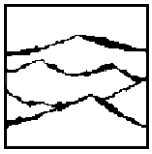


WAVECREST Corporation

DTS-2079
DTS-2077
DTS-2075

Test List Option Programming Guide

WAVECREST Corporation continually engages in research related to product improvement. New material, production methods, and design refinements are introduced into existing products without notice as a routine expression of that philosophy. For this reason, any current *WAVECREST* product may differ in some respect from its published description but will always equal or exceed the original design specifications unless otherwise stated.

***WAVECREST* Corporation**
A Technologies Company
7275 Bush Lake Road
Edina, Minnesota 55439
(612) 831-0030
(800) 733-7128
www.wavecrest.com

CONTENTS

SECTION 1 – TEST LIST DESCRIPTION

1-1 INTRODUCTION

The Test List feature is a set of IEEE-488 (GPIB) utility commands that work with the DTS to expedite repetitive execution of a fixed set of tests as is common in semiconductor testing.

Test List GPIB commands provide the user with the ability to define 1500 tests and a procedural script in memory on the DTS instrument and to then control test operation and receive measured statistics and pass/fail judgments.

Test List consists of the following groups of commands:

1. A group of commands to create:
   a) Test definitions
   b) Control script definitions
2. A group of commands to:
   a) Monitor and control the tests being run in the instrument
   b) Retrieve test results from the instrument.

Test List has been developed to support the user who has a large set of measurements that will be repeated numerous times.

Test List supports the return of results for each measurement or after a series of measurements have been taken.

Test List allows the evaluation and return of Pass/Fail judgments for each measurement or for a group of measurements. Each measurement statistic can have limits assigned to define a Pass/Fail. Test List summarizes Pass/Fail for groups as the logical “or” of the status masks of the measurements taken.

The user defines the parameters for individual measurements, limits for comparisons and how a measurement or group of measurements will return the results.

1-2 SAMPLE APPLICATION

A typical measurement sequence in summarized in the following pseudo-code example:

Test List script:

```
Start of script
    Group 1          Return          Pass/Fail per Group
        Measurement #1
                    #2
                    #3
                    #4
                    #5
    End of Group                     Send Pass/Fail

    Group 2          Return          Pass/Fail per Group
```

	End on Fail - (Go to end of Group return Fail)
Measurement #1	
#2	
#3	
End of Group	Send Pass/Fail
End of script	Send Halt message

The Test Engineer will write a test program to down-load the test measurement parameters and control the execution sequence. A 'C' language example would be:

```
main (   )
{
        Load_List ("filename");                 /* Load Test List definitions */
        Setup_DUT_Interface(Group_1);           /* Set up system for group #1 tests */
        Execute(Group_1);                       /* Run Test List group #1 */
        Wait_for_message_available (   );       /* Wait for response */
        Group_1_status = Read_Status (   );     /* Read response */
            If ( Group_1_Status == PASS )       /* If group #1 testing passed, save data */
                                                {
            Get_Group_Statistics(Group_1);      /* Read statistics back from DTS */
            Writefile(statistics);              /* Save to a file */
            exit (1);                           /* Exit program signaling "1" as grade */
            }
        Setup_DUT_Interface(Group_2);           /* Set up system for group #2 testing */
        Execute(Group_2);                       /* Run group #2 */
        Wait_for_message_available (   );       /* Wait for status response */
        Group_2_status = Read_Status (   );     /* Get the status */
            If(Group_2_status = = PASS)         /* If passed, get and save test data */
                                                {
            Get_Group_statistics(Group_2);      /* Get statistics for group #2 tests */
            Writefile(statistics);              /* Save data */
            exit( 2 );                          /* Exit program signaling grade 2 */
            }
        exit(0);                                /* Exit program and signal "0" as failure */
}       /* End of Main */
```

1-3 DATA STORAGE

Measured data is always stored on a per-test basis. These data are available for reading back to the host. The data remains in storage until a given test is executed again.

1-4 TEST TYPES

Test List supports 5 types of tests operations on the DTS:

1. Time measurement
2. Pulse find measurement
3. Strobe measurement
4. Strobe Pulse Find
5. DC measurement

1-5 DEBUG CAPABILITY

The Test List option has a debug mode that executes single tests and displays the results on the front panel of the DTS. (See section 2-6 Test Execution.)

Scripts and tests are checked for correctness when sent to the DTS with standard GPIB signals used to report problems.

1-6 TEST LIST SIZE SPECIFICATIONS

Test List will support 1500 different tests set-ups in the DTS at one time. This includes the data storage for the results of each test. The script portion of Test List has a limit of 8000 elements.

(This page intentionally left blank.)

SECTION 2 – GPIB COMMANDS

2-1 INTRODUCTION

This section describes the Test List feature of the DTS. A working knowledge of GPIB and the DTS systems is assumed. For a more detailed discussion of the DTS systems and their usage of GPIB please reference the Users Guide and IEEE-488 Interface Guide for the DTS.

In the presentation of the GPIB commands, only the capitalized portion of the key words is required. The lower case portion may be included but is not required. The form of GPIB responses is controlled by :SYStem:HEADer and :SYSTem:LONGform commands which cause the DTS to send or omit the :TL: header and lower-case characters, respectively.

The command syntax definitions are presented in Backus-Nauer form (BNF) with the following symbol definitions:

Symbol	Meaning
::=	One-way equation meaning “is comprised of” or “is an element of”
< >	An instance of a definition or a set of selections with one choice required
[ ]	An optional parameter
|	Selection list separator meaning “OR”

Syntax definitions are presented in a regular font. Examples of GPIB commands and responses are printed in bold face. Each Command must be sent in a single transmission. Some examples are shown on two lines with a carriage return as needed to fit some into the manual.

In this manual, discussion of binary and bit-wise values considers bit 1 to be the lowest bit with decimal value of 1.

2-2 COMMAND USE OVERVIEW

The interaction between a host computer and a DTS is performed in two phases: system set-up and test execution. In set-up, the test definitions and controlling script are loaded to the DTS. Then the host computer directs the DTS to execute testing sequences from the script or individual tests as required.

Test List tests are programmed into the DTS using macro-style commands describing the individual tests. There are five (5) types of tests:

Time measurement
Pulse find measurement
Strobe window measurement
Strobe Pulse Find
DC measurement

Test List requires a script to control the test execution sequence and the interaction between the host and the instrument. A sample script is shown in section 3-1. To define the script, there are six GPIB commands that load the different script elements: clear script, start group, end group, take measurement, conditional branch and script halt.

Script commands are assembled at the instrument in the order received. The script may not be subsequently edited or re-written. If any change needs to be made, the entire script, starting with a clear command, must be reloaded.

The Test List feature includes commands to start and stop execution and to retrieve status and statistical data.

2-3 SUMMARY OF COMMANDS

Test set-up commands:

:TL:TEST <1|2|3...> <test type> <testing mask> <macro>
:TL:LIMits <1|2|3...> <test type> <testing mask> <limit macro>

Script commands:

:TL:SCRipt CLear
:TL:SCRipt GRoup [<group index>] [<execution control parameter>]
:TL:SCRipt END [<group index>] <execution control parameter>
:TL:SCRipt MEASure <1|2|3...> [<execution control parameter>]
:TL:SCRipt COMPare <1|2|3...> [<execution control parameter>]
:TL:SCRipt CONDition <conditional control parameter>
:TL:SCRipt DELay <*time string*>
:TL:SCRipt HALT
:TL:WAIT

Execution and control commands:

:TL:EXE SETUP
:TL:EXE <ALL|GRoup <1|2|3...>| TEST <1|2|3...>>
:TL:CONTinue
:TL:HALT
:TL:DATa <GRoup <1|2|3...> <1|2|3|...>>|<TEST <1|2|3...>>
:TL:DATa TEST

Requested measured data is returned from the DTS in this format:

:TL:DATa <TIMe|PULSe|STRobe|STRobeLEVel|DC> <status code> <data packet>

The status response from the DTS for a measurement or at a group end is:

:TL:<PASS|FAIL|END> <1|2|3...> <1|2|3|...> <1|2|3|...> <status code>

For a script which runs to completion, the final response sent by the DTS is:

:TL:HALT

2-4 :TL:TEST - TEST SET-UP

All test types are defined with a single, variable parameter length command. Each test type is defined by a slash-delimited (“/”) macro similar to the :SYST:MACRO mechanism. For convenience, the at symbol, “@”, may be used in place of slashes to delimit fields. In this macro scheme, not all parameters need to be sent. A null field (“//”) causes that value to be copied from the preceding test of that type. The tests do not have to be loaded in any particular order and may be altered later. The command includes the storage index for the test:

Command syntax: :TL:TEST <1|2|3...> <TIMe|PULSe|STRobe|STRobeLEVel|DC> <*testing mask*> <*macro*>

Where *macro* is the appropriate definition macro for the test type. The macros include all the set-up and limit information required to perform and validate a measurement. Tests must be defined before the script can be loaded.

The *testing mask* is a hexadecimal number giving bit-wise flags for which test results will be compared to the supplied limits for PASS/FAIL judgments. The bits are used in the same order as the limits are supplied so that, for time measurements, bit 1 is average upper limit and bit 10 (decimal) is the samples lower limit. The error status is reported using the same convention with a "1" indicating an exceeded limit. To check all limits of a time test, a mask of 3FF hex (1111111111 binary) would be required. See the definitions of the individual test types for the way the bits are used for pulse, DC and strobe tests.

There is no response packet for the :TL:TEST command. Errors will be posted in the Event Status Register according to GPIB standard as noted in the IEEE-488 Interface Guide. The two errors applicable are command error (CME) which indicates a syntax error in the received command and execution error (EXE) which indicates that the command was not acceptable in this situation or the conditions defined were not compatible.

2-4.1 Time Measurement Macro

The *macro* for defining time measurement parameters is as follows:

macro ::= /<function>/<channel>/<arming mode>/<arming seq.>/<sample size>
/<start % stop %>/<start voltage>/<stop voltage>
/<start count>/<stop count>/<gating>
/<start ext. arm>/<stop ext. arm>
/<ARM1 edge>/<ARM1 voltage>
/<ARM2 edge>/<ARM2 voltage>/<filtering>
/<filter max>/<filter min>
/<avg upper>/<avg lower>/<jitter upper>/<jitter lower>
/<min upper>/<min lower>/<max upper>/<max lower>
/<samples upper>/<samples lower>

function	::=	<TPD++|TPD--|TPD+-|TPD-+|TT+|TT-|PW+|PW-|PER|FREQ>
channel	::=	<1|2>
arming mode	::=	<EXTernal|AUTOmatic>
arming seq	::=	<STARt|STOP|STARTFIRST|STOPFIRST>
sample size	::=	<1|2|3|...|1000000>
start % stop %	::=	<USER|10 90|90 10|50 50|20 80|80 20>
start voltage	::=	<*voltage string*>
stop voltage	::=	<*voltage string*>
start count	::=	<1|2|3|...|256>
stop count	::=	<1|2|3|...|256>
gating	::=	<ON|OFF>
start ext. arm	::=	<1|2>
stop ext. arm	::=	<1|2>
ARM1 edge	::=	< <RISe|POSitive>|<FALL|NEGative>>
ARM1 voltage	::=	<*voltage string*>
ARM2 edge	::=	< <RISe|POSitive>|<FALL|NEGative>>
ARM2 voltage	::=	<*voltage string*>
filtering	::=	<ON|OFF>
filter max	::=	<*time string*>
filter min	::=	<*time string*>

avg upper	::=	<*time string*>	(bit 1 in testing mask and error code)
avg lower	::=	<*time string*>	(bit 2)
jitter upper	::=	<*time string*>	(bit 3)
jitter lower	::=	<*time string*>	(bit 4)
max upper	::=	<*time string*>	(bit 5)
max lower	::=	<*time string*>	(bit 6)
min upper	::=	<*time string*>	(bit 7)
min lower	::=	<*time string*>	(bit 8)
samples upper	::=	<1|2|3|...|1000000>	(bit 9)
samples lower	::=	<1|2|3|...|1000000>	(bit 10)

These two definitions are used across all test types:

time string	::=	up to 15 character ASCII string of time or frequency including engineering unit notation of S, mS, uS, nS, pS and fS for time and Hz, KHz, MHz and GHz for frequency. Values without units are assumed to be in seconds or hertz. (e.g., 5.02nS or 150MHz)
voltage string	::=	seven character ASCII string (e.g., -0.3456)

Example:

:TL:TEST 1 TIM 7 /FREQ/1/AUTO/STOP/1000/USER/0.5/0.5/1/256/OFF ///////OFF//@22.5MHz/18MHz/150/

Setup the DTS to do a frequency measurement on channel 1 with auto arming, 1 start count and 256 stop counts with trip points at 0.5 volts for start and stop, no gating or filtering and 1000 samples. Limits are checked only for average outside of 22.5MHz-18MHz band and jitter of more than 150 Hz with the error mask of 7. In this command the "@" symbol is used to denote, to the user, the start of the limits.

2-4.2 Pulse Find Macro

For Pulse find operations, the *macro* is:

macro ::= /<channels>
/<waveform type>

/<CH1 max peak upper limit>/<CH1 max peak lower limit>	(bits 1&2)
/<CH1 min peak upper limit>/<CH1 min peak lower limit>	(bits 3&4)
/<CH2 max peak upper limit>/<CH2 max peak lower limit>	(bits 5&6)
/<CH2 min peak upper limit>/<CH2 min peak lower limit>	(bits 7&8)
/<CH1 VOH upper limit>/<CH1 VOH lower limit>	(bits 9&10)
/<CH1 VOL upper limit>/<CH1 VOL lower limit>	(bits 11&12)
/<CH2 VOH upper limit>/<CH2 VOH lower limit>	(bits 13&14)
/<CH2 VOL upper limit>/<CH2 VOL lower limit>	(bits 15&16)
/<CH1 VOH overshoot upper limit>/<CH1 VOH overshoot lower limit>	(bits 17&18)
/<CH1 VOL overshoot upper limit>/<CH1 VOL overshoot lower limit>	(bits 19&20)
/<CH2 VOH overshoot upper limit>/<CH2 VOH overshoot lower limit>	(bits 21&22)
/<CH2 VOL overshoot upper limit>/<CH2 VOL overshoot lower limit>	(bits 23&24)

channels ::= <1|2|BOTH>
waveform type ::= <PEAK|FLAT>

All limit parameters are of *voltage string* format:

voltage string ::= seven character ASCII string (e.g., -0.3456)

The “peak” parameters apply to the standard pulse find method. In FLAT mode, the stable portions of a square wave is sought in addition to the absolute extremes. The VOH and VOL limits apply to the flat spot levels. The VOH/VOL overshoot limits are for the difference between peak and flat levels. There are 24 comparisons possible which requires 24 bits of limit mask and error response codes.

Example: :**TL:TEST 2 PULS 99 /BOTH/PEAK@0.525///-0.2/0.525///-0.2/**

Set up a Pulse Find measurement as test #2 of peak voltages with pass/fail limits of 0.525 for upper maximum and -0.2 for lower minimum on both channels 1 and 2. The hexadecimal testing mask for this situation is 99. Again, the ‘@’ denotes the start of the limit section.

2-4.3 Strobe Measurement Macro

For Strobe tests, the *macro* parameters are:

macro ::= /<channel>/<arm>/ <arm edge>/<arm voltage>
/<min delay>/<max delay>/<delay increment>
/<avg voltage upper>/<avg voltage lower> (bits 1&2)
/<std. dev. upper>/<std. dev. lower> (bits 3&4)
/<max voltage upper>/<max voltage lower> (bits 5&6)
/<min voltage upper>/<min voltage lower> (bits 7&8)

channel ::= <1|2>
arm ::= <1|2>
arm edge ::= <<RISe|POSitive>|<FALL|NEGative>>
arm voltage ::= *voltage string*
min delay ::= <8000|8001|...|140000>
max delay ::= <8000|8001|...|140000>
delay incr. ::= <5|6|...|1000>

All limit values are of *voltage string* format:
voltage string ::= seven character ASCII string (e.g., -0.3456)

The delay values are integer picoseconds with the delays in the range 8ns through 140ns. The delay increment is from 5ps through 1ns.

Example:
:TL:TEST 1234 STR 93/1/2/NEG/-0.45/24500/45000/50@0.45/0.4///0.72///-0.25

Set up as test #1234 a strobe test over the range of 24.5nS to 45nS at 50 pS increments on channel 1 using ARM2 as the trigger source tripping at -0.45 volts with a falling edge. The error limits on the average are +0.45 to +0.4 with maximum upper limit of 0.72 and minimum lower limit of -0.25 which requires a testing mask of 93 hexadecimal.

2-4.4 Strobe Pulse Find Macro

For using strobe to find pulse levels, the *macro* parameters are:

macro ::= /<channel>/<arm>/
/<VOH start delay>/<VOH end delay>/< VOH delay increment>
/<VOL start delay>/<VOL end delay>/< VOL delay increment>
/<VOH voltage upper>/<VOH voltage lower> (bits 1&2)
/<VOH max voltage upper>/<VOH max voltage lower> (bits 3&4)
/<VOH min voltage upper>/<VOH min voltage lower> (bits 5&6)
/<VOL voltage upper>/<VOL voltage lower> (bits 7&8)
/<VOL max voltage upper>/<VOL max voltage lower> (bits 9&10)
/<VOL min voltage upper>/<VOL min voltage lower> (bits 11&12)

channel ::= <1|2>
arm ::= <1|2>
start delay ::= <8000|8001|...|140000>
end delay ::= <8000|8001|...|140000>
delay incr. ::= <5|6|...|1000>

All limit values are of *voltage string* format:
voltage string ::= seven character ASCII string (e.g., -0.3456)

The delay values are integer picoseconds with the delays in the range 8ns through 140ns. The delay increment is from 5ps through 1ns.

Example:
:TL:TEST 372 STRLEV 9E7/1/2/8400/8600/20/22000/22200/20@0.75/0.7 /0.8///0.65/-0.7/-0.75/-0.65///-0.8/

Make test #372 a strobed pulse find operation on channel 1 triggered from ARM2 with the upper level (VOH) found between 8.4nS and 8.6 nS with a 20pS increment and the lower level (VOL) found between 22nS and 22.2nS with a 20pS increment. The error mask of 9E7 causes the results the be verified with VOH check for a range of 0.7 to 0.75 volts with no point greater than 0.8 or less than 0.65. VOL is checked for a range of -0.7 to -0.75 volts with no points outside of -0.65 to -0.8.

2-4.5 DC Measurement Macro

For DC tests, the *macro* is:

macro ::= /<channel>/<upper limit>/<lower limit>

channel ::= <1|2>
upper limit ::= <*voltage string*> (bit 1)
lower limit ::= <*voltage string*> (bit 2)

Example: **:TL:TEST 517 DC 3 /2@0.01/-0.01**

Establish as test #517 a DC test of channel 2 with limits of +0.01 and -0.01 volts.

2-5 :TL:LIMITS - LIMIT SET-UP

The :TL:LIMit command allows the limits of a test to be set separately from the test parameters set using the :TL:TEST command. There is no additional functionality beyond that of the :TL:TEST command, this is merely a way to break down the macro into two parts. Use of :TL:LIMit assumes that the test parameters have been previously loaded for the test in question.

Command syntax:

:TL:LIMit <1|2|3...> <TIMe|PULSe|STRobe|STRobeLEVel|DC> <*testing mask*><*limit macro*>

Where *limit macro* is the appropriate definition macro for the test type. The macros include all limit information as described in appendices C and D for the four test types.

The *testing mask* is a hexadecimal number giving bit-wise flags for which test results will be compared to the supplied limits for PASS/FAIL judgments. The bits are used in the same order as the limits are supplied so that, for time measurements, bit 1 is average upper limit and bit 10 (decimal) is the samples lower limit. The error status is reported using the same convention with a "1" indicating an exceeded limit. To check all limits of a time test, a mask of 3FF hex (1111111111 binary) would be required. See the definitions of the individual test types for the way the bits are used for pulse, DC and strobe tests.

2-5.1 Time Limit Macro

The *limit macro* for defining time measurement limits is as follows:

limit macro ::= /<avg upper>/<avg lower>/<jitter upper>/<jitter lower>/<min upper>/
<min lower>/<max upper>/<max lower>/<samples upper>/<samples lower>

avg upper	::=	<*time string*>	(bit 1 in testing mask and error code)
avg lower	::=	<*time string*>	(bit 2)
jitter upper	::=	<*time string*>	(bit 3)
jitter lower	::=	<*time string*>	(bit 4)
max upper	::=	<*time string*>	(bit 5)
max lower	::=	<*time string*>	(bit 6)
min upper	::=	<*time string*>	(bit 7)
min lower	::=	<*time string*>	(bit 8)
samples upper	::=	<1|2|3|...|1000000>	(bit 9)
samples lower	::=	<1|2|3|...|1000000>	(bit 10)

These two definitions are used across all test types:

time string	::=	up to 15 character ASCII string of time or frequency including engineering unit notation of s, ms, us, ns, ps and fs for time and Hz, KHz, MHz and GHz for frequency. Values without units are assumed to be in seconds or hertz. (e.g., 5.02nS or 150MHz)
voltage string	::=	seven character ASCII string (e.g., -0.3456)

Example: **:TL:LIM 1 TIM 7 /22.5MHz/18MHz/150/**

Limits are checked only for average outside of 22.5MHz-18MHz band and jitter of more than 150Hz with the error mask of 7.

2-5.2 Pulse Find Limit Macro

For Pulse find operations, the *limit macro* is:
limit macro ::=

/<CH1 max peak upper limit>/<CH1 max peak lower limit>	(bits 1&2)
/<CH1 min peak upper limit>/<CH1 min peak lower limit>	(bits 3&4)
/<CH2 max peak upper limit>/<CH2 max peak lower limit>	(bits 5&6)
/<CH2 min peak upper limit>/<CH2 min peak lower limit>	(bits 7&8)
/<CH1 VOH upper limit>/<CH1 VOH lower limit>	(bits 9&10)
/<CH1 VOL upper limit>/<CH1 VOL lower limit>	(bits 11&12)
/<CH2 VOH upper limit>/<CH2 VOH lower limit>	(bits 13&14)
/<CH2 VOL upper limit>/<CH2 VOL lower limit>	(bits 15&16)
/<CH1 VOH overshoot upper limit>/<CH1 VOH overshoot lower limit>	(bits 17&18)
/<CH1 VOL overshoot upper limit>/<CH1 VOL overshoot lower limit>	(bits 19&20)
/<CH2 VOH overshoot upper limit>/<CH2 VOH overshoot lower limit>	(bits 21&22)
/<CH2 VOL overshoot upper limit>/<CH2 VOL overshoot lower limit>	(bits 23&24)

All limit parameters are of *voltage string* format:
voltage string ::= seven character ASCII string (e.g., -0.3456)

The “peak” parameters apply to the standard pulse find method. In FLAT mode, the stable portions of a square wave is sought in addition to the absolute extremes. The VOH and VOL limits apply to the flat spot levels. The VOH/VOL overshoot limits are for the difference between peak and flat levels. There are 24 comparisons possible which requires 24 bits of limit mask and error response codes.

Example: :**TL:LIM 2 PULS 99 /0.525///-0.2/0.525///-0.2/**

Set limits for test #2 of 0.525 for upper maximum and -0.2 for lower minimum on both channels 1 and 2. The hexadecimal testing mask for this situation is 99.

2-5.3 Strobe Measurement Limit Macro

For Strobe tests, the *limit macro* parameters are:
limit macro ::=

/<avg voltage upper>/<avg voltage lower>	(bits 1&2)
/<std. dev. upper>/<std. dev. lower>	(bits 3&4)
/<max voltage upper>/<max voltage lower>	(bits 5&6)
/<min voltage upper>/<min voltage lower>	(bits 7&8)

All limit values are of *voltage string* format:
voltage string ::= seven character ASCII string (e.g., -0.3456)

Example:
:TL:LIM 1234 STR 93 /0.45/0.4///0.72///-0.25

Set limits of test #1234. The limit range of the average value is +0.45 to +0.4 with maximum upper limit of 0.72 and minimum lower limit of -0.25 which requires a testing mask of 93 hexadecimal.

2-5.4 Strobe Pulse Find Limits Macro

For using strobe to find pulse levels, the *limit macro* parameters are:
limit macro ::=

/<VOH voltage upper>/<VOH voltage lower>	(bits 1&2)
/<VOH max voltage upper>/<VOH max voltage lower>	(bits 3&4)
/<VOH min voltage upper>/<VOH min voltage lower>	(bits 5&6)
/<VOL voltage upper>/<VOL voltage lower>	(bits 7&8)
/<VOL max voltage upper>/<VOL max voltage lower>	(bits 9&10)
/<VOL min voltage upper>/<VOL min voltage lower>	(bits 11&12)

All limit values are of *voltage string* format:
voltage string ::= seven character ASCII string (e.g., -0.3456)

Example:
:TL:LIM 372 STRLEV 9E7/0.75/0.7/0.8///0.65/-0.7/-0.75/-0.65///-0.8/

Change test #372, a strobed pulse find to have an error mask of 9E7 to have the results verified with VOH check for a range of 0.7 to 0.75 volts with no point greater than 0.8 or less than 0.65. VOL is checked for a range of -0.7 to -0.75 volts with no points outside of -0.65 to -0.8.

2-5.5 DC Measurement Limit Macro

For DC tests, the *limit macro* is:
limit macro ::= /<upper limit>/<lower limit>

upper limit	::= <*voltage string*>	(bit 1)
lower limit	::= <*voltage string*>	(bit 2)

Example: **:TL:LIM 517 DC 3 /0.01/-0.01**

Add to test #517 limits of +0.01 and -0.01 volts.

2-6 :TL:SCRIPT - SCRIPT DEFINITION

This command loads the DTS with the script to control the testing actions of the DTS. The simple constructs provide for accessing a separate resource of test set-ups loaded with :TL:TEST. Conditional evaluation can be done to provide script jumping or branching based on pass/fail evaluations of measurements.

Command syntax:
:TL:SCRipt <CL|GR|END|MEAS|COND|HALT> [<1|2|3...>][<control param>]

The script command is used to load the script lines for the DTS to follow. There are six possible script “language” elements: Start group, End group, take measurement, compare measurement, conditional branch and halt. These can be used to produce a reasonably complex script allowing the DTS to acquire, or not, data according to need as the characteristics of a situation become known. For example, if a DUT fails a criteria for grading the script can jump to the test group for the next lowest grade of device or success the script can jump to a section that will collect more detailed information.

There is no response packet for the :TL:SCRipt command. Errors will be posted in the Event Status Register according to GPIB standard as noted in the IEEE-488 Interface Guide. The command error bit (CME) indicates a syntax error in the received command. The execution error bit (EXE) indicates that the command was not acceptable in this situation or phase or operation or that the conditions defined were not compatible.

2-6.1 :TL:SCRIPT CLEAR - Clear Script Storage

Remove previous script information from storage. This is always the first command when loading a script.

Command syntax: :TL:SCRipt Clear

Example: **:TL:SCR CL**

2-6.2 :TL:SCRIPT HALT - Script Halt

Indicate end of all script items with no more accepted after its receipt. During script execution, the halt statement causes the response :TL:HALT to be sent and for the DTS to be set back to its normal operating mode.

Command syntax: :TL:SCRipt HALT

Example: **:TL:SCR HALT**

2-6.3 :TL:SCRipt GRoup - Start New Group

Start a measurement group. The execution control parameter is used as the default for that group's measurements and is defined below. Processing of the group start statement produces no GPIB transmissions. Errors will be reported through the standard GPIB status procedure. The group index is an optional numerical value to aid in documenting the groups. It is provided as a notation only. Groups are numbered by the DTS in the order received.

Command syntax: :TL:SCRipt GRoup [<group index>] [<execution control parameter>]

Example: **:TL:SCR GR STAT**

Start a new group with STATUS as the default status reporting mode for :TL:SCR MEAS commands in this group that do not have an execution control parameter.

Example: **:TL:SCR GR 321 ONF**

Start a new group with ONFail status reporting as the default for measurements in the group. This is group 321 to the user. The actual number is determined by the loading order of the groups in the script.

2-6.4 :TL:SCRipt END - End Group

End of a measurement group. The pass/fail status of a group is based on the bit-wise “OR” of all the measurements taken since the group started. During test list execution, This status will be returned according to the execution control parameter and may used for conditional branch evaluation. The group index allows the user to enter a numerical notation regarding the terminated group. The group index does not in any way affect the actual number of the group.

Command syntax: :TL:SCRipt END [<group index>] <execution control parameter>

Example: **:TL:SCR END 765 STAT**

End the currently open group and have the overall status reported when executed. This is group “765” to the user.

2-6.4.1 Execution Control Parameters

These parameters specify an action to be taken for :TL:SCRipt: COND, :TL:SCRipt MEAS and :TL:SCRipt END statements. Using these options, the host computer is informed of test results only when necessary.

<execution control parameter> ::= <STATus|NOSTatus|ONPass|ONFail|PASS|FAIL>

STATus - Send status
Pass or fail will be sent after test/group.

NOSTatus - Return no status
Nothing sent to host for this test/group.

ONPass - Return on Pass
Send status to host on pass status only.

ONFail - Return on Fail
Send status to host on fail status only.

PASS - Force Pass
Test/group treated as pass in any case. Status is sent to host.

FAIL - Force Fail
Test/group considered failure in any case. Status is sent to host.

2-6.5 :TL:SCRipt MEASure - Take Measurement

The script measurement statement includes the index of the test to be executed and an ptional parameter for status response. The execution control parameter may be carried from the group start command.

Command syntax: :TL:SCRipt MEASure <1|2|3...> [<execution control parameter>]

Example: **:TL:SCR MEAS 1345 NOSTAT**

Perform test #1345 and return no status to the host computer.

2-6.6 :TL:SCRipt COMPare - Compare Measurement Results

The compare measurement statement does not have the DTS take any new data. The statement prompts the DTS to compare the measurement results from a preceding test with the test limits as defined for the indicated test. The comparison is done with the most recently completed test of the same type, DC, strobe, pulse or time. The control parameter is optional with the group start parameter used if none is provided.

Command syntax: :TL:SCRipt COMPare <1|2|3...> [<execution control parameter>]

Example: **:TL:SCRipt COMP 734 ONP**

Compare the limits of test #734 to the existing results with status returned if the test passes.

2-6.7 :TL:SCRipt DELay - Time Delay

A time delay can be included in script execution through the :TL:SCR DEL command. The delay resolution is 50mS with a maximum delay of 1000 seconds.

Command syntax: :TL:SCRipt DELay <time string>

Example: **:TL:SCR DEL 0.5**

The script execution will pause for 1/2 second when encountering this statement.

2-6.8 :TL:SCR WAIT <time string> - ATE to DTS SYNC (Parallel)

This command is similar to :TL:SCR DEL <time string> except it waits for a signal from the parallel port or for time string seconds, whichever comes first, rather than delaying for <time string>.

A wait for parallel signal can be included in script execution through the TL:SCR WAIT command. The delay resolution is approximately 50ms with a maximum delay of 1000 seconds unless the delay is 0 or not supplied, in which case there is no time out, i.e., waits until parallel signal is received.

Script execution is suspended until signal received over parallel port or until the number of seconds specified in delay has elapsed.

Command syntax- :TL:SCR WAIT <time string>

Examples:	`:TL:SCR WAIT 0`	Wait unconditionally for parallel port signal.
	`:TL:SCR WAIT 120.0`	Wait for parallel signal but no more than two minutes.
Default:	`time string = 0.0`	

The following signals are used:

BUSY* Pin 1 - Parallel Port 25 Pin Connector
Pin 26 - Board Flatcable 26 Pin Connector
Internal Signal Name: LPT Strobe*

The signal used as the DTS BUSY* line output signal is the LPT - Strobe* output signal on pin 26 of the board flatcable connector.

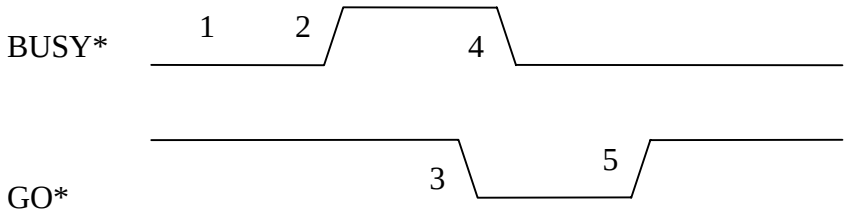
This signal is generated by Bit 0 of PORT - C and is inverted at the output. Thus, writing a 0 to this bit sets the cable line high.

GO* Pin 10 - Parallel Port 25 Pin Connector
Pin 8 - Board Flatcable 26 Pin Connector
Internal Signal Name: LPT-Acknowledge*

The signal used as a DTS GO* Input signal is the LPT - Acknowledge* input signal on pin 8 of the board flatcable connector. This signal can be read as bit 4 of PORT - B. A low input on this line sets LPT-Interrupt to the DTS processor.

OPERATION

The following describes the meaning of the two signals and the relationship between them.



Signals and Polarities on the parallel cable:

BUSY* is an output from the DTS and is Low TRUE

GO* is an input to the DTS and is Low TRUE

Note that BUSY* (LPT - Strobe* signal internally) is driven through an inverter so the DTS sees BUSY as being High TRUE. Thus it sets BUSY by writing a 1 to bit 0 of PORT-C.

Events are numbered on the waveforms above and are described as follows:

1. DTS remains busy from startup until it reaches the point where it waits for an external GO*.

   During this time, the DTS will ignore the GO* line in case it is pulled low by the external system. Note, however, that if the GO* line remains low, the DTS will never raise the BUSY*. See the next item.

2. When the DTS reaches the point where it is waiting for an external GO*, and if the GO* Input Line is high, it will set the PPI - WAITING (Waiting For Interrupt) flag, clear the BUSY Line and wait for an interrupt.

3. On seeing not busy from the DTS, and when it is ready to initiate the next step of the operation, the external system pulls GO* low generating an interrupt in the DTS.

4. On seeing GO* go low, the DTS sets the BUSY line, clears the PPI-WAITING flag, and sets the PPI-GO flag.

   The PPI-GO flag will be seen in the idle loop, which will continue the program execution.

5. It is not necessary for the external system to hold GO* low until it sees the DTS go busy. It only needs to hold it low long enough for it to activate the set input of a FF. The time required for this depends on the characteristics of the cable.

INTERFACE SIGNAL RULES

If the GO* line is pulled low by the external system while the BUSY* line is low, the DTS will not see the signal.

The minimum width of the GO* signal (low going) must be only enough to activate the SET input of the LPT - Interrupt Flip Flop. This requires only 20 - 30 nanoseconds at the Flip Flop but the requirement at the external system depends on the length and other characteristics of the cable.

The GO* signal line can remain low as long as the external system wants to hold it low but the DTS will not raise the BUSY* (go not busy) until the GO* is raised. If timeout is enabled, a timeout will occur if the Go remains low too long value.

A reasonable time for the external system to raise the GO* signal would be when it sees the DTS set the BUSY* signal.

When the external system sets the GO* signal, the DTS will set the BUSY* signal within 20 milliseconds.

2-6.9 :TL:SCRipt CONDition - Conditional Branch

Conditional branching statements can be applied to individual measurements or to groups. The condition of the preceding measurement or group is used to make the decision.

Command syntax: :TL:SCRipt CONDition <conditional control parameter>
[<GRoup|MEASure> <END|1|2|3...>]

Example: **:TL:SCR COND FJUMP GR 6**

If the preceding test failed, jump to group #6 and start executing there.

Example: **:TL:SCR COND PSKIP**

If the preceding test passed, skip the next script item.

2-6.9.1 Conditional Control Parameters

These parameters are used with the :TL:SCRipt CONDiditional command to provide branching capabilities in the Test List script.

<Conditional control parameter> ::= <PJUMP|FJUMP|PSKIP|FSKIP|GOEND>

PJUMP <GRoup|MEASure><index|END> - Jump on Pass
Jump to indicated group or measurement inside of current group if the preceding measurement status was pass. Can not jump to lower measurement index. For tests, following a :TL:MEAS command, END causes a jump to the end of the group.

FJUMP <GRoup|MEASure><index|END> - Jump on Fail
Jump to indicated group or measurement inside of current group if the preceding measurement status was fail. Can not jump to lower measurement index. For tests, the index may be END to jump to the end of the group.

PSKIP - Skip next on Pass
If preceding measurement status was pass, skip over the next measurement/group.

FSKIP - Skip next on Fail
Skip next measurement/group if preceding measurement/group was a failure.

GOEND - Go to End of List
For measurements, go to group end. For groups, terminate execution of script.

2-7 :TL:EXE SETUP - Test Execution Configuration

Before a :TL:EXE TEST command is issued, the :TL:EXE SETUP command may be sent to configure the operation of the execution. If this command is not issued, the execution performs as described in section 2-7.1.

Command syntax- :TL:EXE SETUP <PF|ST|NO>/<ON|OFF>/<AV><JI><MN><MX><SM>/<A|F>

Example- `:TL:EXE SETUP /ST//AVJISM/F`
Return statistics; do not change quick setup mode; return Average, Jitter, and Samples as floating point.

The first parameter determines what is returned when the test has finished. If PF is selected, the pass/fail results will be returned. If ST is selected the statistical data will be returned. When NO is selected, nothing will be returned and the trigger bit will need to be checked to determine if the test has finished.

The second parameter enables or disables quick setup. When this parameter is OFF, the test executes at the normal speed. When this parameter is ON the test executes slightly faster. This is because the current DTS setup is not saved before execution or restored afterwards. That means an Interface Clear or a front panel clear should be performed before any front panel readings are taken when this parameter is set to ON.

The third parameter selects which of the statistical measurements to return. Any combination may be used. Data is always returned in AV JI MN MX SM order.

The last parameter selects the format of the returned data. In ASCII format the data is returned as before, and in float format the data is returned as a binary floating point value.

SAMPLE SESSION

This example demonstrates the setup command. Review Section 3 of the Test List User's Guide for examples of the other test list commands.

```
:TL:EXE SETUP /ST/OFF/AVJI/A
:TL:TEST 1 TIM 0/TPD++//AUTO/STOP/60/50 50///1/1/
:TL:TEST 2 TIM 0/TT+/1/AUTO/STOP/60/10 90///1/1/
```

At this point, the above tests are defined and downloaded to the DTS.

Host transmissions	DTS transmissions
`:TL:EXE TEST 1`	
	`TIM 0 -7.7616374e-012 +7.3153515e-012`
`:TL:EXE SETUP ///AV`	
`:TL:EXE TEST 1`	
	`TIM 0 -1.6815186e-011`
`:TL:EXE SETUP /PF`	
`:TL:EXE TEST 1`	
	`PASS 0 0 1 0`

2-7.1 :TL:EXE - Test Execution

After downloading the tests and the script, test execution can commence. There are four commands to operate the test list feature: start, halt, continue and retrieve data. Once started, the DTS will process script commands, taking measurements and making limit comparisons until required to return data or status to the controlling host computer. After sending the packet, the DTS will wait for host direction to possibly pass back test results and then to continue the script or to halt.

The test list script may de executed either from the beginning or one group at a time. The tests themselves may be run singly with measured results optionally shown on the DTS front panel (except for strobe tests).

Command syntax: :TL:EXE <ALL|GRoup <1|2|3...>| <TEST|SHOW> <1|2|3...> >

Example: **:TL:EXE ALL**

Start script execution with group 1 and continue until halted by the host or by reaching the terminal script halt statement.

Example: **:TL:EXE GR 6**

Run script group #6 and halt execution at group end.

Example: **:TL:EXE TEST 7**

Have the DTS make the measurement defined in test #7.

Example: **:TL:EXE SHOW 376**

Make the measurement for test #376 and have the results put on the front panel.

The DTS performs the selected tests and sends back, pass, fail and completion events with the status response packet.

Response syntax: :TL:<PASS|FAIL|END> <1|2|3...> <1|2|3|...> <1|2|3|...> <status code>, where the three numerical fields specify the group, measurement and test indices, respectively.

Example: **:TL:PASS 2 4 7 0**

This response indicates that in group #2, the fourth measurement is test #7 and it passed.

Example: **:TL:END 6 9 473 79**

At the end of group #6, the status code is 79 (indicating a failure in that group) with the 9 measurements taken and the last being test #473.

For the TEST and SHOW options, there is always a status response sent. When executing the script with GRoup and ALL, status responses are made according by the execution control parameters of the script elements.

After the DTS gives a status response (e.g., :TL:PASS) it waits for further direction from the host. There are three options: to halt, to retrieve data or to continue testing.

2-8 :TL:CONTINUE - CONTINUE SCRIPT

Script processing pauses when the DTS sends status results. The continue command resumes script execution with the next script item.

Command syntax: :TL:CONTinue

Example: **:TL:CONT**

2-9 :TL:HALT - HALT SCRIPT

To halt script execution and return the DTS to normal operation:

Command syntax: :TL:HALT

Example: **:TL:HALT**

2-10 :TL:DATA - REQUEST TEST RESULTS

If the host needs more specific information, the data command is used to retrieve data associated with either a measurement in a group or a specific test:

Command syntax: :TL:DATa <GRoup <1|2|3...><1|2|3|...>>|<TEST <1|2|3...>>

Example: **:TL:DAT TEST 764**

Have the DTS send the information for test #764.

The data is then transmitted with the type of test and the resultant testing status. The measured data is sent in the same order as the limits. The host may retrieve results before issuing a halt or continue command.

Response syntax: :TL:DATa <TIMe|STRobe|STRobeLEVel|PULSe|DC> <status>
<data packet>

<status> ::= hexadecimal pass/fail testing status

For TIMe tests:
<data packet> ::= <average> <jitter> <maximum> <minimum> <samples>

For STRobe tests:
<data packet> ::= <average voltage> <Std. Dev.> <maximum> <minimum>

For STRobeLev tests:
<data packet> ::= <upper average> <upper max> <upper min>
<lower average> <lower max> <lower min>

For PULSe tests:
<data packet> ::= <CH1 upper peak> <CH1 lower peak>
<CH2 upper peak> <CH2 lower peak>
<CH1 upper flat> <CH1 lower flat>
<CH2 upper flat> <CH2 lower flat>

For DC tests:
<data packet> ::= <DC voltage>

Where all parameters are returned as ASCII-encoded floating point numbers. Values are returned in exponential format if very large or very small to reduce transmission time.

Example: **:TL:DAT DC 1 0.3823**

The test queried was a DC test with error status of 1 (upper limit exceeded) for a value of 0.3823 volts.

Example: **:TL:DAT TIM A2 +5.0567e-09 +6.1634e-12 +5.07123e-09 +5.0313e-09 100.0**

This time test found three fail conditions: lower average limit (value 2, bit 2), lower maximum limit (value 20, bit 6) and lower minimum limit (value 80, bit 8). The measured result was 5.0567ns with 6.16ps of jitter. The maximum sample value was 5.071ns with a minimum found of 5.0313ns. There were 100 samples in the measurement burst.

2-10.1 :TL:DATA - Request Multiple Test Results In ASCII Or Float

The host may now request the data for more than one test with DTS returning the data in ASCII or binary form.

Command syntax- :TL:DATa <Group <1|2|3…><1|2|3…>>|<TEST <1|2|3…>-<1|2|3…>/<A|F>>

Example- `:TL:DAT TEST 1-4`

Have the DTS return the data for tests 1 through 4 in ASCII.

Example- `:TL:DAT TEST 2/F`

Have the DTS return the data for test 2 in Float.

Example- `:TL:DAT TEST 2-3/A`

Have the DTS return the data for tests 2-3 in ASCII.

The data is then transmitted as requested. In ASCII format, the data is transmitted in packets as defined in Sec 2-10 of the Test List manual. In float format the data is returned in packets of binary data. The data type identifier is not sent. The format of the packet is as follows:

<status> ::= four byte hexadecimal pass/fail testing status

For TIMe tests:
<data packet> ::= five floating point numbers
<avg><jitter><max><min><samples>

For STRobe tests:
<data packet> ::= four floating point numbers
<avg><Std. Dev.><max><min>

For DC tests:
<data packet> ::= 1 floating point number
<DC voltage>

For PULSe tests:

<data packet> ::= eight floating point numbers
<CH1 upper peak> <CH1 lower peak>
<CH2 upper peak> <CH2 lower peak>
<CH1 upper flat> <CH1 lower flat>
<CH2 upper flat> <CH2 lower flat>

2-11 COM - COMMENT COMMAND

The text following a valid COM command is discarded by the DTS up to the end of the packet or the next command separator (';'). The comment command is a general-purpose command that presented here because of its usefulness if test scripts are written by hand.

Command syntax: COM [<text comment>]

Example: **COM this is a comment**

Example: **:TL:DAT TEST 764 ;COM send back data for test #764**

COM can be used in a compound command to notate individual lines.

(This page intentionally left blank.)

SECTION 3 - TEST LIST EXAMPLES

3-1 SAMPLE SESSION #1

The test list definition is loaded in two phases: individual test set-ups and the test script. The tests must be loaded before being referenced in the script. Notice the use of the comment command COM to add notations to the script.

```
COM This is a sample program
:TL:TEST 1 TIM 3FF/TPD++//START/100/USER/0.212/0.313/...
:TL:TEST 2 TIM 3FF/PW+/1/STOP//USER/0.412/0.43/...
:TL:TEST 3 DC 3/2/0.02/-0.02/0.01/-0.01/
:TL:TEST 4 TIM 3FF/TT-/2/START/200/USER/0.212/0.313/...
:TL:SCRipt CLear
:TL:SCRipt GR STATus
:TL:SCRipt MEASure 1
:TL:SCRipt MEASure 2
:TL:SCRipt MEASure 3;COM This is the DC test
:TL:SCRipt MEASure 4
:TL:SCRipt END STATus
:TL:SCRipt HALT
```

At this point there are four tests and one group defined. The Test list is run as follows:

Host transmissions	DTS transmissions	Comment
:TL:EXE GR 1		Start through group 1
	:TL:PASS 1 1 1 0	First test good
:TL:CONT		Go ahead
	:TL:FAIL 1 2 2 57	Failed in group 1 on measurement 2, which is test #2, on error 57
:TL:DAT GR 1 2		Get data for group 1, measurement 2
	:TL:DAT TIM 57 2567.56 2578.3...	Time data
:TL:CONT		Continue testing
	:TL:PASS 1 3 3 0	Passed test #3
:TL:CONT		Continue testing
	:TL:FAIL 1 4 4 35	Failed in group 1 on measurement 4 (test #4) with error #35

:TL:DAT TEST 4		Get data from test #4
	:TL:DAT TIM 35 234.5 245.6 ...	Time data sent
:TL:CONT		Continue testing
	:TL:END 1 4 4 77	Group end notice with cumulative error of 77
:TL:HALT		Reset DTS to normal operation.

3-2 SAMPLE SESSION #2

This example shows a more complex test with three script groups and 11 tests and includes the host program in "C" pseudo-code. The test procedure attempts to grade parts into three qualities: grade A, grade B and scrap. This is done by using two script groups with those devices passing group #1 considered "A" with those devices passing the second group considered grade "B." The third script group is used to collect data on those parts that failed the quality tests.

3-2.1 Pseudo-code Host Program

```
int main()
        {

/*
** Download test definitions and testing script to DTS
*/
        load_tests();
        load_script();

/*
** Repeat testing as long as parts are made available
*/
        while ( part_ready_for_test() )
                {

/*
** Configure tester for group 1 testing and have tests executed on DTS
*/
                setup_DUT_interface( Group_1 );
                DTS_execute( Group_1 );
                DTS_wait_for_message_available();
                group_1_status = DTS_read_status();

/*
** If group #1 failed, run group #2
*/
                if ( group_1_status == FAILED )
                        {
```

```
/*
** Set-up for and run group #2 tests
*/
                        setup_DUT_interface( Group_2 );
                        DTS_execute( Group_2 );
                        DTS_wait_for_message_available();
                        group_2_status = DTS_read_status();

/*
** If group #2 failed, run group #3 and save that data
*/
                        if ( group_2_status == FAILED )
                                {
                                DTS_execute( Group_3 );
                                DTS_read_statistics( Group_3 );
                                save_statistics();
                                }       /* If failed group 2 */
                        }       /* If failed group 1 */

/*
** Sort device based on test results
*/
                if ( group_1_status == PASSED )
                        bin( grade_A );
                else if ( group_2_status == PASSED )
                        bin( grade_B );
                else
                        bin( Scrap );

                }       /* End while part ready */

        }       /* End main */
```

3-2.2 Sample Test Definitions

To set up the tests, the *load_tests()* routine would send the following commands, which are in bold face, to the DTS:

Test 1: Frequency measurement on channel 1 with auto arming, 1 start count and 256 stop counts tripping at 1.5 volts for start and stop, no gating or filtering and 1000 samples in the burst. Limits are checked only for average outside of 20.5MHz-19.5MHz band and jitter of more than 150 Hz.

:TL:TEST 1 TIM 7 /FREQ/1/AUTO/STOP/USER/1.5/1.5/1000/1/256/OFF
////////OFF///20.5MHz/19.5MHz/150/

Test #2: Frequency measurement as before except for limits of 21.5MHz-18.5MHz on the average reading and jitter maximum limit of 300Hz.
:TL:TEST 2 TIM 7 ////////100//////////////21.5MHz/18.5MHz/300/

Test #3: Positive pulse-width measurement on channel 1 of 100 samples tripping at 1.5v for start and stop with average limits of 17ns-12ns and jitter limit of 30ps.
:TL:TEST 3 TIM 7 /PW+/1//////100//////////////17ps/12ps/3ps/

Test #4: Positive pulse-width with average limits of 25ns - 11ns and 50ps of jitter.
:**TL:TEST 4 TIM 7 ////////////////////25ns/11ns/50ps/**

Test #5: Propagation delay between rising edges of 10 samples with measurement limits of 80ns-0ns and 150ps for jitter.
:TL:TEST 5 TIM 7/TPD++////10////////////////80ns/0/150ps/

Test #6: Propagation delay between falling edges otherwise exactly as test #5.
:TL:TEST 6 TIM 7/TPD--/

Test #7: Pulse find of peak voltages with limits of 0.725 for upper maximum and -0.225 for lower minimum on channel 1 and channel 2.
:**TL:TEST 7 PULS 99 /PEAK/0.725///-0.225/0.725///-0.225/**

Test #8: Rising edge measurement from 20%-80% on channel 1 with arm on stop, 100 samples and no limits checked.
:TL:TEST 8 TIM 0/TT+/1//STOP/100/20 80/

Test #9: TT+ test exactly as #8 except for on channel 2.
:TL:TEST 9 TIM 0//2/

Test #10: Falling edge measurement on channel 1 of 100 samples and 80%-20% levels. No limits checked.
:TL:TEST 10 TIM 0/TT-/1////80 20/

Test #11: TT- like #10 except for on channel 2.
:TL:TEST 11 TIM 0//2/

3-2.3 Sample Script

The script is then sent down to define three test groups. Groups 1 and 2 are used to grade the DUT. If there is a failure in groups 1 and 2, group 3 is executed to collect archival data.

:TL:SCR CL	- Clear all existing groups
:TL:SCR GR NOST	- Start a group (#1) with no status return as default
:TL:SCR MEAS 7	- Measure with test #7 (Pulse find)
:TL:COND FJUMP END	- Jump to end of group if failed
:TL:SCR MEAS 1	- Measure with test #1 (Freq.)
:TL:COND FJUMP END	- Jump to end of group if failed
:TL:SCR MEAS 3	- Measure with test #3 (PW+)
:TL:COND FJUMP END	- Jump to end of group if failed
:TL:SCR MEAS 5	- Measure with test #5 (TPD++)
:TL:COND FJUMP END	- Jump to end of group if failed
:TL:SCR MEAS 6	- Measure with test #6 (TPD--)
:TL:SCR END STAT	- Send status for group
:TL:SCR GR NOST	- Start a group (#2) with no status return as default.
:TL:SCR MEAS 2	- Measure with test #2 (Freq.)
:TL:COND FJUMP END	- Jump to end of group if failed
:TL:SCR MEAS 4	- Measure with test #4 (PW+)
:TL:COND FJUMP END	- Jump to end of group if failed
:TL:SCR MEAS 5	- Measure with test #5 (TPD++)
:TL:COND FJUMP END	- Jump to end of group if failed
:TL:SCR MEAS 6	- Measure with test #6 (TPD--)
:TL:SCR END STAT	- Send status for group
:TL:SCR GR NOST	- Start a group (#3) with no status return as default
:TL:SCR MEAS 8	- Measure with test #8 (TT+ 1)
:TL:SCR MEAS 9	- Measure with test #9 (TT+ 2)
:TL:SCR MEAS 10	- Measure with test #10 (TT- 1)
:TL:SCR MEAS 11	- Measure with test #11 (TT- 2)
:TL:SCR END STAT	- Send status for group
:TL:SCR HALT	- Send HALT notice

After loading, there are three script groups accessing eleven test definitions. The host/DTS interaction could be from the following examples:

Host transmissions:	DTS transmissions:	Comment
:TL:EXE GR 1		Execute group #1.
	:TL:END 1 5 6 0	Group #1 passed after five measurements the last being of test #6.
:TL:HALT		Reset DTS to normal mode. The DUT is found to grade A.

Or, the device could fail grade A and be grade B.

Host transmissions:	DTS transmissions:	Comment
:TL:EXE GR 1		Execute group #1.
	:TL:END 1 4 5 1	Failed test #5 (the fourth measurement in the group) with error 1, average too high.
:TL:EXE GR 2		Execute group #2.
	:TL:END 2 4 6 0	Passed (error of 0) group #2 after completing 4 tests, the last being test #6.
:TL:HALT		Reset DTS to normal mode. The DUT is found to grade B.

Or the DUT could be scrap and fail both test groups with data retrieved on the TT+ and TT- measurements.

Host transmissions:	DTS transmissions:	Comment
:TL:EXE GR 1		Execute group #1.
	:TL:END 1 3 3 4	Failed test #3 (the third measurement in the group) with error 4, jitter too high.
:TL:EXE GR 2		Execute group #2.
	:TL:END 2 2 4 5	Failed (error of 5) group #2 on the second test, #4. The error was average too high and jitter too high.
:TL:EXE GR 3		Execute group #3.
	:TL:END 3 4 11 0	Completed group #3.
:TL:DAT 8		Request data for test #8.
	:TL:DAT TIMe 0 35.34e-012 13.34e-012... Data sent.	
:TL:DAT 9		Request data for test #9.
	:TL:DAT TIMe 0 62.07e-012 10.73e-012... Data sent.	
:TL:DAT 10		Request data for test #10.
	:TL:DAT TIMe 0 47.75e-012 9.46e-012... Data sent.	
:TL:DAT 11		Request data for test #11.
	:TL:DAT TIMe 0 72.83e-012 12.53e-012... Data sent.	
:TL:HALT		Reset DTS to normal mode. The DUT is found to be scrap.

SECTION 4 - DEMONSTRATION PROGRAMS

4-1 INTRODUCTION

There are two sample programs supplied with the Test List option to demonstrate its features. These programs operate under MS-DOS with the National Instruments GPIB driver. However, the supplied “C” source code is general enough to be useful on any hardware or software platform.

There are six files on the diskette:

tl_demo.exe
tl_demo.c
tl_file.exe
tl_file.c
trial1.tst
trial2.tst

4-2 TL_DEMO

The program tl_demo is representative of the way most customers will use the DTS Test List feature. The source code and MS-DOS executable program are supplied on an MS-DOS floppy disk. The tl_demo program is a self-contained demonstration of Test Lists. The code shows how to create the GPIB commands for test definitions from numerical values. It also includes a full-featured script that illustrates the Test List operations. The tl_demo program loads the DTS with the test definitions and the script and then runs the tests in three ways: one test at a time, one group at a time and the entire script from start to end.

4-3 TL_FILE

The tl_file program is a simple program that exercises the basic features of Test List. The program is able to load to the DTS the contents of a file of tests and a script on a line-by-line basis. The program can then execute the script in is entirety, run a single group or execute a range of tests. There are two sample files provided: trial1.tst and trial2.tst. The user can edit these files, or create new ones, and see what results occur. The menu for tl_file is:

Enter Option:

L)oad file	- Read file and send to DTS
R)un tests	- Run tests one-at-a-time
E)xe all	- Execute entire script
G)roup run	- Execute a single group
Q)uit	

The menu waits for a single character command followed by a carriage return or enter.

When L)oading a file, the entire file name including extension must be supplied. The file is then sent, one line at a time, to the DTS and simultaneously shown on the screen. If any errors are reported, they will be displayed on the screen.

The R)un option also requests the count of tests. The :TL:EXE SHOW command is then used to execute tests #1 through the supplied index.

The entire script is run with the :TL:EXE ALL command by selecting the E)xe option.

A single group can be executed using the G)roup run command. The program will ask for the index of the target group and then issues a :TL:EXE GR command.

When executing the script with the G)roup or R)un options, tl_file will prompt the DTS to continue the script execution until a HALT response is received. All GPIB traffic is displayed. Note that the DTS is set to use no headers on transmitted information and to use the short form of response text.

4-3.1 TRIAL1.TST Example

The first trial example is meant to demonstrate the execution control and conditional jump features of Test List. There are sixteen test used with a testing mask of zero for each meaning that no comparisons will be made and that each test will “pass.” It is then possible to predict DTS responses and trace the execution flow. Note the use of the COM GPIB command that allows comments to be placed in the script.

TRIAL1.TST contents:

```
COM Test file TRIAL1.TST
COM Define 16 tests, one of each basic DTS measurement type
:TL:TEST 1 TIM 0/TPD++/
:TL:TEST 2 TIM 0/PER/2/
:TL:TEST 3 TIM 0/FREQ/1/
:TL:TEST 4 TIM 0/TT+/1/
:TL:TEST 5 TIM 0/TT+/2/
:TL:TEST 6 TIM 0/TT-/1/
:TL:TEST 7 TIM 0/TT-/2/
:TL:TEST 8 TIM 0/PW+/1/
:TL:TEST 9 TIM 0/PW+/2/
:TL:TEST 10 TIM 0/PW-/1/
:TL:TEST 11 TIM 0/PW-/2/
:TL:TEST 12 TIM 0/PER/1/
:TL:TEST 13 TIM 0/TPD-+//
:TL:TEST 14 TIM 0/FREQ/2/
:TL:TEST 15 TIM 0/TPD+-/
:TL:TEST 16 TIM 0/TPD--/;COM end of tests
:TL:SCR CL              ;COM Clear all stored script commands
:TL:SCR GR 1 ONF        ;COM Start Group #1 with status returned if tests fail
:TL:SCR DEL 1.0         ;COM Delay one second
:TL:SCR MEAS 1          ;COM There are four tests #1 -> #4
:TL:SCR MEAS 2
:TL:SCR MEAS 3
:TL:SCR MEAS 4
```

```
:TL:SCR END 1 STAT         ;COM Send overall status
:TL:SCR COND PJUMP GR 3    ;COM Jump to Group #3 if PASSED otherwise run Group #2
:TL:SCR GR 2 NOST          ;COM Group #2 has one test
:TL:SCR MEAS 1             ;COM Return no status for test #1
:TL:SCR END 2 STAT         ;COM Send status at group end
:TL:SCR GR 3 ONF           ;COM Start Group #3 with status sent ONFAIL
:TL:SCR MEAS 5 PASS        ;COM Three tests have specific control parameters
:TL:SCR MEAS 6 FAIL        ;COM which take precedence over the ONF status
:TL:SCR MEAS 7 ONP
:TL:SCR MEAS 8
:TL:SCR END 3 STAT         ;COM Return PASS/FAIL at end of group
:TL:SCR COND PSKIP         ;COM Skip the next group (#4) if group #3 PASSED
:TL:SCR GR 4 STAT          ;COM Start Group #4 with status sent for each test
:TL:SCR MEAS 9             ;COM Execute test #9 with status sent
:TL:SCR COND PJUMP MEAS 3  ;COM If previous test PASSED, jump to the third
:TL:SCR MEAS 10            ;COM measurement command in this group, MEAS 11
:TL:SCR MEAS 11
:TL:SCR COND PSKIP         ;COM If previous test PASSED, skip next statement
:TL:SCR MEAS 12
:TL:SCR END 4 STAT         ;COM Send cumulative status for group #4
:TL:SCR GR 5 NOST          ;COM Start Group #5 which will execute 2, 3 or 4 tests
:TL:SCR MEAS 13            ;COM Execute test #13 with no status sent
:TL:SCR COND FJUMP MEAS 4  ;COM If test #13 failed, jump to the fourth
:TL:SCR MEAS 14            ;COM :TL:SCR MEAS which will test with #16
:TL:SCR MEAS 15
:TL:SCR COND FSKIP         ;COM If test #15 FAILED, skip the next statement
:TL:SCR MEAS 16
:TL:SCR END 5 STAT         ;COM Send status of group #5
:TL:SCR COND PJUMP GR END  ;COM If Group #5 PASSED, halt test with jump to end
:TL:SCR GR 6 NOST          ;COM Group #6 executes 1 test
:TL:SCR MEAS 11            ;COM Do not return status of test #11
:TL:SCR END 6 STAT         ;COM Return status at end of group
:TL:SCR HALT
```

The script defines six script groups. When script is executed from start to finish (:TL:EXE ALL) the DTS will make ten responses from executing four of the groups. With the DTS set for short form, no header responses, the system interaction is as follows:

Host transmissions:	DTS transmissions:	Comment:
:TL:EXE ALL		The DTS starts with group 1 and does tests 1 to 4 after delaying one second.
	END 1 4 4 0	Group 1 ended with no error.
:TL:CONT		Go ahead. The DTS then jumps past group 2 to execute group 3.
	PASS 3 1 5 0	Test #5 is the first measurement done in group 3. The PASS parameter forces return of a status of 0.
:TL:CONT		Proceed.
	FAIL 3 2 6 1	Test #6 is the second measurement in group 3. The FAIL parameter forces return of a status of 1.
:TL:CONT		Proceed.

	PASS 3 3 7 0	Test #7 is the third measurement in the group and the ONPass parameter causes the PASS status to be sent.
:TL:CONT		Proceed.
	END 3 4 8 1	At the end of group 3, the cumulative error of 1 is returned.
:TL:CONT		Proceed. Since group 3 status was fail, the PSKIP conditional is ignored.
	PASS 4 1 9 0	As the default execution control parameter for group 4 is STATus, the first measurement returns PASS.
:TL:CONT		Proceed. The DTS now does a jump to the third measurement on the COND PJUMP script item.
	PASS 4 3 11 0	The third measurement is test #11 and the resulting PASS is returned.
:TL:CONT		Proceed. The next item, COND PSKIP, causes the DTS to jump to the end of the group.
	END 4 3 11 0	Group 4 ended with status of zero.
:TL:CONT		Proceed to group 5.
	END 5 4 16 0	Group 5 ended with status of 0.
:TL:CONT		Proceed. The following script item, COND PJUMP GR END, causes the DTS to jump to the end of the groups, skipping group 6.
	HALT	At group end, the final halt is issued and the DTS is reset for normal operations.

4-3.2 TRIAL2.TST Example

The TRIAL2.TST demonstration is designed to be used with the 200mhz calibration signal connected as the inputs to CH1 and CH2. There are several measurements (DC, TPD++, PER, PW+) made each type with three sets of limits. The script then has three groups so that the DTS may be judged as to its level of calibration. I this example, the start of the limit section of each test definition is denoted by the ‘@’ symbol to aid readability.

The first measurement performed in Group #1 is a Pulse Find. If the voltages found are not between 0.5 and 1 volts (or -0.5 and -1) the script jumps to the end and the test halts. The balance of measurements in group 1 judge the DTS against limits of +/- 10 pS to the 200 MHz signal and a DC values of +/- 0.0005 volts. If group 1 passes, the DTS jumps to the end of the script and halts.

If any time or DC test fails in group 1, the script jumps to group 2 where the measurements are repeated, this time with limits of +/- 20ps and a DC limit of +/- 0.001 volts. If group 2 passes, the following group end jump will halt the script.

Likewise, if any group 2 test fails, the script immediately jumps to run group 3. All tests in group 3 are run. The DTS will respond with group end status and the FAILures of individual measurements.

TRIAL2.TST contents:

```
COM test file TRIAL2.TST
:TL:TEST 1 TIM 7/TPD++//AUTO/STOP/250/50 50///1/1/OFF///////OFF//
      @10000fs/-0.1ns/10ps/
:TL:TEST 2 TIM 7/PER/1/AUTO/STOP/250/50 50///1/2/OFF///////OFF//
      @5.01ns/4.99ns/10ps/
:TL:TEST 3 TIM 7/PW+/1/AUTO/STOP/250/50 50///1/1/OFF///////OFF//
      @2.51ns/2.49ns/10ps/
:TL:TEST 4 TIM 7/PER/2/AUTO/STOP/250/50 50///1/2/OFF///////OFF//
      @5.01ns/4.99ns/10ps/
:TL:TEST 5 TIM 7/PW+/2/AUTO/STOP/250/50 50///1/1/OFF///////OFF//
      @2.51ns/2.49ns/10ps/
:TL:TEST 6 TIM 7/TPD++//AUTO/STOP/250/50 50///1/1/OFF///////OFF// @20ps/-
      20ps/20ps/
:TL:TEST 7 TIM 7/PER/1/AUTO/STOP/250/50 50///1/2/OFF///////OFF//
      @5.02ns/4.98ns/20ps/
:TL:TEST 8 TIM 7/PW+/1/AUTO/STOP/250/50 50///1/1/OFF///////OFF//
      @2.52ns/2.48ns/20ps/
:TL:TEST 9 TIM 7/PER/2/AUTO/STOP/250/50 50///1/2/OFF///////OFF//
      @5.02ns/4.98ns/20ps/
:TL:TEST 10 TIM 7/PW+/2/AUTO/STOP/250/50 50///1/1/OFF///////OFF//
      @2.52ns/2.48ns/20ps/
:TL:TEST 11 TIM 7/TPD++//AUTO/STOP/250/50 50///1/1/OFF///////OFF// @30ps/-
      30ps/30ps/
:TL:TEST 12 TIM 7/PER/1/AUTO/STOP/250/50 50///1/2/OFF///////OFF//
      @5.03ns/4.97ns/30ps/
:TL:TEST 13 TIM 7/PW+/1/AUTO/STOP/250/50 50///1/1/OFF///////OFF//
      @2.53ns/2.47ns/30ps/
:TL:TEST 14 TIM 7/PER/2/AUTO/STOP/250/50 50///1/2/OFF///////OFF//
      @5.03ns/4.97ns/30ps/
:TL:TEST 15 TIM 7/PW+/2/AUTO/STOP/250/50 50///1/1/OFF///////OFF//
      @2.53ns/2.47ns/30ps/
:TL:TEST 16 DC 3/1@0.005/-0.005
:TL:TEST 17 DC 3/2@0.005/-0.005
:TL:TEST 18 DC 3/1@0.01/-0.01
:TL:TEST 19 DC 3/2@0.01/-0.01
:TL:TEST 20 DC 3/1@0.02/-0.02
:TL:TEST 21 DC 3/2@0.02/-0.02
:TL:TEST 22 PULSE 0/BOTH/PEAK
```

```
COM limits sent separately
:TL:LIM 22 PULSE FF/1.0/0.5/-0.5/-1.0/1.0/0.5/-0.5/-1.0/
:TL:SCR CL                  ;COM Start Group #1
:TL:SCR GR ONF
:TL:SCR MEAS 22
:TL:SCR COND FJUMP GR END
:TL:SCR MEAS 16
:TL:SCR COND FJUMP GR 2
:TL:SCR MEAS 17
:TL:SCR COND FJUMP GR 2
:TL:SCR MEAS 1
:TL:SCR COND FJUMP GR 2
:TL:SCR MEAS 2
:TL:SCR COND FJUMP GR 2
:TL:SCR MEAS 3
:TL:SCR COND FJUMP GR 2
:TL:SCR MEAS 4
:TL:SCR COND FJUMP GR 2
:TL:SCR MEAS 5
:TL:SCR END STAT
:TL:SCR COND PJUMP GR END
:TL:SCR GR ONF              ;COM Group #2
:TL:SCR MEAS 18
:TL:SCR COND FJUMP GR 3
:TL:SCR MEAS 19
:TL:SCR COND FJUMP GR 3
:TL:SCR MEAS 6
:TL:SCR COND FJUMP GR 3
:TL:SCR MEAS 7
:TL:SCR COND FJUMP GR 3
:TL:SCR MEAS 8
:TL:SCR COND FJUMP GR 3
:TL:SCR MEAS 9
:TL:SCR COND FJUMP GR 3
:TL:SCR MEAS 10
:TL:SCR END ONP
:TL:SCR COND PJUMP GR END
:TL:SCR GR ONF              ;COM Group #3
:TL:SCR MEAS 20
:TL:SCR MEAS 21
:TL:SCR MEAS 11
:TL:SCR MEAS 12
:TL:SCR MEAS 13
:TL:SCR MEAS 14
:TL:SCR MEAS 15
:TL:SCR END STAT
:TL:SCR HALT
```

4-3.3 TL_FILE source code

```
/*************************************************************/
/*
/*  tl_file.c - Version 1.0
/*
/*  10/11/94    GJL Demonstration program for Test List Feature
/*
/*
/*=========================================================== */
#include <stdlib.h>
#include <stdio.h>
#include <dts.h>

#define FALSE   0
#define TRUE    1

int err;

/*
** Print out command and error
*/

void    command_failure( int err, char * str )

{
    switch ( err )
        {

        case DTS_ERR_COMMAND_ERROR :
            printf( "\n Command error: %s", str );
            break;

        case DTS_ERR_EXECUTION_ERROR :
            printf( "\n Execution error: %s", str );
            break;

        case DTS_ERR_OPER_FAIL :
            printf( "\n Operation failed: %s", str );
            break;

        case DTS_ERR_BURST_FAIL :
            printf( "\n Burst failed: %s", str );
            break;
        default:
            printf( "\nERROR: %d %s ", err, str );
            break;
        }
}
```

```
/*
** Transmit the GPIB command and print out any resulting error
*/

void    send_or_error( char * str )

{
    printf( "\n %s", str );
    err = dts_send_488( str );
    if ( err )
        command_failure( err, str );
}

void load_from_file()

{
FILE * file;
char str[500], file_name[30];
int i;
    printf( "\n Enter file name:" );
    if ( scanf( "%s", file_name ) != 1 )
        {
        printf( "\n Bad file name" );
        return;
        }

    if ( ( file = fopen( file_name, "r" ) ) == 0 )
        {
        printf( "\n\n Failure to open file: %s", file_name );
        return;
        }
    err = 0;
    while ( err == 0 && fgets( str, 500, file ) != NULL )
        {
        i = strlen( str );
        while( i && ( str[ i - 1 ] == '\n' || str[ i - 1 ] ==
'\r' ) )
            {
            i--;
            str[ i ] = 0;
            }
        if ( i )
            send_or_error( str );
        }

    fclose( file );

}
```

```
void run_group()

{
int id, igroup;
char cmd[50], resp[ MAX_CHAR_FROM_DTS ], ch0;
int  run, grp;

    printf( "\n Enter group index:" );
    if ( scanf( "%d", &grp ) != 1 || grp < 1 )
        {
        printf( "\n Bad group index" );
        return;
        }

/*
** Clear any residual status and prepare for loop
*/
    send_or_error( "*CLS" );
    err = FALSE;
    run = TRUE;
    sprintf( cmd, ":TL:EXE GROUP %d", grp );

/*
** Execute each group one-at-a-time
*/
    while( run )
        {
        printf( "\n %s", cmd );

/*
** Send command and wait for response, bailing out on an error
*/
        err = dts_request_and_obtain_asc( cmd, resp );
        if ( err )
            {
            cmd[ 40 ] = 0;
            command_failure( err, cmd );
            return;
            }

        ch0 = resp[0];
        printf( "\n %s", resp );

/*
** If HALT was response, exit from loop
*/
        if ( ch0 == 'H' )
            run = FALSE;
```

```
/*
** If END was found, start the next group
*/
        else if ( ch0 == 'E' )
            {
            run = FALSE;
            }
/*
** Otherwise, send continue command to proceed through balance
of group
*/
        else
            strcpy( cmd, ":TL:CONT" );
        }
/*
** Send halt to reset DTS status
*/
    send_or_error( ":TL:HALT" );
}
void run_script()
{
int id, igroup;
char cmd[50], resp[ MAX_CHAR_FROM_DTS ], ch0;
int  run, grp;
/*
** Clear any residual status and prepare for loop
*/
    send_or_error( "*CLS" );
    err = FALSE;
    run = TRUE;
    sprintf( cmd, ":TL:EXE ALL" );      /* Have DTS execute
all scripts from first to last */
    while( run )
        {
        printf( "\n %s", cmd );
/*
** Send command and wait for response, bailing out on an error
*/
        err = dts_request_and_obtain_asc( cmd, resp );
        if ( err )
            {
            cmd[ 40 ] = 0;
            command_failure( err, cmd );
            return;
            }
```

```
        ch0 = resp[0];
        printf( "\n %s", resp );
/*
** If HALT received, exit from loop
*/
        if ( ch0 == 'H' )
            run = FALSE;
/*
** Otherwise, have DTS proceed with balance of scripts
*/
        strcpy( cmd, ":TL:CONT" );
        }
/*
** Reset DTS status
*/
    send_or_error( ":TL:HALT" );

}

void run_tests()

{
int id, itest;
char cmd[50], resp[ MAX_CHAR_FROM_DTS ], ch0;
int  run, test_cnt;
    printf( "\n Enter count of tests:" );
    if ( scanf( "%d", &test_cnt ) != 1 || test_cnt < 1 )
        {
        printf( "\n Bad group index" );
        return;
        }
/*
** Clear any residual status and prepare for loop
*/
    send_or_error( "*CLS" );
    err = FALSE;
    run = TRUE;
    itest = 1;
    sprintf( cmd, ":TL:EXE SHOW 1" );   /* Have DTS execute
                                           one test */
    while( itest <= test_cnt )
        {
        printf( "\n %s", cmd );
```

```
/*
** Send command and wait for response, bailing out on an error
*/
        err = dts_request_and_obtain_asc( cmd, resp );
        if ( err )
            {
            cmd[ 40 ] = 0;
            command_failure( err, cmd );
            return;
            }
        printf( "\n %s", resp );
        itest++;
        sprintf( cmd, ":TL:EXE SHOW %d", itest );
        }
/*
** Reset DTS status
*/
    send_or_error( ":TL:HALT" );
}
main()
{

char str[ MAX_CHAR_FROM_DTS ], ch;
int i, j;
    i = 0;
    if ( ( i = dts_init_488( "GPIB0", "DEV5" ) ) )
        {
        printf( "\n Failed GPIB connection %ld %d %x \n",
(long int) i, i, i );
        exit(-1);
        }
    do
        {
        printf( "\n\n Enter Option:");
        printf( "\n     L)oad file");
        printf( "\n     R)un tests");
        printf( "\n     E)xe all");
        printf( "\n     G)roup run");
        printf( "\n     Q)uit\n  ");
        do  {
            if ( scanf( "%c", &ch ) != 1 )
                exit(0);
            }
        while( ch == '\n' || ch == '\r' );
        ch = toupper( ch );
```

```
        switch ( ch )
            {
            case 'L' :
                load_from_file();
                break;

            case 'R' :
                run_tests();
                break;

            case 'E' :
                run_script();
                break;

            case 'G' :
                run_group();
                break;

            case 'Q' :
            default :
                exit(0);
                break;
            }
        }
    while ( TRUE );
    exit( 0 );
}
```

WAVECREST Corporation

World Headquarters:
7275 Bush Lake Road
Edina, MN 55439
(612) 831-0030
FAX: (612) 831-4474
Toll Free: 1-800-733-7128
www.wavecrestcorp.com

West Coast Office:
1735 Technology Drive, Suite 400
San Jose, CA 95110
(408) 436-9000
FAX: (408) 436-9001
1-800-821-2272

Europe Office:
Lilenthalalle 25
D-80939 Munchen
011-49-89-32225330
FAX: 011-49-89-32225333