1 Change History

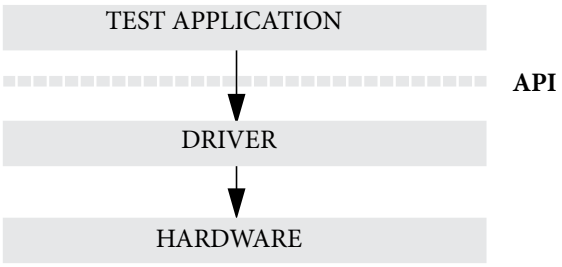
Date	Version	Reason and Change
15 Feb 2000	1.0	Initial release
01 Aug 2000	2.0	New: batch mode, interactive mode, auto-test
09 Aug 2000	2.1	Add a menu before starting the test (the user can set how to run Testtool)
16 Nov 2000	2.2	Add assignment error test
04 Oct 2001	2.3	Add 'If elif else' test. Increase factorial test value.
26 Oct 2001	2.4	Add strings tests.
15 May 2002	2.5	New platform supported.
12 Dec 2002	2.6	New platform supported.
10 Oct 2003	2.7	New platform supported.
01 Apr 2004	2.8	New platform supported.
16 Sep 2004	2.9	New platform supported.
17 Sep 2004	2.9	Add note for OS21 users.
23 Dec 2004	2.10	5105 platform supported.
08 Mar 2005	2.11	7100 platform supported
20 Apr 2005	2.12	5301 platform supported
11 Nov 2005	2.13	7109 platform supported
14 Feb 2006	2.14	5188 & 5525 platforms supported
03 Nov 2006	2.15	Extended the script M_TEST4 to test the command Eval

2 Introduction

This document describe what must be done to test the command line interpreter Testtool which is part of the ST Microelectronics library for Set Top Box.

2.1 Overview

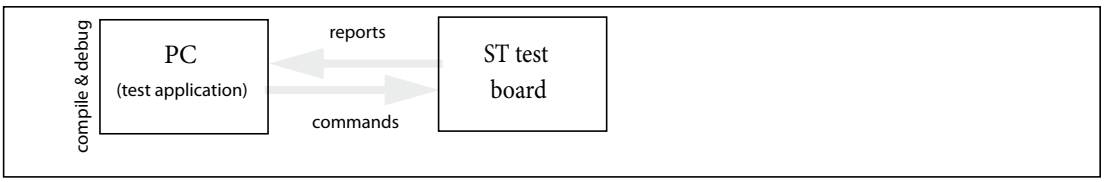
The API includes C-functions which are independents of the hardware component.



The test application need:
- Testtool software,
- DCU Toolset.

2.2 Environment

The test application run on a stand-alone system, with a cross development environment.



Host station	Personal computer with Windows NT 4.0 Pack 6
DCU Toolset	ST20: 1.9.6p6, ST40: 1.8
ST Test Board	Chip STi5510 on MB231
	Chip STi5512 on MB282B
	Chip STi5508 on MB275 rev. A and rev. B
	Chip STi5518 on MB275 rev. A and rev. B
	Chip STi7015 on MB295
	Chip STi7020 on MB295C
	Chip STi5514 on MB314
	Chip STi5516 on MB361
	Chip STi5517 on MB361 / mb382
	Chip ST40GX1 on MB317B
	Chip STi5528 on MB376 with (ST20 and ST40)
	Chip STi5100 on MB390 with ST20C2
	Chip STi7710 on MB391 with ST20C1
	Chip STi5105 on MB400 with ST20C1
	Chip STi7100 on MB411 with ST40
	Chip STi5301 on MB390 with ST200

	Chip STi7109 on MB411 with ST40
	Chip STi5525 on MB428 with ST200
	Chip STi5188 on MB457 with ST20

2.3 Command line interpreter

---> *See TESTTOOL User's Manual*

3 Test plan

3.1 Running test application

1) Open a DOS window, select the directory where the test application is stored, and compile it:

```
CD TESTS\SRC
```

Note for OS21/ST200 users :

```
use cygwin prompt for compile,run and debug applications:

folowing environment variable must be set:

export DVD_OS=OS21

export ARCHITECTURE=ST200

export DVD_HOST=unix
```

or :

```
export DVD_HOST=pc-cygwin
```

Note for OS21/ST40 users :

```
* Define the environnenent variable DVD_POSIX before compilation
If those conditions are not satisfied, it will not be API compliant
because functions related to keyboard input will fail ! As getchar()
function is line buffered on OS21 (it waits until Return key is hit),
it must be replaced by a Posix function. And sh4gdb is mandatory for
Posix function (keyboard input fails with 'make run').

set DVD_POSIX=1

MAKE
```

2) Start the test application in batch mode:

*Important note for OS21/ST40 users :

```
CD OBJS\ST40
```

On OS21 a script must be used.

Here is an example for testtool on board mb376 (6 lines of commands) :

```
exec-file tst_test.exe
```

```
mb376 10.129.101.199
symbol-file tst_test.exe
load tst_test.exe
break main
c
```

Here is an example for testtool on board mb411:

file tst_test.exe

mb411bypass 164.130.60.105

load

break main

c

Adapt the script to the 'exe' file, the board, and IP address of the ST Micro Connect and save it on Objs\ST40.

```
sh4gdb -x start_script -w
```

Note for OS20 users :

```
MAKE RUN
```

Answer to the 2 questions:

```
Do you want to launch the tests stored in default.com    (Y/N) ?
Do you want to start in batch mode (default=interactive) (Y/N) ?
```

If 'Y' or 'y' is answered at the 1st question, an input file will be used for automatic tests.

If 'Y' or 'y' is answered at the 2nd question, the following message appears:

```
System initialised - hit return to enter Testtool or backspace to exit
```

Those questions allow the user to test the different available modes of Testtool (see the chapter about batch/ interactive modes in API documentation) without recompiling the test application.

Non regression tests

Usually, only the automatic tests of the *"main tests"* section, needs to be passed.

The other tests must be passed when the changes may have effects on interactive/batch mode, or on command recall.

Note for OS_LINUX users :

First step :

Load NFS Image : the script "runxxxx" detailed below will be used for this reason :

exemple run7100 :

JEI=IP@ # Name of the JEI, HTI or MicroConnect

TARGETIP= # IP address to be given to the target

SERVERIP= # IP address of your NFS server

GWIP= # IP address of your network gateway

NETMASK= # Local network subnet mask

NAME= # Initial hostname for the target

AUTOCONF=off # Try to determine addresses automatically?

Root of target's file system

SERVERDIR=/opt/STM/STLinux-2.0/devkit/sh4/target

Kernel image

KERNEL=/opt/STM/STLinux-2.0/devkit/kernel/linux-sh4-2.6.11_stm20/vmlinux

/opt/STM/st40load_gdb \

-t $JEI \

-b $KERNEL \

-c mb411bypass \

nwhwconf=device:eth0,hwaddr:00:80:e1:12:02:00 \

console=ttyAS0,115200 \

root=/dev/nfs \

nfsroot=$SERVERIP:$SERVERDIR \

ip=$TARGETIP::$GWIP:$NETMASK:$NAME::$AUTOCONF \

mem=64m \

bigphysarea=8000

#end of run7100

try run this script in order to load kernel image.

Second step :(Compilation and execution)

+Setting all environement variables.

+ Compilation is done like OS21 , Compile the test application and you will find the driver modules generated in the path "\tests\lib\lib\modules".

+ TESTTOOL test application compilation.

+by telnet load them in kernel by (depends degree order) , for exemple loading stcommon module before stavmem module.

```
insmod stcommon_core.ko
```

```
major=`cat /proc/devices | awk "\\$2==\"stcommon_core\" {print \\$1}"`
rm -f /dev/stapi/stcommon_core
mknod /dev/stapi/stcommon_core c $major 0
```

to remove module use the commande below :

```
rmmod stcommon_core.ko
```

then running tst_test.exe

```
./tst_test.exe
```

Then a test menu appears, where you can directly call the macros and set their initial conditions automatically, or you can bypass this facility, by typing another key to exit this menu

At the TestTool prompt load the scripts (command: source <script name>), set the inital condition and type the macros as described in the test plan.

Debug Informations :

+To debug in kernel mode the only way is to use KGDB throw a serial port.

+In user mode,

from the board(telnet/ssh):

```
gdbserver localhost: port number(6789) tst_test.exe
listening
```

on the host under : testtool/tests/src/objs/LINUX :

```
cd Objs\LINUX
sh4-linux-gdb tst_test.exe
(gdb) target remote target IP address : port number(the same given on the
board 6789).
```

3.2 Batch mode with input file (main tests)

The commands of the next test sequence are stored in `../SCRIPTS/DEFAULT.COM` file. Most of the features are covered by those tests.

	Batch mode with input file (automatic tests)
Coverage	- batch mode: no key hit request at start and at end - logging in `DEFAULT.LOG` file with `print` and `STTST_Print()` - **test 1**: macro definition and execution: for & while loops, recursive calls - **test 2**: input tests: pollkey, getkey - **test 3**: intrinsic commands: help, show - **test 4**: expression evaluation: base change, string concatenation, arithmetic and logical operators tests, expression evaluation, string with testtool command evaluation - **test 5**: assignment errors (string too long) - **test 6**: delete in nested macros - **test 7**: string tests. - **test 8**: memory leak tests.
Commands	IF, FOR, WHILE, POLLKEY, GETKEY, HELP, SHOW, DEFINE, DELETE

Test	Run the test application.
	Answer 'Y' and 'Y' to the initial questions.
	Then some commands are automatically launched.
	See the display, and answer to the questions.
	`......`
	`Press on space bar to continue the test sequence`
	`......`
	`Press on 'a' key (minus case)`
	`......`
	`Enter a string and press on Return key`
	`......`
	`Is it ok (Y/N) ?`
	`......`
	`Is it ok (Y/N) ?`
	`......`
	`Is it ok (Y/N) ?`
	`......`
	`Is it ok (Y/N) ?`
	`......`
	
	At the end of the test sequence :
	
	`------ END of TESTTOOL's tests ------`
	
	`RESULT : *** PASSED ***`
	
	Then the test application is automatically finished.
	You can see the traces in DEFAULT.LOG file (see Appendix)

3.3 Batch mode with no input file

	Test batch mode without input file
Coverage	batch mode and lack of input file

Command	EXIT or END
Test	Run the test application. Answer 'N' and 'Y' to the initial questions. The prompt is displayed (without any key to hit). Enter some commands like SHOW or HELP... To quit, type `EXIT` or `END`. Test application is finished (without BackSpace hit).

3.4 Interactive mode with no input file

	Starting interactive mode without input file
Coverage	Interactive modes
Commands	
Test	Start the test application. Answer 'N' and 'N' to the initial questions The following message is displayed: `System initialized - hit return to enter Testtool or backspace to exit` Press a key The prompt `Testtool>` is displayed. Press on Return key. The prompt `Testtool>` is displayed again.

	Help
Coverage	HELP
Command	HELP

Test	Enter: HELP Result: The following commands and macros are defined: DELETE - <symbolnames> Removes named symbols or macros HELP - <commandnames> Displays help string for named commands and macros GETKEY - Waits for one key and returns its value GETSTRING - Waits for a string and returns it HISTORY - <number> Displays the last commands or runs the selected one POLLKEY - Tests if a key was hitten and returns it if any PRINT - Formats and prints variables ... ... Bases = binary (bxx), octal (oxx), decimal (xx), hexadecimal (hxx) print b10+o10+10+h10 is equivalent to print 2+8+10+16 Operators = + - * / (arithmetical) & ^ | ~ ! (logical) Return code = each command returns FALSE if succesful or TRUE if failure some of them returns also a data, like s=getstring Commands = command recall from historic is allowed with control keys ctrl-u=erase last char. ctrl-p=recall last char. ctrl-i=previous command ctrl-o=next command Return code = each command returns FALSE if successful or TRUE if failure some of them also return a data, like s=getstring Commands = command recall from history is allowed with control keys ctrl-u=move cursor left ctrl-p=move cursor right ctrl-i=previous command ctrl-o=next commandVariables = integer, float, string Command and macros can be used as statements or part of expressions, if appropriate, where the assignment is made to a symbol whose value is either defined or modified by this assignment. Generalised assignments can be made to symbols of integer, floating point or string types, including evaluation of arbitrary arithmetic expressions. Commands and assignments can be combined into parameterised macros that use FOR, IF, ELSE and WHILE constructs. Using the DEFINE command these macros can be set up for later invocation. All blocks within these macro constructs are terminated using the END statement. Enter: HELP GE* Result: GETKEY - Waits for one key and returns its value GETSTRING - Waits for a string and returns it

	Command execution
Coverage	Intrinsic and register commands
Command	Each command provided by HELP
Test	For each existing command, type it at the keyboard. Verify if the behavior is like the description given in the user manual.

3.5 Interactive mode with input file

	Test interactive mode with input file
Coverage	interactive mode and input file
Commands	FOR, WHILE, POLLKEY, GETKEY, HELP, SHOW, DEFINE
Test	Start the test application. Answer 'Y' and 'N' to the initial questions Press a key The prompt `Testtool>` is displayed. The automatic test sequence is launched (see previous chapter). Answer to the asked question. The prompt `Testtool>` is displayed. Enter `EXIT` or `END`.

3.6 Changing the run mode

	Change interactive mode to batch mode
Coverage	STTST_SetMode()
Command	TST_SETMODE
Test	Start the test application. Answer 'N' and 'N' to the initial questions Press a key The prompt `Testtool>` is displayed. Type: `TST_SETMODE 1` EXIT The test application is finished

	Change batch mode to interactive mode
Coverage	STTST_SetMode()
Command	TST_SETMODE

Test	Start the test application. Answer 'N' and 'Y' to the initial questions The prompt Testtool> is displayed. Type: `TST_SETMODE 0` EXIT Type Return. The prompt is displayed again. Type `EXIT`. The test application is finished

3.7 Command recall

Here are some other tests to be done in interactive mode, and according the following chronology.

	Command storage
Initial condition	Quit the test application, and start it again.
Coverage	Commands storage
Command	HISTORY

Test	Enter the following commands:
	`PRINT 12`
	`PRINT 123`
	`PRINT 1234`Enter the following commands :
	`HISTORY`
	Display:
	`History of last commands`
	` 0  --`
	` 1  -- PRINT 1`
	` 2  -- PRINT 12`
	` 3  -- PRINT 123`

	Command recall
Initial condition	Previous test passed
Coverage	Command recall with Control keys
Commands	<Ctrl><I> and <Ctrl><O>

Test	Type 4 times on <Ctrl><I> :
	
	The successive displayed lines will be:
	
	`PRINT 1234`
	`PRINT 123`
	`PRINT 12`
	`(empty line)`
	
	Each time, the cursor is at the right end of the line.
	
	Type 4 times on <Ctrl><O> :
	
	The successive displayed lines will be:
	
	`PRINT 12`
	`PRINT 123`
	`PRINT 1234`
	`PRINT 1234`
	
	Each time, the cursor is at the right end of the line.

	Command recall & modification
Initial condition	Previous test passed
Coverage	Command recall with control keys
Commands	<Ctrl><U> and <Backspace>

Test	Press 2 times on <Ctrl><U>, then press <Backspace>, press 8 and <Return>:
	
	The successive displayed lines will be:
	
	`PRINT 1234`
	`PRINT 1234`
	`PRINT 134`
	`PRINT 1834`
	`1834`

	Command recall & modification
Initial condition	Previous test passed
Coverage	Command recall with control keys
Commands	<Ctrl><I> , <Ctrl><P> and <Backspace>

Test	Press <Ctrl><I>, 3 times on <Backspace>, then 3 times on <Ctrl><P>, then press 5 and <Return>:
	
	The successive displayed lines will be:
	
	`PRINT 1834`
	`PRINT 183`
	`PRINT 18`
	`PRINT 1`
	`PRINT 18`
	`PRINT 183`
	`PRINT 1834`
	`PRINT 18345`
	`18345`

	Command recall & modification
Initial condition	Previous test passed
Coverage	Command recall with control keys
Commands	<Ctrl><I>, <Ctrl><U> and <Backspace>

Test	Press <Ctrl><I>, press <Ctrl><U>, press 2 times on <Backspace>:
	
	The successive displayed lines will be:
	
	`PRINT 18345`
	`PRINT 18345`
	`PRINT 1835`
	`PRINT 185`
	Now abort the change with <Ctrl><I> and <Return>. The display is.
	`PRINT 1834`
	`1834`

- APPENDIX -

Example of test results:

```
input from file "../../../scripts/default.com"
Starting execution of DEFAULT.COM file...
Starting the tests sequence...
Logging stopped and log file closed
Logging input and output to file default.log (mode=a)

M_TEST1

------ M_TEST1 ------
Test 2 embedded loops for ... end
Loop 1/5
Loop 1/4
Loop 1/3
Loop 1/2
Loop 1/1
Loop 2/5
Loop 2/4
Loop 2/3
Loop 2/2
Loop 2/1
Loop 3/5
Loop 3/4
Loop 3/3
Loop 3/2
Loop 3/1
Loop 4/5
Loop 4/4
Loop 4/3
Loop 4/2
Loop 4/1
Loop 5/5
Loop 5/4
Loop 5/3
Loop 5/2
Loop 5/1
RESULT: M_FORLOOP SUCCESSFUL (25 loops done)

Test 2 embedded loops while ... end
Loop 1/5
Loop 1/4
Loop 1/3
Loop 1/2
Loop 1/1
Loop 2/5
Loop 2/4
Loop 2/3
Loop 2/2
Loop 2/1
Loop 3/5
Loop 3/4
Loop 3/3
```

```
Loop 3/2
Loop 3/1
Loop 4/5
Loop 4/4
Loop 4/3
Loop 4/2
Loop 4/1
Loop 5/5
Loop 5/4
Loop 5/3
Loop 5/2
Loop 5/1
RESULT: M_WHILELOOP SUCCESSFUL (25 loops done)

Test a recursive function : factorial 12
NUM 2 RES 2
NUM 3 RES 6
NUM 4 RES 24
NUM 5 RES 120
NUM 6 RES 720
NUM 7 RES 5040
NUM 8 RES 40320
NUM 9 RES 362880
NUM 10 RES 3628800
NUM 11 RES 39916800
NUM 12 RES 479001600
RESULT: M_FACTORIAL SUCCESSFUL

Test if .. elif .. elif .. else .. end
RESULT: M_IF_ELIF_ELSE SUCCESSFUL

Test macro return assignment
RESULT: M_RETURN SUCCESSFUL
M_TEST2

------ M_TEST2 ------
Press on space bar to continue the test sequence
You have pressed on 32 key (code for space bar is 32)
RESULT: POLLKEY SUCCESSFUL

Press on 'a' key (minus case)
You have pressed on 97 key (code for letter 'a' is 97)
RESULT: GETKEY SUCCESSFUL
M_TEST3

------ M_TEST3 ------
M_QUESTION (MSG_P) - command macro
M_FORLOOP () - command macro
M_WHILELOOP () - command macro
M_FACTORIAL (VALUE) - command macro
M_IF_ELIF_ELSE () - command macro
M_COMPARISON () - command macro
M_RETURN (VALUE) - command macro
M_TEST1 () - command macro
M_TEST2 () - command macro
M_TEST3 () - command macro
M_TEST4 () - command macro
M_TEST5 () - command macro
```

```
Free blocks size sum : info not available on OS21
Does free blocks size sum and number of symbols at START and END of test the same
(Y/N) ?
RESULT : Memory leak test SUCCESSFUL
print

print "-------------------------------------------------"
-------------------------------------------------
print "Now test with STTST_KEEP_CONTROL_VARIABLE_MODE..."
Now test with STTST_KEEP_CONTROL_VARIABLE_MODE...
print

STTST_MODE=STTST_MODE|STTST_KEEP_CONTROL_VARIABLE_MODE
TST_SetMode STTST_MODE
STTST_SetMode(STTST_BATCH_MODE | STTST_KEEP_CONTROL_VARIABLE_MODE): ok
M_TEST1

------ M_TEST1 ------
Test 2 embedded loops for ... end
Loop 1/5
Loop 1/4
Loop 1/3
Loop 1/2
Loop 1/1
Loop 2/5
Loop 2/4
Loop 2/3
Loop 2/2
Loop 2/1
Loop 3/5
Loop 3/4
Loop 3/3
Loop 3/2
Loop 3/1
Loop 4/5
Loop 4/4
Loop 4/3
Loop 4/2
Loop 4/1
Loop 5/5
Loop 5/4
Loop 5/3
Loop 5/2
Loop 5/1
RESULT: M_FORLOOP SUCCESSFUL (25 loops done)

Test 2 embedded loops while ... end
Loop 1/5
Loop 1/4
```

```
Loop 1/3
Loop 1/2
Loop 1/1
Loop 2/5
Loop 2/4
Loop 2/3
Loop 2/2
Loop 2/1
Loop 3/5
Loop 3/4
Loop 3/3
Loop 3/2
Loop 3/1
Loop 4/5
Loop 4/4
Loop 4/3
Loop 4/2
Loop 4/1
Loop 5/5
Loop 5/4
Loop 5/3
Loop 5/2
Loop 5/1
RESULT: M_WHILELOOP SUCCESSFUL (25 loops done)

Test a recursive function : factorial 12
NUM 2 RES 2
NUM 3 RES 6
NUM 4 RES 24
NUM 5 RES 120
NUM 6 RES 720
NUM 7 RES 5040
NUM 8 RES 40320
NUM 9 RES 362880
NUM 10 RES 3628800
NUM 11 RES 39916800
NUM 12 RES 479001600
RESULT: M_FACTORIAL SUCCESSFUL

Test if .. elif .. elif .. else .. end
RESULT: M_IF_ELIF_ELSE SUCCESSFUL

Test macro return assignment
RESULT: M_RETURN SUCCESSFUL
M_END

------ END of TESTTOOL's tests ------

TST_GetRevision(): STTST-REL_3.3.0

RESULT : *** PASSED ***
close
```

```
M_TEST6 () - command macro
M_TEST7 () - command macro
M_TEST8 () - command macro
M_END () - command macro
There is 16 command macros in the list : is it ok (Y/N) ?

Currently defined symbols:

 0: HEXADECIMAL: 16 - integer constant
 0: DECIMAL: 10 - integer constant
 0: OCTAL: 8 - integer constant
 0: BINARY: 2 - integer constant
 0: TRUE: 1 - integer constant
 0: FALSE: 0 - integer constant
 0: PI: 3.14159 - floating point constant
 0: COMMAND_FILE: "default.com" - string variable
 0: LOG_FILE: "default.log" - string variable
 0: ZERO: 0 - integer constant
 0: BASEADDRESS: -2147483648 - integer variable
 0: RANGE: 256 - integer variable
 0: DATAVALUE: 0 - integer variable
 0: ADDRESSVALUE: -2147483648 - integer variable
 1: STTST_INTERACTIVE_MODE: 0 - integer constant
 1: STTST_BATCH_MODE: 1 - integer constant
 1: STTST_NO_ABBREVIATION_MODE: 2 - integer constant
 1: STTST_HIT_KEY_TO_ENTER_MODE: 4 - integer constant
 1: STTST_KEEP_CONTROL_VARIABLE_MODE: 8 - integer constant
 1: STTST_MODE: 1 - integer variable
 1: ERR_NB: 0 - integer variable
 1: M_QUESTION: (MSG_P) - command macro
 1: M_FORLOOP: () - command macro
 1: M_WHILELOOP: () - command macro
 1: M_FACTORIAL: (VALUE) - command macro
 1: M_IF_ELIF_ELSE: () - command macro
 1: M_COMPARISON: () - command macro
 1: M_RETURN: (VALUE) - command macro
 1: M_TEST1: () - command macro
 1: M_TEST2: () - command macro
 1: M_TEST3: () - command macro
 1: M_TEST4: () - command macro
 1: M_TEST5: () - command macro
 1: M_TEST6: () - command macro
 1: M_TEST7: () - command macro
 1: M_TEST8: () - command macro
 1: M_END: () - command macro
 1: KEYCODE: 121 - integer variable
 2: I: 0 - integer variable
 2: STR: "HELLO" - string variable
The list above should contain :
 0: TRUE: 1 - integer constant
 1: KEYCODE: 121 - integer variable
 2: STR: "HELLO" - string variable
Is it ok (Y/N) ?
RESULT : HELP and SHOW tests sucessful
M_TEST4

------ M_TEST4 ------
Expressions evaluation...
```

```
((1+2)*3)/4=2
5+6*7%%8=47
$10+o10+10+#10=36
b10+o10+10+h10=36
$10*O10*10*#10=2560
B10*O10*10*H10=2560
This string is the concatenation of 8 chains
(b100|b111)+(b100^b111)+(b100&b111)=7+3+4=14
(~b00000000)+(~b11111111)+b01111111+b11111111+!b11111111=-1-256+127+255+0=125
Limit values assignments in each base ...
Is it ok (no error message) (Y/N) ?
UNDEFINED_MACRO
^              ^
Unrecognised command statement
Is there exactly 1 error message above (Y/N) ?
SIMPLE_MACRO macro execution success
SIMPLE_MACRO macro execution success
Are the two previous lines identical (Y/N) ?
PRINT command execution succeeded
PRINT command execution succeeded
Are the two previous lines identical (Y/N) ?
30+40+50=120
30+40+50=120
Are the two previous lines identical (Y/N) ?

Test comparison with OR (||), AND (&&), NOT (!)
RESULT: M_COMPARISON SUCCESSFUL

RESULT : evaluation SUCCESSFUL
M_TEST5

------ M_TEST5 ------
Try to make a too big string...
symbol string value too long
  STR=STR+STR
     ^       ^
Unrecognised assignment statement
Is there exactly 1 error message above (Y/N) ?
RESULT : error test SUCCESSFUL
M_TEST6

------ M_TEST6 ------
Use delete in nested macros...
[NESTED_MACRO] deleted
[NESTED_MACRO] deleted
Is nested macro deleted twice (Y/N) ?
RESULT : deletion test SUCCESSFUL
M_TEST7

------ M_TEST7 ------
Strings print tests ...
 3 - Init.+open+start playing ../../../scripts/susie.m1v from memory
 3 - Init.+open+start playing ../../../scripts/susie.m1v from memory
Are the two previous lines identical (Y/N) ?
<-  10  ->-  10  ->-  10  ->-  10  ->-  10  ->-  10  ->-  10  ->-  10  ->-
10  ->-  10  ->-  10  ->-  10  ->-  10  ->-  10  ->-  10  ->-  10  ->-  10
->-  10  ->-  10  ->-  10  ->-  10  ->-  10  ->-  10  ->-  10  ->-  10  ->Ok
Does the previous line ends by 'Ok' (Y/N) ?
```

```
RESULT : strings test SUCCESSFUL
M_TEST8

------ M_TEST8 ------
Memory leak tests ...

### START of test memory statistics
Free blocks size sum : info not available on OS21

### MIDDLE of test memory statistics
Free blocks size sum : info not available on OS21

### END of test memory statistics
```