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

2026-08-11 - Comments (0) - PRACTICE

Writing Raw Memory

To write to memory locations or memory-mapped registers from a PRACTICE script, use the command:

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

Examples:

```
Data.Set D:0x1000 %Long 0x12345678
```

```
Data.Set D:0x3000 %Long 42.
```

The `format` specifies the width of the data to be written:

- **`%Byte`** — 8-bit
- **`%Word`** — 16-bit
- **`%Long`** — 32-bit
- **`%Quad`** — 64-bit

There are also format specifiers for data widths of 3, 5, 6, and 7 bytes.

Instead of a fixed address, you can use a label if the ELF file has already been loaded. For example:

```
Data.Set _IntVec1 %Long 0x800000
```

Note

For writing high-level variables declared in the C/C++ application, it is more convenient to use the **`Var.set`** command (see below).

Reading Raw Memory

To read memory locations or memory-mapped registers from a PRACTICE script, use the corresponding PRACTICE function:

```
Data.<value width>()
```

`<value width>` can be **`Byte`**, **`Word`**, **`Long`**, or **`Quad`** (see above).

PRACTICE functions must be used within a PRACTICE command or 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 how to perform a read-modify-write operation to set the least significant bit of the 32-bit value at address `0x1000`.

Note

For reading high-level variables declared in the C/C++ application, it is more convenient to use the **`Var.VALUE()`** function (see below)

Writing Variables

To write an application variable from a PRACTICE script, after loading the ELF file, use 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 operation to set bit 8 of the variable `flags`.

Note

Commands starting with **`Var.`** are special in that they operate on **High-Level Language** expressions, essentially using C/C++-like syntax. Therefore, decimal values do not need to be followed by a dot.

See also the chapter **"Change a Variable Value"** in Training Source Level Debugging.

Reading Variables

To read an application variable from a PRACTICE script, after loading the ELF file, use the function:

```
Var.VALUE()
```

Examples:

```
ECHO Var.VALUE(myvar)

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

The second example pauses the script until bit 8 of the variable `flags` is set.

Related documents:

- Training for the PRACTICE script language
- Training Source Level 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())