How can I read/write from/to memory in a PRACTICE script?

2023-03-03 - Comments (0) - PRACTICE

Writing Raw Memory

Writing memory locations (or writing to memory mapped registers) from PRACTICE scripts works with the command

Data.Set < *address* > %< *format* > < *value* >

Note, that hexadecimal values should be written with prefix "0x", while decimal values must end with a dot "."

Examples:

Data.Set D:0x1000 %Long 0x12345678
Data.Set D:0x3000 %Long 42.

Thereby, "format" specifies the width of the data to be written, where...

%Byte stands for 8-bit

%Word stands for 16-bit

%Long stands for 32-bit

%Quad stands for 64-bit

(There are also format specifiers for a width of 3,5,6,7 bytes)

Instead of using a fixed address, you can also use a label if you have loaded your ELF file already. E.g.:

Data.Set _IntVec1 %Long 0x800000

Note, that for writing of high level variables (which have been declared in the C/C++ code of your application), it is more convenient to use command **Var.set**. (See below)

Reading Raw Memory

Reading memory locations (or reading to memory mapped registers) from PRACTICE scripts

works with the PRACTICE function

Data.< *value width* >()

Where **< *value width* >** stands again for **Byte**, **Word**, **Long**, **Quad** (see above).

PRACTICE functions must be used within a PRACTICE command or have to be assigned to a PRACTICE macro.

Examples:

```
ECHO DATA.Long(D:0x1000)

PRIVATE &myvalue
&myvalue=DATA.Word(D:0x4020)
ECHO "My value is " &myvalue

Data.Set D:0x1000 %Long DATA.Long(D:0x1000)|1
```

The third example shows a way to do a read-modify-write with the purpose to set the least significant bit in the 32-bit value at address 0x1000.

Note, that for reading of high level variables (which have been declared in the C/C++ code of your application), it is more convenient to use function **Var.VALUE()** (See below)

Writing Variables

Writing a variable of your application via a PRACTICE script (after loading the ELF) works with the command

Var.set < *variable name* >=< *value* >

Examples:

```
Var.set myvar=0x100
Var.set flags=flags|0x10
```

The second example performs a read-modify-write with the purpose to set bit 8 of a variable named "flags".

Note, that all commands staring with **Var.** are special in the way that they deal with "High Level Language" expressions, which basically means "Like you would do it in C/C++". So here it is not needed that decimal values are followed by a dot. See also chapter **"Change**

a Variable Value" in Training HLL Debugging.

Reading Variables

Reading a variable of your application via a PRACTICE script (after loading the ELF) works with the function

Var.VALUE()

Examples:

```
ECHO Var.VALUE(myvar)
WAIT (Var.VALUE(flags)&0x10)==0x10
```

The second example stalls the execution of the script until in the variable named "flags" the bit 8 is set.

Related documents:

- Training for the PRACTICE script language
- Training HLL Debugging
- General Commands Reference Guide D (with details about command Data.Set)
- General Commands Reference Guide V (with details about command Var.set)
- General Function Reference (with details about functions Data.() and Var.VALUE())