LAUTERBACH
DEVELOPMENT TOOLS

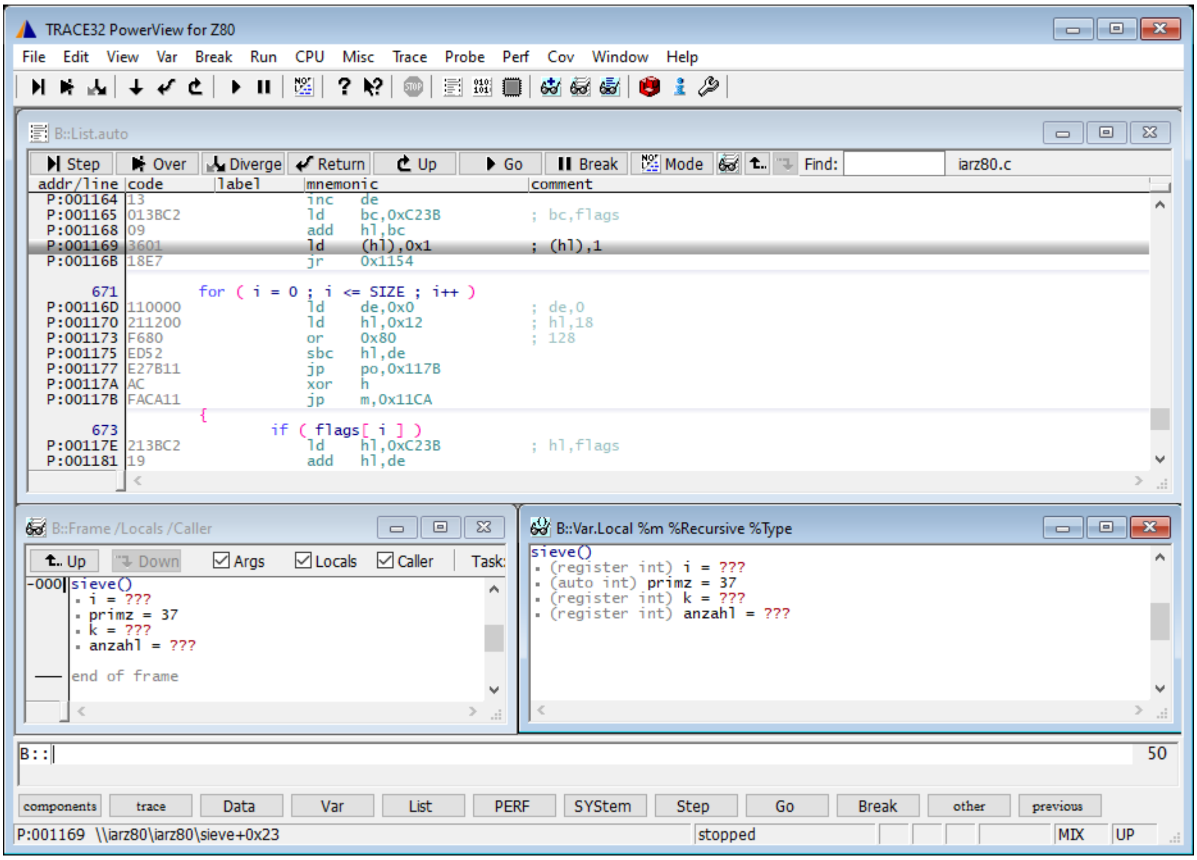
Simulator for Z80+

MANUAL

Simulator for Z80+

Simulator for Z80+

Version 06-Jun-2024


TRACE32 PowerView for Z80
File Edit View Var Break Run CPU Misc Trace Probe Perf Cov Window Help
B::List.auto
Step Over Diverge Return Up Go Break Mode Find: iarz80.c
B::Frame /Locals /Caller
Up Down Args Locals Caller Task:
B::Var.Local %m %Recursive %Type
components trace Data Var List PERF SYStem Step Go Break other previous

```
addr/line |code    |label  |mnemonic           |comment
 P:001164 |13              inc    de
 P:001165 |013BC2          ld     bc,0xC23B      ; bc,flags
 P:001168 |09              add    hl,bc
 P:001169 |3601            ld     (hl),0x1       ; (hl),1
 P:00116B |18E7            jr     0x1154

      671      for ( i = 0 ; i <= SIZE ; i++ )
 P:00116D |110000          ld     de,0x0         ; de,0
 P:001170 |211200          ld     hl,0x12        ; hl,18
 P:001173 |F680            or     0x80           ; 128
 P:001175 |ED52            sbc    hl,de
 P:001177 |E27B11          jp     po,0x117B
 P:00117A |AC              xor    h
 P:00117B |FACA11          jp     m,0x11CA
               {
      673          if ( flags[ i ] )
 P:00117E |213BC2          ld     hl,0xC23B      ; hl,flags
 P:001181 |19              add    hl,de
```

```
-000|sieve()
    | i = ???
    | primz = 37
    | k = ???
    | anzahl = ???

 ---|end of frame
```

```
sieve()
  (register int) i = ???
  (auto int) primz = 37
  (register int) k = ???
  (register int) anzahl = ???
```

```
B::|                                                  50
```

```
P:001169  \\iarz80\iarz80\sieve+0x23    stopped    MIX  UP
```

All general commands are described in the **“PowerView Command Reference”** (ide_ref.pdf) and **“General Commands Reference”**.

TRACE32 Simulator License

[build 68859 - DVD 02/2016]

The extensive use of the TRACE32 Instruction Set Simulator requires a *TRACE32 Simulator License*.

For more information, see **www.lauterbach.com/sim_license.html**.

Quick Start of the Simulator

To start the simulator, proceed as follows:

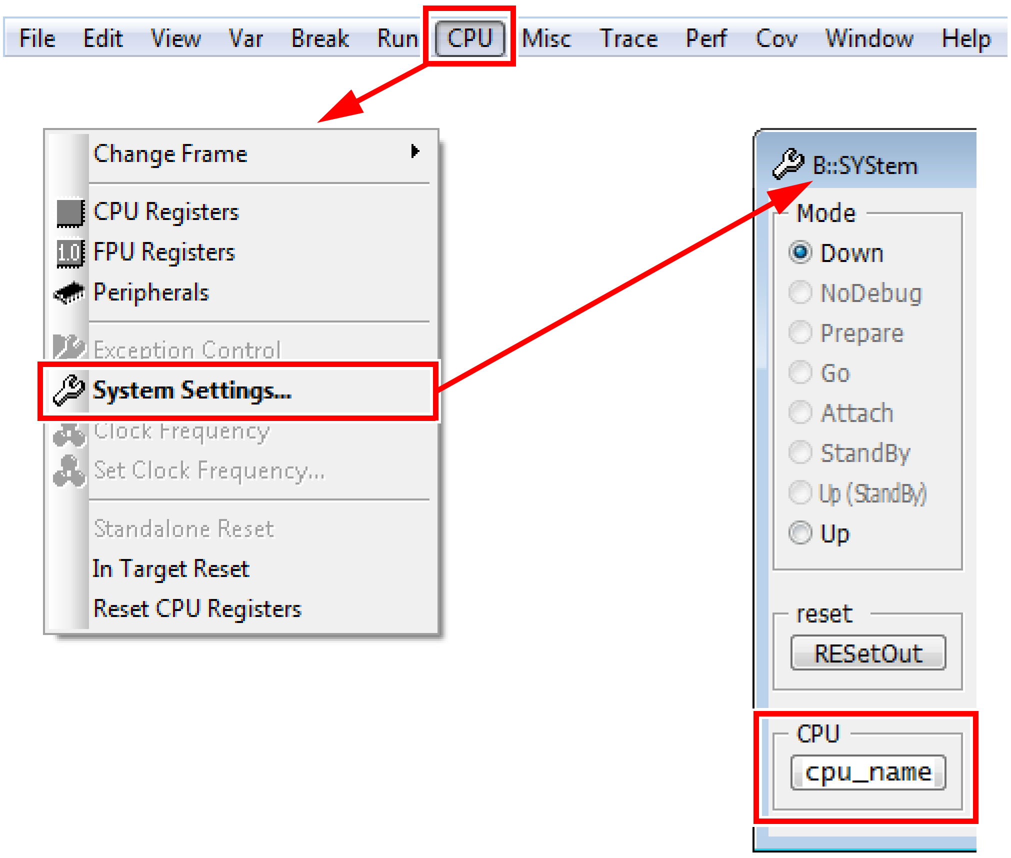
1. Select the device prompt for the Simulator and reset the system.

```
B::

RESet
```

The device prompt `B::` is normally already selected in the TRACE32 command line. If this is not the case, enter `B::` to set the correct device prompt. The **RESet** command is only necessary if you do not start directly after booting TRACE32.

2. Specify the CPU specific settings.



```
SYStem.CPU <cpu_name>
```

The default values of all other options are set in such a way that it should be possible to work without modification. Please consider that this is probably not the best configuration for your target.

3. Enter debug mode.

```
SYStem.Up
```

This command resets the CPU and enters debug mode. After this command is executed it is possible to access memory and registers.

4. Load the program.

```
Data.LOAD.<file_format> <file>      ; load program and symbols
```

See the **Data.LOAD** command reference for a list of supported file formats. If uncertain about the required format, try **Data.LOAD.auto**.

A detailed description of the **Data.LOAD** command and all available options is given in the reference guide.

5. Start-up example

A typical start sequence is shown below. This sequence can be written to a PRACTICE script file (*.cmm, ASCII format) and executed with the command **DO** *<file>*.

```
B::                                 ; Select the ICD device prompt

WinCLEAR                            ; Clear all windows

SYStem.CPU <cpu_name>               ; Select CPU type

SYStem.Up                           ; Reset the target and enter
                                    ; debug mode

Data.LOAD.<file_format> <file>      ; Load the application

Register.Set pc main                ; Set the PC to function main

PER.view                            ; Show clearly arranged
                                    ; peripherals in window      *)

List.Mix                            ; Open source code window    *)

Register.view /SpotLight            ; Open register window       *)

Frame.view /Locals /Caller          ; Open the stack frame with
                                    ; local variables            *)

Var.Watch %Spotlight flags ast      ; Open watch window for
                                    ; variables                  *)
```

*) These commands open windows on the screen. The window position can be specified with the **WinPOS** command.

Peripheral Simulation

For more information, see **“API for TRACE32 Instruction Set Simulator”** (simulator_api.pdf).

Troubleshooting

No information available.

FAQ

Please refer to https://support.lauterbach.com/kb.

Emulation Modes

SYStem.Mode — Establish the communication with the simulator

Format:	**SYStem.Mode** *<mode>* **SYStem.Down** **(alias for SYStem.Mode Down)** **SYStem.Up** **(alias for SYStem.Mode Up)**
<mode>:	**Down** **NoDebug** **Go** **Up**

Default: Down.

Selects the target operating mode.

Down	The CPU is in reset. Debug mode is not active. Default state and state after fatal errors.
NoDebug	The CPU is running. Debug mode is not active. Debug port is tristate. In this mode the target should behave as if the debugger is not connected.
Go	The CPU is running. Debug mode is active. After this command the CPU can be stopped with the break command or if any break condition occurs.
Up	The CPU is not in reset but halted. Debug mode is active. In this mode the CPU can be started and stopped. This is the most typical way to activate debugging.

If the mode **Go** is selected, this mode will be entered, but the control button in the **SYStem.state** window jumps to the mode **Up**.

Format:	**SYStem.CPU** <*mode*>
<*mode*>:	**Z80** | **Z180** | **Z181** | **Z182**

Selects the processor type. The ROM debugger requires also a modification in the debug monitor for different processor types.

SYStem.Option.BASE — Base address of internal registers

Format:	**SYStem.Option.BASE** <*address*>

Defines the base address of the internal registers.

Using the MMU for Z180

This command and the commands **MMU** support the built-in MMU of the Z180 processors.

The analyzer and all memory systems and breakpoints are based on the physical address. The display in the analyzer can be both physical or logical addresses. A logical address can have two formats: smaller than 64K or larger. Smaller addresses are assumed to be an logical address as seen by the CPU in the current MMU configuration. If an address is larger than 64K, the address bits A16 to A23 define the bank base address used for the BBR or CBR register. Logical above 64K addresses should only be used, if the MMU registers were already setup. The following schematic shows these relations for some examples:

```
preset: CBAR=84, BBR=10, CBR=20

logical address:    5    0     4    5    6    7    (Hex)
                    |    |   |    16 bit        |
               CBR/BBR = 50  logical CPU address

                      -->  physical address: 54567


logical address:    0    0     1    5    6    7    (Hex)
                    |    |   |    16 bit        |
                 current-mmu logical CPU address

                      -->  physical address:  1567


logical address:    0    0     4    5    6    7    (Hex)
                    |    |   |    16 bit        |
                 current-mmu logical CPU address

                      -->  MMU Bank Area
                      -->  physical address: 04567
                                  +BBR    10---
                                         =14567


logical address:    0    0     c    d    e    f    (Hex)
                    |    |   |    16 bit        |
                 current-mmu logical CPU address

                      -->  COMMON1 Area
                      -->  physical address: 0cdef
                                  +CBR   +20---
                                         =2cdef
```

To activate the correct address translation for breakpoints, the **MMU** command must be activated. The following example loads a banked application:

```
mmu.off
map.rom 0x0--0x7ffff
…
mmu.on
d.load.u iarz180.dbg
```

The next example loads a banked application in two logical units:

```
CBAR=84, CBR=0, BBR=10 or 20
mmu.reset
symbol.reset
map.rom 0x--0x7ffff
…
mmu.create 0x104000--0x107fff
mmu.create 0x204000--0x207fff
mmu.on
d.load.b bank1.cod 0x104000 /nosymbol
d.load.b bank2.cod 0x204000 /nosymbol
d.load.b common.cod 0x2000 /nosymbol
d.load.sym bank1.sym /noclear
symbol.reloc p:100000 0x4000--0x4fff
d.load.sym bank2.sym /noclear
symbol.reloc p:200000 0x4000--0x4fff
d.load.sym common.sym /noclear
```

Memory Classes

Memory Class	Description
D	Data
P	Program
C	Memory access by CPU
E	Emulation memory access
A	Absolute (physical) memory access