

TestmATE 1.0r2

User manual

Project TestmATE
Author(s) Tjark van Dijk, Sander Pasveer
Date 8 November 2004
Revision 1.1

Tildesign b.v.
Micro Electronics

Krachtenveld 46
3893 CD Zeewolde
The Netherlands

Phone : +31 36 522 11 34
Fax : +31 36 522 13 14
E-mail : info@tildesign.nl
Web site : www.tildesign.nl

Contents

1	INTRODUCTION.....	5
1.1	INTENDED AUDIENCE	5
1.2	SOFTWARE CAPABILITIES.....	5
1.3	TYPOGRAPHY	5
1.4	EXAMPLES.....	5
1.5	HAVE A QUICK LOOK.....	6
1.6	SYSTEM REQUIREMENTS	6
2	INSTALLATION.....	7
2.1	INSTALL THE TESTMATE SOFTWARE.....	7
2.2	CONFIGURE THE KEITHLEY 2700 MULTI METER.....	7
2.3	COMMUNICATION TEST	8
2.4	SWITCHING THE KEITHLEY 2700 MULTI METER BACK TO LOCAL.....	8
3	TESTMATE MAIN DIALOG.....	9
3.1	MENU.....	9
3.1.1	File menu.....	9
3.1.2	View menu.....	9
3.1.3	Operation menu.....	10
3.1.4	Tools menu.....	10
3.1.5	Help menu	10
3.2	SPEED BAR.....	11
3.3	PARAMETER SECTION.....	11
3.4	OPTIONS SECTION.....	11
3.5	LOG LISTING	12
3.6	SCRIPT LOADER DIALOG	12
3.7	SCRIPT DEBUGGER DIALOG.....	13
3.8	SERIAL NUMBER LABEL WRITER DIALOG.....	15
4	HOW TO MAKE YOUR FIRST TESTMATE SCRIPT.....	16
4.1	SCRIPT 1: SHOW A POPUP AND EXIT	16
4.2	SCRIPT 2: RUN A SIMPLE PRODUCTION TEST SCRIPT REPEATEDLY WITH SERIAL NUMBERS	18
4.3	SCRIPT 3: LOOP BACK WITH LED'S	19
4.4	SCRIPT 4: DIODE CHARACTERISTIC.....	19
4.5	SCRIPT 5: CALIBRATION EXAMPLE	19
4.6	SCRIPT 6: CALIBRATION WITH LINEARITY TEST	20
4.7	SCRIPT 7&8: USING A USER DLL	20
4.8	EXAMPLE ERROR SCRIPTS	20
5	SCRIPTING LANGUAGE EXPLAINED	21
5.1	COLUMN DEPENDENCIES	21
5.2	ACTION COLLECTOR AND REDIRECTOR.....	21
5.3	GENERAL LAYOUT CONSIDERATIONS	21
5.4	ACTIONS.....	22
5.5	ARGUMENTS.....	22
5.6	TEST CONDITIONS	22
5.7	COMMENTS	23
5.8	STRING SYNTAX	23
5.9	FORMULA SYNTAX.....	24
5.10	VARIABLE USAGE	24
6	SCRIPTING LANGUAGE REFERENCE.....	25

6.1	ADDINSTRUMENT	25
6.2	CALC	26
6.2.1	<i>Prefixes that can be used in CALC</i>	26
6.2.2	<i>Mathematical operations that can be used in CALC</i>	27
6.3	CALL	29
6.4	CLOSE	29
6.5	CONT	29
6.6	DELAY	30
6.7	DLEXEC	30
6.8	END	30
6.9	ENDTEST	30
6.10	EXEC	31
6.11	HEADER	31
6.12	INPUT	32
6.13	JUMP	32
6.14	LOG	34
6.15	MEASURE	35
6.16	PAUSE	36
6.17	PRINT	36
6.18	RET	37
6.19	SELECT	37
6.20	SAVE	38
6.21	TEST	38
6.22	VAR	39
6.23	SPECIAL GLOBAL VARIABLES	39
APPENDIX A: MAKE THE 2700 SUITED FOR PRODUCTION TESTING		40
APPENDIX B: USER DLL'S		42
INDEX		43
BIBLIOGRAPHY		44

Revision history, copyright, licence

Version	Date	Changes
1.0	23-09-2004	First release
1.1	08-11-2004	Screenshots updated, new features explained, textual adjustments

Copyright

All content included in this document, such as text, graphics, logos, button icons, images, audio clips, video clips, digital downloads, data compilations, and software, is the property of Tildesign bv or its content suppliers and protected by United States and international copyright laws. The compilation of all content in this document is the exclusive property of Tildesign bv and protected by U.S. and international copyright laws. All software provided with this document is the property of Tildesign bv or its software suppliers and protected by United States and international copyright laws.

Trademarks

Keithley is a registered trademark of Keithley Instrument Inc. Both Microsoft and Excel are registered trademarks of Microsoft Corporation Inc.

License

The provided software of TetsmATE is a demo version, which may be distributed without charge. The demo version has several limitations, which are removed from the full product.

The full product will be sold on a "licence per seat" basis with online registration and activation. One licence can be activated three times each year to cover for maintenance purposes like hard disk crashes, system replacements etc.

1 Introduction

This document describes how to use the TestmATE software.

1.1 *Intended audience*

This document is written for those users that need to understand and write their own test scripts where the measurements and actuations are performed by a Keithley 2700 multi meter. This product is suited to perform production tests in an automated way. The term "production test" means in this context "a predefined sequence of actuations and measurements during which the measurement results need to conform to predefined limits in order to produce an 'OK' or 'NOT OK' result of the test".

1.2 *Software capabilities*

The software is designed to be versatile and configurable. Extensions and modifications can be made as desired.

TestmATE (as it is today) can communicate with

- a Keithley 2700 multi meter with a 7706 I/O board (we don't own other boards right now)
- a Peaktech programmable power supply.
- Other instruments through a user DLL. By adding DLL's other "instruments" can be added in the future.
- Microsoft Excel, to read the script and to output the results. The interfacing is "live" so it is possible to define a graph in Excel and have TestmATE fill in the values as the test progresses.

The scripting language enables you to "program" your test procedure. Apart from production tests, you can also "program" a data logger, which not only measures, but can change the outputs on certain moments as well. This kind of data acquisition is very useful for product verification in the engineering department. Data acquisition can be as fast as the 2700 can handle or as slow as you wish.

1.3 *Typography*

By use of different fonts and use of **bold** and *italic* different meanings are indicated in the text.

Plain text is printed like this

Literal scripting text is printed like this

Very important text is shown in a box like this

An important **word** is printed bold.

Remarks are printed in italic.

Cross references are printed in underlined italic.

Paragraph headers are bold and italic and use a larger font size.

We do this to provide the best service to you. As we are the designers of this product, we develop a kind of "product blindness". What is obvious to us, may not be so to the fresh and unbiased user.

If some part is really unclear, please notify us so we can help you and maybe improve the documentation.

1.4 *Examples*

Several scripts are supplied to serve as an example and to demonstrate certain behaviour. Some scripts demonstrate certain features like pop-ups where a user can make a selection. Other scripts let the 2700 behave as a data logger or measure the forward characteristic of a diode.

We also provided some scripts that are intentionally wrong to demonstrate the error handling behaviour.

1.5 Have a quick look

If you don't feel like reading this entire manual, have a quick look and the next chapter to see what the TestmATE program looks like.

Check out the most important controls like Load, Run and the Log list.

Then continue with chapter 4: "How to make your first TestmATE script".

Perform the installation and run the example scripts. If this product is suited for you, you should be getting curious enough to learn about the rest of the possibilities.

1.6 System requirements

To run TestmATE successfully you need:

- A personal computer with at least a Pentium I, 133 MHz, 16 MB of RAM, 10 MB of free disk space and at least one unused RS-232 (COM) port.
- Microsoft Windows, (95, 98, ME, NT, 2000 and XP).
- Microsoft Excel 97 or newer.
- Minimum screen resolution is 800x600. Recommended is 1024x768 or higher.
- Adobe Acrobat reader 4.0 or newer.
- A Keithley 2700 multi meter with a 7706 add-on card.
- An RS-232 cable to connect the multi meter to the PC.
- The TestmATE program.

Preliminary

2 Installation

2.1 Install the TestmATE software

Run setup.exe from the CD or from you're the directory where you copied the installation files.


The default install directory is C:\Program Files\Tildesign\TestmATE.

From that point we use some subdirectories

\bin	program executable and support files
\doc	manual
\examples	example scripts
\userdll	user DLL's
\drv	drivers

As a working directory these directories are created as well:

C:\TestmATE\results	a directory to store your results
C:\TestmATE\scripts	a directory to store your own scripts

You can start the program by clicking on START, Programs, TestmATE, TestmATE.exe or by clicking on the TestmATE icon  on your desktop.

2.2 Configure the Keithley 2700 multi meter

Connect the Keithley 2700 multi meter's RS-232 port with your PC's RS-232 port.

Connect the Keithley 2700 multi meter to the mains and switch it on.

The TestmATE communicates through an RS-232 interface, at 19200 baud without any flow control. Each protocol packet is terminated by a carriage return-linefeed (CR/LF).

How to configure the serial interface for the Keithley 2700 instrument?

- Switch the device on.
- Press the <shift> button and then <enter> (RS-232 setup)
- If the display says "RS 232: OFF", press the <right-arrow> key once and then toggle the ON/OFF state using the <range up> key. Press <enter> when done.
- Now the RS-232 interface is enabled.

- If the display says "BAUD: 19.2k", press <enter>. Otherwise press the <right-arrow> key once and use the <range up> key to select 19.2k. Press <enter> when done.
- Now the RS-232 interface will communicate at 19200 baud.

- If the display says "FLOW: NONE", press <enter>. Otherwise press the <right-arrow> key once and toggle the FLOW state so FLOW is turned off, using the <range up> key. Press <enter> when done.
Now the RS-232 interface will communicate at 19200 baud, without using flow control.

- If the display says "TX TERM: CRLF", press <enter>. Otherwise press the <right-arrow> key once and use the <range up> key to select CRLF. Press <enter> when done.

Now the RS-232 interface setup is complete! The instrument is ready to use.

2.3 Communication test

Start TestmATE and open 'C:\TestmATE\scripts\installtest.xlt'. Hit the F5 button to execute the script. See the next chapter for instructions about running scripts.

This script lets you select a COM port. After that the script tries to connect with the Keithley 2700 through the selected COM port and shows an instrument dialog when the connection was made successfully.


2.4 Switching the Keithley 2700 multi meter back to local

When a test procedure script ends, the Keithley 2700 multi meter is still in remote mode and cannot be operated from the front panel.

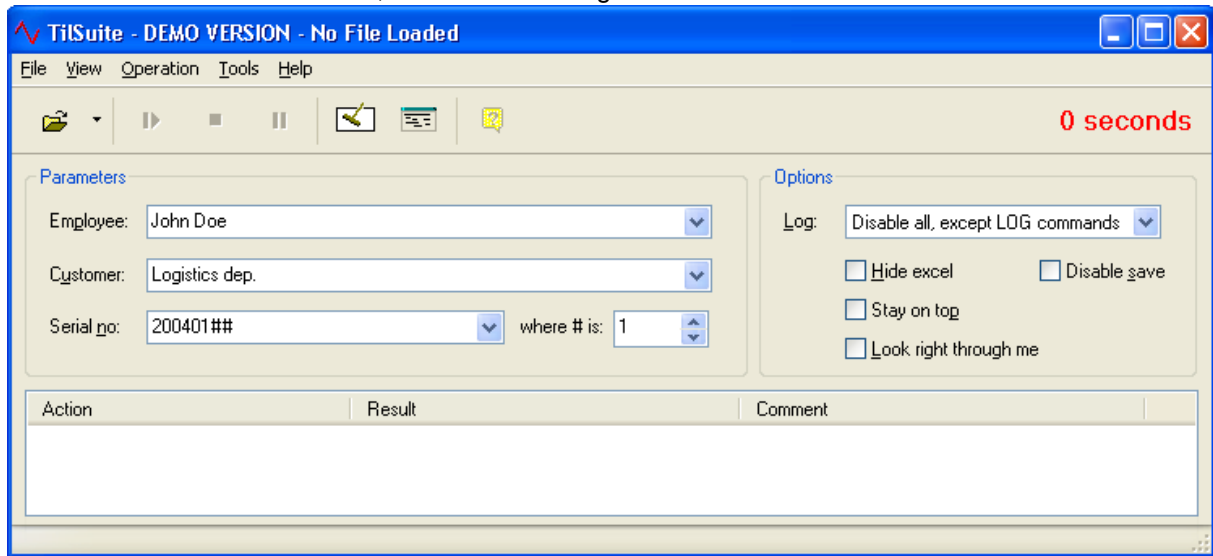
This condition can be ended by pressing the 'local' switch on the front panel of the Keithley 2700 multi meter to switch back to manual operation.

Preliminary

3 TestmATE main dialog

You can start the program by clicking on START | Programs | TestmATE | TestmATE or by clicking on the TestmATE icon  on your desktop.

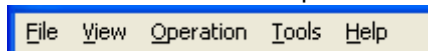
When TestmATE.exe is started, it shows the dialog below.



This dialog contains several areas that are explained below. Note the red time indicator, this shows the execution time of a single script cycle. The timer is reset at the (re-)start of execution of the script.

3.1 Menu

The menu looks like this picture:



From the menu you can select files, modify preferences and execute a script. Click on the menu item with your left mouse button and then click on the sub-item of your choice.

3.1.1 File menu

'File | Open test procedure' opens an Excel template file, which contains the script and an empty result sheet. Empty means in this case: formatted but without measured values.

'File | Close' closes the current script, but does not close the TestmATE program.

'File | Recent files' gives the user access to the most recently used file list. This can save a lot of searching for the correct script.

'File | Exit' closes Excel and exits the TestmATE program.

3.1.2 View menu

'View | Script loader' opens the script loader dialog. In this dialog messages are shown when the parser reads the script. If the parser encounters errors, an error message is shown.

For detailed information, please see chapter [3.6: Script loader dialog](#).

'View | Script debugger' opens the script debugger dialog. This dialog enables the user to debug a script by setting breakpoints, inspecting variables, reading from or writing to an instrument and evaluating formulas. Please see chapter [3.7: Script debugger dialog](#) for detailed information.

3.1.3 Operation menu

'Operation | Run' executes the script which was loaded with 'File | Open test procedure'.

'Operation | Stop' halts the execution upon completion of the current Action. In case of a DELAY or a PAUSE this can take considerable time.

When halted, the script is reset. When the script is run again, it will start at the first Action.

'Operation | Pause/Continue' halts or continues execution. When the script is halted, the current Action will always be completed. In case of a DELAY or a PAUSE this can take considerable time.

'Operation | Save test results to file' will save the Excel result workbook in a selectable directory. This can be useful when a script does not contain a SAVE action. This options is only available when script execution is either stopped or paused.

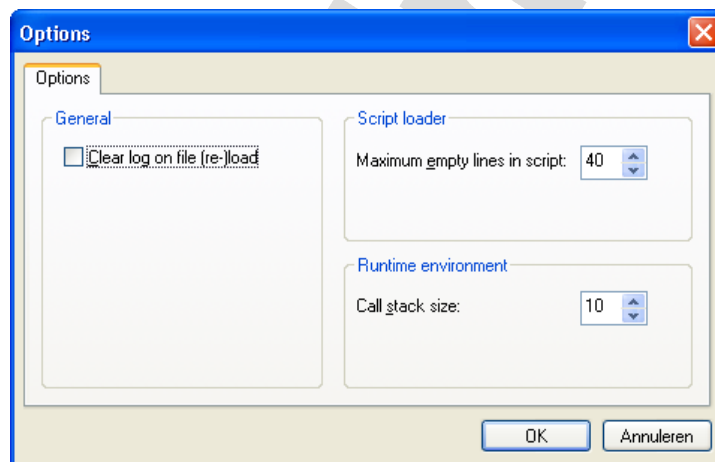
3.1.4 Tools menu

'Tools | Serial number label writer' will open the serial number generation dialog. This feature generates an Excel sheet from a predefined label template, which can be printed by a label printer. Please see [3.8: Serial number label writer dialog](#) for detailed information.

'Tools | Options' opens the options dialog showed below that allows the user to configure the some behavior of the TestmATE application.

At this time the following parameters can be set:

- 'Clear log on file (re-)load': clear the log window when the excel template is loaded.
- 'The maximum empty action lines' in a script: See [21: Action Collector and Redirector](#) for more information about the Action Collector.
- The 'Call stack size': See chapter [29: CALL](#) for more information about the call stack.



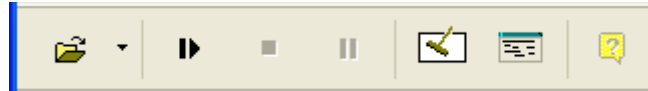
3.1.5 Help menu

'Help | Manual' opens this document.

'Help | About' shows revision information about TestmATE.

3.2 Speed bar

The speed bar looks like this picture:



The speed bar allows you to load and control execution of a script with a single mouse click.

For these functions the following accelerator keys are available:

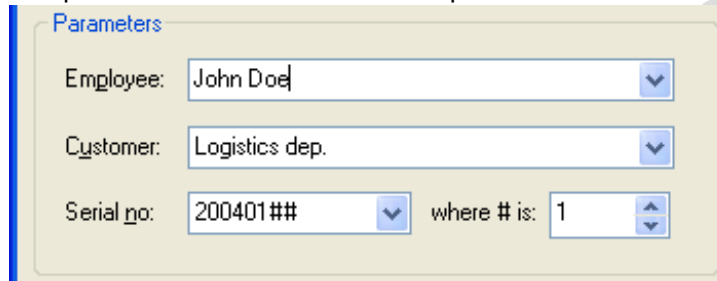
Ctrl + O	Opens a script template
F5	Reload and Run
Pause/Break	Pause / Continue
Ctrl + Pause/Break	Stop
F1	Help

Note that the open icon also contains a list of recently used files.

When the mouse pointer moves over a button, a 'tool tip' is shown to indicate the purpose of the button.

3.3 Parameter section

The parameter section looks like this picture:



The parameter fields are available to the script, so these values can be stored in the result sheet. If the desired Employee, Customer or Serial number format is not available in the drop-down list, a new item can be added by typing the new item in the edit field.

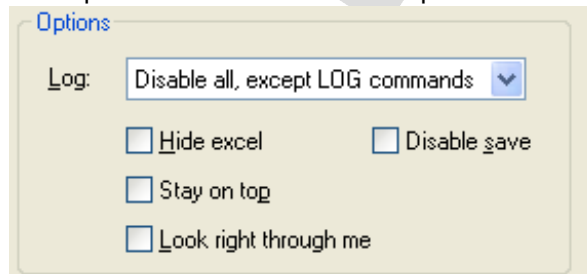
The serial number field is interpreted as plain text in which ##### is replaced by the number in the edit box. For instance the serial number string 'Serial number: ABQ-334-####-XXT' and number '154' is converted to this text: 'Serial number: ABQ-334-0154-XXT'.

Multiple chains of ### are allowed, so 'T#-I##-L###' would become 'T154-I154-L154'.

The number that replaces ### can be incremented by the script by using the `ENDTEST` action with `INCSERIALNR` as parameter. Please see chapters [6.9 ENDTEST](#) and [6.11: HEADER](#) for more details and examples.

3.4 Options section

The Options section looks like this picture:



Selecting the desired Log filter controls the output to the log listing.

When 'Hide excel' is checked, the Excel sheets will remain minimised and operate in the background.

When 'Stay on top' is checked, TestmATE will always remain visible.

When 'Look right through me' is checked, TestmATE becomes transparent. (Exists only on Windows XP)

When 'Disable save' is checked, the `SAVE` action is suppressed. Please see chapter [6.20: SAVE](#).

3.5 Log listing

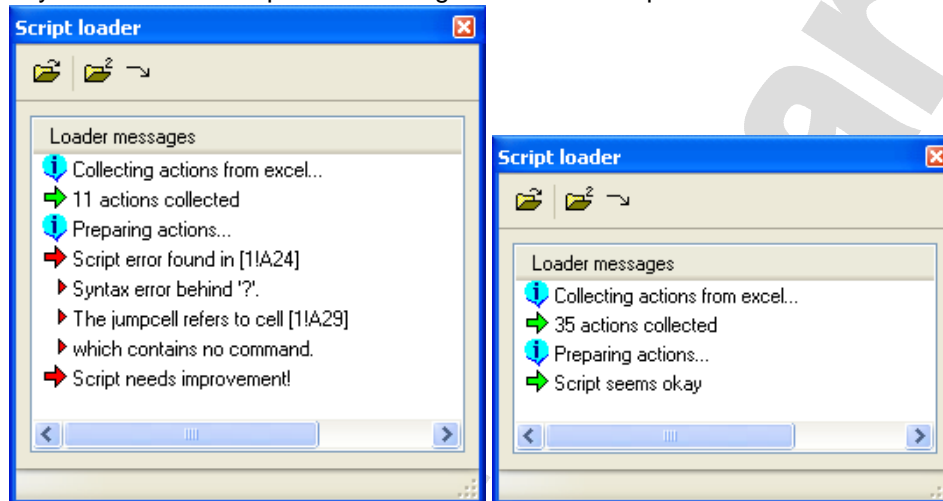
The log listing looks like this picture:

Action	Result	Comment

Messages from the script and from the command executor are output on the log list control. Please see chapter [6.14: LOG](#) for detailed information about the LOG command and log listing use.




3.6 Script loader dialog

This dialog can be opened from the main menu by clicking on 'View | Script loader' or with the shortcut key 'Ctrl + L'. The script loader dialog looks like these pictures:



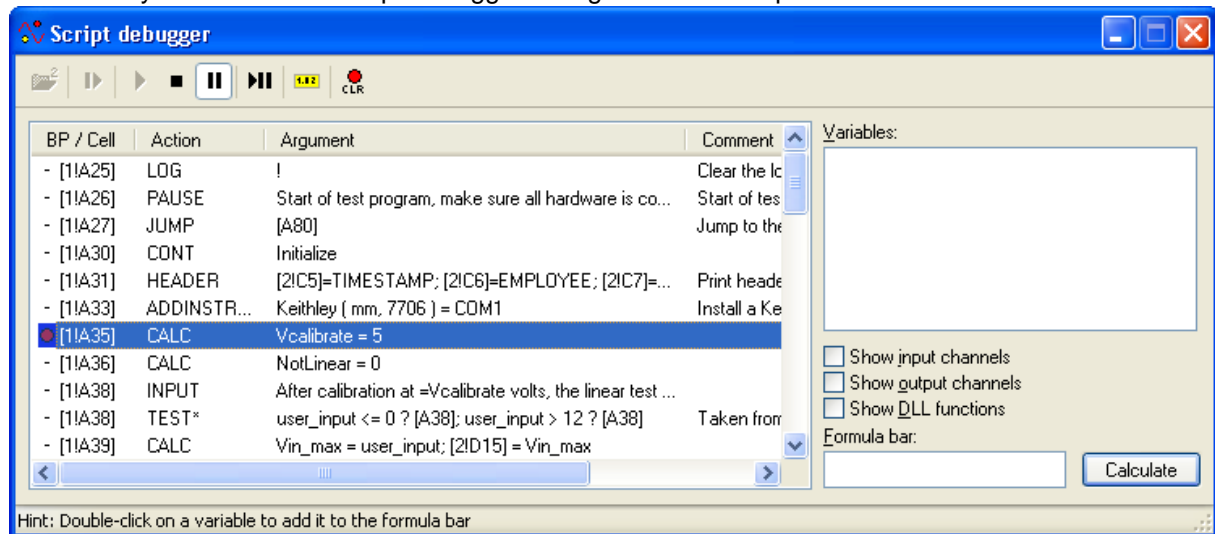
The left script contains syntax errors and will need improvement. The right script was loaded without parser errors. It 'seems okay' because run time errors are not checked during loading.

There are three buttons on the speed bar:

-  Browse for an Excel file and open it
-  Reopen the Excel file, any changes that were made in the Excel file are discarded
-  Import all Actions from the open Excel file

3.7 Script debugger dialog

This dialog can be opened from the main menu by clicking on 'View | Script debugger' or with the shortcut key 'Ctrl + D'. The script debugger dialog looks like this picture:

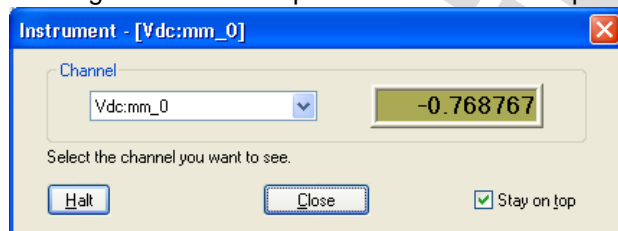


The speed bar offers execution controls like Reload, Restart, Run ('Shift + F5'), Stop, Pause and Singlestep ('F10').

Restart and Singlestep reload the script automatically if needed. The Run button will run the actions, as they were loaded, but if any errors were detected the user is forced to reload the script. If the script needs to be reloaded, click the reload button.

When the script is loaded, the content of the script is shown in the listing.

Note the instrument inspection speed button ('Ctrl + I') that is available when the script is paused. Clicking on this button opens the Instrument Inspection dialog, like the picture below:



In the selection box all channels of all instruments that are connected and that are submitted by ADDINSTRUMENT are accessible in this dialog. When a channel is selected, its reading will be shown continuously. Activating the Halt button stops the measurement, allowing the user to control an instrument's control panel.

Double clicking on a cell reference sets a breakpoint. A red dot appears left from the cell reference like in this picture:

- [1!A48]	CALC	Vadj:mm_23 = Vout; [2!B30 + Step] = Vout; Vloo...	Apply voltage to channel 23 and measure back ...
● [1!A49]	CALC	[2!C30 + Step] = Vloopback	Put read value in result table
- [1!A50]	PRINT	[2!C31 + Step]	Clear the next cell to make current measurement...

Double clicking again disables this breakpoint, but doesn't remove it. The red dot turns yellow like in this picture:

- [1!A48]	CALC	Vadj:mm_23 = Vout; [2!B30 + Step] = Vout; Vloo...	Apply voltage to channel 23 and measure back ...
● [1!A49]	CALC	[2!C30 + Step] = Vloopback	Put read value in result table
- [1!A50]	PRINT	[2!C31 + Step]	Clear the next cell to make current measurement...

Double clicking for the third time removes the breakpoint reference. The dot disappears.

The most right button on the speed bar removes all breakpoints from the script ('Ctrl + B').

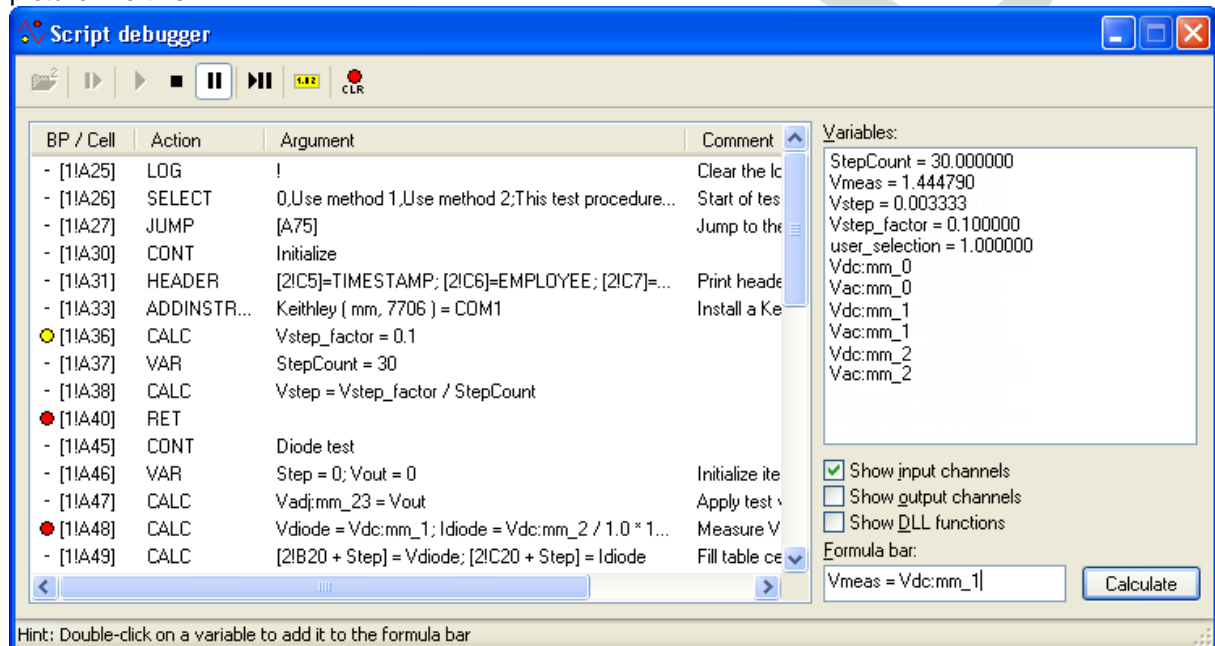
Please note that some cell references appear twice. This happens when a test condition is present for that action. A test condition is evaluated separately, but is coupled to the same cell reference as the Action. Like cell [1!A62] in the picture below:

- [1!A61]	CALC	Bit:mm_21_1 = gt(0, Vloopback)	Turn the green led on when Vloopback is negati...
- [1!A62]	CALC	Step = Step + 1; Vout = Vout - Vstep	Increment iterator and Vout. Test if not ready
- [1!A62]	TEST*	Step <= (StepCount / 2) ? [A58]	Taken from command 'CALC' at [1!A62]: Increm...
- [1!A63]	RET		

Note that the TEST action is marked with an '*' to indicate that is part of another action and can be found in the **condition** column instead of the **command** column.

Please see chapter [5.4: Actions](#) for explanation about actions and chapter [5.6: Test conditions](#) for explanation about test conditions.

When the script has run beyond ADDINSTRUMENT the channels of the instrument can be displayed in the 'Variables' area. So Run the script and then Pause and check 'Show input channels' to see a picture like this:

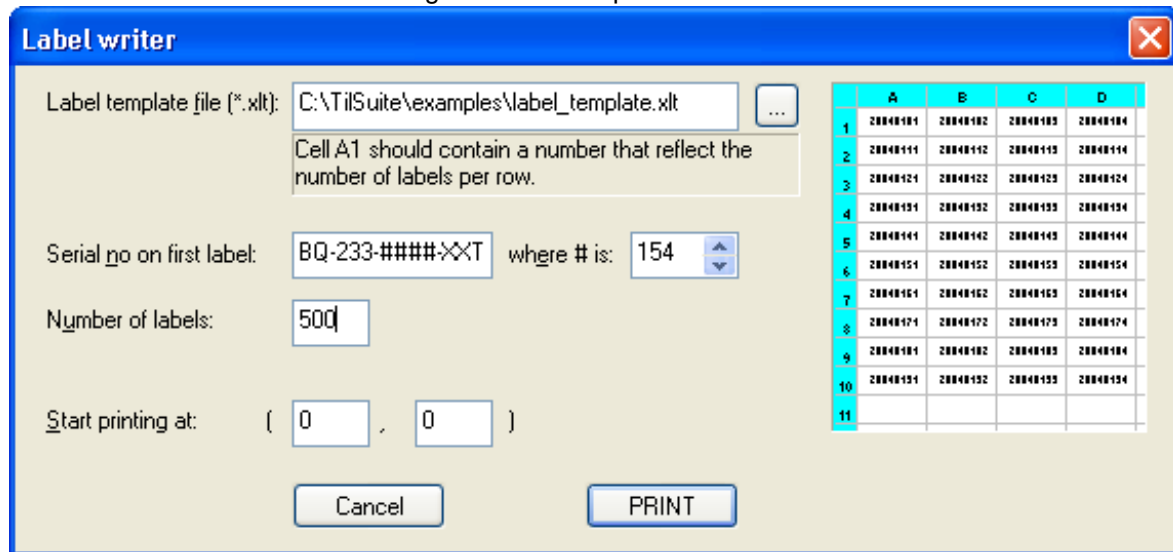


In the variables area please find:

- 'StepCount' up to 'Vstep_factor' these are script defined variables
- 'user_selection' indicates what the SELECT action from cell [1!A26] returned
- 'Vdc:mm_0' up to 'Vac:mm_2' represent channels on 'mm': the Keithley 2700 multi meter
- 'Vmeas' is defined through the formula bar after calculate was clicked. The value of 'Vmeas' represents the DC Voltage measured on channel 1 of the Keithley 2700.

3.8 Serial number label writer dialog

The serial number label writer dialog looks like this picture:



The '*Label template file*' is an Excel template that is used to generate the worksheet with all serial numbers. The template should be formatted to match the labels to be printed. Cell A1 of the template file must contain the number of labels per row.

The serial number text is copied from the parameter section of the main dialog. If necessary, this text can be changed. Indicate the starting serial number in the number field. '*Number of labels*' indicates the number of labels to be printed. '*Start printing at*' gives a starting location in Excel (row, column). This is useful if you need to print on a partially used label sheet.

By clicking on the '*PRINT*' button, the Excel sheet is generated and shows a popup. All printing details (like selecting the desired printer) must be done from Excel. Switch to Excel and print the labels that are generated. After that, switch back to TestmATE and click on the OK button on the popup to close Excel and close this dialog.

4 How to make your first TestmATE script

We truly hope we didn't scare you too much so far. Let's take it one step at a time to get started, see the results and feel comfortable about this.

Open Microsoft Excel, this should open with a blank workbook with 3 worksheets. Select sheet 1 if it is not already selected.

4.1 Script 1: show a popup and exit

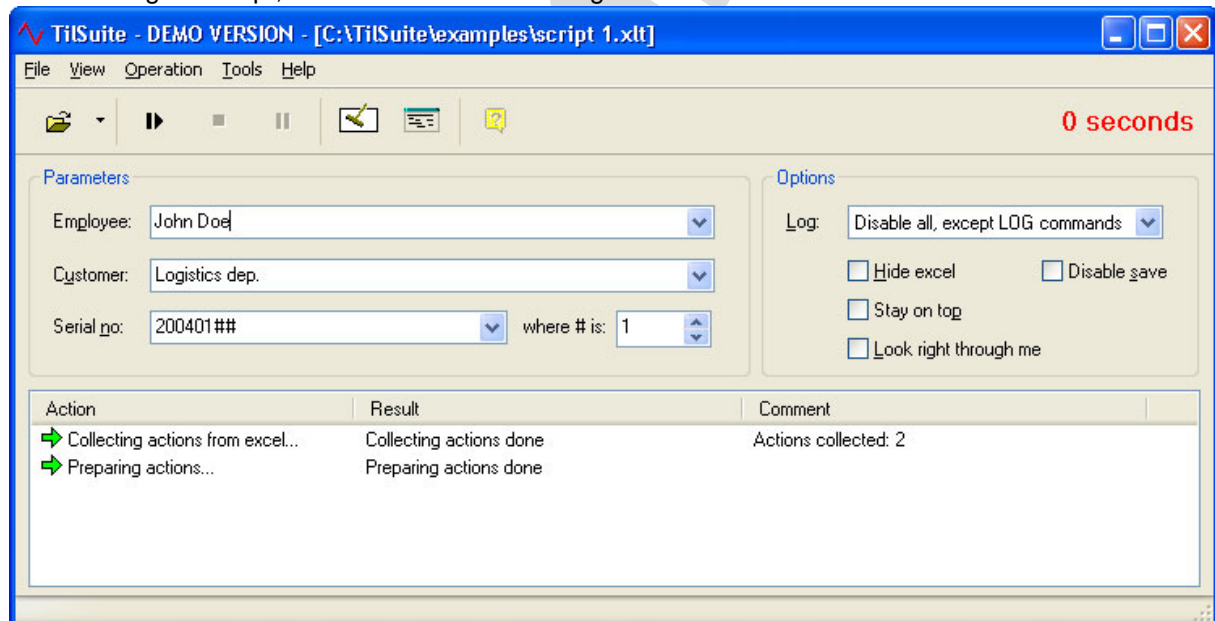
Type these lines:

	A	B	C	D
1	>[A5]			Start at A5
2	Demo script 1	Show a popup that always exits		
3				
4	command	arguments	cond.	comment
5	PAUSE	Q[1!A6] Script question: Abort now?		Ask if the user wants to abort
6	END			
7				

Save the workbook as an Excel Template: "C:\TestmATE\scripts\script1.xlt".

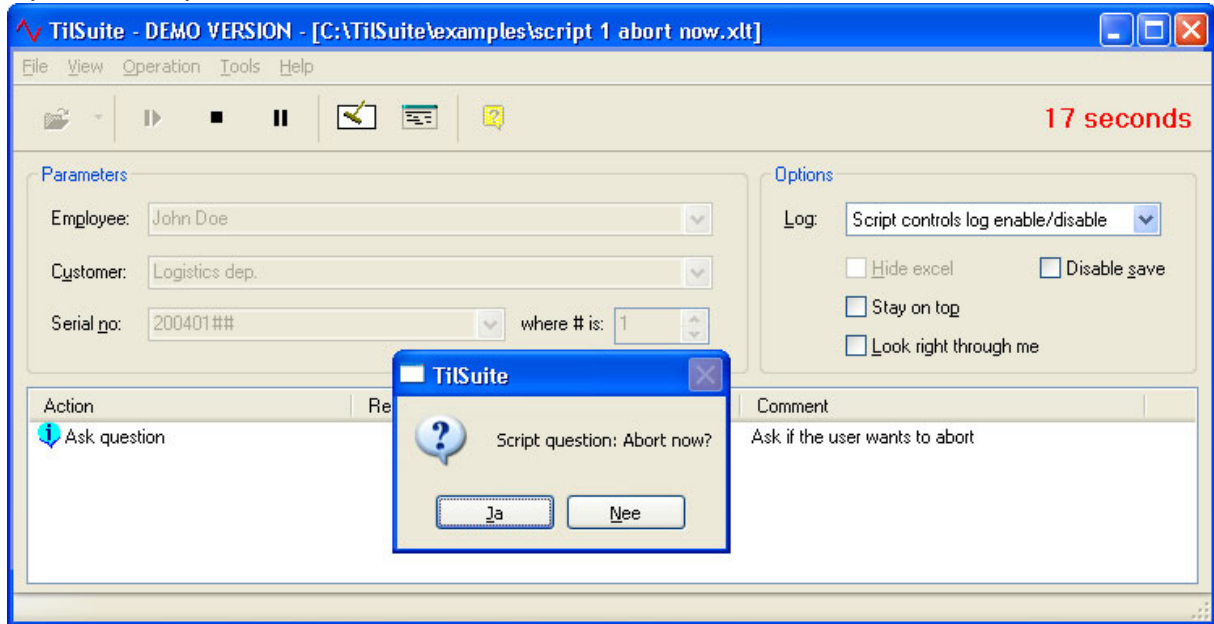
Now let's run the script: Run TestmATE, click 'File | Open test procedure' and select C:\TestmATE\scripts\script1.xlt.

After loading the script, the TestmATE main dialog looks like this:



Click the Run button, choose 'Operation | Run' from the menu or press F5.

If you did well, your screen looks like this:



Click on YES (or NO) and the script will end.

Is this exciting ? Maybe not yet. Let's improve. So far you didn't use the multi meter at all. The next script shows how to get measurements from the Keithley 2700.

Please note that all scripts that are explained here are also stored in:

"C:\Program Files\Tildesign\TestmATE\Examples" as a reference and in

"C:\TestmATE\scripts" to use, modify and try without possibly destroying the original examples.

When your script is complete, save it and make it read-only.
This avoids accidental saving to your test templates.

4.2 Script 2: run a simple production test script repeatedly with serial numbers

Please note that in all example scripts COM1 is used for the Keithley 2700. Modify this parameter to suit your configuration. See ADDINSTRUMENT for details.

	A	B	C	D
1	>[A5]			Start at A5
2	Demo script 2	Measure channel 1 in a popup, in a variable and in a cell		
3				
4	command	arguments	cond.	comment
5	PAUSE	Q[1!A7]Start running?		Shows a popup with abort question, if YES continue in call A7
6	JUMP	[A20]		If NO, goto A20
7	ADDINSTRUMENT	Keithley(mm,7706)=COM1		Define Keithley 2700 as "mm" with a 7706 add-on card on RS-232 port COM1
8	VAR	Vmeas=0		define variable "Vmeas"
9	LOG	s=Run the test		write message in log
10	CALC	Vadj:mm_23 = 1.000		output 1V on DAC1 of "mm"
11	MEASURE	Vdc:mm_1;Adjust potmeter to calibrate, then click Close		Measure value and show in popup to calibrate a potmeter manually
12	CALC	Vmeas=Vdc:mm_1		Fill variable "Vmeas" with the Vdc measurement of channel 1 of mm
13	CALC	[2!C5]=Vdc:mm_1		Fill cell C5 of sheet 2 with the Vdc measurement of channel 1 of mm
14	HEADER	[2!C1]=TIMESTAMP; [2!C2]=SERIALNR		Mark test with timestamp and serial# in sheet 2 cells C1 and C2
15	SAVE	C:\TestmATE\results\ #_test.xls		Save this file
16	SELECT	0,Test another, Repeat,Done;The test has completed.	user_selection == 1 ? [A18]; user_selection == 2 ? [A19]	Test another: A17 Repeat: goto A18 Done: goto A19
17	ENDTEST	RESTART, INC SERIALNR		Restart test, next serial number
18	ENDTEST	RESTART		Restart test, same serial number
19	CLOSE			Close Excel
20	END			end the script
21				

In this script already quite a lot happens.

First a popup is shown, the program waits for the user to tell that the Device Under Test or D.U.T. is connected. Then the Keithley 2700 multi meter is added to the TestmATE (A7). A variable "Vmeas" is declared (A8) to be used later on. In the log control of TestmATE a message is printed (A9).

To give channel 1 a sensible value, DAC 1 is set to 1.00 V (A10).

Then a calibration popup is shown, the input of channel 1 is displayed continuously so a potmeter on the D.U.T. can be calibrated (A11). When the user clicks Close, channel 1 is sampled again. It's value is stored in variable Vmeas (A12). Next channel 1 is sampled again, that result is stored in cell C5 of

sheet 2. Note that channel 1 is sampled a few times. If you need only one sample, you could have used Vmeas to store that value. Replace cell B13 with: "[2 ! C5] =Vmeas" to do so.

Next the time stamp and the serial number are copied to sheet 2 (A14) and the file is saved with the serial number in the filename (A15). This allows for easy retrieval of the test results.

Now the test is done, a popup is shown to let the user select: Test another, Repeat or Done (A16). The default selection is "Test another" because of the 0 at the start of cell B16. 0 indicates the first selection.

When the user selects "Test another" the script continues at A17. The test is restarted with the next serial number.

If the user selects "Repeat" the script continues at A18. The test is restarted with the same serial number.

If the user selects "Done" the script continues at A19. Excel is closed and the script ends.

We strongly advise to store your own scripts in the 'C:\TestmATE\scripts' directory and make your scripts read-only. That way your scripts are protected from accidental Save actions or delete keystrokes.

You can make your scripts read-only using Windows Explorer. Click on the filename with your right mouse button and select Properties. Check the Read-only checkbox and click OK.

If you want to modify your script, then uncheck this checkbox or save your script under a different name.

4.3 Script 3: loop back with LED's

For this script it is necessary to have some components connected to the 7706 add-on board.

Please open "C:\TestmATE\scripts\script 3 loopback with leds.xlt" with Excel. On the Procedure sheet there is a picture showing how to connect the components. More information about connecting demo components is in the Quick Start Manual.

This script will output a triangular voltage on the DACs and measure these signals back.

Please make the connections, mount the 7706 board and power up the Keithley 2700.

Click on the Run button and check out the results.

4.4 Script 4: diode characteristic

Please open script example 4 from:

"C:\TestmATE\scripts\script 4 diode characteristic.xlt".

Note that in all example scripts COM1 is used for the Keithley 2700. Modify this parameter to suit your configuration. See ADDINSTRUMENT for details.

The content of this script is not shown here. Where necessary some explanation is given.

This script measures the diode characteristic. It supports two ways of measuring the diode current.

Method 1 measures the voltage drop over a sense resistor.

Method 2 measures the diode voltage and assumes no error in the DAC output.

The Excel sheet contains additional information as well as the Quick Start Manual.

4.5 Script 5: calibration example

Please open script example 5 from:

"C:\TestmATE\scripts\script 5 calibration.xlt".

This script shows how a voltage can be calibrated by trimming a potentiometer.

The Excel sheet contains additional information as well as the Quick Start Manual.

4.6 Script 6: calibration with linearity test

Please open script example 6 from:

"C:\TestmATE\scripts\script 6 calibration with linear test.xlt".

This script shows how a voltage can be calibrated by trimming a potentiometer.

The Excel sheet contains additional information as well as the Quick Start Manual.

4.7 Script 7&8: using a user DLL

Please open script example 7 from:

"C:\TestmATE\scripts\script 7 userdll.xlt".

This script shows how a user DLL can be used. The example shows a DLL that returns the system time as 'minutes.seconds'. The Instrument Dialog demonstrates that the DLL functions behave like channels on an instrument.

Please open script example 8 from:

"C:\TestmATE\scripts\script 8 beep userdll.xlt".

This script shows how a user DLL can be used. with arguments. The example shows a DLL that asks for a starting frequency and an ending frequency. After that the system beeper of the PC will be controlled by the script through the DLL. Note: this example will possible only function correctly on Windows XP.

4.8 Example Error Scripts

We added some scripts that contain errors to demonstrate the error handling behaviour and to indicate the probable cause.

<u>script e1 syntax error.xlt</u>	contains a syntax error
<u>script e2 unknown command.xlt</u>	contains an unknown command
<u>script e3 runtime error.xlt</u>	contains a run time error
<u>script e4 invalid jump.xlt</u>	contains a jump to an invalid cell
<u>script e5 no end.xlt</u>	misses the END Action

5 Scripting language explained

The script is stored in an Excel workbook and contains:

<i>Actions</i>	what needs to be done,
<i>Arguments</i>	what is the source, what is the target, what text will be shown,
<i>Test conditions</i>	after execution of the Action, test the result and jump accordingly and
<i>Comments</i>	explanation what the designer <i>intended</i> to do and
<i>Redirectors</i>	to let the action collector continue from a different location.

The action collector fetches these when a script is loaded.

Please note; All commands are case insensitive. Most parameters are case sensitive.

5.1 Column dependencies

If the action collector fetches an Action in column A it expects the Arguments in column B, Test conditions in column C and Comments in column D, all on the same line.

We strongly advise that you use the Comment column. Comments explain what the purpose of this Action is and that improves maintainability. Apart from that comments are displayed in the log list and can be of help when you are developing a script.

5.2 Action Collector and Redirector

The Action Collector needs to be 'told' where the script starts and where it ends. This can be done with the 'Collector Redirector' or 'Redirector' for short.

The syntax for the Redirector is '>[target sheet ! target cell]'.

A Redirector must always be in cell A1 of sheet 1 because that is the first cell to be read. In the example below cell A1 contains '>[1!A6]' which redirects the Collector to sheet 1, cell A6. This causes the collector to skip the comment lines.

When you need a lot of spare lines in your Excel sheet then use a Redirector to let the Collector skip these lines. If the Collector encounters 40 or more empty lines it displays an warning. Using the Redirector avoids this error message.

The Redirector also allows you to use different columns than A, B, C and D. You can redirect the Collector to fetch actions from any column or sheet you prefer. This is however deprecated.

The Action Collector stops when it fetches the action END and will never fetch beyond that cell.

Example:

	A	B	C	D
1	>[1!A6]			Start script at cell A6
2	Test script for	testing product X	Rev 1.1	Sept 12, 2004
3	Written by	Sander Pasveer		
4				
5	command	arguments	cond.	comment
6	PAUSE	Install D.U.T.		wait until user clicks OK
7	END			script ends here

5.3 General layout considerations

All actions need to be read from the selected Excel file. Writing scripts is straightforward, but for maintenance purposes it is advised to use the first lines of your script to explain:

- for which purpose or product the script is intended,
- the revision number and revision date,
- who created or maintained the script,
- what equipment or configuration is needed

This way it is clear to everyone who opens the Excel workbook what to expect.

We advise to always use sheet 1 for scripts and sheet 2 for results. Please use always column A for Actions, column B for Arguments, column C for Test conditions and column D for comments. This makes the script easier to read and maintain.

Please write comments for every action. If a script needs modification after a year, you really can't remember every detail of the script. Then you or your successor will need the comments badly.

5.4 Actions

All possible actions are explained in chapter 6 *Scripting language reference*.

Just copy the action text literally into the cell. In the examples it is assumed that Actions are stored in column A. If an action is preceded by '//', this action is omitted (including its test condition). In that way actions can still reside in the script, but are skipped.

Example:

	A	B	C	D
8	command	arguments	cond.	comment
9	CALC			
10	PAUSE			
11	//MEASURE	future use		skip for the time being

5.5 Arguments

The arguments contain source or destination locations for data, strings and formulas. The Action defines what can be used as arguments.

Most Actions can handle a sequence of arguments which must be separated by a semicolon ';'. In the Action definitions this possibility is indicated by this line:

This action supports semicolon (;) separated arguments.

Arguments are stored in the first column right of the Action column.

Example:

	A	B	C	D
8	command	arguments	cond.	comment
9	CALC	Vmeas1 = Vdc:mm_1 ; Vmeas2 = Vdc:mm_2		Read both mm_1 and mm_2 as DC voltage in one transaction
10	CALC	Vmeas1 = Vdc:mm_1		Do the same in 2 lines
11	CALC	Vmeas2 = Vdc:mm_2		

This offers the advantage of grouping transactions that are related. For instance set a DAC to output a voltage and read from an input. The action can't be divided by inserting a line between those actions.

5.6 Test conditions

After an Action has completed, the result(s) may be tested and the script can continue in a cell based on the test results.

An Action supports this feature if this test is mentioned in the Action definition:

This action allows a test condition to be passed in the next column.

Test conditions are stored in the second right column next to the Action column.

Example:

	A	B	C	D
8	command	arguments	condition	comment
9	CALC	Vmeas1 = Vdc:mm_1	Vmeas1 < 2 ? [C11]; Vmeas1 > 3 ? [C11]	Read mm_1 and check if value is between 2 and 3 V
10	JUMP	[C20]		2 < Vmeas1 < 3
11	CONT			Vmeas1 < 2 or Vmeas1 > 3

Syntax: symbol compare_operator symbol ? [cell reference] ; next condition

5.7 Comments

Comments are used to explain what the designer intended to do. Apart from that the comments are displayed in the log list. Comments are stored in the third right column next to the Action column.

Example:

	A	B	C	D
8	command	arguments	cond.	comment
9	PAUSE	Q[1!A30]Abort?		Ask if the user wants to abort
10	LOG	w=Script log: user aborted script		Add log using the warning icon
11	ENDTEST			

Output:

Action	Result	Comment
→ Ask question	User clicked YES, jumping to [1!A30]	Ask if the user wants to abort
→ Log	Script log: user aborted script	Add log using the warning icon
→ End of test reached	Test completed successfully	Test duration: 2 seconds

5.8 String syntax

An Action may expect a string as an argument. It is necessary to understand how strings are interpreted in order to understand why a string is generated as it is.

If a string is plain, without any formula, a string starts after the first separation character (if any) and ends just before the next separation character (if any).

Example:

LOG this is plain text and will be printed as is

A string can be expected as an argument on a certain position in the argument list.

Example:

SELECT 1,First selection,Second selection,Third selection;Make a choice

The argument part contains a number and four strings.

The string popups (Input, Select) may contain a formula. When the parser encounters an '=' sign it checks if the next character is a blank ' '. If so this is treated as literal text. If not this is treated as a formula or a variable. When used in a string the formula ends at the first blank ' '. Therefore the formula itself may not contain any blanks.

Examples:

String: The measured voltage at input 3 = =Vdc:mm_3 Volt

Output: The measured voltage at input 3 = 1.235 Volt

String: The measured power = =Vdc:mm_3*Vdc:mm_4/Rsense Watt

Output: The measured power = 3.733 Watt

5.9 Formula syntax

A formula consists of at least two symbols an operator a destination variable and an assignment operator '='.

```
destination_var = symbol1 operator symbol2
```

A symbol can be a constant, a variable a function or a measured parameter.

The parser removes all white space from an equation before interpretation is started. You are free to use as much white space as you like to improve readability.

Examples:

```
Vmeas = Vdc:mm_1 * Vdc:mm_2
```

Vmeas equals the product of the measured values of channels 1 and 2 of 'mm'.

```
Vmeas = sin(Vdc:mm_1)
```

Vmeas equals the sine of the DC voltage measured at channel 1 of 'mm'.

All of the examples below are valid statements:

```
var1=1
```

```
var1=1.234
```

```
var1 = 1.2345e23
```

```
var1 = - 2.375 e - 15 + 4. 2 4 3 2 e - 14
```

Please note that this is only allowed when using plain formulas. When a formula is used in a string it may not contain any white space.

5.10 Variable usage

A variable name consists of alphanumerical characters, so '0'..'9', 'a'..'z', 'A'..'Z' and '_'.

The use of other characters is depreciated, correct functioning is not guaranteed in that case.

The first character of a variable name may never be a number.

Variables can only contain floating-point numbers.

Variables should be defines before they are used. Some variables are predefined. Define your own variables, as you need them. You can use the `CALC` and `VAR` actions to define variables.

When the interpreter cannot resolve a name (because an instrument is not added), a new variable will be declared. In this case a warning will be shown because this is not what is usually intended.

Variables can be used to store a value or text. Variables can be used in formulas and as parameters in functions.

	A	B	C	D
8	command	arguments	cond.	comment
9	VAR	meas=10		declare variable "meas", fill with 10
10	CALC	sq_meas=pow(meas,2)		declare variable "sq_meas" and fill with meas ²
11				

6 Scripting language reference

6.1 ADDINSTRUMENT

Install an "instrument" and connect to it. This can be a power supply, multi meter, user DLL, etcetera. A user DLL gives the possibility to let the Device Under Test be an "instrument" and let it participate in the test. Currently supported instruments are:

- Keithley 2700 series multi meter, with or without one or two 7706 multiplexer card(s) installed.
- Peaktech power supply.
- Relays card interface, 16 relays each, can be cascaded on a single parallel port.
- User supplied DLL files with exported C functions with prototype:

```
'extern "C" _declspec(dllexport) int DllFunction (char *arg, double *result);'
```

The syntax for all devices is as follows:

```
ADDINSTRUMENT DEVICE(NAME, SUBDEVICE_IN_SLOT1=ENUM1,
                    SUBDEVICE_IN_SLOT2=ENUM2,
                    SUBDEVICE_IN_SLOT3=ENUM3, ...) =PORT
```

Where:

- DEVICE is one of the supported instruments (so far: KEITHLEY, PEAKTECH, RELAISCARD, or DLL).
- NAME is the variable name assigned to this device (e.g. 'multi_meter').
- SUBDEVICE_IN_SLOTx (optional) is used when an instrument can have add-on modules that must be defined (a multiplexer card like the 7706).
- ENUMx (optinal) is used to specify the first ID assigned to the channels of module x. When this is not specified, the assigned ID is simply the serial number of the channel.
- PORT is the serial/parallel the device is connected to (e.g. 'COM1'), or the path to the DLL file.

For the relays card the number of cascaded cards is to be specified as SUBDEVICE_IN_SLOT1.

This action supports semicolon (;) separated arguments.

Example:

	A	B	C	D
8	command	arguments	cond	comment
9	ADDINSTRUMENT	Keithley(mm,7706)=COM5		Install Keithley 2700 with a 7706 add-on card in slot 1 as "mm" on COM5
10	ADDINSTRUMENT	Keithley(mm,7706=101)=COM5		Same here, but slot 1 channels are now accessible through mm_101, mm_102 etc.
11	ADDINSTRUMENT	Peaktech(ps)=COM6		Install Peaktech power supply as "ps" on COM6
12	ADDINSTRUMENT	Relaiscard(rc,4)=LPT1		Install a relays card stack of 4 cards as "rc" on LPT1
13	ADDINSTRUMENT	Dll(my_dll)= c:\dll\instrument.dll		Install "instrument.dll" as "my_dll"
14				
15	CALC	Vadj:ps_0 = 10; Iadj:ps_0 = 0.2		Set power supply to 10V and current limit to 0.2A
16	CALC	Setbit:rc_0 = 8		Switch on relay 8 on relays card 0
17	DLLEXEC	my_dll__get_adc 10		Query D.U.T. for ADC channel 10 through DLL interface
18	CALC	Vmeas = Vdc:mm_0		Measure DC Voltage on channel 0 of the Keithley and store the reading in "Vmeas"

6.2 CALC

Performs a mathematical calculation. Also reads and writes to instrument channels and Excel cells. Before an instrument channel can be accessed, the instrument must be added using `ADDINSTRUMENT`.

The name that is defined for an instrument is prefixed with the unit (e.g. 'vdc:', 'Iac:'), and postfixed by the zero based channel number (e.g. '_10').

Digital channels may contain multiple bits per channel (sub channels). These bits are individually addressable by using another '_X' postfix (e.g. '_5' for bit 5).

This is also zero based: the first bit is '_0'.

Excel cells are recognised by encapsulating them with brackets ('[]'). In the most simple form, constant cell co-ordinates reflect the destination or source cell, in an - optional - specified sheet number.

[2!A1] refers to cell A1 on sheet 2.

[A1] refers to cell A1 on the current visible sheet.

The cell co-ordinate can also be a formula where a variable can function as an offset. This is very useful in loops.

[2!A10 + row_offset] refers to cell A(10+row_offset) on sheet 2.

[2!A + col_offset|10] refers to cell (A+col_offset)10 on sheet 2. Note the '|' symbol to mark the end of the column formula.

It is allowed to nest ONE Excel cell. Thus [2!A10 + [3!A1]] will add the numerical contents of cell A1 on sheet 3 to row 10.

6.2.1 Prefixes that can be used in CALC

Supported prefixes (units) are:

V: or Vdc:	measure / set dc voltage
Vac:	measure / set ac voltage
I: or Idc:	measure / set dc current
Iac:	measure / set ac current
T:	measures temperature using a type T thermocouple
Vdcadj:	get adjusted / set dc voltage
Vacadj:	get adjusted / set ac voltage
Idcadj:	get adjusted / set dc current
Iacadj:	get adjusted / set ac current
Vdcmax:	gets maximum adjustable dc voltage
Vacmax:	gets maximum adjustable ac voltage
Idcmax:	gets maximum adjustable dc current
Iacmax:	gets maximum adjustable ac current
Bit:	sets or clears a digital output bit (a sub channel of a channel)
Setbit:	sets a digital output bit (a sub channel of a channel)
Clrbit:	clears a digital output bit (a sub channel of a channel)
Dll:	executes a dll function and gets the result. The arguments passed are THE SAME as used previously using DLLEXEC !

Example:

Vadj:mm_23 = voltage

If an instrument is added and defined as 'mm', channel 23 (7706's DAC0) will adjusted to <voltage> dc volt. If no instrument is defined as 'mm', a regular variable 'Vadj:mm_23' will be assigned.

Note that this way of defining variables is not recommended and a warning will be shown in the comment column of the log list.

Setbit:mm_21 = 4

Bit:mm_21_4 = 1

Will both set bit-4 of digital output 21 of the Keithley 7706 I/O card.

6.2.2 *Mathematical operations that can be used in CALC*

The following mathematical operations are available:

*	Multiply
/	Divide
+	Add
-	Subtract
sin(x)	Sine of x
cos(x)	Cosine of x
tan(x)	Tangent of x
asin(x)	Arcsine of x
acos(x)	Arccosine of x
atan(x)	Arc tangent of x
sinh(x)	Hyperbolic sine of x
cosh(x)	Hyperbolic cosine of x
tanh(x)	Hyperbolic tangent of x
exp(x)	Exponential of x
eq(x1, x2)	Compares x1 with x2. Result is 1 when equal, 0 when not equal
neq(x1, x2)	Compares x1 with x2. Result is 1 when not equal, 0 when equal
gt(x1, x2)	Compares x1 with x2. Result is 1 when x1 > x2, 0 otherwise
ge(x1, x2)	Compares x1 with x2. Result is 1 when x1 >= x2, 0 otherwise
lt(x1, x2)	Compares x1 with x2. Result is 1 when x1 < x2, 0 otherwise
le(x1, x2)	Compares x1 with x2. Result is 1 when x1 <= x2, 0 otherwise
abs(x)	Absolute value of x
int(x)	With x being positive, x is round down. When x is negative, x is round up
and(x1, x2)	Bitwise 'AND' x1 with x2
AND(x1, x2)	Bitwise 'AND' x1 with x2, but returns 1 if result is not 0
or(x1, x2)	Bitwise 'OR' x1 with x2
inv(x)	One's complement of x
shr(x1, x2)	Bitwise shift-right x1, x2 times
shl(x1, x2)	Bitwise shift-left x1, x2 times
max(x1, x2)	Compares x1 with x2. The result is the largest of the two
min(x1, x2)	Compares x1 with x2. The result is the smallest of the two
pow(x1, x2)	Calculates x1 raised to the power of x2
ln(x)	Natural logarithm of x
log(x)	Logarithm of x
sqr(x)	Square root of x
fact(x)	Factorial of x

Examples:

```
voltage = 1 + 2 * 3
```

Calculates $1 + (2 * 3)$, and stores the result in variable 'voltage'.

```
Vadj:mm_23 = voltage
```

If an instrument is added and defined as 'mm', channel 23 (7706's DAC0) will be adjusted to <voltage> dc volt. If no instrument is defined as 'mm', a regular variable 'Vadj:mm_23' will be assigned.

Note that this way of defining variables is not recommended and you will get a warning.

```
output_voltage = Vac:mm_3
```

If an instrument (a multi meter e.g.) is added and defined as 'mm', an ac voltage measurement will be performed on channel 3. The result is stored in variable 'output_voltage'. If no instrument is defined as 'mm', a regular variable with the name 'Vac:mm_3' should exist.

```
Bit:mm_21_5 = 0
```

If an instrument (with digital outputs) is added and defined as 'mm', bit 5 of digital channel 21 will be cleared.

```
[2!C10] = output_voltage
```

Cell C10 on sheet 2 is filled with variable 'output_voltage'.

This action supports semicolon (;) separated arguments.

This action allows a test condition to be passed in the next column.

Examples:

	A	B	C	D
8	command	arguments	cond.	comment
9	CALC	point = point + 1		increment "point" by 1
10	CALC	[2!C3] = point		store "point" in sheet 2, cell C3
11	CALC	[2!C50+point] = T:mm_9; [2!D50+point] = T:mm_10		measure channels 9 and 10 of "mm" as temperature, store results in sheet 2, cells C50 and D50, offset by "point".
12				

6.3 CALL

Calls a function which entry point is at the specified Excel cell.

The destination cell **MUST** contain a command. See also the CONT action.

The return 'address' is placed on the stack. Use the RET action to return to this point.

Example: CALL [1!A100] Calls function at cell A100 on sheet 1.

Calls can be nested 10 levels deep. Every CALL should be terminated by a RET action.

The cell reference may contain a formula, see JUMP for details.

This action allows a test condition to be passed in the next column.

Example:

	A	B	C	D
8	command	arguments	condition	comment
9	CALL	A13		Call function at A13
10	ENDTEST			Execution ends here
11				
12				
13	CONT			function entry
14	CALC	point = point + 1		increment point
15	RET			return from call
16				
17	END			Script ends here (last line)
18				

On cell A9 a function is called, execution continues at cell A13. Next in A14 a calculation is performed and the function returns at cell A15. Last instruction is A10, which stops the interpreter.

6.4 CLOSE

Closes Excel WITHOUT saving the open file. This action takes no arguments.

Example:

	A	B	C	D
8	command	arguments	condition	comment
9	SAVE	C:\TestmATE\results\ #_test.xls		Save sheet. Filename is <serial number>_test.xls
10	CLOSE			Close excel
11	ENDTEST	RESTART, INCSERIALNR		Increment serial number and restart the test
12	END			Last instruction of this script
13				

The file is saved (A9), the filename contains the serial number followed by '_test.xls'. After saving Excel is closed (A10). The test is restarted with the next serial number (A11). The script ends at cell A12.

6.5 CONT

Creates a dummy action. Useful when jumping outside a loop while there is actually no action defined. This action takes no arguments.

Example: see CALL

6.6 DELAY

Sleep function in milliseconds.

Example: `DELAY 500` Wait half a second and then continue the test program.

Example:

	A	B	C	D
8	Command	arguments	condition	comment
9	DELAY	1000		Wait 1 second
10				

6.7 DLLEXEC

Passes command line parameters to the specified function that resides in a DLL 'instrument'.

The result is stored in variable 'dllexec_result'.

The return code is stored in 'dllexec_returncode'.

Syntax: `DLLEXEC dllname_function arg1 arg2`

This command line consists of the DLL instrument name ('my_dll') as it's defined by the `ADDINSTRUMENT` action, followed by an underscore ('_') and then the DLL function to be executed. Arguments, separated by a white space, come next.

In order to pass values of variables (or Excel cells) use the equal sign ('='). Even formulas are allowed, but note that white spaces break them.

Example: `DLLEXEC my_dll__set_dig_out =channel_no 1`

This example calls function 'set_dig_out' in dll 'my_dll' and passes the value of 'channel_nr' as argument 1 and the number '1' as argument 2.

If the DLL passes a value, it is stored in `dllexec_result`. The DLL function itself will pass an exitcode to `dllexec_returncode` to indicate an error or success.

This action allows a test condition to be passed in the next column.

Example: see chapter [6.1: ADDINSTRUMENT](#)

6.8 END

Use this 'action' to mark the last the script action. When the parser reads this action, it stops collecting. Any action past this action is discarded.

Example: see chapter [6.4: CLOSE](#)

6.9 ENDTEST

Terminates the test procedure immediately. Excel is not saved nor closed. But, when you pass `RESTART` as the argument, the procedure is restarted immediately after it ends. This is very useful for testing a series of products.

Please note that the option `RESTART` always closes and reopens the Excel sheet. If you didn't `SAVE` the file, the results are lost.

You can also pass `INCSERIALNR` as a parameter. This will increment the serial number of the Excel workbook. See also the example scripts.

Example: see chapters [6.4: CLOSE](#) and [3.3 Parameter section](#)

6.10 EXEC

Execute a DOS command line. The return code is stored in variable 'exec_result'.

Example: EXEC c:\flashtool.exe "-fprogram.hex" -pCOM1

Note that the program should not return -1. When it does, 'exec_result' is not set and an - possibly false - error message will popup. A return value of -1 is reserved by the operating system to indicate an unrecoverable error.

This action allows a test condition to be passed in the next column.

Example:

	A	B	C	D
8	Command	arguments	condition	comment
9	EXEC	C:\flashtool.exe "-fprogram.hex" -pCOM1	exec_result==0 ? A13	Execute flashtool with parameters and test for success
10				

6.11 HEADER

Prints header info (like time/date, serial number etc.) in the specified cell.

Examples:

HEADER [2!B3]=TIMESTAMP Prints current time and date in cell B3 on sheet 2.
 HEADER [2!B4]=SERIALNR Prints serial number in cell B4 on sheet 2.
 HEADER [2!B5]=EMPLOYEE Prints employee name in cell B5 on sheet 2.
 HEADER [2!B6]=CUSTOMER Prints customer name in cell B6 on sheet 2.

This action supports semicolon (;) separated arguments. The mentioned parameters are predefined and are filled by the TestmATE main dialog.

Example:

	A	B	C	D
8	command	arguments	condition	comment
9	HEADER	[2!B2]=TIMESTAMP		Place the timestamp in sheet 2, cell B2
10				

Output:

	A	B	C	D
1				
2		Thursday, 18th of March in 2004 at 14:38:24		
3				

6.12 INPUT

Popup a dialog that shows an edit field in which the user should enter a numerical value. The dialog shows some user text as well. By typing \n a newline is inserted in the text.

This text may contain a 'formula' to print variables in the text. See action PRINT for more details.

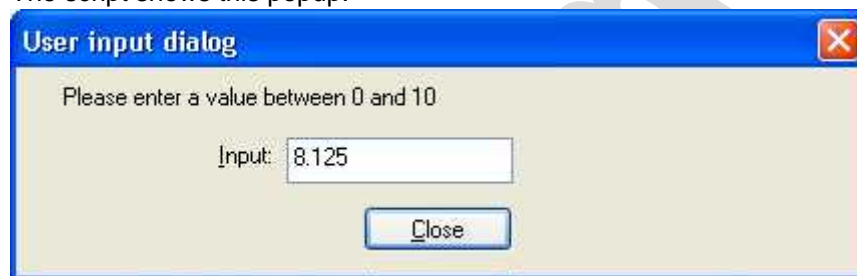
When the user closes the dialog, the entered value item is stored in variable 'user_input'.

This action allows a test condition to be passed in the next column.

Example:

	A	B	C	D
8	command	arguments	condition	comment
9	INPUT	Please enter a value between 0 and 10	user_input > 0?[A12]; user_input < 10?[A12]	Ask for user input and validate 0 < x < 10
10	PAUSE	Value must be 0..10		Print a message
11	JUMP	[A9]		and repeat input
12	CALC	Vadj:mm_23 = user_input		Continue, user_input is written to DAC1 of the Keithley 7706
13				

The script shows this popup:



6.13 JUMP

Invokes an unconditional jump to the action that was declared in the specified cell.

The destination cell MUST contain a command. See also the CONT action.

The destination may be calculated through a formula

Example:

	A	B	C	D
8	Command	arguments	cond.	comment
9	JUMP	[1!A100]		jumps to action at cell A100 in sheet 1
10	SELECT	0,COM1,COM2;Select COM port		Let user select a COM port
11	JUMP	[1!A12 + user_selection * 2]		Jump to: A12 if COM1 or A14 if COM2 was selected
12	ADDINSTRUMENT	Keithley(mm,7706)=COM 1		Connect through COM1
13	JUMP	[A15]		skip A14
14	ADDINSTRUMENT	Keithley(mm,7706)=COM 2		Connect through COM2
15	CONT			Continue

Preliminary

6.14 LOG

LOG adds text to the log list control. Also can suppress messages and clear the log list control.

Log syntax:

```
LOG s=hello world.      Adds text with the success icon
LOG e=This is wrong!   Adds text with the error icon
LOG w=just a warning.  Adds text with the warning icon
LOG 0                  Suppress success logs, show only LOG actions, errors and warnings.
                        This is only valid if the log filter setting allows this.
LOG 1                  From this point: show every action in the log.
                        This is only valid if the log filter setting allows this
```

Example:

```
LOG s=Output voltage = =(output_current*resistance) Volt.
```

Note that the first '=' sign is not printed because it is part of 's=', the second '=' sign will be printed as literal text and the third '=' sign is not printed because it is part of a formula.

With S, E or W in uppercase you get a popup message as well.

Use '0' (zero) to suppress success logs from this point. A '1' (one) re-enables these logs again.

Use '!' to clear the log window.

The text may contain a 'formula' to print variables in the text. See action PRINT for more details.

Example:

	A	B	C	D
8	Command	arguments	condition	comment
9	LOG	s=hello world		log: success with text
10	LOG	e=This is wrong!		log: error with text
11	LOG	w=User aborted script		log: warning with text
12				

Example of a line, produced by a LOG action:




Action	Result	Comment
➔ Ask question	User clicked YES, jumping to [1!A30]	Ask if the user wants to abort
➔ Log	Script log: user aborted script	Add log using the warning icon
➔ End of test reached	Test completed successfully	Test duration: 2 seconds

In the main dialog a log filter can be selected to suppress or allow Actions to be printed in the LOG list.

6.15 MEASURE

Shows a dialog that measures an instrument channel every 0.5 s and shows some user text as well. By typing \n a newline is inserted in the text. The text may contain a 'formula' to print variables in the text. See action PRINT for more details.

The action is extremely suitable for manual calibration purposes. For this, the target value with a maximum deviation can be specified optionally. Three figures in the dialog help the user to calibrate according the specifications:

-  The reading is below the lower limit of the target value. The user must increase the level.
-  The reading is above the upper limit of the target value. The user must decrease the level.
-  The reading meets the desired accuracy. The user may close the dialog.

Examples:

MEASURE Vdc:mm_0;Adjust output voltage with potmeter 1 to =vout Volts. Shows the instrument dialog with channel Vdc:mm_0 preselected.

MEASURE Vdc:mm_1, vout, 0.001;Adjust output Voltage with potmeter 1 to =vout Volts. Shows the instrument dialog in calibration mode with channel Vdc:mm_1 preselected.

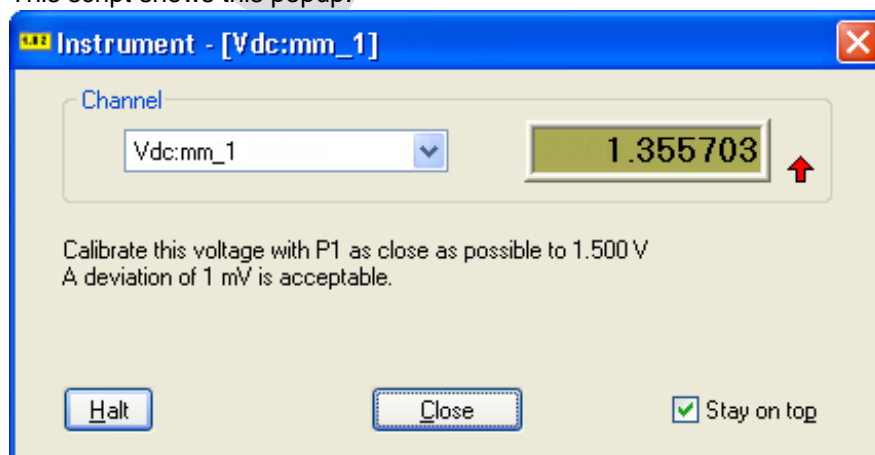
Note that the applied instrument channel MUST EXACTLY match the channel known by the system.

Also dll functions can continuously be executed. Note that you must call DLLEXEC first to initialise the argument string to be used on every 'measurement cycle'.

Example:

	A	B	C	D
8	command	arguments	condition	comment
9	MEASURE	Vdc:mm_1, 1.5, 0.001; Calibrate this voltage with P1 as close as possible to 1.500 V\nA deviation of 1 mV is acceptable.		Show popup with live measurement values to show the parameter to be calibrated.
10				

This script shows this popup:



Notice the red arrow that indicates the desired voltage is too low and should be increased.

6.16 PAUSE

Show a popup message. If started with a 'Q', a YESNO popup is shown. By typing \n a newline is inserted in the text.

The text may contain a 'formula' to print variables in the text. See action PRINT for more details.

Examples:

PAUSE Hello world

Shows text in a standard message box.

PAUSE The voltage is =pos_voltage Volt

The equal sign prints the value of variable 'pos_voltage'.

PAUSE Q[1!A100]Abort now?

Shows a question in a YESNO message box. When YES is clicked, the test continues at cell A100 in sheet 1.

The destination cell MUST contain a command. See also the CONT action.

This action allows a test condition to be passed in the next column.

Example:

	A	B	C	D
8	command	arguments	condition	comment
9	PAUSE	Q[1!A100]Abort now?		Shows a popup with abort question. If YES, continue in cell A100
10				

This script shows this popup:



6.17 PRINT

Prints user text in the specified cell.

The argument should begin with the destination cell ([2!C20+row_offset] e.g.).

It is possible to print variables (or even formulas). This can be achieved by typing the equal sign ('=') followed by the formula. The formula is ended by the first white space found, so don't accidentally put white spaces in your formula! See for more information about formula's the CALC action.

Example:

	A	B	C	D
8	command	arguments	condition	comment
9	PRINT	[2!C20]Output is =vout+voffset Volt.		Print text: "Output is xxxxxx Volt" in cell C20 of sheet 2, where xxxxxx equals the sum of vout and voffset
10	PRINT	[2!C21]=current_time		Print current time in cell C21 of sheet 2
11				

6.18 RET

Returns from a function, which was called by CALL.
This action takes no arguments.

Example: see CALL

6.19 SELECT

Popup a dialog that shows a dropdown box from which the user should select an item. The dialog shows some user text as well. By typing \n a newline is inserted in the text.
This text may contain a 'formula' to print variables in the text. See action PRINT for more details.

Syntax:

```
SELECT defaultno , sel.text0, sel.text1 {, sel.textn};question text
```

The first argument `defaultno` identifies the - zero based - default item that is to be selected when the dialog is shown.

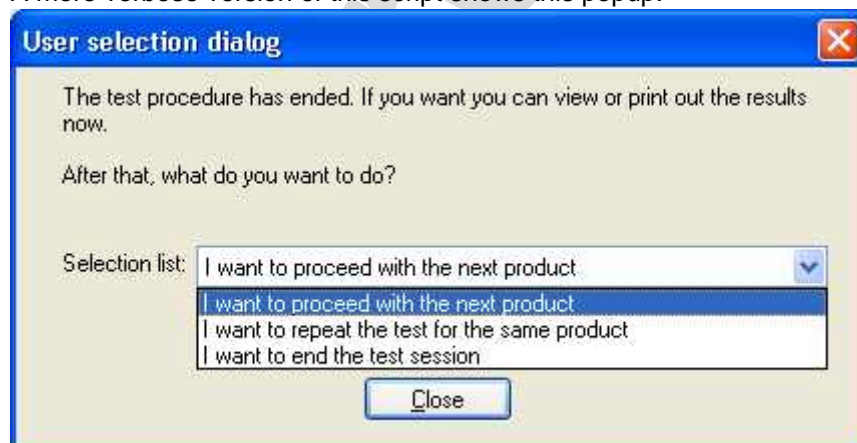
When the user closes the dialog, the index of the selected item is stored in variable 'user_selection'.

This action allows a test condition to be passed in the next column.

Example:

	A	B	C	D
8	command	arguments	condition	comment
9	SELECT	1, Skip, Retry, Abort; Please choose an option		Let user select an option, the default option is 1: Retry. Note: the first option is #0.
10				"user_selection" now contains the selected option number
11				

A more verbose version of this script shows this popup:



6.20 SAVE

Save the Excel file to disk with the specified name.

Example:

```
SAVE      c:\TestmATE\results\product_#.xls
```

Where # is replaced by the serial number. The serial number is set in the main dialog of TestmATE and can be incremented by the script. Please see chapter [6.9: ENDTEST](#).

The destination directory MUST exist, otherwise an error is shown.

Note that the main TestmATE dialog supplies a checkbox that allows the user to disable this SAVE action.

Example: see CLOSE

6.21 TEST

Performs a comparison and jumps to the specified cell when TRUE.

If the comparison is FALSE, the jump is not invoked and procedure just continues.

The follow conditions can be examined:

- greater or equal than >=
- greater than >
- less or equal than <=
- less than <
- equal to ==

Syntax: TEST symbol operator symbol ? [cell reference]

In this case symbol can be a variable, a channel or a formula.

The destination cell MUST contain a command. See also the CONT action.

This action supports semicolon (;) separated arguments.

Examples:

	A	B	C	D
8	command	arguments	condition	comment
9	TEST	abs(Vmeas)>1.001?[1!A40]		If Vmeas is greater than 1.001V goto A40
10	TEST	user_selection==1?[1!A5 0]		if user_selection equals 1 goto A50
11				

6.22 VAR

Explicitly define variables.

For details see chapter [6.2: CALC](#).

Examples:

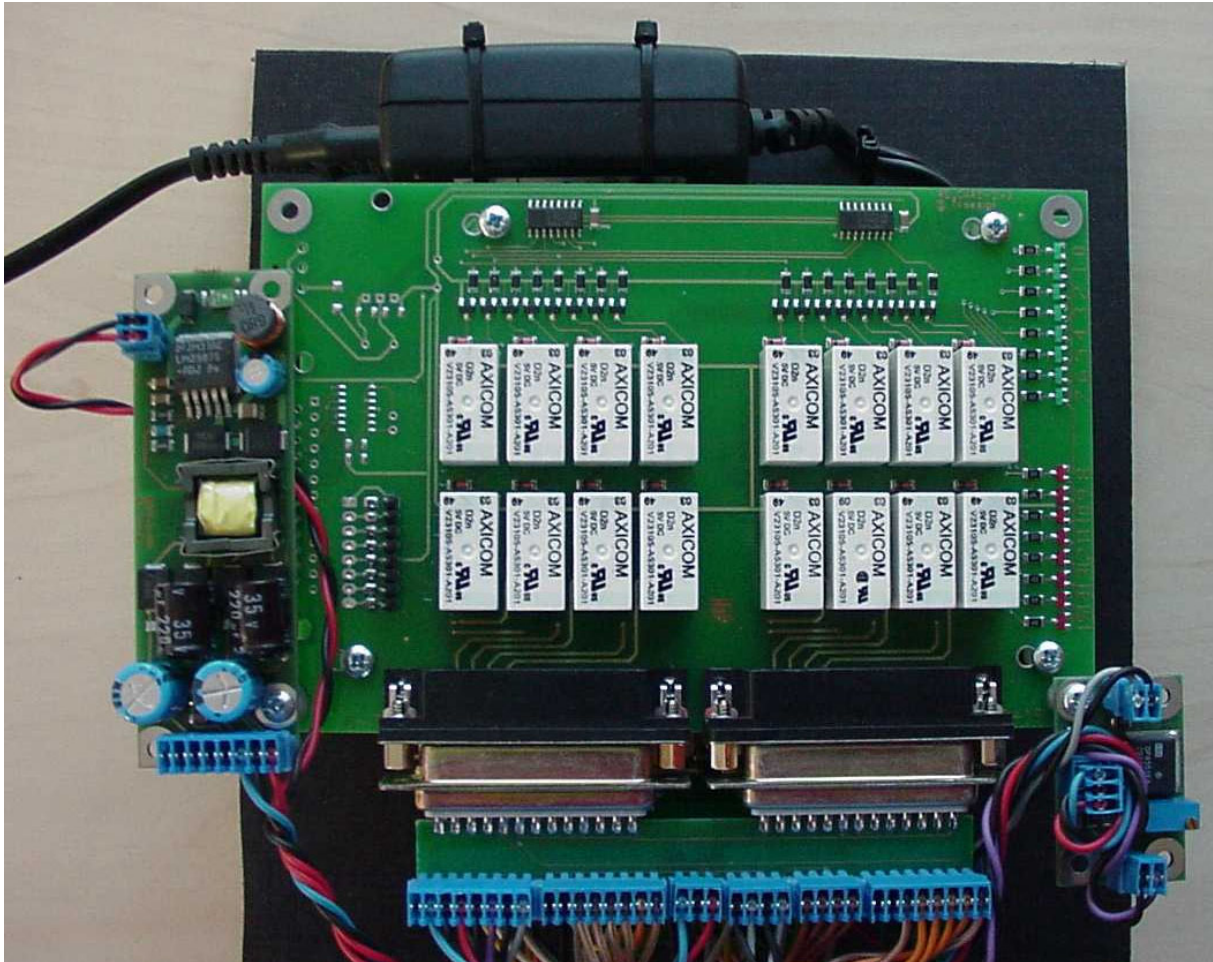
	A	B	C	D
8	command	arguments	condition	comment
9	VAR	vlowlimit = -1		declare variable vlowlimit and initialize vlowlimit with -1
10	VAR	vtemp = 0		declare variable vtemp and set it to 0
11				

6.23 Special global variables

- current_time Get the current time as an integer value.
- exec_result Get the exit code (integer) of an executed command line (DOS) program.
- dllexec_result Get the result argument (double) of the last DLL call.
- dllexec_returncode Get the return code (integer) of the last DLL call.
- user_input Result of INPUT action (double).
- user_selection Result of SELECT action (integer).

Some of our products really have a lot of digital I/O. For that purpose we designed a relaiscard that can be controlled through the PC's parallel port.

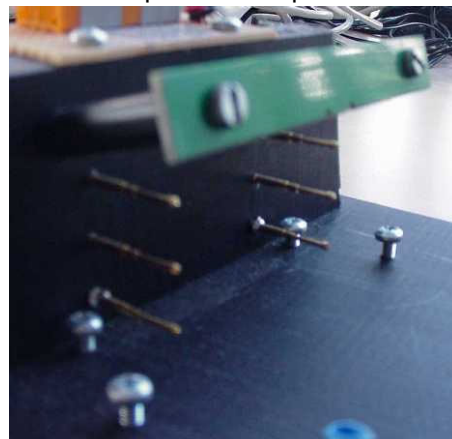
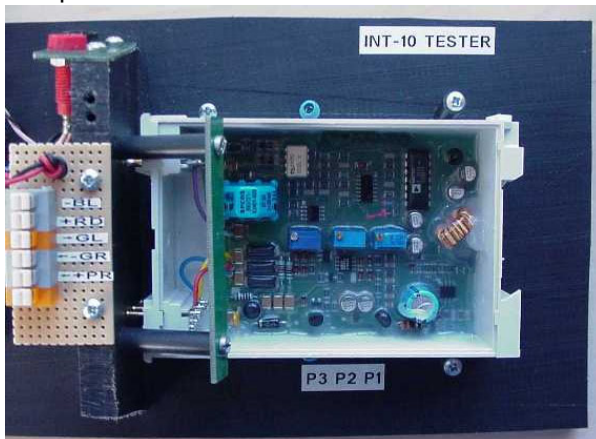
In the set-up shown on the photo, 4 relays cards are stacked.



On the left there is a stamp size power supply added to generate +5, +12, -5 and -12 V. Bottom right is a stamp size V to I converter to act as a current source. The control voltage for this current source is one of the DACs of the 7706 board. We needed so many relays because we had to test a PLC like controller with 40 digital inputs and 16 analogue inputs. As the 7706 has 2 DACs, we use the relays card as a multiplexer to apply the correct voltage to each analogue input.

For every product that has to be tested we make a test board on a plastic base plate. Most of the time this board only contains the connectors to the D.U.T. and the cables to the Keithley and the power supply. Changing the test set-up is done in minutes.

The photos below show a test board with a D.U.T. and a close up of the test pins.



Appendix B: User DLL's

User DLL's provide the interface between TestmATE and an instrument. TestmATE can 'learn' about an instrument like the Device Under Test through the definition of the appropriate DLL.

TestmATE works with 'channels' and an instrument should have at least one 'channel'. The user DLL must translate the channel transactions into the appropriate commands that must be sent over a communication port.

The DLL itself exports instrument properties to TestmATE so the handling of inputs and outputs can be done in a generic way.

Example:

This DLL retrieves the system time and returns it as "mm,ss" to explain the use of DLL's and how to exchange data with DLL's. See also example ['script 7 userdll.xlt'](#)

```
#include <time.h>
#ifdef __cplusplus
extern "C" {
#endif

_declspec(dllexport) int GetTime (char * arg, double * result)
{
    time_t curr_time;
    struct tm * time_struct;

    time( &curr_time );

    time_struct = localtime( &curr_time );

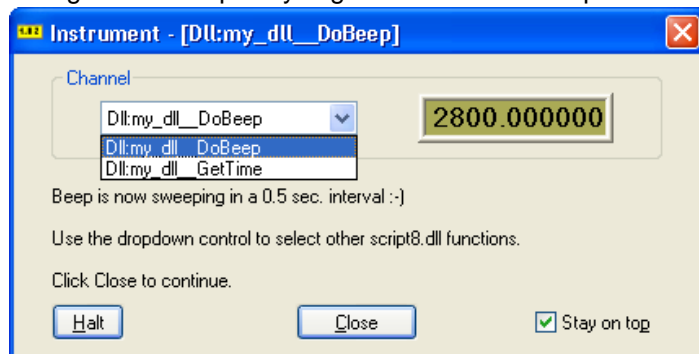
    if (time_struct)
    {
        *result = (double)time_struct->tm_min;
        *result += ((double)time_struct->tm_sec) / 100;
        return 0;
    }

    return -1;
}

#ifdef __cplusplus
}
#endif
```

Note: result will be stored in global variable `dllexec_result`.
the return code (return 0 or -1 in this case) will be stores in `dllexec_returncode`.

The script in ['script 8 beep userdll.xlt'](#) demonstrates the use of arguments and shows this dialog while a frequency is generated on the beeper of the PC.



Index

C

collector redirector, 21
current_time, 38
CUSTOMER, 31

D

D.U.T., 18
datalogger, 5
Device Under Test, 18
DLL, 5
dllexec_result, 38
dllexec_returncode, 38

E

EMPLOYEE, 31
error message, 5
example
 datalogging, 19
examples, 5
Excel, 16
exec_result, 38

G

global variables, 38
 current_time, 35, 38
 CUSTOMER, 31
 dllexec_result, 30, 38
 dllexec_returncode, 30, 38
 EMPLOYEE, 31
 exec_result, 31, 38
 serial number, 29
 SERIALNR, 31, 37
 TIMESTAMP, 31
 user_input, 32, 38
 user_selection, 36, 38

P

production test, 5

R

redirector, 21

S

script

datalogging, 19
formula syntax, 24
how to make your first, 16
parameters, 22
popup, 18
run, 16
serial number, 18
string syntax, 23
time stamp, 19
variable usage, 24
script action
 ADDINSTRUMENT, 25
 CALC, 26
 CALL, 29
 CLOSE, 29
 CONT, 29
 DELAY, 30
 DLLEXEC, 30
 END, 30
 ENDTEST, 30
 EXEC, 31
 HEADER, 31
 INPUT, 32
 JUMP, 32
 LOG, 33
 MEASURE, 34
 PAUSE, 35
 PRINT, 35
 RET, 36
 SAVE, 37
 SELECT, 36
 TEST, 37
 VAR, 38
serial number, 18, 30, 31, 37

T

testprocedure, 5
testscripts, 5
TIMESTAMP, 31

U

user DLL, 5, 41
user_input, 32, 38
user_selection, 36, 38

V

variables, 38
variables, global, 38

Bibliography

This document is based on below mentioned documents, web sites and other sources:

- | | | |
|--|--------------|---------------------------|
| 1. 2700 series Multi meter User's Manual | 2700-0900-01 | Revision A, November 1999 |
| 2. Model 7706 All-in-One module | PA-719 | Revision A |

<LAST PAGE>

Preliminary