

Knowledgebase > PRACTICE > How can I evaluate values displayed within a TRACE32 window in a PRACTICE script?

How can I evaluate values displayed within a TRACE32 window in a PRACTICE script?

2025-07-19 - Comments (0) - PRACTICE

The default method that always works is to print the TRACE32 window into a file, and then parse this file using specific PRACTICE commands and functions.

Example: Parsing a TASK.STacK window

let's consider that you want to get the different values displayed in the following

TASK.STacK window:

B::TASK.STacK																- (
name		high	sp		lowest	spare	max	0	10	20	30	40	50	60	70	80	90
SieveDemo	20002AC8	20002CC8	20002C28	31%	20002BC0	00000F8	51%	-					_				
IDLE	20002F68	20003168	20003120	14%	20003134	000001CC	10%										
StackEater	20002D18	20002F18	20002EA0	23%	20002D44	0000002C	91%										_
QueueCons	20002878	20002A78	200029E0	29%	20002908	00000160	31%										

The first step is to print the window into a file using the **PRinTer.FILE** and **WinPrint** commands:

```
PRinTer.FILE test.txt // redirect the printer output to the file
test.txt
WinPrint.TASK.STack // send the content of the window to the printer
The next step is to parse the file using the PRACTICE commands OPEN, READ and CLOSE
as well as the STRing.* PRACTICE functions
// declare macros
PRIVATE &stack line
PRIVATE &task name &low &high &sp &lowest &spare &max &tmp
// open the file for reading and writing
OPEN #1 test.txt
// read the first line of the file, which corresponds to the command
name, into the macro &stack line
READ #1 %LINE &stack_line
// read the second line of the file, which corresponds to the column
names, into the macro &stack line
READ #1 %LINE &stack_line
```

// iterate over the rest of lines until the end of the file is
reached:

```
RePeaT
(
 // read one line
 READ #1 %LINE &stack line
 // Abort when reading an empty line
 IF "&stack line"==""
   ENDD0
 // extract name and emove unnecessary spaces
 &task name=STRing.SPLIT("&stack line","|",0)
 &task name=STRING.TRIM("&task name")
 // extract low and high values
 &tmp=STRing.SPLIT("&stack_line","|",1)
 &low=STRing.SPLIT("&tmp"," ",0)
 &high=STRing.SPLIT("&tmp"," ",1)
 // extract sp
 &sp=STRing.SPLIT("&stack_line","|",2)
 &sp=STRing.SPLIT("&sp"," ",0)
 // extract lowest, spare and max
 &tmp=STRing.SPLIT("&stack line","|",3)
 &lowest=STRing.SPLIT("&tmp"," ",0)
 &spare=STRing.SPLIT("&tmp"," ",1)
 &max=STRing.SPLIT("&tmp"," ",-1)
 // print results
 AREA.view
 PRINT "------"
 PRINT "Task name:
                                  &task name"
 PRINT "Low stack address:
                                  &low"
 PRINT "High stack address:
                                  &high"
 PRINT "SP value:
                                   &sp"
 PRINT "Lowest used stack address: &lowest"
 PRINT "Spare bytes:
                                   &spare"
 PRINT "Max stack space used:
                                   &max"
```

```
)
```

WHILE !FILE.EOFLASTREAD()

// close the file
CLOSE #1

Final Results

Here is an example of the parsed output displayed in the A**REA.view** window:

= 1	-	A 10 4	A.vi	
_	- R	ARE	- A NO	LOO A
	D		- VI	CV

Task name:QueueConsLow stack address:20002878High stack address:200029E0Lowest used stack address:200029D8Spare bytes:00000160Max stack space used:31%Task name:SieveDemoLow stack address:20002C88High stack address:20002C88Lowest used stack address:20002C88Lowest used stack address:20002C88Lowest used stack address:20002BC0Spare bytes:000000F8Max stack space used:51%Task name:IDLELow stack address:20002F68High stack address:20003168SP value:20003120Low stack address:20003120Low stack address:20003120Lowest used stack address:20003134Spare bytes:000001CC	B::AREA.view		
ow stack address: 20002AC8 righ stack address: 20002C88 P value: 20002BC0 pare bytes: 000000F8 ax stack space used: 51%	ow stack address: igh stack address: P value: owest used stack address: pare bytes:	20002878 20002A78 200029E0 200029D8 00000160	
ow stack address: 20002F68 tigh stack address: 20003168 P value: 20003120 owest used stack address: 20003134	ow stack address: igh stack address: P value: owest used stack address: pare bytes:	20002AC8 20002CC8 20002C88 20002BC0 000000F8	
lax stack space used: 10%	ow stack address: igh stack address: P value: owest used stack address: pare bytes:	20002F68 20003168 20003120 20003134 000001CC	
Fask name: StackEater .ow stack address: 20002D18 High stack address: 20002F18 SP value: 20002EA0 .owest used stack address: 20002D44 Spare bytes: 0000002C Max stack space used: 91%	ow stack address: igh stack address: P value: owest used stack address: pare bytes:	20002D18 20002F18 20002EA0 20002D44 0000002C	

References

The PRACTICE commands and functions used in this script are described in the following documentation:

- PRinTer.FILE and WinPrint: PowerView Command Reference
- OPEN, READ and CLOSE: PRACTICE Script Language Reference Guide
- **STRing.*** PRACTICE functions: <u>PowerView Function Reference</u>

Alternative Approach

It is possible to get the values displayed in some TRACE32 windows using specific PRACTICE functions directly. For example, refer to the description of the sYmbol.List.MAP.<x>() functions in General Function Reference.