LAUTERBACH
DEVELOPMENT TOOLS

Simulator for 68K/ColdFire

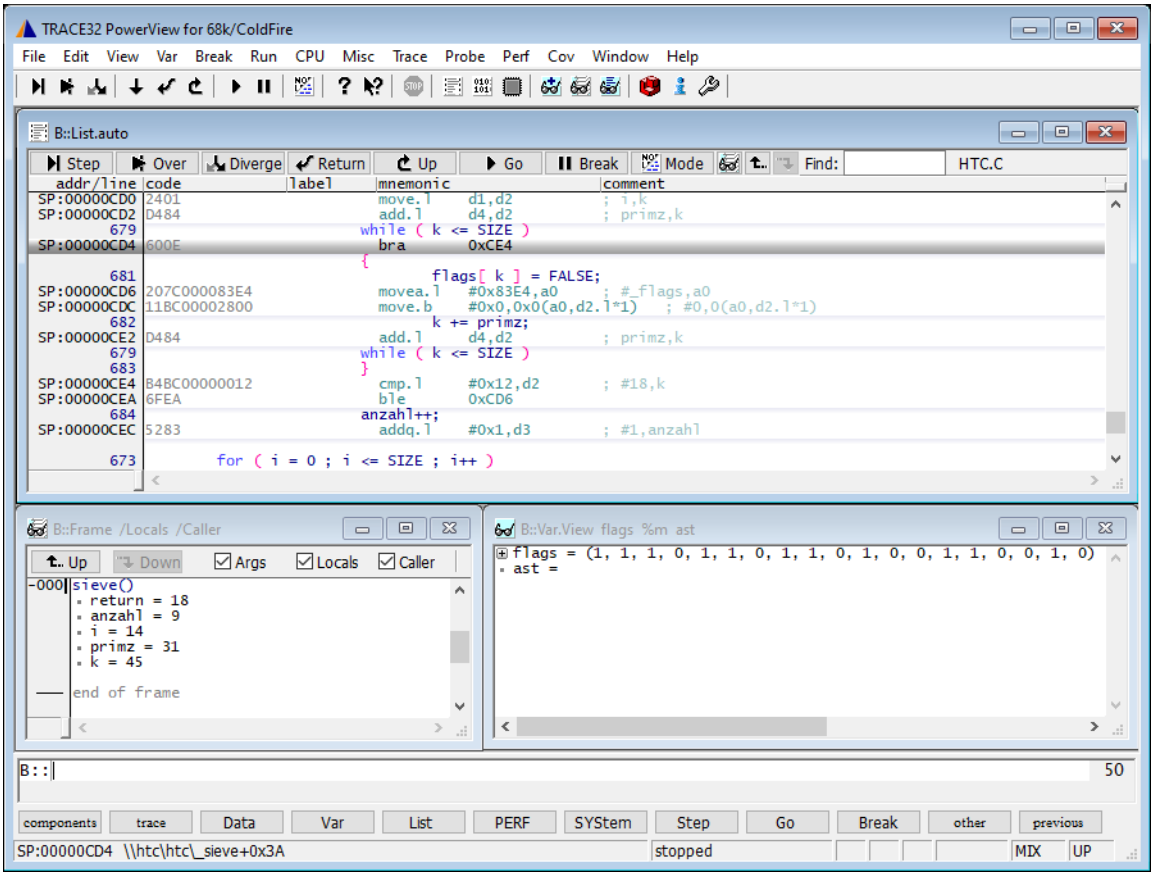
MANUAL

Simulator for 68K/ColdFire

Version 06-Jun-2024

History

20-Jul-22 For the MMU.SCAN ALL command, CLEAR is now possible as an optional second parameter.



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**.

To start the simulator, proceed as follows:

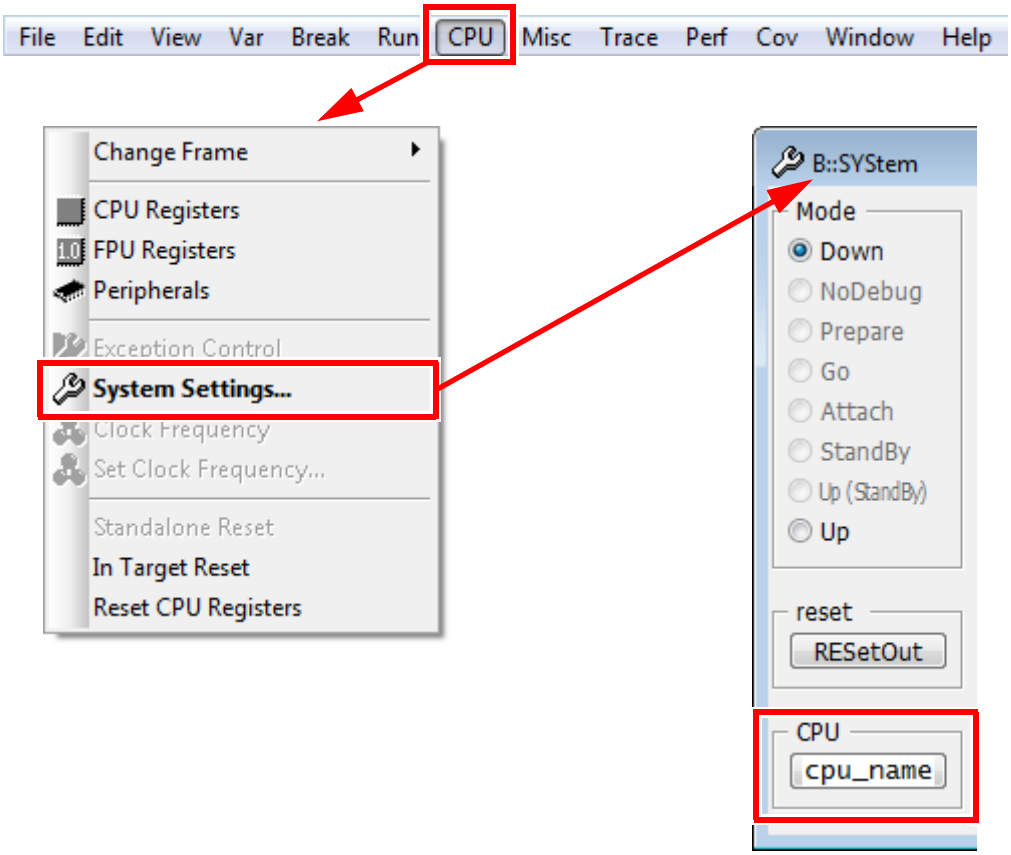
1. Select the device prompt for the Simulator and reset the system.

```
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.

Memory Classes

Memory Class	Description
FC0	Function-Code 0
FC1	USER-DATA
UD	USER-DATA
FC2	USER-PROGRAM
UP	USER-PROGRAM
FC3	Function-Code 3
FC4	Function-Code 4
FC5	SUPERVISOR-DATA
SD	SUPERVISOR-DATA
FC6	SUPERVISOR-PROGRAM
SP	SUPERVISOR-PROGRAM
FC7	Function-Code 7
CPU	CPU Function-Code

U	User
S	Supervisor
D	Data
P	Program

C	Memory access by CPU
E	Emulation memory access

SYStem.CONFIG — Configure debugger according to target topology

The **SYStem.CONFIG** commands have no effect on the simulator. They are only provided to allow the user to run PRACTICE scripts written for the debugger within the simulator without modifications.

SYStem.CPU — Select CPU type

Format: **SYStem.CPU** <*mode*>

<*mode*>: **000** | **010** | **020** | **030** | **040** | **060**
302 | **LC302** | **PM302** | **EN302** | **356** | **306** | **307**
330 … **68336** | **340** | **341** | **349** | **360**

Selects the processor type.

NOTE: ROM debuggers require also a modification in the debug monitor for different processor types.

SYStem.LOCK — Lock and tristate the debug port

Format: **SYStem.LOCK** [**ON** | **OFF**]

The command has no effect for the simulator.

Format: **SYStem.MemAccess Enable | StopAndGo | Denied**
SYStem.ACCESS (deprecated)

Enable Memory access during program execution to target is enabled.
CPU (deprecated)

Denied Memory access during program execution to target is disabled.

StopAndGo Temporarily halts the core(s) to perform the memory access. Each stop takes some time depending on the speed of the JTAG port, the number of the assigned cores, and the operations that should be performed.

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**.

SYStem.Option.BASE Select peripheral base address

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

Defines the base address of the internal IO of some 683xx processors. This should be set to the value used by the target system.

SYStem.Option.MISALIGN Allow mis-alignment in data accesses

Format: **SYStem.Option.MISALIGN** [**ON** | **OFF**]

When set to ON, mis-alignment in data accesses is allowed in the TRACE32 Instruction Set Simulator, otherwise an exception is raised.

SYStem.Option.MMUSPACES Separate address spaces by space IDs

Format: **SYStem.Option.MMUSPACES** [**ON** | **OFF**]
SYStem.Option.MMUspaces [**ON** | **OFF**] (deprecated)
SYStem.Option.MMU [**ON** | **OFF**] (deprecated)

Default: OFF.

Enables the use of space IDs for logical addresses to support **multiple** address spaces.

For an explanation of the TRACE32 concept of address spaces (zone spaces, MMU spaces, and machine spaces), see **"TRACE32 Concepts"** (trace32_concepts.pdf).

NOTE:	**SYStem.Option.MMUSPACES** should not be set to **ON** if only one translation table is used on the target. If a debug session requires space IDs, you must observe the following sequence of steps: 1. Activate **SYStem.Option.MMUSPACES**. 2. Load the symbols with **Data.LOAD**. Otherwise, the internal symbol database of TRACE32 may become inconsistent.

Examples:

```
;Dump logical address 0xC00208A belonging to memory space with
;space ID 0x012A:
Data.dump D:0x012A:0xC00208A

;Dump logical address 0xC00208A belonging to memory space with
;space ID 0x0203:
Data.dump D:0x0203:0xC00208A
```

SYStem.Option.PIPELINE — Pre-fetching simulation

Format: **SYStem.Option.PIPELINE** [**ON** | **OFF**]

Default: off.

Enables pre-fetching simulation.

Format: **SYStem.RESetOut**

Special reset command for 68Ksimulator.

SYStem.RESetOut CPU reset command

The command asserts nRESET on the JTAG connector in the TRACE32 In-Circuit Debugger (ICD) but is ignored by the TRACE32 Instruction Set Simulator. However, the command is allowed in the simulator so that you can run scripts which have actually been made for the debugger. For more information about the effect in the debugger, refer to your **Processor Architecture Manual** (debugger_<*arch*>.pdf).

Format: **SYStem.Option.IMASKASM** [**ON** | **OFF**]

Default: OFF.

If enabled, the bit responsible for ignoring pending interrupts during assembler single-step operations of the CPU will be set. The interrupt routine is not executed during single-step operations.

SYStem.Option.IMASKHLL Disable interrupts while HLL single stepping

Format: **SYStem.Option.IMASKHLL** [**ON** | **OFF**]

Default: OFF.

If enabled, the interrupt mask bits of the CPU will be set during HLL single-step operations. The interrupt routine is not executed during single-step operations. After single step the interrupt mask bits are restored to the value before the step.

MMU.DUMP Page wise display of MMU translation table

Format: **MMU.DUMP** *<table>* [*<range>* | *<address>* | *<range>* *<root>* | *<address>* *<root>*]
MMU.*<table>***.dump** (deprecated)

<table>: **PageTable**
KernelPageTable
TaskPageTable *<task_magic>* | *<task_id>* | *<task_name>* | *<space_id>***:0x0** *<cpu_specific_tables>*

Displays the contents of the CPU specific MMU translation table.

- If called without parameters, the complete table will be displayed.
- If the command is called with either an address range or an explicit address, table entries will only be displayed if their **logical** address matches with the given parameter.

<root>	The *<root>* argument can be used to specify a page table base address deviating from the default page table base address. This allows to display a page table located anywhere in memory.
<range> *<address>*	Limit the address range displayed to either an address range or to addresses larger or equal to *<address>*. For most table types, the arguments *<range>* or *<address>* can also be used to select the translation table of a specific process if a space ID is given.
PageTable	Displays the entries of an MMU translation table. • if *<range>* or *<address>* have a space ID: displays the translation table of the specified process • else, this command displays the table the CPU currently uses for MMU translation.

KernelPageTable	Displays the MMU translation table of the kernel. If specified with the **MMU.FORMAT** command, this command reads the MMU translation table of the kernel and displays its table entries.
TaskPageTable *<task_magic>* | *<task_id>* | *<task_name>* | *<space_id>***:0x0**	Displays the MMU translation table entries of the given process. Specify one of the **TaskPageTable** arguments to choose the process you want. In MMU-based operating systems, each process uses its own MMU translation table. This command reads the table of the specified process, and displays its table entries. • For information about the first three parameters, see **"What to know about the Task Parameters"** (general_ref_t.pdf). • See also the appropriate **OS Awareness Manuals**.

ITLB	Displays the contents of the Instruction Translation Lookaside Buffer.
DTLB	Displays the contents of the Data Translation Lookaside Buffer.

MMU.List — Compact display of MMU translation table

Format: **MMU.List** *<table>* [*<range>* | *<address>* | *<range>* *<root>* | *<address>* *<root>*]
MMU.*<table>***.List** (deprecated)

<table>: **PageTable**
KernelPageTable
TaskPageTable *<task_magic>* | *<task_id>* | *<task_name>* | *<space_id>***:0x0**

Lists the address translation of the CPU-specific MMU table.

- If called without address or range parameters, the complete table will be displayed.
- If called without a table specifier, this command shows the debugger-internal translation table. See **TRANSlation.List**.
- If the command is called with either an address range or an explicit address, table entries will only be displayed if their **logical** address matches with the given parameter.

<root>	The *<root>* argument can be used to specify a page table base address deviating from the default page table base address. This allows to display a page table located anywhere in memory.
<range> *<address>*	Limit the address range displayed to either an address range or to addresses larger or equal to *<address>*. For most table types, the arguments *<range>* or *<address>* can also be used to select the translation table of a specific process if a space ID is given.
PageTable	Lists the entries of an MMU translation table. • if *<range>* or *<address>* have a space ID: list the translation table of the specified process • else, this command lists the table the CPU currently uses for MMU translation.

KernelPageTable	Lists the MMU translation table of the kernel. If specified with the **MMU.FORMAT** command, this command reads the MMU translation table of the kernel and lists its address translation.
TaskPageTable *<task_magic>* | *<task_id>* | *<task_name>* | *<space_id>***:0x0**	Lists the MMU translation of the given process. Specify one of the **TaskPageTable** arguments to choose the process you want. In MMU-based operating systems, each process uses its own MMU translation table. This command reads the table of the specified process, and lists its address translation. • For information about the first three parameters, see **"What to know about the Task Parameters"** (general_ref_t.pdf). • See also the appropriate **OS Awareness Manuals**.

Format:	**MMU.SCAN** *<table>* [*<range>* *<address>*] **MMU.***<table>***.SCAN** (deprecated)
<table>:	**PageTable** **KernelPageTable** **TaskPageTable** *<task_magic>* | *<task_id>* | *<task_name>* | *<space_id>***:0x0** **ALL** [**Clear**] *<cpu_specific_tables>*

Loads the CPU-specific MMU translation table from the CPU to the debugger-internal static translation table.

- If called without parameters, the complete page table will be loaded. The list of static address translations can be viewed with **TRANSlation.List**.
- If the command is called with either an address range or an explicit address, page table entries will only be loaded if their **logical** address matches with the given parameter.

Use this command to make the translation information available for the debugger even when the program execution is running and the debugger has no access to the page tables and TLBs. This is required for the real-time memory access. Use the command **TRANSlation.ON** to enable the debugger-internal MMU table.

PageTable	Loads the entries of an MMU translation table and copies the address translation into the debugger-internal static translation table. • if *<range>* or *<address>* have a space ID: loads the translation table of the specified process • else, this command loads the table the CPU currently uses for MMU translation.

KernelPageTable	Loads the MMU translation table of the kernel. If specified with the **MMU.FORMAT** command, this command reads the table of the kernel and copies its address translation into the debugger-internal static translation table.
TaskPageTable *<task_magic>* | *<task_id>* | *<task_name>* | *<space_id>***:0x0**	Loads the MMU address translation of the given process. Specify one of the **TaskPageTable** arguments to choose the process you want. In MMU-based operating systems, each process uses its own MMU translation table. This command reads the table of the specified process, and copies its address translation into the debugger-internal static translation table. • For information about the first three parameters, see **"What to know about the Task Parameters"** (general_ref_t.pdf). • See also the appropriate **OS Awareness Manual**.
ALL [**Clear**]	Loads all known MMU address translations. This command reads the OS kernel MMU table and the MMU tables of all processes and copies the complete address translation into the debugger-internal static translation table. See also the appropriate **OS Awareness Manual**. **Clear:** This option allows to clear the static translations list before reading it from all page translation tables.

TrOnchip — Onchip triggers

This command group has no effect on the TRACE32 Instruction Set Simulator.