LAUTERBACH
DEVELOPMENT TOOLS

Simulator for PowerPC

MANUAL

Simulator for PowerPC

TRACE32 Online Help

TRACE32 Directory

TRACE32 Index

TRACE32 Documents ..... 🗁

TRACE32 Instruction Set Simulators ..... 🗁

Simulator for PowerPC **1**

TRACE32 Simulator License **5**

Quick Start of the Simulator **6**

Peripheral Simulation **8**

Troubleshooting **8**

FAQ **8**

Memory Classes **9**

Simulated Registers **10**

CPU specific SYStem Commands **12**

SYStem.CPU — Select CPU type — 12
SYStem.MemAccess — Select run-time memory access method — 12
SYStem.Mode — Establish the communication with the simulator — 12
SYStem.Option.Address32 — Define address format display — 13
SYStem.Option.DisMode — Simulator operation mode — 14
SYStem.Option.DUALPORT — Run-time memory access for all windows — 14
SYStem.Option.IMASKASM — Disable interrupts while single stepping — 15
SYStem.Option.IMASKHLL — Disable interrupts while HLL single stepping — 15
SYStem.Option.MMUSPACES — Separate address spaces by space IDs — 16
SYStem.Option.MACHINESPACES — Address extension for guest OSes — 17
SYStem.Option.NOTRAP — Use alternative software breakpoint instruction — 17
SYStem.Option.OVERLAY — Enable overlay support — 18
SYStem.Option.REALTIME — Stall the simulator if faster than real processor — 18
SYStem.Option.TranslationSPACE — Identify user and hypervisor modes — 19
SYStem.Option.ZoneSPACES — Enable symbol management for zones — 20

CPU specific MMU Commands **23**

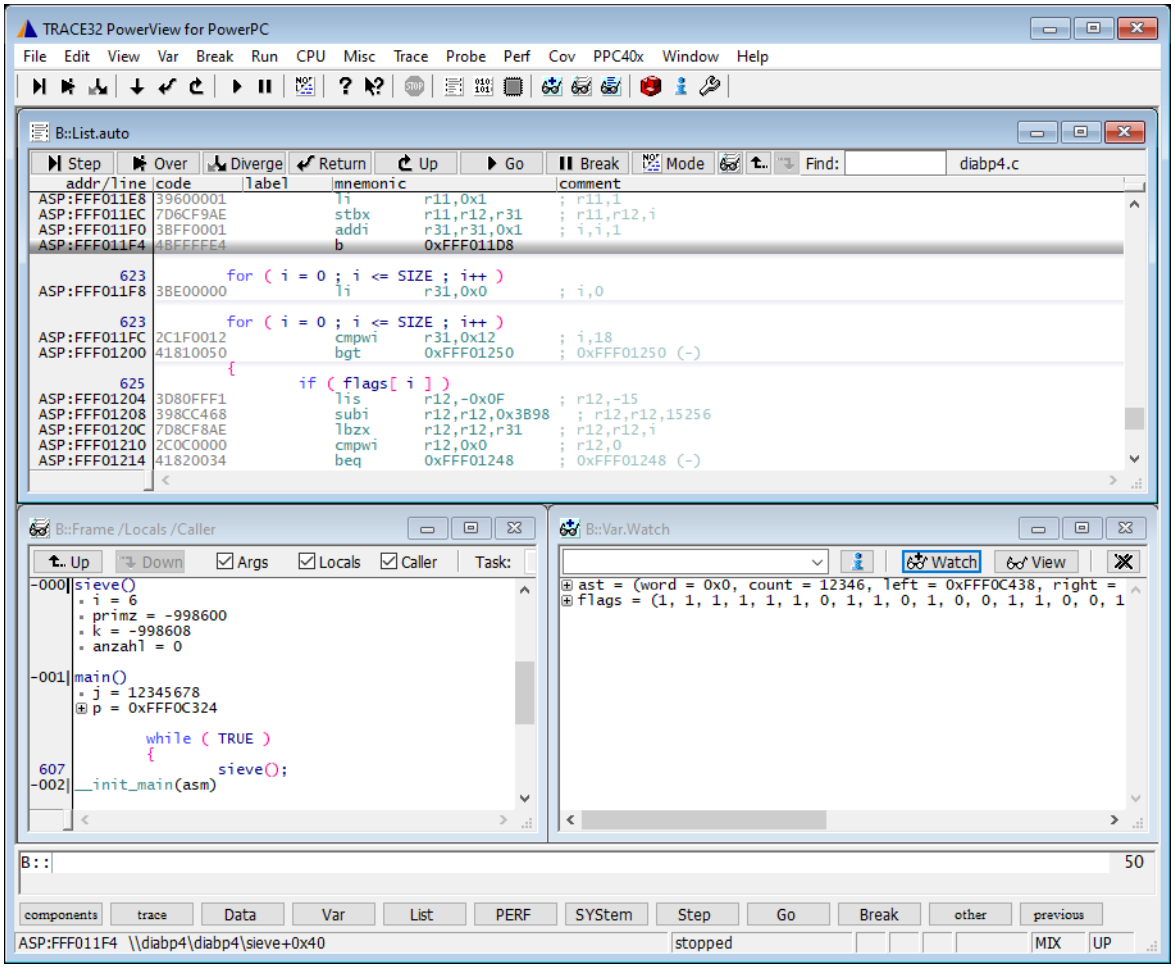
MMU.DUMP — Page wise display of MMU translation table — 23
MMU.FORMAT — Define MMU table structure — 25
MMU.List — Compact display of MMU translation table — 30
MMU.SCAN — Load MMU table from CPU — 32

TrOnchip Commands **34**

TrOnchip.state Display on-chip trigger window 34
TrOnchip.RESet Set on-chip trigger to default state 34

Simulator for PowerPC

Version 06-Jun-2024



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:

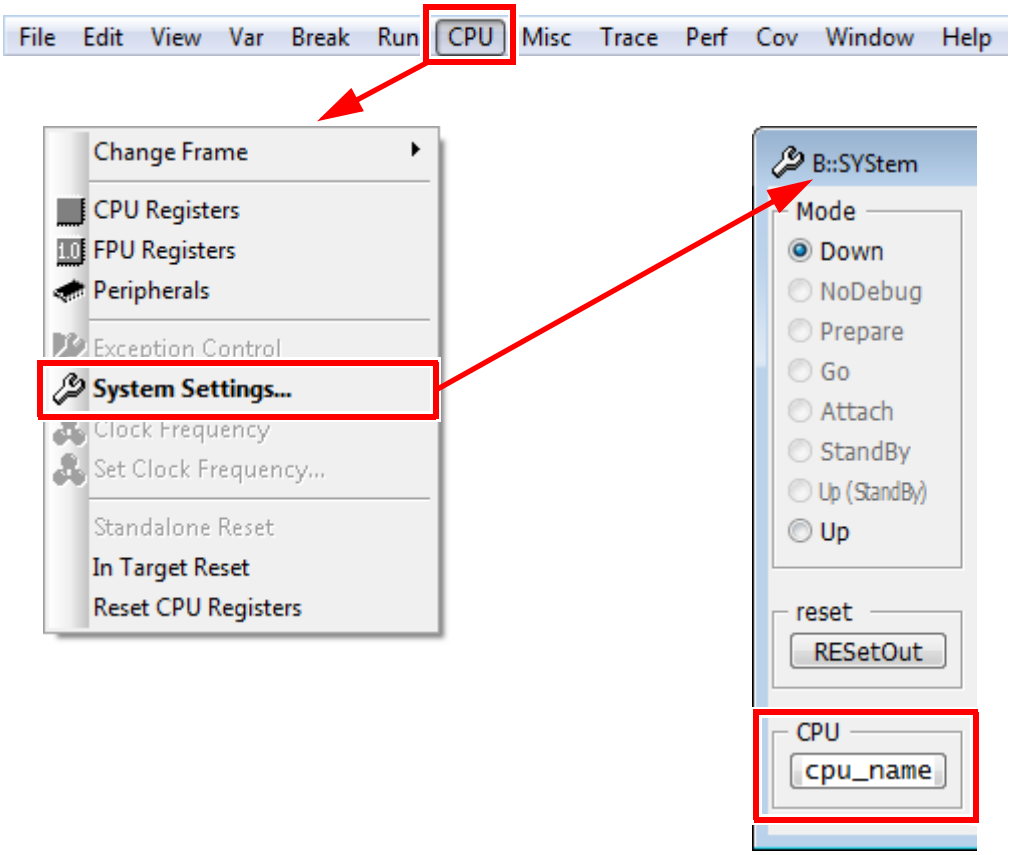
1. Select the device prompt for the Simulator and reset the system.

```
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

The following memory classes are available:

Memory Class	Description
P	Program
D	Data
SPR	Special Purpose Register
DCR	Device Control Register
PMR	Performance Control Register
IC	Instruction Cache
DC	Data Cache
NC	No Cache (only physically memory)

If the cache is disabled, memory accesses to the memory classes IC or DC are realized by TRACE32-ICD as reads and writes to physical memory.

Simulated Registers

The instruction set simulator has built-in support for all core registers. The level of support varies by register. The first table lists all registers and their basic supported features, while the second table lists all registers that have support beyond the basic features.

Register(s)	Basic supported features
GPR[0..31] FPR[0..31] (*) FPSCR VR[0..31] (*) VRSAVE (*) VSCR (*) CR CTR LR XER SPEFSCR (*) ACC (*)	**Full support, e.g.:** View/modify through debugger views and commands Source/destination register of opcodes (including status flags) of standard PowerPC, SPE, EFPU, Altivec and VLE. etc.
SPR DCR PMR	**If not stated otherwise in table with additional features:** View/modify through by debugger Read/write by instruction (MFSPR/MTSPR etc.)

Registers with additional features:

Register(s)	Additional features
MSR	Read/write/modify by instructions modifying MSR (MTMSR, MFMSR, WRTEE, RFI, RFCI, RFMCI, RFDI, MPC5xx: EIE/EID) Modified by exceptions MSR[FP] evaluated for FPU QorIQ; MSR[PR], MSR[GS] evaluated for guest register access PPC4XX, MPX5XXX, QorIQ: MSR[EE] evaluated for fixed interval timer, decrementer (TCR/TSR) MPC74XX, MPC86XX, QorIQ: MSR[VEC] evaluated for Altivec e300 cores: MSR[IP] interrupt prefix
SRR0/1	Modified by exceptions
HID0 (*)	HID0[TBEN]
PVR	Reports processor version, value may differ from actual processor
TBL/TBU (*)	Incrementing with each executed instruction
DEC (*) DECAR (*)	Decrementing, interrupt when passes through zero an MSR[EE] set. Reloaded with DECAR value (if supported by core)
TCR/TSR (*)	Support for decrementer, fixed interval timer and watchdog
MAS[0..6] (*)	MPC5XXX: MMU simulation limited to support mixed FLE/VLE applications

ESR (*)	ESR[PTR] and ESR[VLE]
IVPR/EVPR (*) IVOR0/1/4/6/8/10 (*)	Base address for interrupts handlers
L1CSR0, L1CSR1 (*) L1FINV0, L1FINV1 (*)	MPC5XXX: evaluated for cache simulation (see **SIM.CACHE**)

(*): not available for all PowerPC cores.

SYStem.CPU — Select CPU type

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

<cpu>: **505** | **509** | **555** | **821** | **860** | …

SYStem.MemAccess — Select run-time memory access method

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

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

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.Address32 Define address format display

Format: **SYStem.Option.Address32** [**ON** | **OFF** | **AUTO** | **NARROW**]

Default: AUTO.

Selects the number of displayed address digits in various windows, e.g. **List.auto** or **Data.dump**.

ON	Display all addresses as 32-bit values. 64-bit addresses are truncated.
OFF	Display all addresses as 64-bit values.
AUTO	Number of displayed digits depends on address size.
NARROW	32-bit display with extendible address field.

Format: **SYStem.Option.DisMode** *<mode>*

<mode>: **ACCESS**
AUTO
FLE
VLE

MPC5XXX/SPC5XX only.

This command sets the operation mode for the simulator.

AUTO (default) Behavior depending on CPU selection. VLE/FLE operation for VLE/FLE-only processors, others: see ACCESS

ACCESS Default: Standard PowerPC (FLE) instruction set. Simulator supports mixed FLE/VLE code execution if MMU simulation is enabled.

FLE Simulator is configured to execute code compiled for the standard PowerPC instruction set (fixed length encoding).

VLE Simulator is configured to execute code compiled for VLE (variable length encoding).

SYStem.Option.DUALPORT Run-time memory access for all windows

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

Default: OFF.

Dualport access of memory while simulation in running.

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

Default: OFF.

If enabled, the interrupt mask bits of the CPU will be set during assembler 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.

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.

NOTE: Do not enable this option for code that disables MSR_EE. The debugger will disable MSR_EE while the CPU is running and restore it after the CPU stopped. If a part of the application is executed that disables MSE_EE, the debugger cannot detect this change and will restore MSE_EE.

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
```

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

Default: OFF

Enables the TRACE32 support for debugging virtualized systems. Virtualized systems are systems running under the control of a hypervisor.

After loading a Hypervisor Awareness, TRACE32 is able to access the context of each guest machine. Both currently active and currently inactive guest machines can be debugged.

If **SYStem.Option.MACHINESPACES** is set to **ON**:

- Addresses are extended with an identifier called machine ID. The machine ID clearly specifies to which host or guest machine the address belongs.

  The host machine always uses machine ID 0. Guests have a machine ID larger than 0. TRACE32 currently supports machine IDs up to 30.
- The debugger address translation (**MMU** and **TRANSlation** command groups) can be individually configured for each virtual machine.
- Individual symbol sets can be loaded for each virtual machine.

SYStem.Option.NOTRAP Use alternative software breakpoint instruction

Format: **SYStem.Option.NOTRAP** *<type>*

<type>: **ON** | **OFF** | **FPU** | **ILL**

Defines which instruction is used to implement software breakpoints.

OFF Use TRAP instructions as software breakpoint (default setting).

ON
ILL Using illegal opcode as breakpoint instruction, TRAP instruction available for use in application.

FPU Not supported in simulator. Same effect as ON.

Format: **SYStem.Option.OVERLAY** [**ON** | **OFF** | **WithOVS**]

Default: OFF.

ON Activates the overlay extension and extends the address scheme of the debugger with a 16 bit virtual overlay ID. Addresses therefore have the format *<overlay_id>*:*<address>*. This enables the debugger to handle overlaid program memory.

OFF Disables support for code overlays.

WithOVS Like option **ON**, but also enables support for software breakpoints. This means that TRACE32 writes software breakpoint opcodes to both, the *execution area* (for active overlays) and the *storage area*. This way, it is possible to set breakpoints into inactive overlays. Upon activation of the overlay, the target's runtime mechanisms copies the breakpoint opcodes to the execution area. For using this option, the storage area must be readable and writable for the debugger.

Example:

```
SYStem.Option.OVERLAY ON
Data.List 0x2:0x11c4                    ; Data.List <overlay_id>:<address>
```

SYStem.Option.REALTIME Stall the simulator if faster than real processor

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

Default: OFF.

Prevents the simulator from runnig faster than the frequency set with **VCO.Frequency** (default: 10MHz).

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

Default: ON.

This system option configures the debugger how to distinguish between user and supervisor modes.

In bare-metal applications or uncomplex operating systems typically the MSR[IS] bit is used to isolate user from supervisor address space. There is no way to get the information about this bit within the trace information, the program trace will be decoded using the current context of the cores.

In complex or hypervisor systems, typically the MSR[PR] bit is used to handle user and supervisor modes. This bit will also be included in ownership trace messages. TRACE32 will therefore be able to decode the program trace depending on this privilege information, which doesn't have to be compliant to the current context of cores.

a) **SYStem.Option.TranslationSPACE ON** (default)

Mode	MSR.GS bit	MSR.IS bit
Hypervisor-supervisor mode	0	0
Hypervisor-user	0	1
Guest-supervisor	1	0
Guest-user	1	1

b) **SYStem.Option.TranslationSPACE OFF**

Mode	MSR.GS bit	MSR.PR bit
Hypervisor-supervisor mode	0	0
Hypervisor-user	0	1
Guest-supervisor	1	0
Guest-user	1	1

[Example]

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

Default: OFF.

The **SYStem.Option.ZoneSPACES** command must be set to **ON** if separate symbol sets and MMU translation tables are used for the CPU operation modes:

- Hypervisor-supervisor mode
- Hypervisor-user mode
- Guest-supervisor mode
- Guest-user mode

Within TRACE32, these CPU operation modes are referred to as zones. For information about the status bits controlling these modes, see **SYStem.Option.TranslationSPACE**.

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

In each CPU operation mode (zone), the CPU's TLB may contain separate translations, and a kernel or hypervisor may uses separate MMU translation tables for memory accesses and separate register sets. Consequently, in each zone, different code and data can be visible on the same logical address.

OFF	TRACE32 does not separate symbols by access class. Loading two or more symbol sets with overlapping address ranges will result in unpredictable behavior. Loaded symbols are independent of the CPU mode.
ON	Separate symbol sets can be loaded for each zone, even with overlapping address ranges. Loaded symbols are specific to one of the CPU zones.

SYStem.Option.ZoneSPACES ON in Detail

SYStem.Option.ZoneSPACES is usually set to **ON** if you need to debug virtualized systems with guest and hypervisor. For both guest and hypervisor, TRACE32 also separates between supervisor mode and user mode. Typical scenarios use separate symbol sets for the hypervisor-supervisor mode, the guest-supervisor and the guest-user mode. The hypervisor-user mode is rarely used. The symbol sets are loaded to the access classes HS: (hypervisor-supervisor mode, GS: (guest-supervisor mode) and GU: (guest-user mode).

If **SYStem.Option.ZoneSPACES** is **ON**, TRACE32 enforces any memory address specified in a TRACE32 command to have an access class which clearly indicates to which of the four zones the memory address belongs.

If an address specified in a command uses an anonymous access class such as D:, P: or C:, the access class of the current PC context is used to complete the access class of the addresses. Also, if an incomplete access class where either the guest/hypervisor information is missing (such as SP: or UP:) or the supervisor/user information is missing (such as GP: or HP:), the missing information will automatically be expanded from the access class of the current PC context.

Example: If the CPU is currently in user mode, a memory access with the access class GP: will be expanded by TRACE32 to become GUP:

If a symbol is referenced by name, the associated access class of its zone will be used automatically, so that the memory access is done within the correct CPU mode context. As a result, the symbol's effective address will be translated to the physical address with the correct MMU translation table.

Example 1

In this script, **SYStem.Option.ZoneSPACES** is used for a simple host and guest debugging.

```
SYStem.Option.ZoneSPACES ON

; 1. Load the Xen hypervisor symbols to the hypervisor-supervisor
; access class HS:
Data.LOAD.ELF xen-syms HS:0x0 /NoCODE

; 2. Load the vmlinux kernel symbols to the guest-supervisor access class
; GS:
Data.LOAD.ELF vmlinux GS:0x0 /NoCODE

; 3. Load the guest application symbols (the 'sieve' application in this
; example) to the guest-user access class GU:
Data.LOAD.ELF sieve GU:0x0 /NoCODE
```

Effect on the TRANSlation command group: **SYStem.Option.ZoneSPACES ON** enforces separate address spaces for the four zones HS:, HU:, GS: and GU:. Commands affecting the address translation, such as **TRANSlation.Create**, **TRANSlation.COMMON**, **TRANSlation.Protect** or **MMU.FORMAT**, must be executed individually for each of the four zones.
It is, however, possible to use the generic access classes G: and H: as "joker". This simplifies the scripts if identical translations for GS: and GU: are needed or identical translations for HS: and HU: are needed.

Example 2

```
SYStem.Option.ZoneSPACES ON
; show the list of static translations created by the commands
; TRANSlation.Create and TRANSlation.COMMON
TRANSlation.List

;1. the command
TRANSlation.Create G:0x80000000--0x8FFFFFFF 0x0
; is equivalent to the commands
TRANSlation.Create GS:0x80000000--0x8FFFFFFF 0x0
TRANSlation.Create GU:0x80000000--0x8FFFFFFF 0x0

;2. the command
TRANSlation.Create H:0xA0000000--0xAFFFFFFF 0x0
; is equivalent to the commands
TRANSlation.Create HS:0xA0000000--0xAFFFFFFF 0x0
TRANSlation.Create HU:0xA0000000--0xAFFFFFFF 0x0

;3. the command
TRANSlation.COMMON G:0xC00000000--0xFFFFFFFF
; is equivalent to the commands
TRANSlation.COMMON GS:0xC00000000--0xFFFFFFFF
TRANSlation.COMMON GU:0xC00000000--0xFFFFFFFF
```

MMU.DUMP Page wise display of MMU translation table

Format:	**MMU.DUMP** *<table>* [*<range>* | *<address>* | *<range>* *<root>* | *<address>* *<root>*] [**/***<option>*] **MMU.***<table>***.dump** (deprecated)
<table>:	**PageTable** **KernelPageTable** **TaskPageTable** *<task_magic>* | *<task_id>* | *<task_name>* | *<space_id>***:0x0** *<cpu_specific_tables>*
<option>:	**MACHINE** *<machine_magic>* | *<machine_id>* | *<machine_name>*

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 or a specific machine if a space ID and/or a machine ID is given.
PageTable	Displays the entries of an MMU translation table. • if *<range>* or *<address>* have a space ID and/or machine ID: displays the translation table of the specified process and/or machine • 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**.
MACHINE *<machine_magic>* | *<machine_id>* | *<machine_name>*	This option is only available if **SYStem.Option.MACHINESPACES** is set to **ON**. Dumps a page table of a virtual machine. The **MACHINE** option applies to **PageTable** and **KernelPageTable** and some *<cpu_specific_tables>*. The parameters *<machine_magic>*, *<machine_id>* and *<machine_name>* are displayed in the **TASK.List.MACHINES** window.

CPU specific Tables in MMU.DUMP <table>

If both **SYStem.Option.ZoneSPACES** and **SYStem.Option.MACHINESPACES** are set to **ON**, then these CPU specific tables are available:

SupervisorPT	Displays the supervisor mode page table of the machine specified with option **MACHINE**.
UserPT	Displays the user mode page table of the machine specified with the option **MACHINE**.

If only **SYStem.Option.ZoneSPACES** is set to **ON**, then these CPU specific tables are available:

HSPageTable	Displays the page table which is defined for the hypervisor supervisor mode.
HUPageTable	Displays the page table which is defined for the hypervisor user mode.
GSPageTable	Displays the page table which is defined for the guest supervisor mode.
GUPageTable	Displays the page table which is defined for the guest user mode.

[Examples]

Format:	**MMU.FORMAT** *<format>* [*<base_address>* [*<logical_kernel_address_range>* *<physical_kernel_address>*]] [**/***<option>*]
<option>:	**MACHINE** *<machine_magic>* | *<machine_id>* | *<machine_name>* **Hypervisormode** | **Guestmode** | **Supervisormode** | **Usermode**

Default *<format>*: STD.

Defines the information needed for the page table walks, which are performed by TRACE32 for debugger address translation, page table dumps, or page table scans.

<format>

<format> is to be replaced with a CPU architecture specific keyword which defines the structure of the MMU page tables used by the kernel. By default, TRACE32 assumes that the MMU format is **STD**, unless *you* specify the **MMU.FORMAT** *<format>* explicitly.

<format>	Description
DEOS	DEOS OS (32 bit) specific MMU format
DEOS64	DEOS OS (64 bit) specific MMU format
EXTENSION	Table walk performed by a TRACE32 extension that a) was developed by the customer and b) defines table walk callback functions.
LINUX	Standard format used by Linux
LINUX26	Linux format with physical table pointers
LINUX64_E6	Use LINUX64_E6 for e6500 core devices
LINUXE5	Linux with 64-bit PTEs, e500 core
LINUXEXT	Linux with 64-bit PTEs, no e500 core
LYNXOS	LynxOS format, virtual table pointers
LYNXOSPHYS	LynxOS format, physical table pointers
OSE	OSE format for load modules
PIKEOS.E500	PIKEOS specific format for PowerPC e500 core (formerly named PIKEOSE5).Works for PikeOS 4.1 and older. For e500 cores with PikeOS 4.2 and newer use E500MC format.*/

<format>	Description
PIKEOS.E500MC	PIKEOS specific format for PowerPC e500mc core (PPC64 only).Can also be used with PikeOS 4.2 and newer on PPC32 e500 cores.*/
PIKEOS.E500MC4G	PIKEOS specific format for PowerPC e500mc core addressing 4GB of memory.Has no common address range.*/
PIKEOS.E5500	PIKEOS specific format for PowerPC e5500 core
PIKEOS.OEA	PIKEOS specific format for PowerPC core (formerly named PIKEOS) */
QNX	QNX standard format
QNXBIG	QNX format with 64-bit table entries
STD	Standard format defined by the CPU
VX653	MMU format for VXWORKS 653
VXWORKS.E500	VxWorks specific format for PowerPC e500 core
VXWORKS.E500MC	VxWorks specific format for PowerPC e500mc core with 36 bit physical addresses (PPC64 only)
VXWORKS.E500_64	VxWorks specific format for PowerPC e500 core (PPC64 only)
VXWORKS.E6500	VxWorks specific format for PowerPC e6500 core

<base_address>

<base_address> defines the start address of the default page table which is usually the kernel page table. The kernel page table contains translations for mapped address ranges owned by the kernel.

The debugger address translation uses the default page table if no process specific page table (task page table) is available to translate an address.

<base_address> can be left empty by typing a comma or set to zero if there is no default page table available in the system.

<logical_kernel_address_range> and <physical_kernel_address> for the Default Translation

The arguments *<logical_kernel_address_range>* and *<physical_kernel_address>* define a linear logical-to-physical address translation for the kernel addresses, called *kernel translation* or *default translation*. This translation should cover all statically mapped logical address ranges of kernel code or kernel data.

For the *<physical_kernel_address>* you just need to specify the start address.

NOTE: If no kernel translation is specified for a given memory access, TRACE32 tries to use static address translations defined by the command **TRANSlation.Create**. The kernel translation is shown in the **TRANSlation.List** window.

Supervisormode	If **SYStem.Option.MACHINESPACES** is set to **OFF**: Specifies the format, default page table, and default translation for one or both supervisor zones (access class HS: or GS:). Can be combined with the **Hypervisormode** or **Guestmode** option. If **SYStem.Option.MACHINESPACES** is set to **ON**: Specifies the format, default page table, and default translation for the supervisor mode zone of the machine selected with the **MACHINE** option. For an example, see below.
Usermode	If **SYStem.Option.MACHINESPACES** is set to **OFF**: Specifies the format, default page table, and default translation for one or both user mode zones (access class HU: or GU:). Can be combined with the **Hypervisormode** or **Guestmode** option. If **SYStem.Option.MACHINESPACES** is set to **ON**: Specifies the format, default page table, and default translation for the user mode zone of the machine which is selected with the **MACHINE** option.
Hypervisormode	Specifies the format, default page table, and default translation for one or both hypervisor zones (access class HS: or HU:). Can be combined with the **Supervisormode** or **Usermode** option. For an example, see below.
Guestmode	Specifies the format, default page table, and default translation for one or both guest zones (access class GS: or GU:). Can be combined with the **Supervisormode** or **Usermode** option.
MACHINE	For a description of the **MACHINE** option, see **MMU.DUMP**.

If both **SYStem.Option.ZoneSPACES** and **SYStem.Option.MACHINESPACES** are set to **ON**, then these options are available:

- **MACHINE**
- **Supervisormode**
- **Usermode**

If only **SYStem.Option.ZoneSPACES** is set to **ON**, then these options are available:

- **Hypervisormode**
- **Guestmode**
- **Supervisormode**
- **Usermode**

If only **SYStem.Option.MACHINESPACES** is set to **ON**, then these option is available:

- **MACHINE**

Examples for Page Tables in Virtualized Systems

[Back to MMU.FORMAT]

NOTE:
- The MMU format and default page table base address of each zone can be viewed with the command **TRANSlation.state**.
- The default translation of each zone can be viewed with the command **TRANSlation.List**

Example 1: This script shows how to define separate default page tables and separate default translations for various zones (*without* Hypervisor Awareness and *without* machine IDs).
The backslash \ is used as a line continuation character. No white space permitted after the backslash.

```
; enable symbol management for zones
SYStem.Option.ZoneSPACES ON

; define the format for the hypervisor-supervisor zone (access class HS:)
MMU.FORMAT STD HS:0xC8000000 HS:0x80000000++0x0FFFFFFF                \
                           A:0x00000000 /Hypervisormode /Supervisormode

; define the format for the hypervisor-user zone (access class HU:)
MMU.FORMAT STD HU:0x34000000 HU:0x30000000++0x0FFFFFFF                \
                           A:0x00800000 /Hypervisormode /Usermode

; define the format for guest-supervisor zone (access class GS:)
MMU.FORMAT VX653 GS:0xA4000000 GS:0xA0000000++0x0FFFFFFF              \
                           A:0x10000000 /Guestmode  /Supervisormode

; define the format for guest-user zone (access class GU:)
MMU.FORMAT VX653 GU:0x22000000 GU:0x20000000++0x1FFFFFFF              \
                           A:0x18000000 /Guestmode  /Usermode

; show the result of the format definition
TRANSlation.state
TRANSlation.List
```

Example 2: This script shows how to define separate default page tables and separate default translations for various zones (**with** Hypervisor Awareness and **with** machine IDs).
The backslash \ is used as a line continuation character. No white space permitted after the backslash.

```
; enable symbol management for zones
SYStem.Option.ZoneSPACES ON

; enable address extension for guest OSes
SYStem.Option.MACHINESPACES ON

; define the format for the supervisor zone of machine 0
; (access class HS:)
MMU.FORMAT STD  HS:0:::0xC8000000 HS:0:::0x80000000++0x0FFFFFFF        \
                                  A:0x00000000 /MACHINE 0 /Supervisormode

; define the format for the supervisor zone of machine 1
; (access class GS:)
MMU.FORMAT STD  GS:1:::0xA4000000 GS:1:::0xA0000000++0x0FFFFFFF        \
                                  A:0x10000000 /MACHINE 1 /Supervisormode

; define the format for the guest-user zone of machine 1
; (access class GU:)
MMU.FORMAT VX653 GU:1:::0x22000000 GU:1:::0x20000000++0x1FFFFFFF       \
                                  A:0x18000000 /MACHINE 1 /Usermode

; define the format for both the guest-supervisor zone and the guest-user
; zone of machine 2 concurrently (access class G:)
MMU.FORMAT VX653 G:0xB8000000 G:0xB0000000++0x1FFFFFFF                 \
                                  A:0x40000000 /MACHINE 2

; show the result of the format definition
TRANSlation.state
TRANSlation.List
```

Format:	**MMU.List** *<table>* [*<range>* | *<address>* | *<range>* *<root>* | *<address>* *<root>*] [**/***<option>*]
	MMU.*<table>***.List** (deprecated)
<table>:	**PageTable** **KernelPageTable** **TaskPageTable** *<task_magic>* | *<task_id>* | *<task_name>* | *<space_id>***:0x0** *<cpu_specific_tables>*
<option>:	**MACHINE** *<machine_magic>* | *<machine_id>* | *<machine_name>*

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 or a specific machine if a space ID and/or a machine ID is given.
PageTable	Lists the entries of an MMU translation table. • if *<range>* or *<address>* have a space ID and/or machine ID: list the translation table of the specified process and/or machine • 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**.
MACHINE	For a description of the **MACHINE** option, see **MMU.DUMP**.

CPU specific Tables in MMU.List <table>

If both **SYStem.Option.ZoneSPACES** and **SYStem.Option.MACHINESPACES** are set to **ON**, then these CPU specific tables are available:

SupervisorPT	Displays the supervisor mode page table of the machine specified with option **MACHINE**.
UserPT	Displays the user mode page table of the machine specified with the option **MACHINE**.

If only **SYStem.Option.ZoneSPACES** is set to **ON**, then these CPU specific tables are available:

HSPageTable	Displays the page table which is defined for the hypervisor supervisor mode.
HUPageTable	Displays the page table which is defined for the hypervisor user mode.
GSPageTable	Displays the page table which is defined for the guest supervisor mode.
GUPageTable	Displays the page table which is defined for the guest user mode.

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

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 and/or machine ID: loads the translation table of the specified process and/or machine • 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	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**.
MACHINE	For a description of the **MACHINE** option, see **MMU.DUMP**.

CPU specific Tables in MMU.SCAN <table>

If both **SYStem.Option.ZoneSPACES** and **SYStem.Option.MACHINESPACES** are set to **ON**, then these CPU specific tables are available:

SupervisorPT	Displays the supervisor mode page table of the machine specified with option **MACHINE**.
UserPT	Displays the user mode page table of the machine specified with the option **MACHINE**.

If only **SYStem.Option.ZoneSPACES** is set to **ON**, then these CPU specific tables are available:

HSPageTable	Displays the page table which is defined for the hypervisor supervisor mode.
HUPageTable	Displays the page table which is defined for the hypervisor user mode.
GSPageTable	Displays the page table which is defined for the guest supervisor mode.
GUPageTable	Displays the page table which is defined for the guest user mode.

TrOnchip.state — Display on-chip trigger window

Format: **TrOnchip.state**

Opens the **TrOnchip.state** window.

TrOnchip.RESet — Set on-chip trigger to default state

Format: **TrOnchip.RESet**

Sets the TrOnchip settings and trigger module to the default settings.