* recall = number of setup location to recall */
if( recall < Get2001SaveRecallSize() && recall >= 0)
{
   sprintf( response, "*RCL %d", recall);
   Send2001( response);
}
else
   fprintf( stderr, "Recall Number is too large!\n");
```

5.4 char *Get2001FuncHeader(char *response, int Func);

Description

Returns the proper SENSe subsystem SCPI Header string for the function specified.

Parameters

Func (1-8) See Function Constants, Appendix B.

Returns

Pointer to response with response = NULL if an IEEE-488 timeout error or Func is out of range or:

Func	Returns
1	"VOLT:DC:"
2	"VOLT:AC:"
3	"CURR:DC:"
4	"CURR:AC:"
5	"RES:"
6	"FRES:"
7	"FREQ:"
8	"TEMP:"

Example

```
/* Quick C setup commands called before here */
Get2001FuncHeader( response, DCV); /* response = "VOLT:DC:" */
```

5.5 int Get2001Func(void);

Description

Finds the present 2001 Function.

Returns

The 2001 function number from 1 to 8 as specified by the Function Constants, Appendix B.

-1 if a parameter is out of range.
-2 if an IEEE-488 timeout error occurred.

Example

```
int func;
/* Quick C setup commands called before here */
DCV2001();                    /* Put 2001 in DC Volts mode */
func = Get2001Func();
/* func = 1, which is the CONSTant DCV */
```

5.6 char *Check2001Val(char *response, double CheckVal, double MinVal, double MaxVal, int CheckMinInf);

Description

Checks a given double precision value (CheckVal) against the given minimum allowed value (MinVal) and the maximum allowed value (MaxVal). It is used internally by the 2001/7001 routines to check parameter ranges.

Parameters

CheckVal

Any double precision value or one of the following constants:

INF = 9.9E+37
MINIMUM = 9.8E+37
MAXIMUM = 9.7E+37
DEFAULT = 9.6E+37

CheckMinInf

0 - don't check for MINIMUM, MAXIMUM, DEFAULT, or INF
1 - check for MINIMUM, MAXIMUM, or DEFAULT
2 - check for MINIMUM, MAXIMUM, DEFAULT, or INF

Associated constants:

CHECKNONE = 0
CHECKMINMAX = 1
CHECKINF = 2

Returns

Pointer to response with response =

- a null string if CheckVal is out of range.
- an ASCII string representation of CheckVal if in range.
- "MAX", "MIN", "DEF", or "INF" if CheckVal is equivalent to one of the constants above.

Example

```
/* Quick C setup commands called before here */
/* Checks for a value to be between 100 an 1000 inclusive */
Check2001Val( response, MINIMUM, 100., 1000., CHECKMINMAX);
/* response = "MIN" */
```

5.7 int Hit2001Key(int HitKey);

Description

Presses the specified 2001 front panel key using SYSTEM:KEY.

Parameters

HitKey

1-23, 26-31

Defined keys are as follows:

UPKEY = 1	TEMPKEY = 2	LEFTKEY = 3
MENUKEY = 4	ACIKEY = 5	STOREKEY = 6
LOCALKEY = 7	PREVIOUSKEY = 8	AUTOKEY = 9
RIGHTKEY = 10	EXITKEY = 11	R2KEY = 12
RECALLKEY = 13	CHANKEY = 14	DCVKEY = 15
NEXTKEY = 16	DOWNKEY = 17	ENTERKEY = 18
R4KEY = 19	FILTERKEY = 20	SCANKEY = 21
ACVKEY = 22	RELKEY = 23	FREQKEY = 26
MATHKEY = 27	CONFIGKEY = 28	DCIKEY = 29
TRIGKEY = 30	INFOKEY = 31	

Returns

0 if no errors.
-1 if a parameter is out of range.
-2 if an IEEE-488 timeout error occurred.

Example

```
/* Quick C setup commands called before here */
Hit2001Key(AUTOKEY); /* Hit Auto Range key on 2001 */
```

5.8 int Disp2001(char *line1, char *line2);
int Disp7001(char *line1, char *line2);

Description

Immediately displays line1 on the first line and line2 on the second line of the Model 2001/7001's display.

Parameters

line1 maximum of 20 characters
line2 maximum of 32 characters

Returns

0 if no errors.
-1 if the length of line1 or 2 is out of range.
-2 if an IEEE-488 timeout error occurred.

Example

```
int error_code;
char line1[] = "2001/7001 Support Software";
/* Quick C setup commands called before here */
error_code = Disp2001( Line1, "(c) 1992 Keithley Instruments");
/* error_code = -2 since line1 is too long (>20 characters)*/
NoDisp2001();              /* turn off user display */
```

5.9 int NoDisp2001(void); int NoDisp7001(void);

Description

Turns off the user's displayed messages on the Model 2001/7001.

Returns

0 if no errors
-2 if an IEEE-488 timeout error occurred.

5.10 char *strins(char *dst, char *src);

Description

Inserts the a null terminated source string into the beginning of the destination string.

Parameters

dst string that is to have src inserted into it
src string to insert into dst

Returns

Pointer to dst if successful, NULL otherwise.

Example

```
char resp[255] = “World!”;
/* Quick C setup commands called before here */
strins( resp, “Hello “);
/* resp == “Hello World!” */
```

5.11 char *str_ltrim(char *trim_string);

Description

Strips leading spaces from the given null terminated string.

Parameters

trim_string

String that needs to have leading spaces stripped from it.

Returns

A pointer to trim_string if successful, NULL otherwise.

Example

```
/* Quick C setup commands called before here */
char trim_string[] = “ 3 leading spaces”;
str_ltrim( trim_string);
/* trim_string == “3 leading spaces” */
```

5.12 char *str_rtrim(char *trim_string);

Description

Strips trailing spaces from the given null terminated string.

Parameters

trim_string

String that needs to have leading spaces stripped from it.

Returns

A pointer to trim_string if successful, NULL otherwise.

Example

```
char trim_string[] = “3 trailing spaces “;
```

```
/* Quick C setup commands called before here */
str_rtrim( trim_string);
/* trim_string == "3 trailing spaces" */
```

5.13 char *dtoa(double value, char *result);

Description

Converts the given double precision value into a result and returns a pointer to it or NULL if unsuccessful.

Parameters

value

Double precision value to convert to a string.

result

Null terminated string representation of value.

Returns

Pointer to result if successful, otherwise NULL.

Example

```
double value = 45.342323;
char result[255];
/* Quick C setup commands called before here */
dtoa( value, result);
/* result = "45.342323" */
```

5.14 char *ftoa(float value, char *result);

Description

Converts the given single precision value into result and returns a pointer to it or NULL is unsuccessful.

Parameters

value

Single precision value to convert to a string.

result

Null terminated string representation of value.

Returns

Pointer to result if successful, otherwise NULL.

Example

```
float value = 45.342323f;
char result[255];
/* Quick C setup commands called before here */
ftoa( value, result);
/* result = “45.34232” */
```

Section 6
Model 2001 SENSe[1] Subsystem Commands

See Model 2001 Minimum and Maximum Sense Constants in Appendix B for CONSTants for use with the Model 2001 SENSe[1] Subsystem commands.

6.1 Model 2001 Function Change Subroutines

Theses subroutines change the present function of the Model 2001.

6.1.1 int DCV2001(void);

Description

Puts Model 2001 into DC Volts mode.

Returns

0 if no errors.
-2 if an IEEE-488 timeout error occurred.

Example

```
/* Quick C setup commands called before here */
/* All three of the following put the 2001 in DC Volts: */
DCV2001();
Set2001Function(DCV);
Send2001( "FUNC 'VOLT:DC'");
```

6.1.2 int ACV2001(void);

Description

Puts Model 2001 into AC Volts mode.

Returns

0 if no errors.
-2 if an IEEE-488 timeout error occurred.

6.1.3 int DCA2001(void);

Description

Puts 2001 into DC Current mode.

Returns

0 if no errors.
-2 if an IEEE-488 timeout error occurred.

6.1.4 int ACA2001(void);

Description

Puts 2001 into AC Current mode.

Returns

0 if no errors.
-2 if an IEEE-488 timeout error occurred.

6.1.5 int RES2_2001(void);

Description

Puts 2001 into 2-wire Resistance mode.

Returns

0 if no errors.
-2 if an IEEE-488 timeout error occurred.

6.1.6 int RES4_2001(void);

Description

Puts 2001 into 4-wire Resistance mode.

Returns

0 if no errors.
-2 if an IEEE-488 timeout error occurred.

6.1.7 int FREQ2001(void);

Description

Puts 2001 into Frequency mode.

Returns

0 if no errors.
-2 if an IEEE-488 timeout error occurred.

6.1.8 int TEMP2001(void);

Description

Puts 2001 into Temperature Mode.

Returns

0 if no errors.
-2 if an IEEE-488 timeout error occurred.

6.2 Return Model 2001 Reading Functions

Theses subroutines return the latest reading on the specified function of the Model 2001. All routines use the SCPI command, MEASure:(FunctionName)?, except Get2001Rdg which uses the SCPI command, "FETCh?".

6.2.1 char *Get2001Rdg(char *response);

Description

FETCHes a 2001 reading in the present Mode and FORMat.

Returns

Pointer to response with response = "Error!" if an IEEE-488 timeout error occurred, or Reading String.

Example

```
/* Quick C setup commands called before here */
/* The following two statements are equivalent: */
Get2001Rdg( response);
Q2001( response, "FETCH?");
```

6.2.2 char *GetDCV2001(char *response);

Description

Gets a 2001 DC Volts reading.

Returns

Pointer to response with response = “Error!” if an IEEE-488 timeout error occurred, or DC Volts Reading String.

Example

```
/* Quick C setup commands called before here */
/* The following two statements are equivalent: */
GetDCV2001( response);
Q2001( response, “MEASURE:VOLT:DC?”);
```

6.2.3 char *GetACV2001(char *response);

Description

Gets a 2001 AC Volts reading.

Returns

Pointer to response with response = “Error!” if an IEEE-488 timeout error occurred, or AC Volts Reading String.

6.2.4 char *GetDCA2001(char *response);

Description

Gets a 2001 DC Current reading.

Returns

Pointer to response with response = “Error!” if an IEEE-488 timeout error occurred, or DC Current Reading String.

6.2.5 char *GetACA2001(char *response);

Description

Gets a 2001 AC Current reading.

Returns

Pointer to response with response = “Error!” if an IEEE-488 timeout error occurred, or AC Current Reading String.

6.2.6 char *Get2RES2001(char *response);

Description

Gets a 2001 2-wire Resistance reading.

Returns

Pointer to response with response = “Error!” if an IEEE-488 timeout error occurred, or 2-wire Resistance Reading String.

6.2.7 char *Get4RES2001(char *response);

Description

Gets a 2001 4-wire Resistance reading.

Returns

Pointer to response with response = “Error!” if an IEEE-488 timeout error occurred, or 4-wire Resistance Reading String.

6.2.8 char *GetFREQ2001(char *response);

Description

Gets a 2001 Frequency reading.

Returns

Pointer to response with response = “Error!” if an IEEE-488 timeout error occurred, or Frequency Reading String.

6.2.9 char *GetTEMP2001(char *response);

Description

Gets a 2001 Temperature reading.

Returns

Pointer to response with response = “Error!” if an IEEE-488 timeout error occurred, or Temperature Reading String in response.

6.3 DC Voltage Functions

The functions set up and return the settings of the configurable options of the Model 2001's DC Voltage measurement function.

6.3.1 int Set2001DCV(double Range, double Time, double Digits);

Description

Sets the Model 2001's DC Voltage Range, Aperture Time, and Number of Digits.

Parameters

Range

0 to +1100
MAXIMUM, MINIMUM, or DEFAULT

Time (Aperture Time)

166.667e-6s to 166.666667e-3s (60Hz); 2e-4 to .2 (50Hz)
MAXIMUM, MINIMUM, or DEFAULT

Digits

3.5 to 8.4999 (7.5-8.4999=>7.5d, 3.5-4.4999=>3.5d)
MAXIMUM, MINIMUM, or DEFAULT

Returns

0 if no errors
-1 if a parameter is out of range.
-2 if an IEEE-488 timeout error occurred.

Example

```
/* Quick C setup commands called before here */
/* Set DC Voltage MAXIMUM range, 1 Power Line Cycle */
/* Integration, and 6-1/2 digits */
Set2001DCV( MAXIMUM, 0.01666667, 7.0);
```

6.3.2 char *Set2001DCVQ(char *response);

Description

Queries the Model 2001 for its DC Voltage Range, Aperture Time, and Number of Digits settings.

Returns

Pointer to response with response =

"Error!" if an IEEE-488 timeout error occurred, or
Query 1: Range: 0 to +1100
Query 2: Time: 166.667e-6 to .2
Query 3: Digits: 3.5 to 8.4999 (4=3.5d, 8=7.5d, etc.)

Use ParseQuery to separate return string into components.

Example

```
char Range[20], AperTime[20], Digits[20];
/* Quick C setup commands called before here */
Set2001DCVQ(response);                  /* Get 2001 DC Voltage Settings */
ParseQuery( Range, response, 1);        /* Extract Range setting */
ParseQuery( AperTime, response, 2);     /* Extract Aperture Time */
ParseQuery( Digits, response, 3);       /* Extract Number of Digits */
```

6.3.3 int Auto2001DCV(int AutoRange, int AutoTime, int AutoDigits);

Description

Sets the Model 2001's DC Voltage Auto Range, Auto Time, and Auto Digits settings.

Parameters

AutoRange, AutoTime, AutoDigits:

-1 = Don't set, 0 = Off, +1 = On, +2 = Once

The following constants can also be used:

NO = -1, TOFF = 0, TON = +1, ONCE = +2

Returns

0 if no errors
-1 if a parameter is out of range.
-2 if an IEEE-488 timeout error occurred.

Example

```
/* Quick C setup commands called before here */
/* Turn 2001's DC Voltage Auto Range ON, Time OFF, */
/* and Digits ONCE: */
Auto2001DCV( TON, TOFF, ONCE);
```

6.3.4 char *Auto2001DCVQ(char *response);

Description

Queries the Model 2001 for its DC Voltage Auto Range, Auto Time, and Auto Digits settings.

Returns

Pointer to response with response =

"Error!" if an IEEE-488 timeout error occurred, or
Query 1: Auto Range: 1 or 0 (1=ON, 0=OFF)
Query 2: Auto Time: 1 or 0 (1=ON, 0=OFF)
Query 3: Auto Digits: 1 or 0 (1=ON, 0=OFF)

Use ParseQuery to separate return string into components.

Example

```
char AutoRange[20], AutoTime[20], AutoDigits[20];
/* Quick C setup commands called before here */
Auto2001DCVQ( response);                    /* Get 2001 DC Voltage Auto Settings */
ParseQuery( AutoRange, response, 1);        /* Extract Auto Range setting */
ParseQuery( AutoTime, response, 2);         /* Extract Auto Time */
ParseQuery( AutoDigits, response, 3);       /* Extract Auto Digits */
```

6.4 AC Voltage Functions

The functions set up and return the settings of the configurable options of the Model 2001's AC Voltage measurement function.

6.4.1 int Set2001ACV(double Range, double Time, double Digits);

Description

Sets the Model 2001's AC Voltage Range, Aperture Time, and Number of Digits.

Parameters

Range

0 to +775 (+1100 if Peak, +Peak or -Peak On)
MAXIMUM, MINIMUM, or DEFAULT

Time (Aperture Time)

166.667e-6s to 166.666667e-3s (60Hz); 2e-4 to .2 (50Hz)
MAXIMUM, MINIMUM, or DEFAULT

Digits

3.5 to 8.4999 (7.5-8.4999=>7.5d, 3.5-4.4999=>3.5d)
MAXIMUM, MINIMUM, or DEFAULT

Returns

0 if no errors
-1 if a parameter is out of range.
-2 if an IEEE-488 timeout error occurred.

Example

```
/* Quick C setup commands called before here */
/* Set AC Voltage MINIMUM range, 10 Power Line Cycle */
/* Integration, and 6-1/2 digits */
Set2001ACV( MAXIMUM, 0.1666667, 6.5);
```

6.4.2 char *Set2001ACVQ(char *response);

Description

Queries the Model 2001 for its AC Voltage Range, Aperture Time, and Number of Digits settings.

Returns

Pointer to response with response =

"Error!" if an IEEE-488 timeout error occurred, or
Query 1: Range: 0 to +1100
Query 2: Time: 166.667e-6 to .2
Query 3: Digits: 3.5 to 8.4999 (4=3.5d, 8=7.5d, etc.)

Use ParseQuery to separate return string into components.

Example

```
#include <stdlib.h>
double Range, AperTime, Digits;
char temp[64];
/* Quick C setup commands called before here */
Set2001ACVQ( response);                          /* Get 2001 AC Voltage Settings */
Range = atof(ParseQuery( temp, response, 1));    /* Extract Range setting */
AperTime = atof(ParseQuery( temp, response, 2)); /* Extract Aperture Time */
Digits = atof(ParseQuery( temp, response, 3));   /* Extract Number of Digits */
```

6.4.3 int Auto2001ACV(int AutoRange, int AutoTime, int AutoDigits);

Description

Sets the Model 2001's AC Voltage Auto Range, Auto Time, and Auto Digits settings.

Parameters

AutoRange, AutoTime, AutoDigits

-1 = Don't set, 0 = Off, +1 = On, +2 = Once

The following constants can also be used:

NO = -1, TOFF = 0, TON = +1, ONCE = +2

Returns

0 if no errors
-1 if a parameter is out of range.
-2 if an IEEE-488 timeout error occurred.

Example

```
/* Quick C setup commands called before here */
/* Turn 2001's AC Voltage Auto Range ON, Time OFF, */
/* and Digits unaffected: */
Auto2001ACV( TON, TOFF, NO);
```

6.4.4 char *Auto2001ACVQ(char *response);

Description

Queries the Model 2001 for its AC Voltage Auto Range, Auto Time, and Auto Digits settings.

Returns

Pointer to response with response =

"Error!" if an IEEE-488 timeout error occurred, or
Query 1: Auto Range: 1 or 0 (1=ON, 0=OFF)
Query 2: Auto Time: 1 or 0 (1=ON, 0=OFF)
Query 3: Auto Digits: 1 or 0 (1=ON, 0=OFF)

Use ParseQuery to separate return string into components.

Example

```
char AutoRange[20], AutoTime[20], AutoDigits[20];
/* Quick C setup commands called before here */
Auto2001ACVQ( response);                    /* Get 2001 AC Voltage Auto Settings */
```

```
ParseQuery( AutoRange, response, 1);        /* Extract Auto Range setting */
ParseQuery( AutoTime, response, 2);         /* Extract Auto Time */
ParseQuery( AutoDigits, response, 3);       /* Extract Auto Digits */
```

6.5 DC Current Functions

The functions set up and return the settings of the configurable options of the Model 2001's DC Current measurement function.

6.5.1 int Set2001DCA(double Range, double Time, double Digits);

Description

Sets the Model 2001's DC Current Range, Aperture Time, and Number of Digits.

Parameters

Range

0 to +2.1, ignored if In Circuit Current Mode is enabled.
MAXIMUM, MINIMUM, or DEFAULT

Time (Aperture Time)

166.667e-6s to 166.666667e-3s (60Hz); 2e-4 to .2 (50Hz)
MAXIMUM, MINIMUM, or DEFAULT

Digits

3.5 to 8.4999 (7.5-8.4999=>7.5d, 3.5-4.4999=>3.5d)
MAXIMUM, MINIMUM, or DEFAULT

Returns

0 if no errors
-1 if a parameter is out of range.
-2 if an IEEE-488 timeout error occurred.

Example

```
/* Quick C setup commands called before here */
/* Set DC Current to .2A range, 10ms Integration, & 6-1/2d */
Set2001DCA( 0.2, 0.01, 7.1);
```

6.5.2 char *Set2001DCAQ(char *response);

Description

Queries the Model 2001 for its DC Current Range, Aperture Time, and Number of Digits settings.

Returns

Pointer to response with response =

"Error!" if an IEEE-488 timeout error occurred, or
Query 1: Range: 0 to +2.1
Query 2: Time: 166.667e-6 to .2
Query 3: Digits: 3.5 to 8.4999 (4=3.5d, 8=7.5d, etc.)

Use ParseQuery to separate return string into components.

Example

```
char Range[20], AperTime[20], Digits[20];
/* Quick C setup commands called before here */
Set2001DCAQ(response);                  /* Get 2001 DC Current Settings */
ParseQuery( Range, response, 1);        /* Extract Range setting */
ParseQuery( AperTime, response, 2);     /* Extract Aperture Time */
ParseQuery( Digits, response, 3);       /* Extract Number of Digits */
```

6.5.3 int Auto2001DCA(int AutoRange, int AutoTime, int AutoDigits);

Description

Sets the Model 2001's DC Current Auto Range, Auto Time, and Auto Digits settings.

Parameters

AutoRange, AutoTime, AutoDigits

-1 = Don't set, 0 = Off, +1 = On, +2 = Once

The following constants can also be used:

NO = -1, TOFF = 0, TON = +1, ONCE = +2

Returns

0 if no errors
-1 if a parameter is out of range.
-2 if an IEEE-488 timeout error occurred.

Example

```
/* Quick C setup commands called before here */
/* Turn DC Current Auto Range ON, Time OFF, and Digits ONCE: */
Auto2001DCA( 1, 0, 2);
```

6.5.4 char *Auto2001DCAQ(char *response);

Description

Queries the Model 2001 for its DC Current Auto Range, Auto Time, and Auto Digits settings.

Returns

Pointer to response with response =

"Error!" if an IEEE-488 timeout error occurred, or
Query 1: Auto Range: 1 or 0 (1=ON, 0=OFF)
Query 2: Auto Time: 1 or 0 (1=ON, 0=OFF)
Query 3: Auto Digits: 1 or 0 (1=ON, 0=OFF)

Use ParseQuery to separate return string into components.

Example

```
char AutoRange[20], AutoTime[20], AutoDigits[20];
/* Quick C setup commands called before here */
Auto2001DCAQ( response);                    /* Get 2001 DC Current Auto Settings */
ParseQuery( AutoRange, response, 1);        /* Extract Auto Range setting */
ParseQuery( AutoTime, response, 2);         /* Extract Auto Time */
ParseQuery( AutoDigits, response, 3);       /* Extract Auto Digits */
```

6.6 AC Current Functions

The functions set up and return the settings of the configurable options of the Model 2001's AC Current measurement function.

6.6.1 int Set2001ACA(double Range, double Time, double Digits);

Description

Sets the Model 2001's AC Current Range, Aperture Time, and Number of Digits.

Parameters

Range

0 to +2.1
MAXIMUM, MINIMUM, or DEFAULT

Time (Aperture Time)

166.667e-6s to 166.666667e-3s (60Hz); 2e-4 to .2 (50Hz)
MAXIMUM, MINIMUM, or DEFAULT

Digits

3.5 to 8.4999 (7.5-8.4999=>7.5d, 3.5-4.4999=>3.5d)
MAXIMUM, MINIMUM, or DEFAULT

Returns

0 if no errors
-1 if a parameter is out of range.
-2 if an IEEE-488 timeout error occurred.

Example

```
/* Quick C setup commands called before here */
/* Set AC Current to 2mA range, .1 msec */
/* Integration, and 7-1/2 digits */
Set2001ACA( 2e-3, 1e-4, 7.5);
```

6.6.2 char *Set2001ACAQ(char *response);

Description

Queries the Model 2001 for its AC Current Range, Aperture Time, and Number of Digits settings.

Returns

Pointer to response with response =

"Error!" if an IEEE-488 timeout error occurred, or
Query 1: Range: 0 to +2.1
Query 2: Time: 166.667e-6 to .2
Query 3: Digits: 3.5 to 8.4999 (4=3.5d, 8=7.5d, etc.)

Use ParseQuery to separate return string into components.

Example

```
char Range[20], AperTime[20], Digits[20];
/* Quick C setup commands called before here */
Set2001ACAQ(response);                  /* Get 2001 AC Current Settings */
ParseQuery( Range, response, 1);        /* Extract Range setting */
ParseQuery( AperTime, response, 2);     /* Extract Aperture Time */
ParseQuery( Digits, response, 3);       /* Extract Number of Digits */
```

6.6.3 int Auto2001ACA(int AutoRange, int AutoTime, int AutoDigits);

Description

Sets the Model 2001's AC Current Auto Range, Auto Time, and Auto Digits settings.

Parameters

AutoRange, AutoTime, AutoDigits

-1 = Don't set, 0 = Off, +1 = On, +2 = Once

The following constants can also be used:

NO = -1, TOFF = 0, TON = +1, ONCE = +2

Returns

0 if no errors
-1 if a parameter is out of range.
-2 if an IEEE-488 timeout error occurred.

Example

```
/* Quick C setup commands called before here */
/* Turn 2001's AC Current Auto Range ON, Time OFF, */
/* and Digits unaffected: */
Auto2001ACA( TON, 0, -1);
```

6.6.4 char *Auto2001ACAQ(char *response);

Description

Queries the Model 2001 for its AC Current Auto Range, Auto Time, and Auto Digits settings.

Returns

Pointer to response with response =

"Error!" if an IEEE-488 timeout error occurred, or
Query 1: Auto Range: 1 or 0 (1=ON, 0=OFF)
Query 2: Auto Time: 1 or 0 (1=ON, 0=OFF)
Query 3: Auto Digits: 1 or 0 (1=ON, 0=OFF)

Use ParseQuery to separate return string into components.

Example

```
char AutoRange[20], AutoTime[20], AutoDigits[20];
/* Quick C setup commands called before here */
Auto2001ACAQ( response);                /* Get 2001 AC Current Auto Settings */
ParseQuery( AutoRange, response, 1);    /* Extract Auto Range setting */
ParseQuery( AutoTime, response, 2);     /* Extract Auto Time */
ParseQuery( AutoDigits, response, 3);   /* Extract Auto Digits */
```

6.7 2-Wire Resistance Functions

The functions set up and return the settings of the configurable options of the Model 2001's 2-wire Resistance measurement function.

6.7.1 int Set2001R2(double Range, double Time, double Digits);

Description

Sets the Model 2001's 2-wire Resistance Range, Aperture Time, and Number of Digits.

Parameters

Range

0 to +1.05e9 or 2.1e5 if SENSe1:RESistance:OCOMpensated ON is set.
MAXIMUM, MINIMUM, or DEFAULT

Time (Aperture Time)

166.667e-6s to 166.666667e-3s (60Hz); 2e-4 to .2 (50Hz)
MAXIMUM, MINIMUM, or DEFAULT

Digits

3.5 to 8.4999 (7.5-8.4999=>7.5d, 3.5-4.4999=>3.5d)
MAXIMUM, MINIMUM, or DEFAULT

Returns

0 if no errors
-1 if a parameter is out of range.
-2 if an IEEE-488 timeout error occurred.

Example

```
/* Quick C setup commands called before here */
/* Set 2-wire Resistance to 200kΩ range, 100 msec */
/* Integration, and 5-1/2 digits */
Set2001R2( 1e5, 0.1, 6.49);
```

6.7.2 char *Set2001R2Q(char *response);

Description

Queries the Model 2001 for its 2-wire Resistance Range, Aperture Time, and Number of Digits settings.

Returns

Pointer to response with response =

"Error!" if an IEEE-488 timeout error occurred, or
Query 1: Range: 0 to +1.05e9
Query 2: Time: 166.667e-6 to .2
Query 3: Digits: 3.5 to 8.4999 (4=3.5d, 8=7.5d, etc.)

Use ParseQuery to separate return string into components.

Example

```
char Range[20], AperTime[20], Digits[20];
/* Quick C setup commands called before here */
Set2001R2Q(response);                    /* Get 2001 2-wire Resistance settings */
ParseQuery( Range, response, 1);         /* Extract Range setting */
ParseQuery( AperTime, response, 2);      /* Extract Aperture Time */
ParseQuery( Digits, response, 3);        /* Extract Number of Digits */
```

6.7.3 int Auto2001R2(int AutoRange, int AutoTime, int AutoDigits);

Description

Sets the Model 2001's 2-wire Resistance Auto Range, Auto Time, and Auto Digits settings.

Parameters

AutoRange, AutoTime, AutoDigits

-1 = Don't set, 0 = Off, +1 = On, +2 = Once

The following constants can also be used:

NO = -1, TOFF = 0, TON = +1, ONCE = +2

Returns

0 if no errors
-1 if a parameter is out of range.
-2 if an IEEE-488 timeout error occurred.

Example

```
/* Quick C setup commands called before here */
/* Turn 2001's 2-wire Resistance Auto Range ON, Time OFF, */
/* and Digits ONCE: */
Auto2001R2( TON, TOFF, ONCE);
```

6.7.4 char *Auto2001R2Q(char *response);

Description

Queries the Model 2001 for its 2-wire Resistance Auto Range, Auto Time, and Auto Digits settings.

Returns

Pointer to response with response =

"Error!" if an IEEE-488 timeout error occurred, or
Query 1: Auto Range: 1 or 0 (1=ON, 0=OFF)
Query 2: Auto Time: 1 or 0 (1=ON, 0=OFF)
Query 3: Auto Digits: 1 or 0 (1=ON, 0=OFF)

Use ParseQuery to separate return string into components.

Example

```
char AutoRange[20], AutoTime[20], AutoDigits[20];
/* Quick C setup commands called before here */
Auto2001R2Q( response);                    /* Get 2001 2-wire Resistance Auto Settings */
ParseQuery( AutoRange, response, 1);       /* Extract Auto Range setting */
ParseQuery( AutoTime, response, 2);        /* Extract Auto Time */
ParseQuery( AutoDigits, response, 3);      /* Extract Auto Digits */
```

6.8 4-Wire Resistance Functions

The functions setup and return the settings of the configurable options of the 2001's 4-wire Resistance measurement function.

6.8.1 int Set2001R4(double Range, double Time, double Digits);

Description

Sets the Model 2001's 4-wire Resistance Range, Aperture Time, and Number of Digits.

Parameters

Range

0 to 2.1e5
MAXIMUM, MINIMUM, or DEFAULT

Time (Aperture Time)

166.667e-6s to 166.666667e-3s (60Hz); 2e-4 to .2 (50Hz)
MAXIMUM, MINIMUM, or DEFAULT

Digits

3.5 to 8.4999 (7.5-8.4999=>7.5d, 3.5-4.4999=>3.5d)
MAXIMUM, MINIMUM, or DEFAULT

Returns

0 if no errors
-1 if a parameter is out of range.
-2 if an IEEE-488 timeout error occurred.

Example

```
/* Quick C setup commands called before here */
/* Set 4-wire Resistance to 20kΩ range, 100 msec */
/* Integration, and 5-1/2 digits */
Set2001R4( 15000.0, 100e-3, 5.649);
```

6.8.2 char *Set2001R4Q(char *response);

Description

Queries the Model 2001 for its 4-wire Resistance Range, Aperture Time, and Number of Digits settings.

Returns

Pointer to response with response =

"Error!" if an IEEE-488 timeout error occurred, or
Query 1: Range: 0 to +2.1e5
Query 2: Time: 166.667e-6 to .2
Query 3: Digits: 3.5 to 8.4999 (4=3.5d, 8=7.5d, etc.)

Use ParseQuery to separate return string into components.

Example

```
char Range[20], AperTime[20], Digits[20];
/* Quick C setup commands called before here */
Set2001R4Q(response);                    /* Get 2001 4-wire Resistance settings */
ParseQuery( Range, response, 1);         /* Extract Range setting */
ParseQuery( AperTime, response, 2);      /* Extract Aperture Time */
ParseQuery( Digits, response, 3);        /* Extract Number of Digits */
```

6.8.3 int Auto2001R4(int AutoRange, int AutoTime, int AutoDigits);

Description

Sets the Model 2001's 4-wire Resistance Auto Range, Auto Time, and Auto Digits settings.

Parameters

AutoRange, AutoTime, AutoDigits

-1 = Don't set, 0 = Off, +1 = On, +2 = Once

The following constants can also be used:

NO = -1, TOFF = 0, TON = +1, ONCE = +2

Returns

0 if no errors
-1 if a parameter is out of range.
-2 if an IEEE-488 timeout error occurred.

Example

```
/* Quick C setup commands called before here */
/* Turn 2001's 4-wire Resistance Auto Range ON, Time OFF, */
/* and Digits ONCE: */
Auto2001R4( TON, TOFF, ONCE);
```

6.8.4 char *Auto2001R4Q(char *response);

Description

Queries the Model 2001 for its 4-wire Resistance Auto Range, Auto Time, and Auto Digits settings.

Returns

Pointer to response with response =

"Error!" if an IEEE-488 timeout error occurred, or
Query 1: Auto Range: 1 or 0 (1=ON, 0=OFF)
Query 2: Auto Time: 1 or 0 (1=ON, 0=OFF)
Query 3: Auto Digits: 1 or 0 (1=ON, 0=OFF)

Use ParseQuery to separate return string into components.

Example

```
char AutoRange[20], AutoTime[20], AutoDigits[20];
/* Quick C setup commands called before here */
Auto2001R4Q( response);                  /* Get 2001 4-wire Resistance Auto Settings */
ParseQuery( AutoRange, response, 1);     /* Extract Auto Range setting */
ParseQuery( AutoTime, response, 2);      /* Extract Auto Time */
ParseQuery( AutoDigits, response, 3);    /* Extract Auto Digits */
```

6.9 Frequency Functions

The functions set up and return the settings of the configurable options of the Model 2001's Frequency measurement function.

6.9.1 int Set2001F(double Digits, int Source);

Description

Sets the Model 2001's Frequency Number of Digits and Source settings.

Parameters

Digits

3.5 to 5.4999 (4.5-5.4999=>4.5d, 3.5-4.4999=>3.5d)
MAXIMUM, MINIMUM, or DEFAULT

Source

0 - Current
1 - Voltage

Returns

0 if no errors
-1 if a parameter is out of range.
-2 if an IEEE-488 timeout error occurred.

Example

```
/* Quick C setup commands called before here */
/* Set Frequency to 5 digits, Current triggered */
Set2001F( 5.49, 0);
```

6.9.2 char *Set2001FQ(char *response);

Description

Queries the Model 2001 for its Frequency Number of Digits and Measurement Source settings.

Returns

Pointer to response with response =

"Error!" if an IEEE-488 timeout error occurred, or
Query 1: Digits: 3.5 to 5.4999 (4=3.5d, 5=4.5d)
Query 2: Source: VOLT or CURR

Use ParseQuery to separate return string into components.

Example

```
char Digits[20], Source[20];
/* Quick C setup commands called before here */
Set2001FQ(response);                    /* Get 2001 Frequency settings */
```

```
ParseQuery( Digits, response, 1);          /* Extract Number of Digits */
ParseQuery( Source, response, 2);          /* Extract Frequency Source */
```

6.10 Temperature Functions

The functions set up and return the settings of the configurable options of the Model 2001's Temperature measurement function.

6.10.1 int Set2001T(double Time, double Digits);

Description

Sets the Model 2001's Temperature Aperture Time and Number of Digits.

Parameters

Time (Aperture time):

166.667e-6s to 166.666667e-3s (60Hz); 2e-4 to .2 (50Hz)
MAXIMUM, MINIMUM, or DEFAULT

Digits

3.5 to 7.4999 (6.5-7.4999=>6.5d, 3.5-4.4999=>3.5d)
MAXIMUM, MINIMUM, or DEFAULT

Returns

0 if no errors
-1 if a parameter is out of range.
-2 if an IEEE-488 timeout error occurred.

Example

```
/* Quick C setup commands called before here */
/* Set Temperature to 100 msec Integration, and 5-1/2 digits */
Set2001T( 0.1, 6.49);
```

6.10.2 char *Set2001TQ(char *response);

Description

Queries the Model 2001 for its Temperature Aperture Time and Number of Digits settings.

Returns

Pointer to response with response =

"Error!" if an IEEE-488 timeout error occurred, or

Query 1: Time: 166.667e-6 to .2
Query 2: Digits: 3.5 to 7.4999 (4=3.5d, 7=6.5d, etc.)

Use ParseQuery to separate return string into components.

Example

```
char AperTime[20], Digits[20];
/* Quick C setup commands called before here */
Set2001TQ(response);                    /* Get 2001 Temperature settings */
ParseQuery( AperTime, response, 1);     /* Extract Aperture Time */
ParseQuery( Digits, response, 2);       /* Extract Number of Digits */
```

6.10.3 int Auto2001T(int AutoTime, int AutoDigits);

Description

Set the Model 2001's Temperature Auto Time and Auto Digits settings.

Parameters

AutoTime, AutoDigits

-1 = Don't set, 0 = Off, +1 = On, +2 = Once

The following constants can also be used:

NO = -1, TOFF = 0, TON = +1, ONCE = +2

Returns

0 if no errors
-1 if a parameter is out of range.
-2 if an IEEE-488 timeout error occurred.

Example

```
/* Quick C setup commands called before here */
/* Turn 2001's Temperature Auto Time OFF, and Digits ONCE: */
Auto2001T( TOFF, ONCE);
```

6.10.4 char *Auto2001TQ(char *response);

Description

Queries the Model 2001 for its Temperature Auto Time and Auto Digits settings.

Returns

Pointer to response with response =

"Error!" if an IEEE-488 timeout error occurred, or
Query 1: Auto Time: 1 or 0 (1=ON, 0=OFF)
Query 2: Auto Digits: 1 or 0 (1=ON, 0=OFF)

Use ParseQuery to separate return string into components.

Example

```
char AutoTime[20], AutoDigits[20];
/* Quick C setup commands called before here */
Auto2001TQ( response);                  /* Get 2001 Temperature Auto Settings */
ParseQuery( AutoTime, response, 1);     /* Extract Auto Time */
ParseQuery( AutoDigits, response, 2);   /* Extract Auto Digits */
```

6.10.5 int Set2001RTD(int Mode, int RType, double Alpha, double Beta, double Delta, double RZero);

Description

Configure and use RTD's to make temperature measurements.

Parameters

Mode	2 or 4 (2-wire or 4-wire RTD)
RType	0 - PT385
	1 - PT3916
	2 - USER
Alpha	0.00 to 0.01 (ignored if not USER)
Beta	0.00 to 1.00 (ignored if not USER)
Delta	0.00 to 5.00 (ignored if not USER)
RZero	0 to 1000 (ignored if not USER)

Returns

0 if no errors
-1 if a parameter is out of range.
-2 if an IEEE-488 timeout error occurred.

Example

```
/* Quick C setup commands called before here */
/* Set RTD Mode to 4-wire RTD, User, Alpha=.005, Beta= .5 */
/* Delta=2.4, and RZero=500 */
Set2001RTD( 4, 2, 0.005, 0.5, 2.4, 500.0);
```

6.10.6 char *Set2001RTDQ(char *response);

Description

Queries the Model 2001 for the Temperature transducer Type, the RTD Type, Alpha, Beta, Delta, and RZero settings.

Returns

Pointer to response with response =

"Error!" if an IEEE-488 timeout error occurred, or

Query 1: Mode: RTD or FRTD or TC
Query 2: RType: USER, PT385, or PT3916
Query 3: Alpha: 0.00 to 0.01
Query 4: Beta: 0.00 to 1.00
Query 5: Delta: 0.00 to 5.00
Query 6: RZero: 0 to 1000

Use ParseQuery to separate return string into components.

Example

```
#include <stdlib.h>
double Alpha, Beta, Delta, RZero;
char Mode[20], RType[20], temp[20];
/* Quick C setup commands called before here */
Set2001RTDQ( response);                          /* Get 2001 RTD settings */
ParseQuery( Mode, response, 1);                  /* Extract Temp. Device */
ParseQuery( RType, response, 2);                 /* Extract RTD Type */
Alpha= atof(ParseQuery( temp, response, 3));     /* Extract Alpha setting */
Beta = atof(ParseQuery( temp, response, 4));     /* Extract Beta setting */
Delta= atof(ParseQuery( temp, response, 5));     /* Extract Delta setting */
RZero= atof(ParseQuery( temp, response, 6));     /* Extract RZero setting */
```

6.10.7 int Set2001TC(char TType);

Description

Sets the thermocouple type and uses TC's for temperature measurement.

Parameters

TType

'J', 'K', 'T', 'E', 'R', 'S', or 'B'

Returns

0 if no errors
-1 if a parameter is out of range.
-2 if an IEEE-488 timeout error occurred.

Example

```
/* Quick C setup commands called before here */
/* Use Type J thermocouples */
Set2001TC( 'J');
```

6.10.8 char *Set2001TCQ(char *response);

Description

Queries the Model 2001 for the Thermocouple Type.

Returns

Pointer to response with response =

"Error!" if an IEEE-488 timeout error occurred, or
"J", "K", "T", "E", "R", "S", or "B"

Example

```
/* Quick C setup commands called before here */
Set2001TCQ(response);          /* Get 2001 Thermocouple Type */
```

Section 7
Model 2001 / 7001 Status Commands

These routines control the SCPI Status Model of the Model 2001/7001. Constants (see Status Model Constants in Appendix B) are defined for all of the registers and their bits to make programming the Model 2001/7001 status model simpler. Please refer to the IEEE-488 Reference Section of the Model 2001 or 7001 manual for more details on the specific bit patterns of these registers and bit patterns.

NOTE

These functions and subroutines also support the Model 7002 Switch System. Just send Model 7001 commands to a Model 7002.

7.1 int Stat2001(int Event2001, unsigned int PTF, unsigned int NTF, unsigned int SEN);
int Stat7001(int Event7001, unsigned int PTF, unsigned int NTF, unsigned int SEN);

Description

Sets the Model 2001/7001's Operation, Trigger, Arm, Sequence, Questionable, or Measurement Event's Positive Transition Filter, Negative Transition Filter, and Status Enable Registers.

Parameters

Event2001, Event7001

1 = Operation Event
2 = Trigger Event
3 = Arm Event
4 = Sequence Event
5 = Questionable Event
6 = Measurement Event (2001 Only)

PTF (Positive Transition Filter)	0 to 32767
NTF (Negative Transition Filter)	0 to 32767
SEN (Status Enable Register)	0 to 32767

Returns

0 if no errors.
-1 if a parameter is out of range.
-2 if an IEEE-488 timeout error occurred.

Example

```
/* Quick C setup commands called before here */
/* Cause the TRIG and/or ARM bits to be set in the 2001's */
/* Operation Event Register when the 2001 enters or exits */
/* Triggering or Arming */
Stat2001( OPERATION, (TRIG | ARM), (TRIG | ARM), (TRIG | ARM));
```

7.2 char *Stat2001Q(char *response, int Event2001); char *Stat7001Q(char *response, int Event7001);

Description

Queries the Model 2001/7001's Operation, Trigger, Arm, Sequence, Questionable, or Measurement Event's Positive Transition Filter, Negative Transition Filter, and Status Enable Registers.

Parameters

Event2001, Event7001

1 = Operation Event
2 = Trigger Event
3 = Arm Event
4 = Sequence Event
5 = Questionable Event
6 = Measurement Event (2001 Only)

Returns

Pointer to response with response =

"Error!" if an IEEE-488 timeout error occurred, or NULL if Event2001 or Event7001 is out of range, or

Query 1: Status Register (0 to 32767)
Query 2: Condition Register (0 to 32767)
Query 3: Positive Transition Filter (0 to 32767)
Query 4: Negative Transition Filter (0 to 32767)
Query 5: Status Enable Register (0 to 32767)

Use ParseQuery to separate return string into components.

Example

```
char StatReg[20], CondReg[20], PTranReg[20];
char NTranReg[20], SEnableReg[20];
/* Quick C setup commands called before here */
Stat7001Q( response, OPERATION);          /* Read the Model 7001's Operation Event Registers */
ParseQuery( StatReg, response, 1);        /* Extract Status Register */
ParseQuery( CondReg, response, 2);        /* Extract Condition Register */
ParseQuery( PTranReg, response, 3);       /* Extract +Transition Reg. */
ParseQuery( NTranReg, response, 4);       /* Extract -Transition Reg. */
ParseQuery( SEnableReg, response, 5);     /* Status Enable Reg. */
```

7.3 int SRE2001(unsigned int mask);
int SRE7001(unsigned int mask);

Description

Enables the Model 2001/7001 to generate a Service Request when the indicated bit(s) of the Status Byte Register are set. If a bit is already set when this command is given, no Service Request is generated.

Parameters

mask

0 to 255

Returns

0 if no errors.
-1 if a parameter is out of range.
-2 if an IEEE-488 timeout error occurred.

Example

```
int Poll;
/* Quick C setup commands called before here */
Poll = Poll2001();                  /* Clear and pending SRQ's */
SRE2001( MAV);                      /* Set up to SRQ on MAV */
Send2001( "FETCH?");                /* fetch a 2001 reading */
do                                  /* Set up an uninterruptable wait */
loop( !Wait2001RQS());
Q2001( response, "");               /* Get reading */
```

7.4 int ESE2001(unsigned int mask);
int ESE7001(unsigned int mask);

Description

Sets which bits of the Model 2001/7001's Standard Event Status Register cause the Event Summary Bit (ESB) of the Status Byte Register to be set.

Parameters

mask

0 to 255

Returns

0 if no errors.
-1 if a parameter is out of range.
-2 if an IEEE-488 timeout error occurred.

Example

```
int RQS2001;
/* Quick C setup commands called before here */
/* Setup an SRQ on OPeration Complete */
ESE2001( OPC);
SRE2001( ESB);
Send2001( "VOLTage:DC:RANGe 200;*OPC");
RQS2001 = Wait2001RQS();                    /* Wait until range change complete */
```

7.5 char *OPC2001(char *response, int UnInterruptable); char *OPC7001(char *response, int UnInterruptable);

Description

Performs a Model 2001/7001 *OPC? command, which returns a "1" when the present operation is complete. However, the "1" may not be output by the 2001/7001 for a long time, which would cause an IEEE timeout on reading data immediately after sending the *OPC?. Thus, this routine waits for a Model 2001/7001 SRQ on MAV after sending the query. The Model 2001/7001 may wait forever to send the "1" out if :INITiate:CONTinuous ON was explicitly set as its is in the Factory Defaults. Thus, the :ABORt command would have to be issued before calling these routines.

Parameters

UnInterruptable

FALSE (0)	the command can be aborted by pressing the Esc key.
TRUE (non-zero)	the command cannot be aborted.

Returns

Pointer to response with response =

"Error!" if an IEEE-488 timeout error occurred, or
"1" if operation was completed
"Cancel" if aborted

Example

```
#include <string.h>
/* Quick C setup commands called before here */
/* Guarantee that *OPC? will not wait forever. */
if( !strcmp( Q2001( response, "INIT:CONT?"),"1")) Send2001( "ABORT");
Send2001( "VOLTage:DC:RANGe 200");
OPC2001( response, TRUE);                   /* Wait until range change complete */
```

7.6 int Clear2001(void);
int Clear7001(void);

Description

Sends the SCPI command *CLS (Clear Status Model) to the 2001/7001.

Returns

0 if no errors.
-2 if an IEEE-488 timeout error occurred.

Section 8
Model 2001 / 7001 Scanning Commands

These routines control the SCPI Trigger Model and Scanning functions of the Model 2001/7001. Please refer to the IEEE-488 Reference Section of the Model 2001 or 7001 manual for more details.

NOTE

These functions and subroutines also support the Model 7002 Switch System. Just send Model 7001 commands to a Model 7002.

See Model 2001/7001 Scanning Minimum and Maximum Constants in Appendix B for CONSTants for use with the Model 2001/7001 Scanning Commands.

8.1 int Close2001(int Channel);

Description

Closes a single channel on the Model 2001 scanner card.

Parameters

Channel

1 to 10

Returns

0 if no errors.
-1 if a parameter is out of range.
-2 if an IEEE-488 timeout error occurred.

Example

```
/* Quick C setup commands called before here */
Close2001( 3);                  /* Take Readings on Channel 3 */
```

8.2 int Close7001(char *ChanList);

Description

Closes the specified channels on the Model 7001 scanner card.

Parameters

ChanList

Any valid SCPI channel-list like (@1!1, 1!2, 1!4:1!10, 2!1!2) with valid channels depending on the scanner card selected. The Model 7001 EAV bit in the serial poll register must be checked to see if the command was accepted or look at the front panel of the instrument for an error message.

Returns

0 if no errors.
-1 if a parameter is out of range.
-2 if an IEEE-488 timeout error occurred.

Example

```
/* Quick C setup commands called before here */
Close7001( "(@1!3:1!6)");                    /* Close channels 3 to 6 on card 1 */
```

8.3 char *Close2001Q(char *response);
char *Close7001Q(char *response);

Description

Queries the Model 2001/7001 for a list of closed channels. The Model 2001 can have only one channel closed at a time on its internal scanner, whereas the Model 7001 may have none or all closed.

Returns

Pointer to response with response =

"Error!" if an IEEE-488 timeout error occurred, or a list of closed channels.

Example

```
/* Quick C setup commands called before here */
Close2001( 2);                               /* Close Channel 2 */
Close2001Q( response);                       /* response = "(@2)" */
```

8.4 int Open2001(int Channel);

Description

Opens one or all channel(s) on the 2001 scanner card.

Parameters

Channel

0 – Open all channels
1 to 10

Returns

0 if no errors.
-1 if a parameter is out of range.
-2 if an IEEE-488 timeout error occurred.

Example

```
/* Quick C setup commands called before here */
Open2001( 1);                    /* Open channel 1 */
Open2001( 0);                    /* Open all 2001 channels */
```

8.5 int Open7001(char *ChanList);

Description

Opens the specified channels on the Model 7001 scanner card.

Parameters

ChanList

Any valid SCPI channel-list with valid channels depending on the scanner card selected. The Model 7001 EAV bit in the serial poll register must be checked to see if the command was accepted or look at the front panel of the instrument for an error message.

Returns

0 if no errors.
-2 if an IEEE-488 timeout error occurred.

Example

```
/* Quick C setup commands called before here */
Open7001( "(@1!3:1!6)")          /* Open channels 3 to 6 on card 1 */
Open7001( "ALL")                 /* Open all 7001 channels */
```

8.6 int Scan2001(char *ChanList);
int Scan7001(char *ChanList);

Description

Defines the Model 2001's Internal ScanList or the Model 7001's scanlist.

Parameters

ChanList

Any valid SCPI channel-list like (@1, 2, 4:10) with channels ranging from 1 to 10 for the Model 2001, or like (@1!1:1!40, 2!1:2!40) for the Model 7001. The Model 2001/7001 EAV bit in the serial poll register must be checked to see if the command was accepted or look at the front panel of the instrument for an error message.

Returns

0 if no errors
-2 if an IEEE-488 timeout error occurred.

Example

```
/* Quick C setup commands called before here */
Scan2001( "(@1:10)");                    /* Scan All 2001 channels */
Scan7001( "(@1!1:1!40, 2!1:2!40)");      /* Scan All 7001 channels */
```

8.7 int Arm2001(double Count1, char *Source1, double Count2, double Delay2, char *Source2);
int Arm7001(double Count1, char *Source1, double Count2, double Delay2, char *Source2);

Description

Sets up the Model 2001/7001's Arm Layer 1 (Arm Layer) and Arm Layer 2 (Trigger Layer).

Parameters

Count1 and Count2

1 to 99999, 9.9e37
MAXIMUM, MINIMUM, DEFAULT, or INF

Source1 and Source2

HOLD	Hold
MANual	Manual
IMMediate	Immediate
TIMEr	Timer (Source2 only)
BUS	IEEE-488 Bus (GET or *TRG)
TLINk	Trigger Link
EXTernal	External

Delay2

0 to 999999.999 seconds
MAXIMUM, MINIMUM, or DEFAULT

Returns

0 if no errors
-1 if a parameter is out of range.
-2 if an IEEE-488 timeout error occurred.

Example

```
/* Quick C setup commands called before here */
/* Setup the 7001 to do 3 sets of 5 scans. Each scan starts */
/* immediately at 1 hour intervals. */
Arm7001( 3.9, "IMM", 5.0, MINIMUM, "TIMER");
Timers7001( 3600.0, 1.5);
```

8.8 char *Arm2001Q(char *response);
char *Arm7001Q(char *response);

Description

Queries the Model 2001/7001 for its Arm Layer 1 (Arm Layer) and Arm Layer 2 (Scan LAyer) settings.

Returns

Pointer to response with response =

"Error!" if an IEEE-488 timeout error occurred, or
Query 1: Arm Layer 1 Count (1 to 99999, 9.9e+37)
Query 2: Arm Layer 1 Source (see Arm2001/7001, short form)
Query 3: Arm Layer 2 Count (1 to 99999, 9.9e+37)
Query 4: Arm Layer 2 Source (see Arm2001/7001, short form)
Query 5: Arm Layer 2 Delay (see Arm2001/7001)

Use ParseQuery to separate return string into components.

Example

```
#include <stdlib.h>
double Count1, Count2, Delay2;
char temp[20], Source1[20], Source2[20];
/* Quick C setup commands called before here */
Arm7001Q( response);                          /* Read 7001's Arm Layers 1&2 Setup */
Count1 = atof(ParseQuery( temp, response, 1)); /* Get Arm Layer 1 Count */
ParseQuery( Source1, response, 2);            /* Get Arm Layer 1 Source */
Count2 = atof(ParseQuery( temp, response, 3)); /* Get Arm Layer 2 Count */
Delay2 = atof(ParseQuery( temp, response, 4)); /* Get Arm Layer 2 Delay */
ParseQuery( Source2, response, 5);            /* Get Arm Layer 2 Source */
```

8.9 int Trig2001(double Count1, double Delay1, char *Source1); int Trig7001(double Count1, double Delay1, char *Source1);

Description

Sets up the Model 2001/7001's Trigger Sequence (Trigger Layer).

Parameters

Count1

1 to 99999, 9.9e37
MAXIMUM, MINIMUM, DEFAULT or INF

Delay1

0 to 999999.999 seconds
MAXIMUM, MINIMUM, or DEFAULT

Source1

HOLD	Hold
MANual	Manual
IMMediate	Immediate
BUS	IEEE-488 Bus (GET or *TRG)
TIMer	Timer
TLINK	Trigger Link
EXTernal	External

Returns

0 if no errors.
-1 if a parameter is out of range.
-2 if an IEEE-488 timeout error occurred.

Example

```
/* Quick C setup commands called before here */
/* Setup the 7001 to scan 40 channels with no delay at */
/* 1.5 second intervals. */
Trig7001( 40.0, MINIMUM, "TIM");
Timers7001( 3600.0, 1.5);
```

8.10 char *Trig2001Q(char *response); char *Trig7001Q(char *response);

Description

Queries the Model 2001/7001 for its Trigger Sequence (Trigger Layer) settings.

Returns

Pointer to response with response =

"Error!" if an IEEE-488 timeout error occurred, or
Query 1: Trigger Sequence 1 Count (1 to 99999, 9.9e+37)
Query 2: Trigger Sequence 1 Source (see Trig2001/7001)
Query 3: Trigger Sequence 1 Delay (see Trig2001/7001)

Use ParseQuery to separate return string into components.

Example

```
#include <stdlib.h>
char temp[20], Source1[20];
double Count1, Delay1;
/* Quick C setup commands called before here */
Trig7001Q( response);                          /* Read 7001's Trigger Setup */
Count1 = atof(ParseQuery( temp, response, 1)); /* Get Trigger Count */
Delay1 = atof(ParseQuery( temp, response, 2)); /* Get Trigger Delay */
ParseQuery( Source1, response, 3);             /* Get Trigger Source */
```

8.11 int Timers2001(double ArmTimer2, double TrigTimer1); int Timers7001(double ArmTimer2, double TrigTimer1);

Description

Sets the Model 2001/7001's Trigger Model timers in Arm Layer 2 (Scan Layer) and Trigger Sequence 1 (Trigger Layer).

Parameters

ArmTimer2, TrigTimer1

.001 to 999999.999
MAXIMUM, MINIMUM, or DEFAULT

Returns

0 if no errors.
-1 if a parameter is out of range.
-2 if an IEEE-488 timeout error occurred.

Example

```
/* Quick C setup commands called before here */
/* Scan at 1 hour intervals, trigger at 1.5 sec intervals. */
Timers7001( 3600.0, 1.5);
```

8.12 char *Timers2001Q(char *response);
char *Timers7001Q(char *response);

Description

Queries the Model 2001/7001 for its Arm Layer 2 (Scan Layer) and Trigger Sequence 1 (Trigger Layer) timer settings.

Returns

Pointer to response with response =

"Error!" if an IEEE-488 timeout error occurred, or
Query 1: Arm Layer 2 Timer (see Timers2001/7001)
Query 2: Trigger Sequence Timer (see Timers2001/7001)

Use ParseQuery to separate return string into components.

Example

```
#include <stdlib.h>
char temp[20];
double ArmTimer, TrigTimer;
/* Quick C setup commands called before here */
Timers7001Q( response);                          /* Read Arm Layer 1 and Trigger Timers */
ArmTimer =atof(ParseQuery(temp,response, 1));    /* Get Arm Layer 1 Timer */
TrigTimer=atof(ParseQuery(temp,response, 2));    /* Get Trigger Timer */
```

8.13 int ArmTcon2001(char *Dir1, int ILine1, int Oline1, char *Dir2, int ILine2, int Oline2);
int ArmTcon7001(char *Dir1, int ILine1, int Oline1, char *Dir2, int ILine2, int Oline2);

Description

Sets the Model 2001/7001's Arm Layer 1 (Arm Layer) and Arm Layer 2 (Scan Layer) Trigger configurations. Note that OLine can not be the same as ILine. If they are, the Oline will be made 1 line number (wrapping around to 1 if necessary) higher then the Iline.

Parameters

Dir1 and Dir2

ACCeptor Disable Source Bypass
SOURce Enable Source Bypass

ILine1 and Iline2

1 to 6 Trigger Link input line number

OLine1 and Oline2

1 to 6 Trigger Link output line number

Returns

0 if no errors.
-1 if a parameter is out of range.
-2 if an IEEE-488 timeout error occurred.

Example

```
/* Quick C setup commands called before here */
/* Don't bypass Arm Layer 1 source and use Trigger Link Lines */
/* 1 and 2 as input and output. Bypass Arm Layer 2 source */
/* and use Trigger Link lines 3 and 4 as I/O. */
ArmTcon7001( "ACC", 1, 2, "SOURCE", 3, 4);
```

8.14 char *ArmTcon2001Q(char *response); char *ArmTcon7001Q(char *response);

Description

Queries the Model 2001/7001 for its Arm Layer 1 (Arm Layer) and Arm Layer 2 (Scan Layer) trigger configuration settings.

Returns

Pointer to response with response =

"Error!" if an IEEE-488 timeout error occurred, or
Query 1: Arm Layer 1 Direction (ACC or SOUR)
Query 2: Arm Layer 1 Input Line (1-6)
Query 3: Arm Layer 1 Output Line (1-6)
Query 4: Arm Layer 2 Direction (ACC or SOUR)
Query 5: Arm Layer 2 Input Line (1-6)
Query 6: Arm Layer 2 Output Line (1-6)

Use ParseQuery to separate return string into components.

Example

```
#include <stdlib.h>
char temp[20], Dir1[20], Dir2[20];
int In1, In2, Out1, Out2;
/* Quick C setup commands called before here */
ArmTcon7001Q( response);                    /*Get Arm Layers 1&2 Trigger Configuration*/
ParseQuery( Dir1, response, 1);             /* Get Arm Layer 1 Direction */
In1 = atoi(ParseQuery( temp, response, 2)); /* Get Arm Layer 1 Input Line */
Out1 = atoi(ParseQuery( temp, response, 3)); /* Get Arm Layer 1 Output Line*/
ParseQuery( Dir2, response, 4);             /* Get Arm Layer 2 Direction */
```

```
In2 = atoi(ParseQuery( temp, response, 5));        /* Get Arm Layer 2 Input Line */
Out2 = atoi(ParseQuery( temp, response, 6));       /* Get Arm Layer 2 Output Line*/
```

8.15 int TrigTcon2001(char *Dir1, char *Synch1, int ILine1, int Oline1); int TrigTcon7001(char *Dir1, char *Synch1, int ILine1, int Oline1);

Description

Sets the Model 2001/7001's Trigger Sequence 1 and Trigger Sequence 2 trigger configurations. Note that OLine1 can not be the same as ILine1 if Synch1="ASYN". If they are, the Oline1 will be made 1 line number (wrapping around to 1 if necessary) higher then the Iline1.

Parameters

Dir1

ACCeptor	Disable Source Bypass
SOURce	Enable Source Bypass

Synch1

ASYNchronous	Asynchronous Trigger Link
SSYNchronous	Semi-Synchronous Link

ILine1

1-6	Trigger Link input line number (I/O if SSYN)

OLine1

1-6	Trigger Link output line number (ignored if SSYN)

Returns

0 if no errors.
-1 if a parameter is out of range.
-2 if an IEEE-488 timeout error occurred.

Example

```
/* Quick C setup commands called before here */
/* Don't bypass Trigger Sequence 1 source, use the Semi- */
/* Synchronous Trigger Link, and use Trigger Link Line 5 as */
/* both input and output. */
TrigTcon7001( "ACC", "SSYN", 5, 6);
```

8.16 char *TrigTcon2001Q(char *response);
char *TrigTcon7001Q(char *response);

Description

Queries the Model 2001/7001 for its Trigger Sequence 1 (Trigger Layer) trigger configuration settings.

Returns

Pointer to response with response =

"Error!" if an IEEE-488 timeout error occurred, or
Query 1: Trigger Sequence 1 Direction (ACC or SOUR)
Query 2: Trigger Sequence 1 Trigger Link Mode (ASYN or SSYN)
Query 3: Trigger Sequence 1 Input Line (1-6)
Query 4: Trigger Sequence 1 Output Line (1-6, 0 if SSYN)

Use ParseQuery to separate return string into components.

Example

```
#include <stdlib.h>
char temp[20], Dir[20];
int In1, Out1;
/* Quick C setup commands called before here */
TrigTcon7001Q( response);                    /* Get Trigger Layers 1 Trigger Configuration */
ParseQuery( Dir1, response, 1);              /* Get Trigger Layer 1 Direction */
In1=atoi(ParseQuery( temp, response, 2));    /* Get Trigger Layer 1 Input Line */
Out1=atoi(ParseQuery( temp, response, 3));   /*Get Trigger Layer 1 Output Line*/
```

Section 9
Model 2001 Calculate Commands

These routines control the Model 2001's Calculate Subsystem capabilities, which include limit testing, mX+B, percent, and data buffer statistics. See Model 2001 Minimum and Maximum Calculate Constants in Appendix B for CONSTants for use with the Model 2001 Calculate Commands.

9.1 int Set2001Calc1MXB(double MMFactor, double MBFactor);

Description

Sets the Model 2001's CALCulate1 Subsystem's Mx+B coefficients and enables the Mx+B mode of operation.

Parameters

MMFactor

-9.999999999e20 to +9.999999999e20

MBFactor

-9.999999999e30 to +9.999999999e30

Returns

0 if no errors
-1 if a parameter is out of range.
-2 if an IEEE-488 timeout error occurred.

Example

```
/* Quick C setup commands called before here */
/* Set and use mX+b with slope of 10 and intercept of 5 */
Set2001Calc1MXB( 10.0, 5.0);
```

9.2 int Set2001Calc1PERC(double Percent);

Description

Sets the Model 2001's CALCulate1 Subsystem's Percentage target and enables the percentage mode of operation.

Parameters

Percent

-9.999999999e35 to +9.999999999e35

Returns

0 if no errors.
-1 if a parameter is out of range.
-2 if an IEEE-488 timeout error occurred.

Example

```
/* Quick C setup commands called before here */
/* Set 100 of full scale at 15.24 */
Set2001Calc1PERC( 15.24);
```

9.3 int Calc1_2001(int State);

Description

Sets the state of the Model 2001's CALCulate1 Subsystem.

Parameters

State

0, TOFF = Turn CALC1 Off
+1, TON = Turn CALC1 On

Returns

0 if no errors.
-1 if a parameter is out of range.
-2 if an IEEE-488 timeout error occurred.

Example

```
/* Quick C setup commands called before here */
Calc1_2001( TOFF);                    /* Turn off CALC1 System */
```

9.4 char *Set2001Calc1Q(char *response);

Description

Queries the Model 2001 for its CALCulate1 Subsystem's State, MMFactor, MBFactor, and Percent settings.

Returns

Pointer to response with response =

"Error!" if an IEEE-488 timeout error occurred, or
Query 1: CALCulate1's State (1 or 0)
Query 2: CALCulate1's MMFactor (see Set2001Calc1MXB)
Query 3: CALCulate1's MBFactor (see Set2001Calc1MXB)
Query 4: CALCulate1's Percent (see Set2001Calc1PERC)

Use ParseQuery to separate return string into components.

Example

```
#include <stdlib.h>
char Calc1State[20], temp[20];
double MMFactor, MBFactor, Percent;
/* Quick C setup commands called before here */
Set2001Calc1Q( response);                          /* Get 2001 CALC1 settings */
ParseQuery( Calc1State, response, 1);              /* Get CALC1's State */
MMFactor = atof(ParseQuery( temp, response, 2));   /* Get slope of mX+b */
MBFactor = atof(ParseQuery( temp, response, 3));   /* Get intercept of mX+b */
Percent = atof(ParseQuery( temp, response, 4));    /* Get 100 value */
```

9.5 char *Calc1_2001Q(char *response);

Description

Queries the Model 2001 for the result of the latest CALCulate1 calculation.

Returns

Pointer to response with response =

"Error!" if an IEEE-488 timeout error occurred, or 2001's present CALCulate1 calculation.

Example

```
/* Quick C setup commands called before here */
Set2001Calc1MXB( 10.0, 5.0);
/* Reading from SENSe:DATA? is 10.4 */
Calc1_2001Q( response); /* response = "109" */
```

9.6 int Set2001Calc2(char *Format);

Description

Sets up the Model 2001 Data Buffer's Format and activates the CALCulate2 subsystem.

Parameters

Format

NONE
MEAN
SDEViation (Standard Deviation)
MAXimum
MINimum
PKPK (Peak to Peak)

Returns

0 if no errors.
-1 if a parameter is out of range.
-2 if an IEEE-488 timeout error occurred.

Example

```
/* Quick C setup commands called before here */
Set2001Calc2( “MAXIMUM”);              /* calculate maximum of data buffer */
Calc2_2001 (TON);
/* Data buffer contains 1.122, 10.211, 10.2222 */
Calc2_2001Q(response);
/* response = “10.2222” */
```

9.7 int Calc2_2001(int State);

Description

Sets the state of the Model 2001’s CALCulate2 Subsystem.

Parameters

State

0, TOFF = Turn CALC2 Off
+1, TON = Turn CALC2 On

Returns

0 if no errors.
-1 if a parameter is out of range.
-2 if an IEEE-488 timeout error occurred.

Example

```
/* Quick C setup commands called before here */
Calc2_2001( 0);                    /* Turn off CALC2 System */
```

9.8 char *Set2001Calc2Q(char *response);

Description

Queries the Model 2001 for its CALCulate2 subsystem's Format and State.

Returns

Pointer to response with response =

"Error!" if an IEEE-488 timeout error occurred, or
Query 1: CALCulate 2 Format (see Set2001Calc2 above)
Query 2: CALCulate 2 State (0 or 1)

Use ParseQuery to separate return string into components.

Example

```
char Format[20], Calc2State[20];
/* Quick C setup commands called before here */
Set2001Calc2Q( response);                    /* Get 2001 CALC2 settings */
ParseQuery( Format, response, 1);            /* Get CALC2's Format */
ParseQuery( Calc2State, response, 2);        /* Get CALC2's State */
```

9.9 char *Calc2_2001Q(char *response);

Description

Queries the Model 2001 for the result of the present CALCulate2 subsystem's calculation.

Returns

Pointer to response with response =

"Error!" if an IEEE-488 timeout error occurred, or the result of the present CALC2 calculation.

Example

```
/* Quick C setup commands called before here */
Calc2_2001Q( response); /* Get 2001 CALC2's last result */
```

9.10 int Set2001Calc3(double Upper1, double Lower1, double Upper2, double Lower2);

Description

Sets the Model 2001's CALCulate3 subsystems Limit Testing HI/LO Limits.

Parameters

Upper1, Upper2, Lower1, Lower2

-9.999999999e35 to +9.999999999e35
MAXIMUM, MINIMUM, or DEFAULT

Returns

0 if no errors.
-1 if a parameter is out of range.
-2 if an IEEE-488 timeout error occurred.

Example

```
/* Quick C setup commands called before here */
/* Setup Limit Testing Ranges */
/* Limit 1 Range: -100.56 to -10.5 or 10.5 to 100.56 */
/* Limit 2 Range: <-100.56 or >100.56 */
/* Pass Range: -10.5 to 10.5 */
Set2001Calc3( 10.5, -10.5, 100.56, -100.56);
```

9.11 char *Set2001Calc3Q(char *response);

Description

Queries the Model 2001 for its CALCulate3 subsystems Limit Testing Hi/Lo Limits.

Returns

Pointer to response with response =

"Error!" if an IEEE-488 timeout error occurred, or
Query 1: limit testing Upper Limit 1
Query 2: limit testing Lower Limit 1
Query 3: limit testing Upper Limit 2
Query 4: limit testing Lower Limit 2

Use ParseQuery to separate return string into components.

Example

```
#include <stdlib.h>
double Upper1, Lower1, Upper2, Lower2;
char temp[20];
/* Quick C setup commands called before here */
Set2001Calc3Q( response);                          /* Get 2001 CALC3 settings */
Upper1 = atof(ParseQuery( temp, response, 1));     /* Get Upper Limit 1 */
Lower1 = atof(ParseQuery( temp, response, 2));     /* Get Lower Limit 1 */
Upper2 = atof(ParseQuery( temp, response, 3));     /* Get Upper Limit 2 */
Lower2 = atof(ParseQuery( temp, response, 4));     /* Get Lower Limit 2 */
```

9.12 int Calc3_2001(int State);

Description

Sets the state of the Model 2001's CALCulate3 Subsystem.

Parameters

State

0, TOFF = Turn CALC3 Off
+1, TON = Turn CALC3 On

Returns

0 if no errors.
-1 if a parameter is out of range.
-2 if an IEEE-488 timeout error occurred.

Example

```
/* Quick C setup commands called before here */
Calc3_2001( TON); /* Turn on CALC3 System */
```

9.13 int Set2001Calc3Dig(int Du1, int Dl1, int Du2, int Dl2);

Description

Sets the Model 2001's CALCulate3's Limit Testing Hi/Lo Digital Output values.

Parameters

Du1 Value (0-15) to place on digital output to indicate Upper 1 Limit Reached
Dl1 Value (0-15) to place on digital output to indicate Lower 1 Limit Reached
Du2 Value (0-15) to place on digital output to indicate Upper 2 Limit Reached
Dl2 Value (0-15) to place on digital output to indicate Lower 2 Limit Reached

Returns

0 if no errors.
-1 if a parameter is out of range.
-2 if an IEEE-488 timeout error occurred.

Example

```
/* Quick C setup commands called before here */
/* Set 1st bit of digital output if Upper Limit 1 Reached */
/* Set 2nd bit of digital output if Lower Limit 1 Reached */
/* Set 3rd bit of digital output if Upper Limit 2 Reached */
```

```
/* Set 4th bit of digital output if Lower Limit 2 Reached */
Set2001Calc3Dig( 1, 2, 4, 8);
```

9.14 char *Set2001Calc3DigQ(char *response);

Description

Queries the Model 2001 for its CALCulate3 subsystem's Limit Testing Hi/Lo Digital Output Values.

Returns

Pointer to response with response =

"Error!" if an IEEE-488 timeout error occurred, or
Query 1: digital output Upper Limit 1 value
Query 2: digital output Lower Limit 1 value
Query 3: digital output Upper Limit 2 value
Query 4: digital output Lower Limit 2 value

Use ParseQuery to separate return string into components.

Example

```
#include <stdlib.h>
char temp[20];
int Du1, Dl1, Du2, Dl2;
/* Quick C setup commands called before here */
Set2001Calc3DigQ( response);                    /* Get 2001 CALC3 Digital I/O settings */
Du1= atoi(ParseQuery( temp, response, 1));      /* Get UpLim1 DigI/O Value */
Dl1= atoi(ParseQuery( temp, response, 2));      /* Get LoLim1 DigI/O Value */
Du2= atoi(ParseQuery( temp, response, 3));      /* Get UpLim2 DigI/O Value */
Dl2= atoi(ParseQuery( temp, response, 4));      /* Get LoLim2 DigI/O Value */
```

9.15 char *Calc3_2001Q(char *response);

Description

Queries the Model 2001 for the result of the present CALCulate3 subsystem's calculation.

Returns

Pointer to response with response =

"Error!" if an IEEE-488 timeout error occurred, or
Query 1: Upper/Lower Limit Range 1 result (0=PASS or 1=FAIL)
Query 2: Upper/Lower Limit Range 2 result (0=PASS or 1=FAIL)

Use ParseQuery to separate return string into components.

Example

```
/* Quick C setup commands called before here */
char Test1[20], Test2[20];
Calc3_2001Q( response);                 /* Get 2001 CALC3 Limit Test Results */
ParseQuery( Test1, response, 1);        /* Get Upper/Lower 1 Test Result */
ParseQuery( Test2, response, 2);        /* Get Upper/Lower 2 Test Result */
```

Appendix A
Model 2001 / 7001 Global Variables

The following is a list of all the global variables used by the 2001/7001 Support Software for Microsoft Quick C 2.5:

KI2001 Global variable that contains the Model 2001's IEEE address.

KI7001 Global variable that contains the Model 7001's IEEE address.

brd0 The location of board "GPIB0" for National Instruments.

Nat2001Addr Integer value of the actual Model 2001 IEEE Address for use with National Instruments board commands.

Nat7001Addr Integer value of the actual Model 7001 IEEE Address for use with National Instruments board commands.

ieee Integer value of the file handle used for IOTech Driver 488 input/output operations.

IeeeInterface contains a number from 1 to 4 indicating the IEEE interface as defined by the constants below:

IEEECEC (1)	Capital Equipment Corp.
IEEEIOTECH (2)	IOTech Driver 488
IEEENATIONALOLD (3)	National Instruments NI-488 (C.10 or C.11)
IEEENATIONALNEW (4)	NI-488 (C.12 or greater) or NI-488.2

NOTE

These variables also support the Model 7002 Switch System.

Appendix B
Model 2001 / 7001 Constants

The following 2001/7001 Support Software #defines are defined to make using the support software easier. The use of these constants are defined below and in the appropriate group of routines that use them.

NOTE

These constants also support the Model 7002 Switch System.

B.1 Function Constants

DCV (1)	DC Voltage	ACV (2)	AC Voltage
DCA (3)	DC Current	ACA (4)	AC Current
R2 (5)	2-wire Resistance	R4 (6)	4-wire Resistance
FREQ (7)	Frequency	TEMP (8)	Temperature

B.2 Status Model Constants

For use with Stat2001, Stat2001Q, Stat7001, Stat7001Q:

OPERATION (1)	Operation Event
TRIGGER1 (2)	Trigger Event
ARM1 (3)	Arm Event
SEQUENCE (4)	Sequence Event
QUESTIONABLE (5)	Questionable Event
MEASUREMENT (6)	Measurement Event

Constants for use with *STB?, *SRE, *SRE?, and serial poll:

MSB (1)	Measurement Summary Bit (Model 2001 Only)
EAV (4)	Error Available
QSB (8)	Questionable Summary Bit(in Model 7001 only for SCPI)
MAV (16)	Message Available
ESB (32)	Event Summary Bit
RQS1 (64)	Request for Service (Serial Poll)
MSS (64)	Master Summary Status (Status Byte)
OSB (128)	Operation Summary Bit

Constants for use with *ESR?, *ESE, and *ESE?:

OPC (1)	Operation Complete
RQC (2)	Request Control (not used in Model 2001/7001)

QYE (4)	Query Error
DDE (8)	Device Specific Error
EXE (16)	Execution Error
CME (32)	Command Error
URQ (64)	User Request
PON (128)	Power On

Constants for use with the Operation Event Condition Register, Positive Transition Filter, Negative Transition Filter, Status Register, and Status Enable Register:

CAL (1)	Calibrating (Model 2001 only)
SET (2)	Settling
TRIG (32)	Waiting for Trigger
ARM (64)	Waiting for Arm
CALC (512)	Calculating (Model 2001 only)
SCAN (1024)	Scanning (Model 7001 only)
IDLE (1024)	in IDLE layer (Model 2001 only)

Constants for use with the Trigger Event Condition Register, Positive Transition Filter, Negative Transition Filter, Status Register, and Status Enable Register:

SEQ1 (2)	Model 2001/7001 is in the Trigger Layer of Sequence 1.

Constants for use with the Arm Event Condition Register, Positive Transition Filter, Negative Transition Filter, Status Register, and Status Enable Register:

SEQ1 (2)	Model 2001/7001 is in the ARM Layer of Sequence 1.

Constants for use with the Sequence Event Condition Register, Positive Transition Filter, Negative Transition Filter, Status Register, and Status Enable Register:

LAY1 (2)	Model 2001 or 7001 is in the Arm Layer 1 of Sequence 1.
LAY2 (4)	Model 2001 or 7001 is in the Arm Layer 2 of Sequence 1.

Constants for use with the 2001 Questionable Event Condition Register, Positive Transition Filter, Negative Transition Filter, Status Register, and Status Enable Register:

QTEMP (16)	Questionable Calibration Process
QCAL (128)	Questionable Calibration Process
WARN (16384)	Command Warning

Constants for use with the Model 2001 Measurement Event Condition Register, Positive Transition Filter, Negative Transition Filter, Status Register, and Status Enable Register:

ROF (1)	Reading OverFlow
LL1 (2)	Low Limit 1

HL1 (4)	High Limit 1
LL2 (8)	Low Limit 2
HL2 (16)	High Limit 2
RAV (32)	Reading AVailable
BAV (128)	Buffer AVailable
BHF (256)	Buffer Half Full
BFL (512)	Buffer FuLl
BPT (2048)	Buffer PreTrigger event occurred

B.3 Automatic Constants

Use these constants with the 2001 Auto measurement functions, like AutoDCV, or any Model 2001/7001 function that requires an ON or OFF state.

NO (-1)	Don't set automatic parameter.
TOFF (+0)	Turn off automatic parameter.
TON (+1)	Turn on automatic parameter.
ONCE (+2)	Set auto parameter for the next measurement.

B.4 Model 2001 Minimum and Maximum Sense Constants

Constant	Value	Constant	Value
MINACA	-2.1	MAXACA	2.1
MINDCA	-2.1	MAXDCA	2.1
MINACV	-775.0	MAXACV	775.0
MINDCV	-1100.0	MAXDCV	1100.0
MINR2	0.0	MAXR2	1050000000.0
INR4	0.0	MAXR4	21000.0
MINFREQ	0.0	MAXFREQ	1500000.0
MINTEMPF	-328.0	MAXTEMPF	3310.0
MINTEMPC	-200.0	MAXTEMPC	1821.0
MINTEMPK	73.0	MAXTEMPK	2094.0
MINSIMF	32.0	MAXSIMF	122.0
MINSIMC	0.0	MAXSIMC	50.0
MINSIMK	273.0	MAXSIMK	323.0
MINTCO	-0.09999	MAXTCO	0.09999
MINALPHA	0.0	MAXAPLHA	0.01
MINBETA	0.0	MAXBETA	1.0
MINDELTA	0.0	MAXDELTA	5.0
MINRZERO	0.0	MAXRZERO	1000.0
MINDIG	3.5	MAXDIG	8.4999
MINNPLC	0.01	MAXNPLC	10.0
MINAVGCNT	1.0	MAXAVGCNT	100.0
MINNTOL	1.0	MAXNTOL	100.0
MINPWIN	0.1	MAXPWIN	9.9

MAXTEMPDIG	7.4999	MAXFREQDIG	5.4999
MAXFREQV	1000.0	MINFREQV	-1000.0
MAXFREQI	1.0	MINFREQI	-1.0
MAXFREQPERC	0.6	MINRANGE	0.0

B.5 Model 2001 Minimum and Maximum Calculate Constants

Constant	Value	Constant	Value
MINCALC	-9.999999E+35	MAXCALC	9.999999E+35
MINPERC	-9.999999E+35	MAXPERC	9.999999E+35
MINMMF	-9.999999E+20	MAXMMF	9.999999E+20
MINMBF	-9.999999E+30	MAXMBF	9.999999E+30
MINDIGITAL	0	MAXDIGITAL	15

B.6 Model 2001/7001 Scanning Minimum and Maximum Constants

Constant	Value	Constant	Value
MINCOUNT	1.0	MAXCOUNT	99999.0
MINDELAY	0.0	MAXDELAY	999999.999
MINLINE	1	MAXLINE	6
MINCHANNEL	1	MAXCHANNEL	10
MINTIMER	0.001	MAXTIMER	999999.999

B.7 Read2001Buffer Constants

Constant	Value
BUFRDGS	1
BUFTIMESTAMP	2
BUFCHANNELS	3
BUFSTATUS	4
BUFUNITS	5

B.8 Examples

To set 2001 to generate a Service Request on EAV or MAV use the following:

```
SRE2001( EAV | MAV);
```

To check for MAV being set in the Status Byte:

```
int poll;
poll = atoi(Q2001( response, "*STB?"));
if( poll & MAV) puts("MAV is set.");
```

Appendix C
Model 2001 / 7001 Quick C 2.5 Support Software File Names

C.1 2001/7001 IEEE Interface Independent Files

20017001.H — 2001/7001 Header File #define's & routine declares

27001 .C — 2001/7001 General Routines:

KI_Disp, KI_NoDisp, Check2001Val, IOTECH, ParseQuery, strins, str_ltrim, str_rtrim, dtoa, ftoa, KI_number

2001 .C — 2001 Function & Buffer routines:

Get2001Func, Get2001FuncHeader, Get2001Rdg, Get2001SaveRecallSize, Get2001Units, Get2RES2001, Get4RES2001, GetACA2001, GetACV2001, GetDCA2001, GetDCV2001, GetFREQ2001, GetTEMP2001, Hit2001Key, Hit2001Key1, BufferSize2001, Take2001BufferReadings

AUTO2001.C — 2001 Automatic Configuration routines:

Auto2001, Auto2001Q, Auto2001TQ

CALC2001.C — 2001 CALCulate Subsytem routines:

Calc1_2001, Calc1_2001Q, Calc2_2001, Calc2_2001Q, Calc3_2001, Calc3_2001Q, Set2001Calc1MXB, Set2001Calc1PERC, Set2001Calc1Q, Set2001Calc2, Set2001Calc2Q, Set2001Calc3, Set2001Calc3Dig, Set2001Calc3DigQ, Set2001Calc3Q

SETS2001.C — 2001 configuration routines:

Set2001, Set2001Q, Set2001ACA, Set2001DCA, Set2001F, Set2001FQ, Set2001R2, Set2001Function, Set2001RTD, Set2001RTDQ, Set2001T, Set2001TC, Set2001TCQ, Set2001TQ

SCPISTAT.C — 2001/7001 Status Model routines:

Clear2001, KI_ESE, KI_SRE, KI_Stat, KI_StatQ

SCPITRIG.C — 2001/7001 Trigger Model routines:

Close2001, KI_Arm, KI_ArmQ, KI_ArmTcon, KI_ArmTconQ, KI_Close, KI_CloseQ, KI_Timers, KI_TimersQ, KI_Trig, KI_TrigQ, KI_TrigTcon, KI_TrigTconQ, Open2001, KI_Open, KI_Scan

GENERALS.C — 2001/7001 Single Precision General Routines:

_DataViewS, _XYGraphS

GENERALD.C — 2001/7001 Double Precision General Routines:

_DataViewD, _XYGraphD

C.2 Capital Equipment Corp. (CEC) IEEE-488 Interface Files

CEC .LIB — Quick C 2.5 CEC 2001/7001 library

CEC .C — Quick C 2.5 CEC 2001/7001 source

Gpiberror, KI_OPC, KI_Poll, KI_Q, KI_Send, KI_SendGET, KI_SendSDC, KI_WaitRQS, SetupIEEE, _Read2001Buffer

CEC .H — Modified CEC Quick C 2.5 include file

MAKECEC .BAT — Makes all *.OBJ and *.LIB files for Capital Equipment Corp.

C.3 IOTech Driver 488 Interface Files

IOTECH .LIB — Quick C 2.5 IOTech 2001/7001 driver library

IOTECH .C —- Quick C 2.5 IOTech 2001/7001 source

Gpiberror, KI_OPC, KI_Poll, KI_Q, KI_Send, KI_SendGET, KI_SendSDC, KI_WaitRQS, SetupIEEE, _Read2001Buffer

IOTECHIO.H — IOTech Input/Output Functions header file

IOTECHIO.C — IOTech Input/Output Functions source code

IOTERROR.H — IOTech Error Handler header file

IOTERROR.OBJ — IOTech Error Handler Object File

MAKEIOT .BAT — Makes all *.OBJ and *.LIB files for IOTech Driver 488.

C.4 National Instruments NI-488 rev C.11 Files

NAT488_1.C — Quick C 2.5 old NI 2001/7001 source

Gpiberror, KI_OPC, KI_Poll, KI_Q, KI_Send, KI_SendGET, KI_SendSDC, KI_WaitRQS, SetupIEEE, _Read2001Buffer

NAT488_1.LIB — Quick C 2.5 old NI 2001/7001 driver library

NI4881. H — Old NI header file

NI4881. OBJ — Old NI object file

MAKENAT1.BAT — Makes all *.OBJ and *.LIB files for National Instruments NI- 488 (rev C.11 and older)

C.5 National Instruments NI-488 rev C.12(and newer) and NI-488.2 Files

NAT488_2.C — Quick C 2.5 new NI 2001/7001 source

Gpiberror, KI_OPC, KI_Poll, KI_Q, KI_Send, KI_SendGET, KI_SendSDC, KI_WaitRQS, SetupIEEE, _Read2001Buffer

NAT488_2.LIB — Quick C 2.5 new NI 2001/7001 driver library

NI4882QB.H — New NI header file

NI4882QB.OBJ — New NI object file.

MAKENAT2.BAT — Makes all *.OBJ and *.LIB files for NI-488.2 and NI-488 (rev C.12 and newer)

Turbo Pascal 6.0

Table of Contents

Section 1
Using the Library Routines

1.1 Installation

1.1.1 National Instruments GPIB.COM

You must have at least Rev C.4 of the National Instruments NI-488 Software or at least Rev 1.0 of the National Instruments NI-488.2 Software to use the Model 2001/7001 Support Software. Some older revisions may work, but they are not guaranteed by Keithley to do so. Use the following settings to set up your National Instruments Card (AT-GPIB card listed here, use all that are applicable) when you are configuring GPIB.COM with IBCONF.EXE.

Primary GPIB Address..... 0
Secondary GPIB Address ..... NONE
Timeout setting T10s
EOS byte..... 0AH
Terminate Read on EOS..... no
Set EOI with EOS on Write..... yes
Type of compare on EOS 7-bit
Set EOI w/last byte of Write..... yes
System Controller..... yes
Assert REN when SC ..... yes
Enable Auto Serial Polling ..... no
Timing 350nsec
Enable 488.2 Protocols ..... yes
CIC Protocol ..... no
Disable Device Unaddressing..... no

Also, for the old revs of NI-488 (C.4 to C.11), "KI2001" and "KI7001" must be devices defined in GPIB.COM, using IBCONF.EXE.

1.1.2 Capital Equipment Corp.

You must have at least Rev 2.14 of the Capitial Equipment Corp. Software to use the Model 2001/7001 Support Software. All older revisions will not work without the removal of SETATNMODE() from SetupIEEE() in CECQBX.BAS for in CEC.BAS for Turbo Pascal 6.0.

1.1.3 IOTech Driver 488

You must have at least Rev 2.6 of the IOTech Driver 488 Software to use the Model 2001/7001 Support Software. Some older revisions may work, but they are not guaranteed to do so.

1.1.4 Turbo Pascal 6.0

Type A:INSTALL or B:INSTALL to install the Turbo Pascal 6.0 Model 2001/7001 support software. The installation program will prompt you for certain information that it needs to copy files and build the make files MAKECEC.BAT, MAKEIOT.BAT, MAKENAT1.BAT, and MAKENAT2.BAT. These files will build the *.OBJ, *.LIB, and *.QLB needed to use the routines.

Installation Notes:

1. BC.EXE, LINK.EXE, and LIB.EXE must be installed in the Turbo Pascal 6.0 Compiler Directory.
2. BQLB45.LIB and IEEEQB.LIB (if installing CEC) must reside in the Library Files Directory.
3. The Model 2001/7001 Support Software directory must exist and you will be prompted to build it if does not.

1.2 General Instructions

1.2.1 Turbo Pascal 6.0

To use the Model 2001/7001 Support Software the following setup must be used:

```
uses cec (or iotech, nat488_1, or nat488_2);
{N+}
{E+}

var
{ variable declarations }
SetupErr: Boolean;
begin
SetupErr:=SetupIEEE(0,0);              { Setup IEEE Interface }
SetupErr:=SetupIEEE(2001,16);          { Setup 2001 at IEEE Address 16 }
SetupErr:=SetupIEEE(7001,7);           { Setup 7001 at IEEE Address 7 }
{
User Program Follows
}
end;
```

1.2.2 Documentation Notes

1. Query forms of a function have the format of function-nameQ which returns a query string containing the results of the specified Model 2001/7001 parameters. Automatic values are returned as 0 for OFF and 1 for ON.
2. If a function query returns more than one parameter, the responses will be separated by a semicolon (;) within the return string. Use the ParseQuery routine to return the specified response string (parameter number 1, 2, 3, etc.) from the query return string.
3. Any Model 2001/7001 command parameters out of range will not be sent to the Model 2001 leaving the previously set or default values in tact. Also, the global variable, OutOfRange : Integer; will be set to TRUE (-1).

4. Most Model 2001/7001 parameters that are double precision values will accept the constants MAXIMUM, MINIMUM, and DEFAULT. INF will be accepted for Trigger Model Count1# and Count2# parameters.
5. All example programs assume that Turbo Pascal 6.0 setup commands listed above were used before issuing any commands.
6. All string parameters that have their valid parameters listed with mixed case, like IMMediate, accept either the short form (IMM) or the long form (IMMEDIATE) in any combination of case. This is comparable to the short and long form notation used for SCPI commands.
7. See Appendix A for a description of the Global Variables used in the Model 2001/7001 support software.
8. See Appendix B for a description the defined CONSTants used in the Model 2001/7001 support software.
9. See Appendix C for a list of all Turbo Pascal 6.0 file names used by the Model 2001/7001 support software.
10. All Model 7001 routines can be used with the Model 7002.

Section 2
2001 / 7001 IEEE-488 Interface Routines

These functions and subroutines control the Models 2001/7001 with low-level IEEE-488 bus commands specific to each IEEE-488 interface manufacturer. If any IEEE-488 timeout errors occur, an error message will be displayed. Also, the TimeOutError global variable (see Global Variables in Appendix A) will be set TRUE.

NOTE

These functions and subroutines also support the Model 7002 Switch System. Just send Model 7001 commands to a Model 7002.

2.1 function SetupIEEE(Device, Address : Integer) : Boolean;

Description

Sets up the specific IEEE-488 interface to correctly handle data transfers between the Models 2001/7001 and the host PC computer. Also, initializes either the 2001 and 7001 at the specified addresses. CEC controllers will be at IEEE address 21. IOtech controllers are usually at IEEE address 21. National Instruments controllers are usually at IEEE address 0.

Parameters

Device

0	Initialize Interface Only
2001	Change the IEEE address of the Model 2001 Only
7001	Change the IEEE address of the Model 7001 Only

Address

0-30	if Device is 2001 or 7001
ignored	if Device is 0

Returns

TRUE (-1) if a TimeOutError occurred or a parameter is OutOfRange.

Global Variables Affected (see Global Variables in Appendix A)

KI2001, KI7001, TimeOutError, OutOfRange, IeeeInterface
brd0, Nat2001Addr, Nat7001Addr (Nat'l. Instruments only)
IeeeIn, IeeeOut (IOTech Driver 488 only)

Example

Before using any of the Model 2001 routines, the following commands must be issued:

```
uses cec;
var
SetupErr:Boolean;

begin
SetupErr:=SetupIEEE(0,0);              { Setup IEEE Interface }
SetupErr:=SetupIEEE(2001,16);          { Setup 2001 at IEEE Address 16 }
SetupErr:=SetupIEEE(7001,7);           { Setup 7001 at IEEE Address 7 }
.
.
end;
```

2.2 procedure SendSDC2001;
procedure SendSDC7001;

Description

Sends the IEEE bus command SDC (Selected Device Clear) to the Model 2001/7001.

Global Variables Affected (see Global Variables in Appendix A)

TimeOutError

2.3 function Q2001(Cmd : String) : String;
function Q7001(Cmd : String) : String;

Description

Gets a query response from the Model 2001/7001. Cmd must be a valid Model 2001/7001 query or else the instrument will TimeOut. Multiple queries are allowed and responses can be separated using the ParseQuery function. If Cmd:='', then the function will still try to read data from the instrument. This is good for reading large amounts of data from the Model 2001/7001 since the most that can be read at one time is 2048 bytes.

Parameters

Cmd := '' - try to read the 2001/7001
<> '' - send query and try to read 2001/7001

Returns

Query response from the Model 2001/7001 if Cmd was valid 'Error!' if Cmd not valid (TimeOutError occurred)

Global Variables Affected

TimeOutError, Resp

Example

```
var
A : String;
{ TP6 setup commands called before here}
A := Q2001('VOLT:DC:RANGE?');     { return DC Voltage Range }
```

2.4 procedure Send2001(Cmd : String); procedure Send7001(Cmd : String);

Description

Sends IEEE-488.2 and SCPI command strings to the Model 2001/7001.

Parameters

Cmd Valid Model 2001/7001 488.2 or SCPI command or query.

Returns

Nothing

Must check the Model 2001/7001 EAV bit in the serial poll register to see if a command was accepted or look at the front panel of the instrument for an error message.

Global Variables Affected (see Global Variables in Appendix A)

TimeOutError, OutOfRange

Example

```
{ TP6 setup commands called before here }
Send7001('ROUTE:CLOSE (@1!1:1!40)');       {Close channels 1-40 on card 1 of 7001 }
```

2.5 procedure SendGET2001; procedure SendGET7001;

Description

Sends the IEEE bus command GET to the 2001/7001.

Global Variables Affected (see Global Variables in Appendix A)

TimeOutError

Example

```
var
Reading : String;
{ TP6 setup commands called before here }
SendGET2001;                    { Trigger the 2001 to take reading }
Reading := Q2001('DATA?');      { Get 2001 reading }
```

2.6 function Wait2001RQS : Boolean; function Wait7001RQS : Boolean;

Description

Waits for the Model 2001/7001 to generate a Request for Service. The routine serial polls the Model 2001/7001 to verify that the instrument is indeed generating a Request for Service. The wait can be aborted by pressing the Esc key.

Returns

FALSE	if aborted.
TRUE	if a 2001/7001 Request for Service.

Example

```
var
Poll : Byte
{ TP6 setup commands called before here }
Poll := Poll2001;               { Clear and pending SRQ's }
SRE2001(MAV);                   { Set up to SRQ on MAV }
Send2001( 'FETCH?');            { fetch a 2001 reading }
repeat                          { Set up an uninterruptable wait }
until Wait2001RQS;
Reading := Q2001('');           { Get reading }
```

Section 3
2001 Buffer Routines

These routines are used to acquire readings in the Model 2001's data buffer. Up to 30,092 readings can be stored in the Model 2001 with the MEM2 option and the compact format.

All routines will set the Global Variable TimeOutError if an IEEE-488 Timeout error occurs. Functions that have bounded parameters (maximum and/or minimum parameters) will set the Global Variable OutOfRange if a parameter is not within the limits specified (see Global Variables in Appendix A).

3.1 function BufferSize2001 : Integer;

Description

Finds the actual number of data points in the 2001 buffer. This function should be used since TRACe:POINts? may not return the correct number of data points if the data buffer acquisition was aborted.

Returns

The actual number of data points in the 2001 buffer (anywhere from 2 to 30,092). Note that the largest variable on the heap can only be 65,521 bytes in size which limits the number of single precision readings that Turbo Pascal can handle to 16,380 and double precision to 8,190. However, if succesive calls are made to GetMem, this will guarantee a contiguous block of memory to store all of the 2001's data buffer.

Example

```
var
   BufferData, BufferData1 : Pointer;
   NumPoints : Integer;

{ TP6 setup commands called before here }
NumPoints := BufferSize2001;
 { Read Back Readings in single precision }
if( NumPoints > 16380 ) then
begin
   GetMem(BufferData, sizeof(single) * 16380);
   GetMem(BufferData1, sizeof(single) * ( NumPoints - 16380));
end
else
   GetMem(BufferData, sizeof(single) * NumPoints);
end;
NumPoints:=Read2001Buffer(BufferData,sizeof(single),0,BUFRDGS);
```

3.2 function Read2001Buffer(var ArrayName : Pointer; ElementSize, DMA, Fmt : Integer) : Integer;

Description

Retrieves all of the readings (up to 30,092 IEEE754 single or double precision readings) from the 2001's data buffer and stores them into a single or double precision array.

Parameters

ArrayName

Single or double precision pointer to either an array or a block of memory large enough to handle the number of data points returned by BufferSize2001 or Take2001BufferReadings.

ElementSize

Size of an element in ArrayName, either sizeof(single) or sizeof(double).

DMA

0	don't use DMA
1	use DMA (IOtech or National Instruments)
1-7	use DMA Channel configured on CEC IEEE-488 Interface Card

Fmt

1	Return Readings (FULL or COMPact Format)
2	Return TimeStamp (Full Format only)
3	Return Channel (Full Format only)
4	Return Status (Full Format only)
5	Return Units (Full Format only)

Also, see Read2001Buffer Constants in Appendix B.

Returns

The actual number of data points transferred to the array. If 0, then either a TimeOutError occurred, an OutOfRange error occurred, or ArrayName!() or ArrayName() is dimensioned too small.

Example

```
var
   BufferData, BufferData1, Times, Times1 : Pointer;
   NumPoints : Integer;
 { TP6 setup commands called before here }
{ Take 5000 AC Volt Full Format Mode Readings: }
NumPoints := Take2001BufferReadings(ACV,FALSE,5000,FALSE);
{ Read Back Readings and TimeStamp in single precision }
if( NumPoints > 16380 ) then
begin;
   GetMem(BufferData, sizeof(single) * 16380);
```

```
   GetMem(BufferData1, sizeof(single) * ( NumPoints - 16380));
   GetMem(Times, sizeof(single) * 16380);
   GetMem(Times1, sizeof(single) * ( NumPoints - 16380));
end
else
begin
   GetMem(BufferData, sizeof(single) * NumPoints);
   GetMem(Times, sizeof(single) * NumPoints);
end;
NumPoints:=Read2001Buffer(BufferData,sizeof(single),0,BUFRDGS);
NumPoints:=Read2001Buffer(Times,sizeof(single),0,BUFTIMESTAMP);
DataViewS( BufferData, NumPoints);
DataViewS( Times, NumPoints);
```

3.3 function Take2001BufferReadings (Func : Integer; BurstMode : Boolean; NumDataPoints1 : Integer; Compact : Boolean) : Integer;

Description

Sets up and acquires up to 30,092 readings in the Model 2001's data buffer. (Use with Read2001Buffer to bring read data). The data acquisition may be aborted by pressing the Esc key.

Parameters

Func, (also, see Function Constants, in Appendix B):

0 — Current Function
1 — DCV
2 — ACV
3 — DCA
4 — ACA
5 — 2-wire Resistance
6 — 4-wire Resistance
7 — Frequency
8 — Temperature

BurstMode

If TRUE, uses the fast Burst Mode of Reading acquisition available from the 2001. BurstMode is only applicable if Func is 1 to 5.

NumDataPoints1

2 to TRACE:POINTS? MAX (depends on the setting of Compact and the memory option installed in the 2001).

Compact

TRUE	use the COMPACT buffer format (readings only).
FALSE	use the FULL buffer format (reading, time stamp, channel number, status, and units).

NOTE

The COMPACT format allows 5 times as many readings as does FULL. If BurstMode is set, only COMPACT format is valid.

Returns

The actual number of data points read from the Model 2001. If 0, then either a TimeOutError occurred or an OutOfRange error occurred for Func or NumDataPoints < 2.

Example

See Read2001Buffer, paragraph 3.2.

Section 4
General Routines

These routines provide data display, graphing, and data manipulating functions that make the handling of the returned data from the Models 2001 and 7001 easier to handle.

NOTE

These functions and subroutines also support the Model 7002 Switch System. Just send Model 7001 commands to a Model 7002.

All functions that have bounded parameters (maximum and/or minimum parameters) will set the Global Variable OutOfRange if a parameter is not within the limits specified (see Global Variables p.54).

4.1 function ParseQuery(Quer : String; QuerNum : Integer) : String;

Description

Returns the query specified by QuerNum inside Quer. Queries are separated by either commas or semi-colons. Also, leading and trailing spaces are removed.

Parameters

Quer a string returned by any of the 2001/7001 query functions.
QuerNum the number of the query to retrieve

Returns

'Error!' if Quer := '',
'Invalid Query Number.' if QuerNum <:= 0,
QuerNum query in Quer, or
last query if QuerNum too large

Example

```
var
A, B : String;
{ TP6 setup commands called before here }
A := '100; FFF; oii, 4423.223; 4,000';
B := ParseQuery(A, 1); { B := '100' }
B := ParseQuery(A, 2); { B := 'FFF' }
B := ParseQuery(A, 3); { B := 'oii' }
B := ParseQuery(A, 4); { B := '4423.223' }
```

B := ParseQuery(A, 5); { B := '4' }
B := ParseQuery(A, 6); { B := '000' }
B := ParseQuery(A, 7); { B := '000' }
B := ParseQuery(A, 0); { B := 'Invalid Query Number.' }

4.2 function IOTechAddr(Address : Integer) : String;

Description

Returns a two digit number string with a leading 0 for use with IOTech's IEEE Addressable commands.

Parameters

Address 0 to 30

Returns

'00','01',..,'09','10','11',..,'30'

Example

```
{ TP6 setup commands called before here }
WriteLn( IeeeOut, 'ENTER ' + IOTechAddr( KI2001));
```

4.3 procedure XYGraphS(var XArray : Pointer; NumX : Word; var YArray : Pointer; NumY : Word; StartNum, StopNum : Word; XTitle, YTitle, Title : String; UseCGA2, MaxMinScale : Boolean);

procedure XYGraphD(var XArray : Pointer; NumX : Word; var YArray : Pointer; NumY : Word; StartNum, StopNum : Word; XTitle, YTitle, Title : String; UseCGA2, MaxMinScale : Boolean);

Description

Produces a simple X and Y auto-scaled graph using single (XYGraphS) or double (XYGraphD) precision data. These routines will plot YArray versus XArray if the size of the two arrays are equal. If they are not the same, only the YArray is plotted versus its corresponding data point number. Use YStart and YStop to zoom in on a particular area of the graph. Both the X and Y axes are scalable to the maximum and minimum of the arrays within the specified YStart and YStop interval.

NOTE

The first point of the array is point 0. Thus, if there are 100 points in an array, StopNum should not be greater than 99.

Parameters

XArray	Pointer to either a Single or a Double-precision X-axis data array
NumX	Number of points in XArray
YArray	Pointer to either a Single or a Double-precision Y-axis data array
NumY	Number of points in YArray
StartNum	First Y data point to plot
StopNum	Last Y data point to plot
XTitle	X-axis Title
YTitle	Y-axis Title
Title	Graph Title
UseCGA2	If TRUE, use CGA 640x200 mode so that GRAPHICS.COM can do a screen dump.
MaxMinScale	If TRUE, scale to the minimum and maximum values of the Y-axis data if all of the data points are of the same sign.

Example

```
{ TP6 setup commands called before here }
{ See Read2001Buffer, paragraph 3.2 on how to get data. }
{ Plot AC Voltage vs. Time scaled to AC Data }
XYGraphS( Times, BufferData, 0, NumPoints-1, 'Time (sec)', 'AC Voltage (Vrms)', 'AC Voltage vs. Time',
FALSE, TRUE);
```

4.4 procedure DataViewS(var DataArray:Pointer; NumDataPoints:Word); procedure DataViewD(var DataArray:Pointer; NumDataPoints:Word);

Description

Views a single or double precision array using PageUp, PageDn, Home, End, and the Arrow Keys. Pressing the Esc key aborts the data display.

Parameters

DataArray	Either a pointer to a Single or a Double precision data array to display
NumDataPoints	Number of data points to display.

Example

See Read2001Buffer, paragraph 3.2.

4.5 function str_ltrim(trim_string : String) : String;

Description

Strips leading spaces from the given string.

Parameters

trim_string String that needs to have leading spaces stripped from it.

Returns

A version of trim_string with its leading spaces stripped.

Example

```
var
trim_string : String;
{ TP6 setup commands called before here }
trim_string := ' 3 leading spaces';
trim_string := str_ltrim( trim_string);
{ trim_string := '3 leading spaces' }
```

4.6 function str_rtrim(trim_string : String) : String;

Description

Strips trailing spaces from the given string.

Parameters

trim_string String that needs to have trailing spaces stripped from it.

Returns

A version of trim_string with its trailing spaces stripped.

Example

```
var
trim_string : String;
{ TP6 setup commands called before here }
trim_string := '3 trailing spaces ';
trim_string := str_rtrim( trim_string);
{ trim_string := '3 trailing spaces' }
```

4.7 function str_upr(convert_string : String) : String;

Description

Convert a string to all upper case.

Parameters

convert_string String that needs to be converted to upper case.

Returns

An all upper case version of convert_string.

Example

```
var
convert_string : String;
{ TP6 setup commands called before here }
convert_string := 'abCDefGH';
convert_string := str_upr( convert_string);
{ convert_string := 'ABCDEFGH' }
```

4.8 function str_low(convert_string : String) : String;

Description

Convert a string to all lower case.

Parameters

convert_string String that needs to be converted to lower case.

Returns

An all lower case version of convert_string.

Example

```
var
convert_string : String;
{ TP6 setup commands called before here }
convert_string := 'abCDefGH';
convert_string := str_low( convert_string);
{ convert_string := 'abcdefgh' }
```

Section 5
General Model 2001 / 7001 Routines

The following routines perform some extra functions that are not in the Model 2001/7001 and manipulate the Model 2001/7001's front panel.

NOTE

These functions and subroutines also support the Model 7002 Switch System. Just send Model 7001 commands to a Model 7002.

All of these routines will set the Global Variable TimeOutError if an IEEE-488 Timeout Error occurs. Functions that have bounded parameters (maximum and/or minimum parameters) will set the Global Variable OutOfRange if a parameter is not within the limits specified (see Global Variables in Appendix A).

5.1 procedure Set2001Function(Func : Integer);

Description

Puts the Model 2001 into the specified measurement function.

Parameters

Func (1-8) See Function Constants, in Appendix B.

Example

```
{ TP6 setup commands called before here }
 { All three of the following put the 2001 in AC Volts: }
Set2001Function(ACV);
ACV2001;
Send2001('FUNC "VOLT:AC"');
```

5.2 function Get2001Units(Func : Integer) : String;

Description

Gets the proper Model 2001 units for the function specified.

Parameters

Func 1-8, See Function Constants, in Appendix B.

Returns

'Error!' if a TimeOutError occurred, or
Func := 1 (DC Volts): 'VDC '
Func := 2 (AC Volts):
<5-character string> + <0-4 character string>, where:
<5-character string> := ' dB ', ' dBm ', or 'VAC '
<0-4 character string> := '', 'Avg', 'Peak', '+Pk', '-Pk', or 'RMS'
Func := 3 (DC Current): 'ADC' or 'ADC ICkt'
Func := 4 (AC Current): 'AAC ' or 'AAC Avg' or 'AAC RMS'
Func := 5 (2-wire Resistance): 'Ω2W' or 'Ω2W Ocmp'
Func := 6 (4-wire Resistance): 'Ω4W' or 'Ω4W Ocmp'
Func := 7 (Frequency): 'Hz'
Func := 8 (Temperature): '°F', '°C', or ' K'

Example

```
var
A : String;
{ TP6 setup commands called before here }
Send2001('*RST');
{ AC Voltage with the detector set to RMS will }
{ return 'VAC RMS' }
A := Get2001Units(ACV);
```

5.3 function Get2001SaveRecallSize : Integer;

Description

Finds the number of Model 2001 Save/Recall (*SAV, *RCL) locations available for storing Model 2001 configurations.

Returns

Model 2001 Save/Recall Sizes

Model	SaveRecallSize
2001	1
2001/MEM1	5
2001/MEM2	10

Example

```
var
RecallNumber : Integer;
recall : String;
{ TP6 setup commands called before here }
{ Model 2001 with MEM1 option }
```

```
{ RecallNumber := number of setup location to recall }
if (RecallNumber <= Get2001SaveRecallSize) then
begin
   Str( RecallNumber, recall);
   Send2001('*RCL' + recall);
end
else
   WriteLn('Recall Number is too large!');
```

5.4 funct ion Get2001FuncHeader(Func : Integer) : String;

Descr iption

Returns the proper SENSe Subsystem SCPI Header string for the function specified.

Parameters

Func (1-8) See Function Constants, Appendix B.

Returns

Func	Returns
1	VOLT:DC:
2	VOLT:AC:
3	CURR:DC:
4	CURR:AC:
5	RES:
6	FRES:
7	FREQ:
8	TEMP:

Example

```
var
A : String;
{ TP6 setup commands called before here }
A := Get2001FuncHeader(DCV); { A := 'VOLT:DC:' }
```

5.5 function Get2001Func : Integer;

Description

Finds the present Model 2001 Function.

Returns

The Model 2001 function number from 1 to 8 as specified by the Function Constants, Appendix B.

Example

```
var
Func : Integer;
{ TP6 setup commands called before here }
DCV2001; { Put 2001 in DC Volts mode }
Func := Get2001Func;
{ Func := 1, which is the CONSTant DCV }
```

5.6 function Check2001Val(CheckVal, MinVal, MaxVal: Double; CheckMinInf : Integer) : String;

Description

Checks a given double precision value (CheckVal) against the given minimum allowed value (MinVal) and the maximum allowed value (MaxVal). It is used internally by the Model 2001/7001 routines to check parameter ranges.

Parameters

CheckVal

Any double precision value or one of the following constants:

INF := 9.9e+37
MINIMUM := 9.8e+37
MAXIMUM := 9.7e+37
DEFAULT := 9.6e+37

CheckMinInf

0 - don't check for MINIMUM, MAXIMUM, DEFAULT, or INF
1 - check for MINIMUM, MAXIMUM, or DEFAULT
2 - check for MINIMUM, MAXIMUM, DEFAULT, or INF

Associated constants:

CHECKNONE := 0
CHECKMINMAX := 1
CHECKINF := 2

Returns

- a null string and sets OutOfRange global variable to TRUE (-1) if CheckVal is out of range.
An ASCII string representation of CheckVal if in range.
'MAX', 'MIN', 'DEF', or 'INF' if CheckVal is equivalent to one of the constants above.

Example

```
var
Value : String;
{ TP6 setup commands called before here }
{ Checks for a value to be between 100 an 1000 inclusive }
Value := Check2001Val(MINIMUM, 100, 1000, CHECKMINMAX);
{ Value := 'MIN' }
```

5.7 procedure Hit2001Key(HitKey : Integer);

Description

Presses the specified Model 2001 front panel key using SYSTEM:KEY.

Parameters

HitKey 1-23, 26-31 (OutofRange set and does nothing otherwise)

Defined keys are as follows:

UPKEY := 1	TEMPKEY := 2	LEFTKEY := 3
MENUKEY := 4	ACIKEY := 5	STOREKEY := 6
LOCALKEY := 7	PREVIOUSKEY := 8	AUTOKEY := 9
RIGHTKEY := 10	EXITKEY := 11	R2KEY := 12
RECALLKEY := 13	CHANKEY := 14	DCVKEY := 15
NEXTKEY := 16	DOWNKEY := 17	ENTERKEY := 18
R4KEY := 19	FILTERKEY := 20	SCANKEY := 21
ACVKEY := 22	RELKEY := 23	FREQKEY := 26
MATHKEY := 27	CONFIGKEY := 28	DCIKEY := 29
TRIGKEY := 30	INFOKEY := 31	

Example

```
{ TP6 setup commands called before here }
Hit2001Key(AUTOKEY); { Hit Auto Range Key on 2001 }
```

5.8 procedure Disp2001(line1 : String; line2 : String); procedure Disp7001(line1 : String; line2 : String);

Description

Immediately displays line1 on the first line and line2 on the second line of the Model 2001/7001's display.

Parameters

line1	maximum of 20 characters
line2	maximum of 32 characters

Example

```
var
Line1 : String;
{ TP6 setup commands called before here }
Line1 := '2001/7001 Support Software';
Disp2001( Line1,'(c) 1992 Keithley Instruments');
NoDisp2001; { turn off user display }
```

5.9 procedure NoDisp 2001; procedure NoDisp 7001;

Description

Turns off the user's displayed messages on the Model 2001/7001.

Section 6
Model 2001 SENSe[1] Subsystem Commands

All of these routines will set the Global Variable TimeOutError if an IEEE-488 Timeout Error occurs. Functions that have bounded parameters (maximum and/or minimum parameters) will set the Global Variable OutOfRange if a parameter is not within the limits specified (see Global Variables, Appendix A).

See Model 2001 Minimum and Maximum Sense Constants on p.56 for CONSTants for use with the 2001 SENSe[1] Subsystem Commands.

6.1 2001 Function Change Subroutines

Theses subroutines change the present function of the Model 2001.

6.1.1 procedure DCV2001;

Description

Puts 2001 into DC Volts Mode.

Example

```
{ TP6 setup commands called before here }
{ All three of the following put the 2001 in DC Volts: }
DCV2001;
Set2001Function(DCV);
Send2001('FUNC "VOLT:DC"');
```

6.1.2 procedure ACV2001;

Description

Puts Model 2001 into AC Volts mode.

6.1.3 procedure DCA2001;

Description

Puts Model 2001 into DC Current mode.

6.1.4 procedure ACA2001;

Description

Puts Model 2001 into AC Current mode.

6.1.5 procedure RES2_2001;

Description

Puts Model 2001 into 2-wire Resistance mode.

6.1.6 procedure RES4_2001;

Description

Puts Model 2001 into 4-wire Resistance mode.

6.1.7 procedure FREQ2001;

Description

Puts Model 2001 into Frequency mode.

6.1.8 procedure TEMP2001;

Description

Puts Model 2001 into Temperature mode.

6.2 Return Model 2001 Reading Functions

Theses subroutines return the latest reading on the specified function of the Model 2001. All routines use the SCPI Command, MEASure:(FunctionName)?, except Get2001Rdg which uses the SCPI command, 'FETCh?'.

6.2.1 function Get2001Rdg : String;

Description

FETCHes a Model 2001 reading in the present Mode and FORMat.

Returns

'Error!' if a TimeOutError occurred, or Reading String.

Example

```
var
A, B : String;
{ TP6 setup commands called before here }
{ The following two statements are equivalent: }
A := Get2001Rdg;
B := Q2001('FETCH?');
```

6.2.2 function GetDCV2001 : String;

Description

Gets a Model 2001 DC Volts Reading.

Returns

'Error!' if a TimeOutError occurred, or DC Volts Reading String.

Example

```
var
A, B : String;
{ TP6 setup commands called before here }
{ The following two statements are equivalent: }
A := GetDCV2001;
B := Q2001('MEASURE:VOLT:DC?');
```

6.2.3 function GetACV2001 : String;

Description

Gets a Model 2001 AC Volts Reading.

Returns

'Error!' if a TimeOutError occurred, or AC Volts Reading String.

6.2.4 function GetDCA2001 : String;

Description

Gets a Model 2001 DC Current Reading.

Returns

'Error!' if a TimeOutError occurred, or DC Current Reading String.

6.2.5 function GetACA2001 : String;

Description

Gets a Model 2001 AC Current Reading.

Returns

'Error!' if a TimeOutError occurred, or AC Current Reading String.

6.2.6 function Get2RES2001 : String;

Description

Gets a Model 2001 2-wire Resistance Reading.

Returns

'Error!' if a TimeOutError occurred, or 2-wire Resistance Reading String.

6.2.7 function Get4RES2001 : String;

Description

Gets a Model 2001 4-wire Resistance Reading.

Returns

'Error!' if a TimeOutError occurred, or 4-wire Resistance Reading String.

6.2.8 function GetFREQ2001 : String;

Description

Gets a Model 2001 Frequency Reading

Returns

'Error!' if a TimeOutError occurred, or Frequency Reading String.

6.2.9 function GetTEMP2001 : String;

Description

Gets a 2001 Temperature Reading

Returns

'Error!' if a TimeOutError occurred, or Temperature Reading String.

6.3 DC Voltage Functions

The functions setup and return the settings of the configurable options of the Model 2001's DC Voltage measurement function.

6.3.1 procedure Set2001DCV(Range, Time, Digits: Double);

Description

Sets the Model 2001's DC Voltage Range, Aperture Time, and Number of Digits.

Parameters

Range

0 to +1100 MAXIMUM, MINIMUM, or DEFAULT

Time (Aperture Time)

166.667e-6s to 166.666667e-3s (60Hz); 2e-4 to .2 (50Hz) MAXIMUM, MINIMUM, or DEFAULT

Digits

3.5 to 8.4999 (7.5-8.4999:=>7.5d, 3.5-4.4999:=>3.5d) MAXIMUM, MINIMUM, or DEFAULT

Example

```
{ TP6 setup commands called before here }
{ Set DC Voltage MAXIMUM range, 1 Power Line Cycle }
{ Integration, and 6-1/2 digits }
Set2001DCV( MAXIMUM, 1/60, 7);
```

6.3.2 function Set2001DCVQ : String;

Description

Queries the Model 2001 for its DC Voltage Range, Aperture Time, and Number of Digits settings.

Returns

'Error!' if a TimeOutError occurred, or
Query 1: Range: 0 to +1100
Query 2: Time: 166.667e-6 to .2
Query 3: Digits: 3.5 to 8.4999 (4:=3.5d, 8:=7.5d, etc.)

Use ParseQuery to separate return string into components.

Example

```
var
A, Range, AperTime, Digits : String;
{ TP6 setup commands called before here }
A := Set2001DCVQ; { Get 2001 DC Voltage Settings }
Range := ParseQuery(A, 1); { Extract Range setting }
AperTime := ParseQuery(A, 2); { Extract Aperture Time }
Digits := ParseQuery(A, 3); { Extract Number of Digits }
```

6.3.3 procedure Auto2001DCV(AutoRange, AutoTime, AutoDigits : Integer);

Description

Sets the Model 2001's DC Voltage Auto Range, Auto Time, and Auto Digits settings.

Parameters

AutoRange, AutoTime, AutoDigits:

-1 := Don't set, 0 := Off, +1 := On, +2 := Once

The following constants can also be used:

NO := -1, TOFF := 0, TON := +1, ONCE := +2

Example

```
{ TP6 setup commands called before here }
{ Turn 2001's DC Voltage Auto Range ON, Time OFF, }
{ and Digits ONCE: }
Auto2001DCV( TON, TOFF, ONCE);
```

6.3.4 function Auto2001DCVQ : String;

Description

Queries the Model 2001 for its DC Voltage Auto Range, Auto Time, and Auto Digits settings.

Returns

'Error!' if a TimeOutError occurred, or
Query 1: Auto Range: 1 or 0 (1:=ON, 0:=OFF)
Query 2: Auto Time: 1 or 0 (1:=ON, 0:=OFF)
Query 3: Auto Digits: 1 or 0 (1:=ON, 0:=OFF)

Use ParseQuery to separate return string into components.

Example

```
var
A, AutoRange, AutoTime, AutoDigits : String;
{ TP6 setup commands called before here }
A := Auto2001DCVQ; { Get 2001 DC Voltage Auto Settings }
AutoRange :=ParseQuery(A, 1); { Extract Auto Range setting }
AutoTime := ParseQuery(A, 2); { Extract Auto Time }
AutoDigits := ParseQuery(A, 3); { Extract Auto Digits }
```

6.4 AC Voltage Functions

The functions setup and return the settings of the configurable options of the Model 2001's AC Voltage measurement function.

6.4.1 procedure Set2001ACV(Range, Time, Digits: Double);

Description

Sets the Model 2001's AC Voltage Range, Aperture Time, and Number of Digits.

Parameters

Range

0 to +775 (+1100 if Peak, +Peak or -Peak On) MAXIMUM, MINIMUM, or DEFAULT

Time (Aperture Time)

166.667e-6s to 166.666667e-3s (60Hz); 2e-4 to .2 (50Hz) MAXIMUM, MINIMUM, or DEFAULT

Digits

3.5 to 8.4999 (7.5-8.4999:=>7.5d, 3.5-4.4999:=>3.5d) MAXIMUM, MINIMUM, or DEFAULT

Example

```
{ TP6 setup commands called before here }
{ Set AC Voltage MINIMUM range, 10 Power Line Cycle }
{ Integration, and 6-1/2 digits }
Set2001ACV( MAXIMUM, 1/6, 6.5);
```

6.4.2 function Set2001ACVQ : String;

Description

Queries the Model 2001 for its AC Voltage Range, Aperture Time, and Number of Digits settings.

Returns

'Error!' if a TimeOutError occurred, or
Query 1: Range: 0 to +1100
Query 2: Time: 166.667e-6 to .2
Query 3: Digits: 3.5 to 8.4999 (4:=3.5d, 8:=7.5d, etc.)

Use ParseQuery to separate return string into components.

Example

```
var
A : String;
Range, AperTime, Digits, Code : Integer;
{ TP6 setup commands called before here }
A := Set2001ACVQ; { Get 2001 AC Voltage Settings }
VAL(ParseQuery(A, 1),Range,Code); {Extract Range setting}
VAL(ParseQuery(A, 2),AperTime,Code); {Extract Aperture Time}
VAL(ParseQuery(A, 3),Digits,Code); {Extract # of Digits}
```

6.4.3 procedure Auto2001ACV(AutoRange, AutoTime, AutoDigits : Integer);

Description

Sets the Model 2001's AC Voltage Auto Range, Auto Time, and Auto Digits settings.

Parameters

AutoRange, AutoTime, AutoDigits

-1 := Don't set, 0 := Off, +1 := On, +2 := Once

The following constants can also be used:

NO := -1, TOFF := 0, TON := +1, ONCE := +2

Example

```
{ TP6 setup commands called before here }
{ Turn 2001's AC Voltage Auto Range ON, Time OFF, }
{ and Digits unaffected: }
Auto2001ACV( TON, TOFF, NO);
```

6.4.4 function Auto2001ACVQ : String;

Description

Queries the Model 2001 for its AC Voltage Auto Range, Auto Time, and Auto Digits settings.

Returns

'Error!' if a TimeOutError occurred, or
Query 1: Auto Range: 1 or 0 (1:=ON, 0:=OFF)
Query 2: Auto Time: 1 or 0 (1:=ON, 0:=OFF)
Query 3: Auto Digits: 1 or 0 (1:=ON, 0:=OFF)

Use ParseQuery to separate return string into components.

Example

```
var
A, AutoRange, AutoTime, AutoDigits : String;
{ TP6 setup commands called before here }
A := Auto2001ACVQ; { Get 2001 AC Voltage Auto Settings }
AutoRange := ParseQuery(A, 1); { Extract Auto Range setting}
AutoTime := ParseQuery(A, 2); { Extract Auto Time }
AutoDigits := ParseQuery(A, 3); { Extract Auto Digits }
```

6.5 DC Current Functions

The functions setup and return the settings of the configurable options of the Model 2001's DC Current measurement function.

6.5.1 procedure Set2001DCA(Range, Time, Digits: Double);

Description

Sets the Model 2001's DC Current Range, Aperture Time, and Number of Digits.

Parameters

Range

0 to +2.1, ignored if In Circuit Mode is enabled. MAXIMUM, MINIMUM, or DEFAULT

Time (Aperture Time)

166.667e-6s to 166.666667e-3s (60Hz); 2e-4 to .2 (50Hz) MAXIMUM, MINIMUM, or DEFAULT

Digits

3.5 to 8.4999 (7.5-8.4999:=>7.5d, 3.5-4.4999:=>3.5d) MAXIMUM, MINIMUM, or DEFAULT

Example

```
{ TP6 setup commands called before here }
{ Set DC Current to .2A range, 10ms Integration, & 6-1/2d }
Set2001DCA( 0.2, 0.01, 7.1);
```

6.5.2 function Set2001DCAQ : String;

Description

Queries the Model 2001 for its DC Current Range, Aperture Time, and Number of Digits settings.

Returns

'Error!' if a TimeOutError occurred, or
Query 1: Range: 0 to +2.1
Query 2: Time: 166.667e-6 to .2
Query 3: Digits: 3.5 to 8.4999 (4:=3.5d, 8:=7.5d, etc.)

Use ParseQuery to separate return string into components.

Example

```
var
A, Range, AperTime, Digits : String;
{ TP6 setup commands called before here }
A := Set2001DCAQ; { Get 2001 DC Current Settings }
Range := ParseQuery(A, 1); { Extract Range setting }
AperTime := ParseQuery(A, 2); { Extract Aperture Time }
Digits := ParseQuery(A, 3); { Extract Number of Digits }
```

6.5.3 procedure Auto2001DCA(AutoRange, AutoTime, AutoDigits : Integer);

Description

Sets the Model 2001's DC Current Auto Range, Auto Time, and Auto Digits settings.

Parameters

AutoRange, AutoTime, AutoDigits

-1 := Don't set, 0 := Off, +1 := On, +2 := Once

The following constants can also be used:

NO := -1, TOFF := 0, TON := +1, ONCE := +2

Example

```
{ TP6 setup commands called before here }
{ Turn DC Current Auto Range ON, Time OFF, and Digits ONCE:}
Auto2001DCA( 1, 0, 2);
```

6.5.4 function Auto2001DCAQ : String;

Description

Queries the Model 2001 for its DC Current Auto Range, Auto Time, and Auto Digits settings.

Returns

'Error!' if a TimeOutError occurred, or
Query 1: Auto Range: 1 or 0 (1:=ON, 0:=OFF)
Query 2: Auto Time: 1 or 0 (1:=ON, 0:=OFF)
Query 3: Auto Digits: 1 or 0 (1:=ON, 0:=OFF)

Use ParseQuery to separate return string into components.

Example

```
var
A, AutoRange, AutoTime, AutoDigits : String;
{ TP6 setup commands called before here }
A := Auto2001DCAQ; { Get 2001 DC Current Auto Settings }
AutoRange := ParseQuery(A, 1); { Extract Auto Range setting}
AutoTime := ParseQuery(A, 2); { Extract Auto Time }
AutoDigits := ParseQuery(A, 3); { Extract Auto Digits }
```

6.6 AC Current Functions

The functions setup and return the settings of the configurable options of the Model 2001's AC Current measurement function.

6.6.1 procedure Set2001ACA(Range, Time, Digits: Double);

Description

Sets the Model 2001's AC Current Range, Aperture Time, and Number of Digits.

Parameters

Range

0 to +2.1
MAXIMUM, MINIMUM, or DEFAULT

Time (Aperture Time)

166.667e-6s to 166.666667e-3s (60Hz); 2e-4 to .2 (50Hz) MAXIMUM, MINIMUM, or DEFAULT

Digits

3.5 to 8.4999 (7.5-8.4999:=>7.5d, 3.5-4.4999:=>3.5d) MAXIMUM, MINIMUM, or DEFAULT

Example

```
{ TP6 setup commands called before here }
{ Set AC Current to 2mA range, .1 msec }
{ Integration, and 7-1/2 digits }
Set2001ACA( 2e-3, 1e-4, 7.5);
```

6.6.2 function Set2001ACAQ : String;

Description

Queries the Model 2001 for its AC Current Range, Aperture Time, and Number of Digits settings.

Returns

'Error!' if a TimeOutError occurred, or
Query 1: Range: 0 to +2.1
Query 2: Time: 166.667e-6 to .2
Query 3: Digits: 3.5 to 8.4999 (4:=3.5d, 8:=7.5d, etc.)

Use ParseQuery to separate return string into components.

Example

```
var
A, Range, AperTime, Digits : String;
{ TP6 setup commands called before here }
A := Set2001ACAQ; { Get 2001 AC Current Settings }
Range := ParseQuery(A, 1); { Extract Range setting }
AperTime := ParseQuery(A, 2); { Extract Aperture Time }
Digits := ParseQuery(A, 3); { Extract Number of Digits }
```

6.6.3 procedure Auto2001ACA(AutoRange, AutoTime, AutoDigits : Integer);

Description

Sets the Model 2001's AC Current Auto Range, Auto Time, and Auto Digits settings.

Parameters

AutoRange, AutoTime, AutoDigits

-1 := Don't set, 0 := Off, +1 := On, +2 := Once

The following constants can also be used:

NO := -1, TOFF := 0, TON := +1, ONCE := +2

Example

```
{ TP6 setup commands called before here }
{ Turn 2001's AC Current Auto Range ON, Time OFF, }
{ and Digits unaffected: }
Auto2001ACA(TON, 0, -1);
```

6.6.4 function Auto2001ACAQ : String;

Description

Queries the Model 2001 for its AC Current Auto Range, Auto Time, and Auto Digits settings.

Returns

'Error!' if a TimeOutError occurred, or
Query 1: Auto Range: 1 or 0 (1:=ON, 0:=OFF)
Query 2: Auto Time: 1 or 0 (1:=ON, 0:=OFF)
Query 3: Auto Digits: 1 or 0 (1:=ON, 0:=OFF)

Use ParseQuery to separate return string into components.

Example

```
var
A, AutoRange, AutoTime, AutoDigits : String;
{ TP6 setup commands called before here }
A := Auto2001ACAQ; { Get 2001 AC Current Auto Settings }
AutoRange := ParseQuery(A, 1); { Extract Auto Range setting}
AutoTime := ParseQuery(A, 2); { Extract Auto Time }
AutoDigits := ParseQuery(A, 3); { Extract Auto Digits }
```

6.7 2-Wire Resistance Functions

The functions setup and return the settings of the configurable options of the Model 2001's 2-wire Resistance measurement function.

6.7.1 procedure Set2001R2(Range, Time, Digits: Double);

Description

Sets the Model 2001's 2-wire Resistance Range, Aperture Time, and Number of Digits.

Parameters

Range

0 to +1.05e9 or 2.1e5 if SENSe1:RESistance:OCOMpensated ON is set
MAXIMUM, MINIMUM, or DEFAULT

Time (Aperture Time)

166.667e-6s to 166.666667e-3s (60Hz); 2e-4 to .2 (50Hz) MAXIMUM, MINIMUM, or DEFAULT

Digits

3.5 to 8.4999 (7.5-8.4999:=>7.5d, 3.5-4.4999:=>3.5d) MAXIMUM, MINIMUM, or DEFAULT

Example

```
{ TP6 setup commands called before here
 { Set 2-wire Resistance to 200kΩ range, 100 msec }
{ Integration, and 5-1/2 digits }
Set2001R2( 1e5, 0.1, 6.49);
```

6.7.2 function Set2001R2Q : String;

Description

Queries the Model 2001 for its 2-wire Resistance Range, Aperture Time, and Number of Digits settings.

Returns

'Error!' if a TimeOutError occurred, or
Query 1: Range: 0 to +1.05e9
Query 2: Time: 166.667e-6 to .2
Query 3: Digits: 3.5 to 8.4999 (4:=3.5d, 8:=7.5d, etc.)

Use ParseQuery to separate return string into components.

Example

```
var
A, Range, AperTime, Digits : String;
{ TP6 setup commands called before here }
A := Set2001R2Q; { Get 2001 2-wire Resistance settings }
Range := ParseQuery(A, 1); { Extract Range setting }
AperTime := ParseQuery(A, 2); { Extract Aperture Time }
Digits := ParseQuery(A, 3); { Extract Number of Digits }
```

6.7.3 procedure Auto2001R2(AutoRange, AutoTime, AutoDigits : Integer);

Description

Sets the Model 2001's 2-wire Resistance Auto Range, Auto Time, and Auto Digits settings.

Parameters

AutoRange, AutoTime, AutoDigits

-1 := Don't set, 0 := Off, +1 := On, +2 := Once

The following constants can also be used:

NO := -1, TOFF := 0, TON := +1, ONCE := +2

Example

```
{ TP6 setup commands called before here }
{ Turn 2001's 2-wire Resistance Auto Range ON, Time OFF, }
{ and Digits ONCE: }
Auto2001R2( TON, TOFF, ONCE);
```

6.7.4 function Auto2001R2Q : String;

Description

Queries the Model 2001 for its 2-wire Resistance Auto Range, Auto Time, and Auto Digits settings.

Returns

'Error!' if a TimeOutError occurred, or
Query 1: Auto Range: 1 or 0 (1:=ON, 0:=OFF)
Query 2: Auto Time: 1 or 0 (1:=ON, 0:=OFF)
Query 3: Auto Digits: 1 or 0 (1:=ON, 0:=OFF)

Use ParseQuery to separate return string into components.

Example

```
var
A, AutoRange, AutoTime, AutoDigits : String;
{ TP6 setup commands called before here }
A := Auto2001R2Q; {Get 2001 2-wire Resistance Auto Settings}
AutoRange := ParseQuery(A, 1); { Extract Auto Range setting}
AutoTime := ParseQuery(A, 2); { Extract Auto Time }
AutoDigits := ParseQuery(A, 3); { Extract Auto Digits }
```

6.8 4-Wire Resistance Functions

The functions setup and return the settings of the configurable options of the Model 2001's 4-wire Resistance measurement function.

6.8.1 procedure Set2001R4(Range, Time, Digits: Double);

Description

Sets the Model 2001's 4-wire Resistance Range, Aperture Time, and Number of Digits.

Parameters

Range

0 to 2.1e5 MAXIMUM, MINIMUM, or DEFAULT

Time (Aperture Time)

166.667e-6s to 166.666667e-3s (60Hz); 2e-4 to .2 (50Hz) MAXIMUM, MINIMUM, or DEFAULT

Digits

3.5 to 8.4999 (7.5-8.4999:=>7.5d, 3.5-4.4999:=>3.5d) MAXIMUM, MINIMUM, or DEFAULT

Example

```
{ TP6 setup commands called before here }
{ Set 4-wire Resistance to 20kΩ range, 100 msec }
{ Integration, and 5-1/2 digits }
Set2001R2( 15000, 100e-3, 5.649);
```

6.8.2 function Set2001R4Q : String;

Description

Queries the Model 2001 for its 4-wire Resistance Range, Aperture Time, and Number of Digits settings.

Returns

'Error!' if a TimeOutError occurred, or
Query 1: Range: 0 to +2.1e5
Query 2: Time: 166.667e-6 to .2
Query 3: Digits: 3.5 to 8.4999 (4:=3.5d, 8:=7.5d, etc.)

Use ParseQuery to separate return string into components.

Example

```
var
A, Range, AperTime, Digits : String;
{ TP6 setup commands called before here }
A := Set2001R4Q; { Get 2001 4-wire Resistance settings }
Range := ParseQuery(A, 1); { Extract Range setting }
AperTime := ParseQuery(A, 2); { Extract Aperture Time }
Digits := ParseQuery(A, 3); { Extract Number of Digits }
```

6.8.3 procedure Auto2001R4(AutoRange, AutoTime, AutoDigits : Integer);

Description

Sets the Model 2001's 4-wire Resistance Auto Range, Auto Time, and Auto Digits settings.

Parameters

AutoRange, AutoTime, AutoDigits

-1 := Don't set, 0 := Off, +1 := On, +2 := Once

The following constants can also be used:

NO := -1, TOFF := 0, TON := +1, ONCE := +2

Example

```
{ TP6 setup commands called before here }
{ Turn 2001's 4-wire Resistance Auto Range ON, Time OFF, }
{ and Digits ONCE: }
Auto2001R4( TON, TOFF, ONCE);
```

6.8.4 function Auto2001R4Q : String;

Description

Queries the Model 2001 for its 4-wire Resistance Auto Range, Auto Time, and Auto Digits settings.

Returns

'Error!' if a TimeOutError occurred, or
Query 1: Auto Range: 1 or 0 (1:=ON, 0:=OFF)
Query 2: Auto Time: 1 or 0 (1:=ON, 0:=OFF)
Query 3: Auto Digits: 1 or 0 (1:=ON, 0:=OFF)

Use ParseQuery to separate return string into components.

Example

```
var
A, AutoRange, AutoTime, AutoDigits : String;
{ TP6 setup commands called before here }
A:=Auto2001R4Q; { Get 2001 4-wire Resistance Auto Settings }
AutoRange := ParseQuery(A, 1); { Extract Auto Range setting}
AutoTime := ParseQuery(A, 2); { Extract Auto Time }
AutoDigits := ParseQuery(A, 3); { Extract Auto Digits }
```

6.9 Frequency Functions

The functions setup and return the settings of the configurable options of the Model 2001's Frequency measurement function.

6.9.1 procedure Set2001F(Digits: Double; Source : Integer);

Description

Sets the Model 2001's Frequency Number of Digits and Source settings.

Parameters

Digits

3.5 to 5.4999 (4.5-5.4999:=>4.5d, 3.5-4.4999:=>3.5d) MAXIMUM, MINIMUM, or DEFAULT

Source

0 Current
1 Voltage

Example

```
{ TP6 setup commands called before here }
{ Set Frequency to 5 digits, Current triggered }
Set2001F( 5.49, 0);
```

6.9.2 function Set2001FQ : String;

Description

Queries the Model 2001 for its Frequency Number of Digits and Measurement Source settings.

Returns

'Error!' if a TimeOutError occurred, or
Query 1: Digits: 3.5 to 5.4999 (4:=3.5d, 5:=4.5d)

Query 2: Source: VOLT or CURR

Use ParseQuery to separate return string into components.

Example

```
var A, Source, Digits : String;
{ TP6 setup commands called before here }
A := Set2001FQ; { Get 2001 Frequency settings }
Digits := ParseQuery(A, 1); { Extract Number of Digits }
Source := ParseQuery(A, 2); { Extract Frequency Source }
```

6.10 Temperature Functions

The functions setup and return the settings of the configurable options of the Model 2001's Temperature measurement function.

6.10.1 procedure Set2001T(Time, Digits: Double);

Description

Sets the Model 2001's Temperature Aperture Time and Number of Digits.

Parameters

Time (Aperture time)

166.667e-6s to 166.666667e-3s (60Hz); 2e-4 to .2 (50Hz) MAXIMUM, MINIMUM, or DEFAULT

Digits

3.5 to 7.4999 (6.5-7.4999:=>6.5d, 3.5-4.4999:=>3.5d) MAXIMUM, MINIMUM, or DEFAULT

Example

```
{ TP6 setup commands called before here }
{ Set Temperature to 100 msec Integration, and 5-1/2 digits}
Set2001T( 0.1, 6.49);
```

6.10.2 function Set2001TQ : String;

Description

Queries the Model 2001 for its Temperature Aperture Time and Number of Digits settings.

Returns

'Error!' if a TimeOutError occurred, or
Query 1: Time: 166.667e-6 to .2
Query 2: Digits: 3.5 to 7.4999 (4:=3.5d, 7:=6.5d, etc.)

Use ParseQuery to separate return string into components.

Example

```
var
A, AperTime, Digits : String;
{ TP6 setup commands called before here }
A := Set2001TQ; { Get 2001 Temperature settings }
AperTime := ParseQuery(A, 1); { Extract Aperture Time }
Digits := ParseQuery(A, 2); { Extract Number of Digits }
```

6.10.3 procedure Auto2001T(AutoTime, AutoDigits : Integer);

Description

Set the Model 2001's Temperature Auto Time and Auto Digits settings.

Parameters

AutoTime, AutoDigits

-1 := Don't set, 0 := Off, +1 := On, +2 := Once

The following constants can also be used:

NO := -1, TOFF := 0, TON := +1, ONCE := +2

Example

```
{ TP6 setup commands called before here }
{ Turn 2001's Temperature Auto Time OFF, and Digits ONCE: }
Auto2001T( TOFF, ONCE);
```

6.10.4 function Auto2001TQ : String;

Description

Queries the Model 2001 for its Temperature Auto Time and Auto Digits settings.

Returns

'Error!' if a TimeOutError occurred, or

Query 1: Auto Time: 1 or 0 (1:=ON, 0:=OFF)
Query 2: Auto Digits: 1 or 0 (1:=ON, 0:=OFF)

Use ParseQuery to separate return string into components.

Example

```
var
A, AutoTime, AutoDigits : String;
{ TP6 setup commands called before here }
A := Auto2001TQ; { Get 2001 Temperature Auto Settings }
AutoTime := ParseQuery(A, 1); { Extract Auto Time }
AutoDigits := ParseQuery(A, 2); { Extract Auto Digits }
```

6.10.5 procedure Set2001RTD(Mode, RType : Integer; Alpha, Beta, Delta, RZero: Double)

Description

Configure and use RTD's to make temperature measurements.

Parameters

Mode	2 or 4 (2-wire or 4-wire RTD)	
RType	0 PT385	
	1 PT3916	
	2 USER	
Alpha	0.00 to 0.01	(ignored if not USER)
Beta	0.00 to 1.00	(ignored if not USER)
Delta	0.00 to 5.00	(ignored if not USER)
RZero	0 to 1000	(ignored if not USER)

Example

```
{ TP6 setup commands called before here }
{ Set RTD Mode to 4-wire RTD, User, Alpha:=.005, Beta:= .5 }
{ Delta:=2.4, and RZero:=500 }
Set2001RTD( 4, 2, 0.005, .5, 2.4, 500);
```

6.10.6 function Set2001RTDQ : String;

Description

Queries the Model 2001 for the Temperature transducer Type, the RTD Type, Alpha, Beta, Delta, and RZero settings.

Returns

'Error!' if a TimeOutError occurred, or
Query 1: Mode: RTD or FRTD or TC
Query 2: RType: USER, PT385, or PT3916
Query 3: Alpha: 0.00 to 0.01
Query 4: Beta: 0.00 to 1.00
Query 5: Delta: 0.00 to 5.00
Query 6: RZero: 0 to 1000

Use ParseQuery to separate return string into components.

Example

```
var
A, Mode, RType: String;
Code : Integer;
Alpha, Beta, Delta, RZero : Double;
{ TP6 setup commands called before here }
A := Set2001RTDQ; { Get 2001 RTD settings }
Mode := ParseQuery(A, 1); { Extract Temp. Device }
RType := ParseQuery(A, 2); { Extract RTD Type }
VAL(ParseQuery(A, 3), Alpha, Code); { Extract Alpha setting }
VAL(ParseQuery(A, 4), Beta, Code); { Extract Beta setting }
VAL(ParseQuery(A, 5), Delta, Code); { Extract Delta setting }
VAL(ParseQuery(A, 6), RZero, Code); { Extract RZero setting }
```

6.10.7 procedure Set2001TC(TType : Char);

Description

Sets the thermocouple type and uses TC's for temperature measurement.

Parameters

TType: 'J', 'K', 'T', 'E', 'R', 'S', or 'B'

Example

```
{ TP6 setup commands called before here }
{ Use Type J thermocouples }
Set2001TC( 'J');
```

6.10.8 function Set2001TCQ : String;

Description

Queries the Model 2001 for the Thermocouple Type.

Returns

'Error!' if a TimeOutError occurred, or 'J', 'K', 'T', 'E', 'R', 'S', or 'B'

Example

```
var
A : String;
{ TP6 setup commands called before here }
A := Set2001TCQ; { Get 2001 Thermocouple Type }
```

Section 7
Model 2001 / 7001 Status Commands

These Routines control the SCPI Status Model of the Model 2001/7001. Constants (see Status Model Constants on page 55) are defined for all of the registers and their bits to make programming the Model 2001/7001 status model simpler. Please refer to the IEEE-488 Reference Section of the Model 2001 or 7001 manual for more details on the specific bit patterns of these registers and bit patterns.

NOTE

These functions and subroutines also support the Model 7002 Switch System. Just send Model 7001 commands to a Model 7002.

All of these routines will set the Global Variable TimeOutError if an IEEE-488 Timeout Error occurs. Functions that have bounded parameters (maximum and/or minimum parameters) will set the Global Variable OutOfRange if a parameter is not within the limits specified (see Global Variables, Appendix A).

7.1 procedure Stat2001(Event2001 : Integer; PTF, NTF, SEN : Word); procedure Stat7001(Event7001 : Integer; PTF, NTF, SEN : Word);

Description

Sets the Model 2001/7001's Operation, Trigger, Arm, Sequence, Questionable, or Measurement Events{ Positive Transition Filter, Negative Transition Filter, and Status Enable Registers. }

Parameters

Event2001, Event7001

1 := Operation Event
2 := Trigger Event
3 := Arm Event
4 := Sequence Event
5 := Questionable Event
6 := Measurement Event (2001 Only)

PTF (Positive Transition Filter): 0 to 32767

NTF (Negative Transition Filter): 0 to 32767

SEN (Status Enable Register): 0 to 32767

Example

{ TP6 setup commands called before here }

{ Cause the TRIG and/or ARM bits to be set in the 2001's }
{ Measurement Event Register when the 2001 enters or exits }
{ Triggering or Arming }
Stat2001(MEASUREMENT,TRIG or ARM,TRIG or ARM,TRIG or ARM);

7.2 function Stat2001Q(Event2001 : Integer) : String; function Stat7001Q(Event7001 : Integer) : String;

Description

Queries the Model 2001/7001's Operation, Trigger, Arm, Sequence, Questionable, or Measurement Events{ Positive Transition Filter, Negative Transition Filter, and Status Enable Registers. }

Parameters

Event2001, Event7001

1 := Operation Event
2 := Trigger Event
3 := Arm Event
4 := Sequence Event
5 := Questionable Event
6 := Measurement Event (2001 Only)

Returns

'Error!' if a TimeOutError occurred, or
Query 1: Status Register (0 to 32767)
Query 2: Condition Register (0 to 32767)
Query 3: Positive Transition Filter (0 to 32767)
Query 4: Negative Transition Filter (0 to 32767)
Query 5: Status Enable Register (0 to 32767)

Use ParseQuery to separate return string into components.

Example

```
var
A, StatReg, CondReg, PTranReg, NTranReg, SEnableReg : String;
{ TP6 setup commands called before here }
A:=Stat7001Q(OPERATION); {Read 7001's Operation Event Registers}
StatReg := ParseQuery(A, 1); { Extract Status Register }
CondReg := ParseQuery(A, 2); { Extract Condition Register }
PTranReg := ParseQuery(A, 3); { Extract +Transition Reg. }
NTranReg := ParseQuery(A, 4); { Extract -Transition Reg. }
SEnableReg := ParseQuery(A, 5); { Status Enable Reg. }
```

7.3 procedure SRE2001(mask : Word);
procedure SRE7001(mask : Word);

Description

Enables the 2001/7001 to generate a Service Request (SRQ) when the indicated bit(s) of the Status Byte Register are set. If a bit is already set when this command is given, no Service Request is generated.

Parameters

mask

0 to 255

Example

```
var
Poll : Byte;
Reading : String;
{ TP6 setup commands called before here }
Poll := Poll2001; { Clear and pending SRQ's }
SRE2001( MAV); { Set up to SRQ on Message Available}
Send2001( 'FETCH?'); { fetch a 2001 reading }
repeat { Set up an uninterruptable wait }
until Wait2001RQS;
Reading := Q2001(''); { Get reading }
```

7.4 function Poll2001 : Byte;
function Poll7001 : Byte;

Description

Serial Polls the 2001/7001.

Returns

Serial poll byte from 0 to 255.

Example

See SRE2001/7001 above.

7.5 procedure ESE2001(mask : Word);
procedure ESE7001(mask : Word);

Description

Sets which bits of the Model 2001/7001's Standard Event Status Register cause the Event Summary Bit (ESB) of the Status Byte Register to be set.

Parameters

mask

0 to 255

7.6 function OPC2001(UnInterruptable : Boolean) : String;
function OPC7001(UnInterruptable : Boolean) : String;

Description

Performs a Model 2001/7001 *OPC? command which returns a '1' when the present operation is complete. However, the '1' may not be output by the Model 2001/7001 for a long time which would cause an IEEE timeout on reading data immediately after sending the *OPC?. Thus, this routine waits for a 2001/7001 SRQ (Service Request) on MAV (Message Available) after sending the query. The Model 2001/7001 may wait forever to send the '1' out if :INITiate:CONTinuous ON was explicitly set as its is in the Factory Defaults. Thus, the :ABORt command would have to be issued before calling these routines.

Parameters

UnInterruptable

FALSE	the command can be aborted by pressing the Esc key.
TRUE	the command cannot be aborted.

Returns

'Error!' if a TimeOutError occurred, or
'1' if operation was completed
'Cancel' if aborted

Example

```
var
OPC : String;
{ TP6 setup commands called before here }
{ Guarantee that *OPC? will not wait forever. }
if (Q2001('INIT:CONT?') = '1') then Send2001('ABORT');
Send2001('VOLTage:DC:RANGe 200');
OPC:=OPC2001(TRUE); { Wait until range change complete }
```

Section 8
Model 2001 / 7001 Scanning Commands

These Routines control the SCPI Trigger Model and Scanning functions of the Model 2001/7001. Please refer to the IEEE-488 Reference Section of the Model 2001 or 7001 manual for more details.

All of these routines will set the Global Variable TimeOutError if an IEEE-488 Timeout Error occurs. Functions that have bounded parameters (maximum and/or minimum parameters) will set the Global Variable OutOfRange if a parameter is not within the limits specified (see Global Variables, Appendix A).

NOTE

These functions and subroutines also support the Model 7002 Switch System. Just send Model 7001 commands to a Model 7002.

See Model 2001/7001 Scanning Minimum and Maximum Constants in Appendix B for CONSTants for use with then Model 2001/7001 Scanning Commands.

8.1 procedure Close2001(Channel : Integer);

Description

Closes a single channel on the 2001 scanner card.

Parameters

Channel

1 to 10

Example

```
{ TP6 setup commands called before here }
Close2001(3); { Take Readings on Channel 3 }
```

8.2 procedure Close7001(ChanList : String);

Description

Closes the specified channels on the 7001 scanner card.

Parameters

ChanList

Any valid SCPI channel-list like (@1!1, 1!2, 1!4:1!10, 2!1!2) with valid channels depending on the scanner card selected. The Model 7001 EAV bit in the serial poll register must be checked to see if the command was accepted or look at the front panel of the instrument for an error message.

Example

```
{ TP6 setup commands called before here }
Close7001('(@1!3:1!6)'); { Close channels 3 to 6 on card 1 }
```

8.3 function Close2001Q : String; function Close7001Q : String;

Description

Queries the Model 2001/7001 for a list of closed channels. The Model 2001 can have only one channel closed at a time on its internal scanner, whereas the Model 7001 many have none or all closed.

Returns

'Error!' if a TimeOutError occurred, or a list of closed channels.

Example

```
var
A : String;
{ TP6 setup commands called before here }
Close2001(2); { Close Channel 2 }
A := Close2001Q; { A := '(@2)' }
```

8.4 procedure Open2001(Channel : Integer);

Description

Opens one or all channel(s) on the Model 2001 scanner card.

Parameters

Channel

0 Open all channels
1 to 10

Example

```
{ TP6 setup commands called before here }
Open2001(1); { Open channel 1 }
Open2001(0); { Open all 2001 channels }
```

8.5 procedure Open7001(ChanList : String);

Description

Opens the specified channels on the 7001 scanner card.

Parameters

ChanList

Any valid SCPI channel-list with valid channels depending on the scanner card selected. The Model 7001 EAV bit in the serial poll register must be checked to see if the command was accepted or look at the front panel of the instrument for an error message.

Example

```
{ TP6 setup commands called before here }
Open7001( '(@1!3:1!6)'); { Open channels 3 to 6 on card 1 }
Open7001( 'ALL'); { Open all 7001 channels }
```

8.6 procedure Scan2001(ChanList : String); procedure Scan7001(ChanList : String);

Description

Defines the Model 2001's Internal ScanList or the Model 7001's scanlist.

Parameters

ChanList

Any valid SCPI channel-list like (@1, 2, 4:10) with channels ranging from 1 to 10 for the Model 2001, or like (@1!1:1!40, 2!1:2!40) for the Model 7001. The Model 2001/7001 EAV bit in the serial poll register must be checked to see if the command was accepted or look at the front panel of the instrument for an error message.

Example

```
{ TP6 setup commands called before here }
Scan2001('(@1:10)'); {Scan All Model 2001 channels }
Scan7001('(@1!1:1!40, 2!1:2!40)'); {Scan All Model 7001 channels }
```

8.7 procedure Arm2001(Count1: Double; Source1 : String; Count2, Delay2: Double; Source2 : String);
procedure Arm7001(Count1: Double; Source1 : String; Count2, Delay2: Double; Source2 : String);

Description

Sets up the Model 2001/7001's Arm Layer 1 (Arm Layer) and Arm Layer 2 (Scan Layer)

Parameters

Count1 and Count2

1 to 99999, 9.9e37 MAXIMUM, MINIMUM, DEFAULT, or INF

Source1 and Source2

HOLD	Hold
MANual	Manual
IMMediate	Immediate
TIMer	Timer (Source2 only)
BUS	IEEE-488 Bus (GET or *TRG)
TLINk	Trigger Link
EXTernal	External

Delay2

0 to 999999.999 seconds MAXIMUM, MINIMUM, or DEFAULT

Example

{ TP6 setup commands called before here }
{Setup the 7001 to do 3 sets of 5 scans. Each scan starts immediately at 1 hour intervals. }
Arm7001(3, 'IMM', 5, MINIMUM, 'TIMER');
Timers7001(3600, 1.5);

8.8 function Arm2001Q : String;
function Arm7001Q : String;

Description

Queries the Model 2001/7001 for its Arm Layer 1 (Arm Layer) and Arm Layer 2 (Scan Layer) settings.

Returns

'Error!' if a TimeOutError occurred, or
Query 1: Arm Layer 1 Count (1 to 99999, 9.9e+37)
Query 2: Arm Layer 1 Source (see Arm2001/7001, short form)
Query 3: Arm Layer 2 Count (1 to 99999, 9.9e+37)
Query 4: Arm Layer 2 Source (see Arm2001/7001, short form)

Query 5: Arm Layer 2 Delay (see Arm2001/7001)

Use ParseQuery to separate return string into components.

Example

```
var
A, Source1, Source2 : String;
Count1, Count2 : Double;
Code : Integer;

{ TP6 setup commands called before here }
A := Arm7001Q; { Read 7001's Arm Layers 1&2 Setup }
VAL(ParseQuery(A, 1), Count1, Code); { Get Arm Layer 1 Count }
Source1 := ParseQuery(A, 2); { Get Arm Layer 1 Source }
VAL(ParseQuery(A, 3), Count2, Code); { Get Arm Layer 2 Count }
VAL(ParseQuery(A, 4), Delay2, Code); { Get Arm Layer 2 Delay }
Source2 := ParseQuery(A, 5); { Get Arm Layer 2 Source }
```

8.9 procedure Trig2001(Count1, Delay1: Double; Source1 : String); procedure Trig7001(Count1, Delay1: Double; Source1 : String);

Description

Sets up the Model 2001/7001's Trigger Sequence (Trigger Layer, i.e., Model 2001 Measure Layer and Model 7001 Channel Layer).

Parameters

Count1

1 to 99999, 9.9e37 MAXIMUM, MINIMUM, DEFAULT or INF

Delay1

0 to 999999.999 seconds MAXIMUM, MINIMUM, or DEFAULT

Source1

HOLD	Hold
MANual	Manual
IMMediate	Immediate
BUS	IEEE-488 Bus (GET or *TRG)
TIMer	Timer
TLINK	Trigger Link
EXTernal	External

Example

```
{ TP6 setup commands called before here }
```

```
{ Setup the 7001 to scan 40 channels with no delay at }
{ 1.5 second intervals. }
Trig7001(40, MINIMUM, 'TIM');
Timers7001( 3600, 1.5);
```

8.10 function Trig2001Q : String; function Trig7001Q : String;

Description

Queries the Model 2001/7001 for its Trigger Sequence (Trigger Layer) settings.

Returns

'Error!' if a TimeOutError occurred, or
Query 1: Trigger Sequence 1 Count (1 to 99999, 9.9e+37)
Query 2: Trigger Sequence 1 Source (see Trig2001/7001)
Query 3: Trigger Sequence 1 Delay (see Trig2001/7001)

Use ParseQuery to separate return string into components.

Example

```
var
A, Source1, Source2 : String;
Count1, Count2 : Double;
Code : Integer;

{ TP6 setup commands called before here }
A := Trig7001Q; { Read 7001's Trigger Layer Setup }
VAL(ParseQuery(A, 1), Count1, Code); { Get Trigger Count }
VAL(ParseQuery(A, 2), Delay1, Code); { Get Trigger Delay }
Source1 := ParseQuery(A, 3); { Get Trigger Source }
```

8.11 procedure Timers2001(ArmTimer2, TrigTimer1: Double); procedure Timers7001(ArmTimer2, TrigTimer1: Double);

Description

Sets the Model 2001/7001's Trigger Model timers in Arm Layer 2 (Scan Layer) and Trigger Sequence 1 (Trigger Layer).

Parameters

ArmTimer2, TrigTimer1

.001 to 999999.999 MAXIMUM, MINIMUM, or DEFAULT

Example

```
{ TP6 setup commands called before here }
{ Scan at 1 hour intervals, trigger at 1.5 sec intervals. }
Timers7001( 3600, 1.5);
```

8.12 function Timers2001Q : String; function Timers7001Q : String;

Description

Queries the Model 2001/7001 for its Arm Layer 2 (Scan Layer) and Trigger Sequence 1 (Trigger Layer) timer settings.

Returns

'Error!' if a TimeOutError occurred, or
Query 1: Arm Layer 2 Timer (see Timers2001/7001)
Query 2: Trigger Sequence Timer (see Timers2001/7001)

Use ParseQuery to separate return string into components.

Example

```
var
A : String;
Code : Integer;
ArmTimer, TrigTimer : Double;
{ TP6 setup commands called before here }
A := Timers7001Q { Read Arm Layer 1 and Trigger Timers }
VAL(ParseQuery(A, 1), ArmTimer, Code); { Get Arm Layer 1 Timer }
VAL(ParseQuery(A, 2), TrigTimer, Code); { Get Trigger Timer }
```

8.13 procedure ArmTcon2001(Dir1 : String; ILine1, OLine1 : Integer; Dir2 : String; ILine2, OLine2 : Integer); procedure ArmTcon7001(Dir1 : String; ILine1, OLine1 : Integer; Dir2 : String; ILine2, OLine2 : Integer);

Description

Sets the Model 2001/7001's Arm Layer 1 (Arm Layer) and Arm Layer 2 (Scan Layer) Trigger configurations. Note that OLine can not be the same as ILine. If they are, the Oline will be made 1 line number (wrapping around to 1 if necessary) higher then the Iline.

Parameters

Dir1 and Dir2

ACCeptor	Disable Source Bypass
SOURce	Enable Source Bypass

ILine1 and ILine2

1 to 6	Trigger Link input line number

OLine1 and OLine2

1 to 6	Trigger Link output line number

Example

{ TP6 setup commands called before here }
{Don't bypass Arm Layer 1 source and use Trigger Link Lines 1 and 2 as input and output. Bypass Arm Layer 2 source and use Trigger Link lines 3 and 4 as I/O. }
ArmTcon7001('ACC', 1, 2, 'SOURCE', 3, 4);

8.14 function ArmTcon2001Q : String; function ArmTcon7001Q : String;

Description

Queries the Model 2001/7001 for its Arm Layer 1 (Arm Layer) and Arm Layer 2 (Scan Layer) trigger configuration settings.

Returns

'Error!' if a TimeOutError occurred, or
Query 1: Arm Layer 1 Direction (ACC or SOUR)
Query 2: Arm Layer 1 Input Line (1-6)
Query 3: Arm Layer 1 Output Line (1-6)
Query 4: Arm Layer 2 Direction (ACC or SOUR)
Query 5: Arm Layer 2 Input Line (1-6)
Query 6: Arm Layer 2 Output Line (1-6)

Use ParseQuery to separate return string into components.

Example

```
var
A, Dir1, Dir2 : String;
In1, Out1, In2, Out2, Code : Integer;
{ TP6 setup commands called before here }
A := ArmTcon7001Q; { Get Arm Layers 1&2 Trigger Configuration }
Dir1 := ParseQuery(A, 1); { Get Arm Layer 1 Direction }
```

```
VAL(ParseQuery(A, 2), In1, Code); { Get Arm Layer 1 Input Line }
VAL(ParseQuery(A, 3), Out1, Code); { Get Arm Layer 1 Output Line}
Dir2 := ParseQuery(A, 4); { Get Arm Layer 2 Direction }
VAL(ParseQuery(A, 5), In2, Code); { Get Arm Layer 2 Input Line }
VAL(ParseQuery(A, 6), Out2, Code); { Get Arm Layer 2 Output Line}
```

8.15 procedure TrigTcon2001(Dir1, Synch1 : String; ILine1, Oline1 : Integer); procedure TrigTcon7001(Dir1, Synch1 : String; ILine1, Oline1 : Integer);

Description

Sets the Model 2001/7001's Trigger Sequence 1 and Trigger Sequence 2 trigger configurations. Note that OLine1 can not be the same as ILine1 if Synch1:='ASYN'. If they are, the Oline1 will be made 1 line number (wrapping around to 1 if necessary) higher then the Iline1.

Parameters

Dir1

ACCeptor	Disable Source Bypass
SOURce	Enable Source Bypass

Synch1

ASYNchronous	Asynchronous Trigger Link
SSYNchronous	Semi-Synchronous Link

ILine1

1-6 - Trigger Link input line number (I/O if SSYN)

OLine1

1-6 - Trigger Link output line number (ignored if SSYN)

Example

```
{ TP6 setup commands called before here }
{ Don't bypass Trigger Sequence 1 source, use the Semi-Synchronous Trigger Link, and use Trigger Link
Line 5 as both input and output. }
TrigTcon7001( 'ACC', 'SSYN', 5, 6);
```

8.16 function TrigTcon2001Q : String; function TrigTcon7001Q : String;

Description

Queries the Model 2001/7001 for its Trigger Sequence 1 (Trigger Layer) trigger configuration settings.

Returns

'Error!' if a TimeOutError occurred, or
Query 1: Trigger Sequence 1 Direction (ACC or SOUR)
Query 2: Trigger Sequence 1 Trigger Link Mode (ASYN or SSYN
Query 3: Trigger Sequence 1 Input Line (1-6)
Query 4: Trigger Sequence 1 Output Line (1-6, 0 if SSYN)

Use ParseQuery to separate return string into components.

Example

```
var
A, Dir1 : String; In1, Out1, Code : Integer;
{ TP6 setup commands called before here }
A := TrigTcon7001Q; { Get Trigger Layer Trigger Configuration }
Dir1 := ParseQuery(A, 1); { Get Trigger Layer Direction}
VAL(ParseQuery(A, 2), In1, Code); { Get Trigger Layer Input Line}
VAL(ParseQuery(A, 3), Out1, Code);{Get Trigger Layer Output Line}
```

Section 9
Model 2001 Calculate Commands

These Routines control the Model 2001's Calculate Subsystem capabilities, which include limit testing, mX+B, percent, and data buffer statistics.

All of these routines will set the Global Variable TimeOutError if an IEEE-488 Timeout Error occurs. Functions that have bounded parameters (maximum and/or minimum parameters) will set the Global Variable OutOfRange if a parameter is not within the limits specified (see Global Variables in Appendix A).

See Model 2001 Minimum and Maximum Calculate Constants in Appendix B for CONSTants for use with then Model 2001 Calculate Commands.

9.1 procedure Set2001Calc1MXB(MMFactor, MBFactor: Double);

Description

Sets the Model 2001's CALCulate1 Subsystem's Mx+B coefficients and enables the Mx+B mode of operation.

Parameters

MMFactor

-9.999999999e20 to +9.999999999e20

MBFactor

-9.999999999e30 to +9.999999999e30

Example

```
{ TP6 setup commands called before here }
{ Set and use mX+b with slope of 10 and intercept of 5 }
Set2001Calc1MXB( 10.0, 5.0);
```

9.2 procedure Set2001Calc1PERC(Percent: Double);

Description

Sets the Model 2001's CALCulate1 Subsystem's Percentage target and enables the percentage mode of operation.

Parameters

Percent

-9.999999999e35 to +9.999999999e35

Example

```
{ TP6 setup commands called before here }
{ Set 100 of full scale at 15.24 }
Set2001Calc1PERC( 15.24);
```

9.3 procedure Calc1_2001(State : Boolean);

Description

Sets the state of the Model 2001's CALCulate1 Subsystem.

Parameters

State

FALSE := Turn CALC1 Off
TRUE := Turn CALC1 On

Example

```
{ TP6 setup commands called before here }
Calc1_2001( FALSE); { Turn off CALC1 System }
```

9.4 function Set2001Calc1Q : String;

Description

Queries the Model 2001 for its CALCulate1 Subsystem's State, MMFactor, MBFactor, and Percent settings.

Returns

'Error!' if a TimeOutError occurred, or
Query 1: CALCulate1's State (1 or 0)
Query 2: CALCulate1's MMFactor (see Set2001Calc1MXB)
Query 3: CALCulate1's MBFactor (see Set2001Calc1MXB)
Query 4: CALCulate1's Percent (see Set2001Calc1PERC)

Use ParseQuery to separate return string into components.

Example

```
var
A, Calc1State : String;
code : Integer;
MMFactor, MBFactor, Percent : Double;
{ TP6 setup commands called before here }
A := Set2001Calc1Q; { Get 2001 CALC1 settings }
Calc1State := ParseQuery(A, 1); { Get CALC1's State }
VAL(ParseQuery(A, 2), MMFactor, code); { Get slope of mX+b }
VAL(ParseQuery(A, 3), MBFactor, code); { Get intercept of mX+b }
VAL(ParseQuery(A, 4), Percent, code); { Get 100 value }
```

9.5 function Calc1_2001Q : String;

Description

Queries the Model 2001 for the result of the latest CALCulate1 calculation.

Returns

'Error!' if a TimeOutError occurred, or 2001's present CALCulate1 calculation.

Example

```
{ TP6 setup commands called before here }
Set2001Calc1MXB( 10, 5);
{ Reading from SENSe:DATA? is 10.4 }
WriteLn( Calc1_2001Q); { Outputs 109 }
```

9.6 procedure Set2001Calc2(Format : String);

Description

Sets up the Model 2001 Data Buffer's Format and activates the CALCulate2 subsystem.

Parameters

Format

NONE
MEAN
SDEViation (Standard Deviation)
MAXimum
MINimum
PKPK (Peak to Peak)

Example

```
var
A : String;
{ TP6 setup commands called before here }
Set2001Calc2( 'MAXIMUM'); {calculate maximum of data buffer}
{ Data buffer contains 1.122, 10.211, 10.2222 }
A := Calc_2001Q; { A := '10.2222' }
```

9.7 procedure Calc2_2001(State : Boolean);

Description

Sets the state of the Model 2001's CALCulate2 Subsystem.

Parameters

State

FALSE := Turn CALC2 Off
TRUE := Turn CALC2 On

Example

```
{ TP6 setup commands called before here }
Calc2_2001( FALSE); { Turn off CALC2 System }
```

9.8 function Set2001Calc2Q : String;

Description

Queries the Model 2001 for its CALCulate2 subsystem's Format and State.

Returns

'Error!' if a TimeOutError occurred, or
Query 1: CALCulate 2 Format (see Set2001Calc2 above)
Query 2: CALCulate 2 State (0 or 1)

Use ParseQuery to separate return string into components.

Example

```
var
A, Format, Calc2State : String;
{ TP6 setup commands called before here }
A := Set2001Calc2Q; { Get 2001 CALC2 settings }
Format := ParseQuery(A, 1); { Get CALC2's Format }
Calc2State := ParseQuery(A, 2); { Get CALC2's State }
```

9.9 function Calc2_2001Q : String;

Description

Queries the Model 2001 for the result of the present CALCulate2 subsystem's calculation.

Returns

'Error!' if a TimeOutError occurred, or the result of the present CALC2 calculation.

Example

```
var
A : String;
{ TP6 setup commands called before here }
A := Calc2_2001Q; { Get 2001 CALC2's last result }
```

9.10 procedure Set2001Calc3(Upper1, Lower1, Upper2, Lower2: Double);

Description

Sets the Model 2001's CALCulate3 subsystems Limit Testing Hi/Lo Limits.

Parameters

Upper1, Upper2, Lower1, Lower
2 -9.999999999e35 to +9.999999999e35
MAXIMUM, MINIMUM, or DEFAULT

Example

```
{ TP6 setup commands called before here }
{ Setup Limit Testing Ranges }
{ Limit 1 Range: -100.56 to -10.5 or 10.5 to 100.56 }
{ Limit 2 Range: <-100.56 or >100.56 }
{ Pass Range: -10.5 to 10.5 }
Set2001Calc3( 10.5, -10.5, 100.56, -100.56);
```

9.11 function Set2001Calc3Q : String;

Description

Queries the Model 2001 for its CALCulate3 subsystems Limit Testing Hi/Lo Limits.

Returns

'Error!' if a TimeOutError occurred, or

Query 1: limit testing Upper Limit 1
Query 2: limit testing Lower Limit 1
Query 3: limit testing Upper Limit 2
Query 4: limit testing Lower Limit 2

Use ParseQuery to separate return string into components.

Example

```
var
A : String;
Upper1, Lower1, Upper2, Lower2 : Double; code : Integer;
{ TP6 setup commands called before here }
A := Set2001Calc3Q; { Get 2001 CALC3 settings }
VAL(ParseQuery(A, 1), Upper1, code); { Get Upper Limit 1 }
VAL(ParseQuery(A, 2), Lower1, code); { Get Lower Limit 1 }
VAL(ParseQuery(A, 3), Upper2, code); { Get Upper Limit 2 }
VAL(ParseQuery(A, 4), Lower2, code); { Get Lower Limit 2 }
```

9.12 procedure Calc3_2001(State : Boolean);

Description

Sets the state of the Model 2001's CALCulate3 Subsystem.

Parameters

State

FALSE := Turn CALC3 Off
TRUE := Turn CALC3 On

Example

```
{ TP6 setup commands called before here }
Calc3_2001(TRUE); { Turn on CALC3 System }
```

9.13 procedure Set2001Calc3Dig(Du1, Dl1, Du2, Dl2 : Integer);

Description

Sets the Model 2001's CALCulate3's Limit Testing Hi/Lo Digital Output values.

Parameters

Du1 Value (0-15) to place on digital output to indicate Upper 1 Limit Reached
Dl1 Value (0-15) to place on digital output to indicate Lower 1 Limit Reached

Du2 Value (0-15) to place on digital output to indicate Upper 2 Limit Reached
Dl2 Value (0-15) to place on digital output to indicate Lower 2 Limit Reached

Example

```
{ TP6 setup commands called before here }
{ Set 1st bit of digital output if Upper Limit 1 Reached }
{ Set 2nd bit of digital output if Lower Limit 1 Reached }
{ Set 3rd bit of digital output if Upper Limit 2 Reached }
{ Set 4th bit of digital output if Lower Limit 2 Reached }
Set2001Calc3Dig( 1, 2, 4, 8);
```

9.14 function Set2001Calc3DigQ : String;

Description

Queries the Model 2001 for its CALCulate3 subsystem's Limit Testing Hi/Lo Digital Output Values.

Returns

'Error!' if a TimeOutError occurred, or
Query 1: digital output Upper Limit 1 value
Query 2: digital output Lower Limit 1 value
Query 3: digital output Upper Limit 2 value
Query 4: digital output Lower Limit 2 value

Use ParseQuery to separate return string into components.

Example

```
var
A : String; Du1, Dl1, Du2, Dl2, code : Integer;
{ TP6 setup commands called before here }
A := Set2001Calc3DigQ; { Get 2001 CALC3 Digital I/O settings }
VAL(ParseQuery(A, 1), Du1, code); { Get UpLim1 DigI/O Value }
VAL(ParseQuery(A, 2), Dl1, code); { Get LoLim1 DigI/O Value }
VAL(ParseQuery(A, 3), Du2, code); { Get UpLim2 DigI/O Value }
VAL(ParseQuery(A, 4), Dl2, code); { Get LoLim2 DigI/O Value }
```

9.15 function Calc3_2001Q : String;

Description

Queries the Model 2001 for the result of the present CALCulate3 subsystem's calculation.

Returns

'Error!' if a TimeOutError occurred, or
Query 1: Upper/Lower Limit Range 1 result (0=PASS or 1=FAIL)
Query 2: Upper/Lower Limit Range 2 result (0=PASS or 1=FAIL)

Use ParseQuery to separate return string into components.

Example

```
var
A, Test1, Test2 : String;
{ TP6 setup commands called before here }
A := Calc3_2001Q; { Get 2001 CALC3 Limit Test Results }
Test1 := ParseQuery(A, 1); { Get Upper/Lower 1 Test Result }
Test2 := ParseQuery(A, 2); { Get Upper/Lower 2 Test Result }
```

Appendix A
Model 2001 / 7001 Global Variables

The following is a list of all the global variables used by the Model 2001/7001 Support Software for Borland Turbo Pascal 6.0:

KI2001	An integer global variable that contains the Model 2001's IEEE address.
KI7001	An integer global variable that contains the Model 7001's IEEE address.
TimeOutError	A Boolean variable set to TRUE when a timeout error occurs.
brd0	An integer variable that holds the location of board 'GPIB0' for National Instruments.
Nat2001Addr	Integer value of the actual Model 2001 IEEE Address for use with National Instruments board commands.
Nat7001Addr	Integer value of the actual Model 7001 IEEE Address for use with National Instruments board commands.
IeeeIn	Integer value of the file handle used for IOTech Driver 488 input operations.
IeeeOut	Integer value of the file handle used for IOTech Driver 488 output operations.
Resp	String used for the input string on all IEEE-488 read functions.
IeeeInterface	contains a number from 1 to 4 indicating the IEEE interface as defined by the constants below:

IEEECEC	(1) Capital Equipment Corp. or KPC488.2
IEEEIOTECH	(2) IOTech Driver 488
IEEENATIONALOLD	(3) National Instruments NI-488 (C.10 or C.11)
IEEENATIONALNEW	(4) NI-488 (C.12 or greater) or NI-488.2

OutOfRange	A Boolena variable set to TRUE whenever a parameter given to a Set Model 2001/7001 routine is out of range.

Appendix B
Model 2001 / 7001 Constants

The following Model 2001/7001 Support Software CONSTants are defined to make using the support software easier. The use of these constants are defined below and in the appropriate group of routines that use them

NOTE

These constants also support the Model 7002 Switch System.

B.1 Function Constants

DCV (1)	DC Voltage
ACV (2)	AC Voltage
DCA (3)	DC Current
ACA (4)	AC Current
R2 (5)	2-wire Resistance
R4 (6)	4-wire Resistance
FREQ (7)	Frequency
TEMP (8)	Temperature

B.2 Status Model Constants

For use with Stat2001, Stat2001Q, Stat7001, Stat7001Q:

OPERATION (1)	Operation Event
TRIGGER1 (2)	Trigger Event
ARM1 (3)	Arm Event
SEQUENCE (4)	Sequence Event
QUESTIONABLE (5)	Questionable Event
MEASUREMENT (6)	Measurement Event

Constants for use with *STB?, *SRE, *SRE?, and serial poll:

MSB (1)	Measurement Summary Bit (2001 Only)
EAV (4)	Error Available
QSB (8)	Questionable Summary Bit(in 7001 only for SCPI)
MAV (16)	Message Available
ESB (32)	Event Summary Bit
RQS1 (64)	Request for Service (Serial Poll)
MSS (64)	Master Summary Status (Status Byte)
OSB (128)	Operation Summary Bit

Constants for use with *ESR?, *ESE, and *ESE?:

OPC (1)	Operation Complete
RQC (2)	Request Control (not used in 2001/7001)
QYE (4)	Query Error
DDE (8)	Device Specific Error
EXE (16)	Execution Error
CME (32)	Command Error
URQ (64)	User Request
PON (128)	Power On

Constants for use with the Operation Event Condition Register, Positive Transition Filter, Negative Transition Filter, Status Register, and Status Enable Register:

CAL (1)	Calibrating (2001 only)
SET (2)	Settling
TRIG (32)	Waiting for Trigger
ARM (64)	Waiting for Arm
CALC (512)	Calculating (2001 only)
SCAN (1024)	Scanning (7001 only)
IDLE (1024)	in IDLE layer (2001 only)

Constants for use with the Trigger Event Condition Register, Positive Transition Filter, Negative Transition Filter, Status Register, and Status Enable Register:

SEQ1 (2)	Model 2001/7001 is in the Trigger Layer of Sequence 1.

Constants for use with the Arm Event Condition Register, Positive Transition Filter, Negative Transition Filter, Status Register, and Status Enable Register:

SEQ1 (2)	Model 2001/7001 is in the ARM Layer of Sequence 1.

Constants for use with the Sequence Event Condition Register, Positive Transition Filter, Negative Transition Filter, Status Register, and Status Enable Register:

LAY1 (2)	Model 2001 or 7001 is in the Arm Layer 1 of Sequence 1.
LAY2 (4)	Model 2001 or 7001 is in the Arm Layer 2 of Sequence 1.

Constants for use with the Model 2001 Questionable Event Condition Register, Positive Transition Filter, Negative Transition Filter, Status Register, and Status Enable Register:

QTEMP (16)	Questionable Calibration Process
QCAL (128)	Questionable Calibration Process
WARN (16384)	Command Warning

Constants for use with the Model 2001 Measurement Event Condition Register, Positive Transition Filter, Negative Transition Filter, Status Register, and Status Enable Register:

ROF (1)	Reading OverFlow
LL1 (2)	Low Limit 1
HL1 (4)	High Limit 1
LL2 (8)	Low Limit 2
HL2 (16)	High Limit 2
RAV (32)	Reading AVailable
BAV (128)	Buffer AVailable
BHF (256)	Buffer Half Full
BFL (512)	Buffer FuLl
BPT (2048)	Buffer PreTrigger event occurred

B.3 Automatic Constants

Use these constants with the Model 2001 Auto measurement functions, like AutoDCV, or any Model 2001/7001 function that requires an ON or OFF state.

NO (-1)	Don't set automatic parameter.
TOFF (+0)	Turn off automatic parameter.
TON (+1)	Turn on automatic parameter.
ONCE (+2)	Set auto parameter for the next measurement.

B.4 2001 Minimum and Maximum Sense Constants

MINACA := -2.1	MAXACA := 2.1
MINDCA := -2.1	MAXDCA := 2.1
MINACV := -775	MAXACV := 775
MINDCV := -1100	MAXDCV := 1100
MINR2 := 0	MAXR2 := 1050000000
MINR4 := 0	MAXR4 := 21000
MINFREQ := 0	MAXFREQ := 1500000
MINTEMPF := -328	MAXTEMPF := 3310
MINTEMPC := -200	MAXTEMPC := 1821
MINTEMPK := 73	MAXTEMPK := 2094
MINSIMF := 32	MAXSIMF := 122
MINSIMC := 0	MAXSIMC := 50
MINSIMK := 273	MAXSIMK := 323
MINTCO := -.09999	MAXTCO := .09999
MINALPHA := 0	MAXAPLHA := .01
MINBETA := 0	MAXBETA := 1
MINDELTA := 0	MAXDELTA := 5
MINRZERO := 0	MAXRZERO := 1000
MINDIG := 3.5	MAXDIG := 8.4999
MINNPLC := .01	MAXNPLC := 10

MINAVGCNT := 1	MAXAVGCNT := 100
MINNTOL := 1	MAXNTOL := 100
MINPWIN := .1	MAXPWIN := 9.9
MAXTEMPDIG := 7.4999	MAXFREQDIG := 5.4999
MAXFREQV := 1000	MINFREQV := -1000
MAXFREQI := 1	MINFREQI := -1
MAXFREQPERC := .6	MINRANGE := 0

B.5 2001 Minimum and Maximum Calculate Constants

MINCALC := -9.999999E+35	MAXCALC := 9.999999E+35
MINPERC := -9.999999E+35	MAXPERC := 9.999999E+35
MINMMF := -9.999999E+20	MAXMMF := 9.999999E+20
MINMBF := -9.999999E+30	MAXMBF := 9.999999E+30
MINDIGITAL := 0	MAXDIGITAL := 15

B.6 2001/7001 Scanning Minimum and Maximum Constants

MINCOUNT := 1	MAXCOUNT := 99999
MINDELAY := 0	MAXDELAY := 999999.999
MINLINE := 1	MAXLINE := 6
MINCHANNEL := 1	MAXCHANNEL := 10
MINTIMER := .001	MAXTIMER := 999999.999

B.7 Read2001BufferS and Read2001BufferD Constants

BUFRDGS := 1
BUFTIMESTAMP := 2
BUFCHANNELS := 3
BUFSTATUS := 4
BUFUNITS := 5

B.8 Examples

To set Model 2001 to generate a Service Request on EAV or MAV use the following:

SRE2001(EAV OR MAV);

To check for MAV being set in the Status Byte:

```
val(Q2001('*STB?'), poll, code);
if (poll and MAV) then WriteLn( 'MAV is set.');
```

Appendix C
Borland Turbo Pascal 6.0 File Names:

C.1 2001/7001 IEEE Interface Independent Files

20017001.PAS - 2001/7001 Header File w/CONST & Routine declares 27001 .PAS - 2001/7001 Routines:

2001/7001 Display and General routines:

Disp2001, Disp7001, KIDisp, KINoDisp, NoDisp2001, NoDisp7001, Check2001Val, KIQ, KISend

2001 Function & Buffer routines:

ACA2001, ACV2001, BufferSize2001, DCA2001, DCV2001, FREQ2001, Get2001Func, Get2001FuncHeader, Get2001Rdg, Get2001SaveRecallSize, Get2001Units, Get2RES2001, Get4RES2001, GetACA2001, GetACV2001, GetDCA2001, GetDCV2001, GetFREQ2001, GetTEMP2001, Hit2001Key, Hit2001Key1, RES2.2001, RES4.2001, Set2001Function, Take2001BufferReadings, TEMP2001

2001 Automatic Configuration routines:

Auto2001, Auto2001ACA, Auto2001ACAQ, Auto2001ACV, Auto2001ACVQ, Auto2001DCA, Auto2001DCAQ, Auto2001DCV, Auto2001DCVQ, Auto2001Q, Auto2001R2, Auto2001R2Q, Auto2001R4, Auto2001R4Q, Auto2001T, Auto2001TQ

2001 CALCulate Subsytem routines:

Calc1_2001, Calc1_2001Q, Calc2_2001, Calc2_2001Q, Calc3_2001, Calc3_2001Q, Set2001Calc1MXB, Set2001Calc1PERC, Set2001Calc1Q, Set2001Calc2, Set2001Calc2Q, Set2001Calc3, Set2001Calc3Dig, Set2001Calc3DigQ, Set2001Calc3Q

2001 configuration routines:

Set2001, Set2001ACA, Set2001ACAQ, Set2001ACV, Set2001ACVQ, Set2001DCA, Set2001DCAQ, Set2001DCV, Set2001DCVQ, Set2001F, Set2001FQ, Set2001Q, Set2001R2, Set2001R2Q, Set2001R4, Set2001R4Q, Set2001RTD, Set2001RTDQ, Set2001T, Set2001TC, Set2001TCQ, Set2001TQ

2001/7001 Status Model routines:

Clear2001, Clear7001, ESE2001, ESE7001, KIESE, KISRE, KIStat, KIStatQ, SRE2001, SRE7001, Stat2001, Stat2001Q, Stat7001, Stat7001Q

2001/7001 Trigger Model routines:

Arm2001, Arm2001Q, Arm7001, Arm7001Q, ArmTcon2001, ArmTcon2001Q, ArmTcon7001, ArmTcon7001Q, Close2001, Close2001Q, Close7001, Close7001Q, KIArm, KIArmQ, KIArmTcon, KI-ArmTconQ, KICloseQ, KITimers, KITimersQ, KITrig, KITrigQ, KITrigTcon, KITrigTconQ, Open2001, Open7001, Scan2001, Scan7001, Timers2001, Timers2001Q, Timers7001, Timers7001Q, Trig2001, Trig2001Q, Trig7001, Trig7001Q, TrigTcon2001, TrigTcon2001Q, TrigTcon7001, TrigTcon7001Q

GENRLQBX.PAS - 2001/7001 General Routines:

AutoGraphicsMode, IOTechAddr, ParseQuery, Str1, DataViewS, XYGraphS, DataViewD, XYGraphD, Abort, SetGraphicsMode, str_low, str_upr, str_ltrim, str_rtrim

MAKEFILE. - File for MAKE program to create various the Turbo Pascal Unit files.

C.2 Capital Equipment Corp. (CEC) IEEE-488 Interface Files

CEC .TPU - Turbo Pascal 6.0 CEC 2001/7001 library

CEC .PAS - Turbo Pascal 6.0 CEC 2001/7001 source

DataInvalid, Gpiberror, OPC2001, OPC7001, Poll2001, Poll7001, Q2001, Q7001, Send2001, Send7001, SendGET2001, SendGET2001, SendSDC2001, SendSDC7001, Wait2001RQS, Wait7001RQS, SetupIEEE, Read2001Buffer

CECIEEE .PAS - Modified CEC Turbo Pascal 6.0 Unit

CECIEEE .OBJ - Modified CEC Turbo Pascal 6.0 Link to IEEE488.LIB

ERRORDIS.CEC - CEC interface detection is disabled if this file exists

MAKECEC .BAT - Makes CEC.TPU

C.3 IOTech Driver 488 Interface Files

IOTECH .TPU - Turbo Pascal 6.0 IOTech 2001/7001 driver library

IOTECH .PAS - Turbo Pascal 6.0 IOTech 2001/7001 source

DataInvalid, Gpiberror, OPC2001, OPC7001, Poll2001, Poll7001, Q2001, Q7001, Send2001, Send7001, SendGET2001, SendGET2001, SendSDC2001, SendSDC7001, Wait2001RQS, Wait7001RQS, SetupIEEE, Read2001Buffer

IOTECHIO.PAS - Modified IOtech Turbo Pascal 6.0 Unit

MAKEIOT .BAT - Makes IOTECH.TPU

C.4 National Instruments NI-488 rev C.11 Files

NAT488_1.PAS - Turbo Pascal 6.0 old NI 2001/7001 source

DataInvalid, Gpiberror, OPC2001, OPC7001, Poll2001, Poll7001, Q2001, Q7001, Send2001, Send7001, SendGET2001, SendGET2001, SendSDC2001, SendSDC7001, Wait2001RQS, Wait7001RQS, SetupIEEE, Read2001Buffer

NAT488_1.TPU - Turbo Pascal 6.0 old NI 2001/7001 driver library

NI4881 .PAS - old NI Turbo Pascal Unit FIle

NI4881 .OBJ - old NI object file link to GPIB.COM

MAKENAT1.BAT - Makes NAT488_1.TPU

C.5 National Instruments NI-488 rev C.12(and newer) and NI-488.2 Files

NAT488_2.PAS - Turbo Pascal 6.0 new NI 2001/7001 source

DataInvalid, Gpiberror, OPC2001, OPC7001, Poll2001, Poll7001, Q2001, Q7001, Send2001, Send7001, SendGET2001, SendGET2001, SendSDC2001, SendSDC7001, Wait2001RQS, Wait7001RQS, SetupIEEE, Read2001Buffer

NAT488_2.TPU - Turbo Pascal 6.0 new NI 2001/7001 driver library

NI4882 .PAS - new NI Turbo Pascal Unit File

NI4882 .OBJ - New NI object file link to GPIB.COM

MAKENAT2.BAT - Makes NAT488_2.TPU

KEITHLEY

Test Instrumentation Group, Keithley Instruments, Inc. • 28775 Aurora Road • Cleveland, Ohio 44139 • (216) 248-0400 • Fax: (216) 248-6168

AUSTRIA:	**Keithley Instruments GesmbH** • Rosenhügelstrasse 12 • A-1120 Wien • 0222-804-6548 • Fax: 0222-804-3597
FRANCE:	**Keithley Instruments SARL** • 3 Allée des Garays • B.P. 60 • 91121 Palaiseau Cédex • 01-60-11-51-55 • Fax: 01-60-11-77-26
GERMANY:	**Keithley Instruments GmbH** • Landsberger Str. 65 • D-8034 Germering • 089-849307-0 • Fax: 089-84930759
GREAT BRITAIN:	**Keithley Instruments, Ltd.** • The Minster • 58 Portman Road • Reading, Berkshire RG3 1EA • 0734-575666 • Fax: 0734-596469
ITALY:	**Keithley Instruments SRL** • Viale S. Gimignano 38 • 20146 Milano • 02-48303008 • Fax: 02-48302274
JAPAN:	**Keithley Instruments Far East KK** • Sumiyoshi 24 Bldg., Room 201 • 2-24-2 Sumiyoshi-cho • Naka-ku, Yokohama 231 • 81-45-201-2246 • Fax: 81-45-201-2247
NETHERLANDS:	**Keithley Instruments BV** • Avelingen West 49 • 4202 MS Gorinchem • Postbus 559 • 4200 AN Gorinchem • 01830-35333 • Fax: 01830-30821
SWITZERLAND:	**Keithley Instruments SA** • Kriesbachstrasse 4 • 8600 Dübendorf • 01-821-9444 • Fax: 01-820-3081
TAIWAN:	**Keithley Instruments Taiwan** • 3rd Floor, Spring Plaza 6 • Section 3, Min Chuan East Road • Taipei, R.O.C. • 886-2-509-4465 • Fax: 886-2-509-4473