



[Knowledgebase](#) > [PRACTICE](#) > [How can I evaluate values displayed within a TRACE32 window in a PRACTICE script?](#)

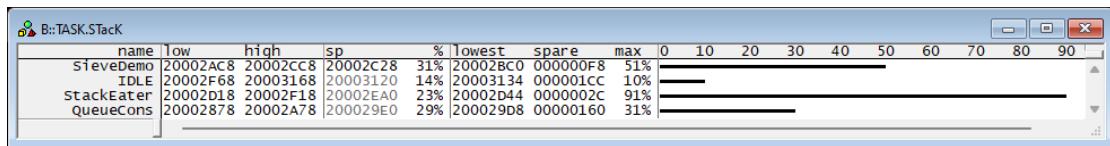
How can I evaluate values displayed within a TRACE32 window in a PRACTICE script?

2026-02-24 - [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:



The screenshot shows a Windows-style window titled "B::TASK.STack". The window contains a table with the following data:

name	low	high	sp	%	lowest	spare	max	0	10	20	30	40	50	60	70	80	90
SteveDemo	20002AC8	20002C88	20002C28	31%	20002BC0	000000F8	51%	0	10	20	30	40	50	60	70	80	90
IDLE	20002F68	20003168	20003120	14%	20003134	000001CC	10%	0	10	20	30	40	50	60	70	80	90
StackEater	20002D18	20002F18	20002EA0	23%	20002D44	0000002C	91%	0	10	20	30	40	50	60	70	80	90
QueueCons	20002878	20002A78	200029E0	29%	200029D8	00000160	31%	0	10	20	30	40	50	60	70	80	90

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"==""
        ENDDO

    // extract name and remove 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"

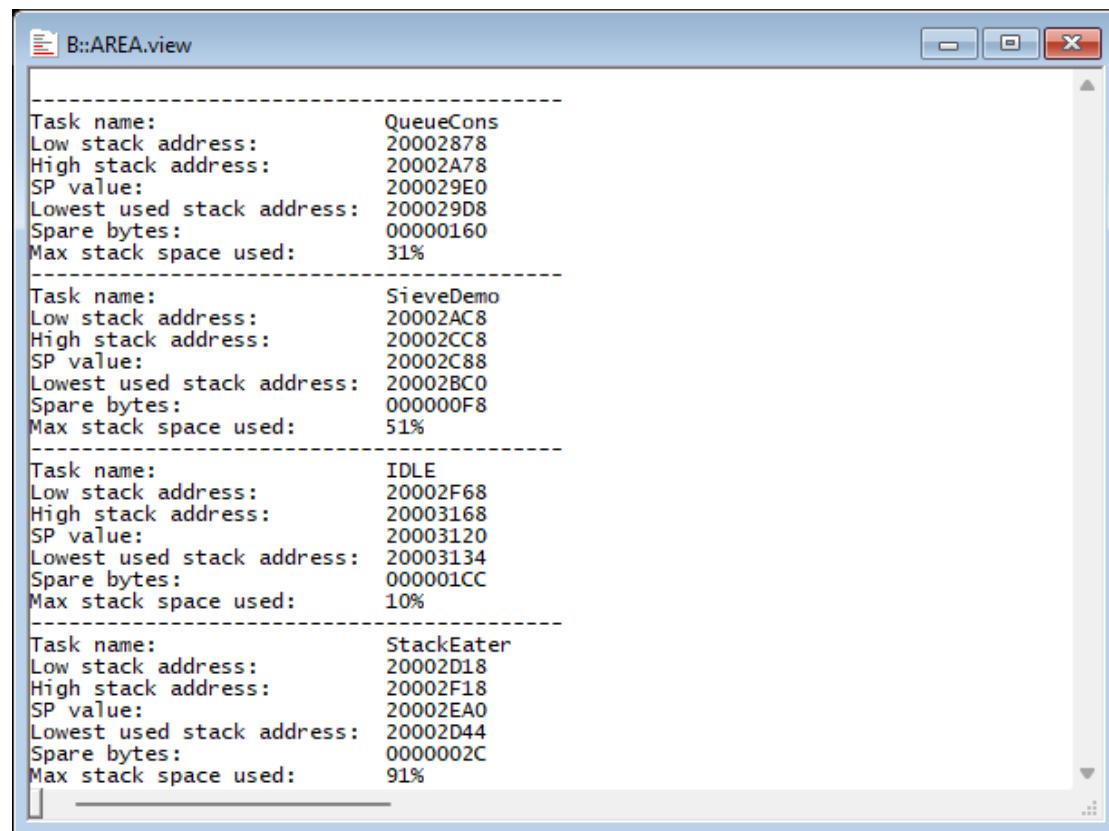
)
WHILE !FILE.EOFLASTREAD()

// close the file
CLOSE #1

```

Final Results

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



The screenshot shows a window titled "B::AREA.view" containing the parsed output of the script. The output is organized into four sections, each representing a task. Each section contains the following information:

- Task name: (e.g., QueueCons, SieveDemo, IDLE, StackEater)
- Low stack address: (e.g., 20002878, 20002AC8, 20002F68, 20002D18)
- High stack address: (e.g., 20002A78, 20002CC8, 20003168, 20002F18)
- SP value: (e.g., 200029E0, 20002C88, 20003120, 20002EA0)
- Lowest used stack address: (e.g., 200029D8, 20002BC0, 20003134, 20002D44)
- Spare bytes: (e.g., 00000160, 000000F8, 000001CC, 0000002C)
- Max stack space used: (e.g., 31%, 51%, 10%, 91%)

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: [PowerView Function Reference](#)

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](#).