

# MPC56x NEXUS Debugger and Trace

MANUAL

# MPC56x NEXUS Debugger and Trace

---

[TRACE32 Online Help](#)

[TRACE32 Directory](#)

[TRACE32 Index](#)

[TRACE32 Documents](#) .....



[ICD In-Circuit Debugger](#) .....



[Processor Architecture Manuals](#) .....



[PQ/MPC500](#) .....



**MPC56x NEXUS Debugger and Trace** .....

1

[Brief Overview of Documents for New Users](#) .....

6

[Warning](#) .....

7

[Quick Start](#) .....

8

[Target Design Requirement/Recommendations](#) .....

10

    General .....

10

    Correct Start-up Sequence for the NEXUS Debugger .....

10

    Special Warning for MPC561 and MPC563 .....

11

    AXIOM Evaluation Board .....

11

    Troubleshooting .....

12

        SYStem.Up Errors .....

12

[FAQ](#) .....

12

[Configuration](#) .....

13

[Breakpoints](#) .....

15

    Software Breakpoints .....

15

    On-chip Breakpoints .....

15

        On-chip Breakpoints on InstructionsROM or FLASH .....

16

        On-chip Breakpoints on Read or Write Accesses .....

16

        Example for Breakpoints .....

16

        Simultaneous FLASH Programming for MPC56x .....

17

[Memory Classes](#) .....

18

[General SYStem Commands](#) .....

19

    SYStem.CONFIG .....

        Configure debugger according to target topology .....

19

    SYStem.CPU .....

        Select CPU type .....

19

    SYStem.LOCK .....

        Lock and tristate the debug port .....

19

    SYStem.MemAccess .....

        Select run-time memory access method .....

19

    SYStem.Mode .....

        Establish the communication with the CPU .....

21

<b>CPU specific SYStem Commands .....</b>	<b>23</b>
SYStem.LOADVOC	Load vocabulary for code compression
SYStem.Option.CCOMP	Enable code compression
SYStem.Option.CLEARBE	Clear MSR[BE] on step/go
SYStem.Option.DCFREEZE	Freeze contents of cache while debugging
SYStem.Option.DUALPORT	Run-time memory access for all windows
SYStem.Option.EXTVECTORS	Workaround for revision C silicon
SYStem.Option.ETASFIX	Workaround for MPC555 RiscTrace
SYStem.Option.FAILSAVE	Special error handling for debug port
SYStem.Option.FREEZE	Stop timer in debug mode
SYStem.Option.HighMemory	Switch on high memory
SYStem.Option.ICFLUSH	Flush branch target cache before program start
SYStem.Option.IMASKASM	Disable interrupts while single stepping
SYStem.Option.IMASKHLL	Disable interrupts while HLL single stepping
SYStem.Option.LittleEnd	Selection of little endian mode
SYStem.Option.NOTRAP	Use alternative instruction to enter debug mode
SYStem.Option.OVERLAY	Enable overlay support
SYStem.Option.PPCLittleEnd	Control for PPC little endian
SYStem.Option.SCRATCH	Scratch for FPU access
SYStem.Option.SLOWRESET	Activate SLOWRESET
SYStem.Option.TriState	Control for NEXUS lines
SYStem.Option.VECTORS	Define ranges for not-standard interrupt vectors
SYStem.Option.WATCHDOG	Enable software watchdog after SYStem.Up
SYStem.state	Display SYStem window
<b>CPU specific NEXUS Commands .....</b>	<b>30</b>
NEXUS.BTM	Program trace messaging enable
NEXUS.DTM	Enable data trace messaging
NEXUS.OFF	Switch the NEXUS trace port off
NEXUS.ON	Switch the NEXUS trace port on
NEXUS.OTM	Ownership trace messaging enable
NEXUS.PortSize	Set trace port width
NEXUS.PTM	Enable an exnhanced method of program trace
NEXUS.PTSM	Enable program trace sync mode
NEXUS.QFM	Configure queue flush mode
NEXUS.Register	Display NEXUS trace control registers
NEXUS.RESet	Reset NEXUS trace port settings
NEXUS.state	Display NEXUS port configuration window
<b>CPU specific TrOnchip Commands .....</b>	<b>34</b>
TrOnchip.BusTrigger	Generate a trigger for the internal trigger bus
TrOnchip.CONVert	Adjust range breakpoint in on-chip resource
TrOnchip.DISable	Disable NEXUS trace register control
TrOnchip.ENABLE	Enable NEXUS trace register control
TrOnchip.EVTI	Allow the EVTI signal to stop the program execution

TrOnchip.EVTO	Use EVTO signal for runtime measurement	38
TrOnchip.EXternal	Enable trace trigger input of NEXUS adapter	38
TrOnchip.G/H	Define data selector	39
TrOnchip.IWx	I-Bus watchpoint	39
TrOnchip.IWx.Count	Event counter for I-Bus watchpoint	39
TrOnchip.IWx.Ibus	Instructions address for I-Bus watchpoint	40
TrOnchip.IWx.Watch	Activate I-Bus watchpoint pin	40
TrOnchip.LWx	L-Bus watchpoint	40
TrOnchip.LW0.Count	Event counter for L-Bus watchpoint	40
TrOnchip.LW0.CYcle	Cycle type for L-Bus watchpoint	41
TrOnchip.LW0.Data	Data selector for L-Bus watchpoint	41
TrOnchip.LW0.Ibus	Instructions address for I-Bus watchpoint	41
TrOnchip.LW0.Lbus	Instructions address for L-Bus watchpoint	42
TrOnchip.LW0.Watch	Activate L-Bus watchpoint pin	42
TrOnchip.RESet	Reset on-chip trigger unit	42
TrOnchip.Set	Stop program execution at specified exception	43
TrOnchip.TEnable	Set filter for the trace	44
TrOnchip.TOFF	Switch the sampling to the trace to OFF	44
TrOnchip.TON	Switch the sampling to the trace to ON	44
TrOnchip.TTrigger	Set a trigger for the trace	45
TrOnchip.VarCONVert	Adjust HLL breakpoint in on-chip resource	45
TrOnchip.state	Display on-chip trigger window	45
<b>Technical Data .....</b>		<b>47</b>
Mechanical Dimension		47
Operation Voltage		47

# MPC56x NEXUS Debugger and Trace

Version 06-Jun-2024

The screenshot displays the Trace32 PowerPC debugger interface with three main windows:

- B::Data.List**: Shows assembly code with line numbers 621, 623, 625, and 627. The code includes loops and an if-statement. The assembly source is shown in the right column.
- B::Var.Frame /Locals /Caller**: Shows variable values for the current frame. It includes local variables *i*, *primz*, *k*, and *anzahl*, and caller variables *j* and *p*.
- B::Trace.List**: Shows a table of trace events. The columns are record, run, address, cycle, d.l, symbol, and ti.back. The table lists four entries related to the sieve() function.

record	run	address	cycle	d.l	symbol	ti.back
-00000142		r12,r12,0x5000			subi r12,r12,0x5000 ; r12,r12,20688	0.140us
-00000139		li r11,0x1			P:003FA97C execute ; r11,1	0.160us
-00000136		stbx r11,r12,r31			P:003FA980 execute ; r11,r12,i	0.140us
-00000132		addi r31,r31,0x1			P:003FA984 execute ; i,i,1	0.200us
		P:003FA988 execute			\diabp555\diabc1\sieve+0x40	

# Brief Overview of Documents for New Users

---

## Architecture-independent information:

- “[Training Basic Debugging](#)” (training\_debugger.pdf): Get familiar with the basic features of a TRACE32 debugger.
- “[T32Start](#)” (app\_t32start.pdf): T32Start assists you in starting TRACE32 PowerView instances for different configurations of the debugger. T32Start is only available for Windows.
- “[General Commands](#)” (general\_ref\_<x>.pdf): Alphabetic list of debug commands.

## Architecture-specific information:

- “[Processor Architecture Manuals](#)”: These manuals describe commands that are specific for the processor architecture supported by your debug cable. To access the manual for your processor architecture, proceed as follows:
  - Choose **Help** menu > **Processor Architecture Manual**.
- “[OS Awareness Manuals](#)” (rtos\_<os>.pdf): TRACE32 PowerView can be extended for operating system-aware debugging. The appropriate OS Awareness manual informs you how to enable the OS-aware debugging.

<b>WARNING:</b>	<p>To prevent debugger and target from damage it is recommended to connect or disconnect the Debug Cable only while the target power is OFF.</p> <p>Recommendation for the software start:</p> <ol style="list-style-type: none"><li>1. Disconnect the Debug Cable from the target while the target power is off.</li><li>2. Connect the host system, the TRACE32 hardware and the Debug Cable.</li><li>3. Power ON the TRACE32 hardware.</li><li>4. Start the TRACE32 software to load the debugger firmware.</li><li>5. Connect the Debug Cable to the target.</li><li>6. Switch the target power ON.</li><li>7. Configure your debugger e.g. via a start-up script.</li></ol> <p>Power down:</p> <ol style="list-style-type: none"><li>1. Switch off the target power.</li><li>2. Disconnect the Debug Cable from the target.</li><li>3. Close the TRACE32 software.</li><li>4. Power OFF the TRACE32 hardware.</li></ol>
-----------------	------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

# Quick Start

---

Starting up the Nexus Debugger is done by the following steps:

1. Select the device prompt B: for the TRACE32 ICD-Debugger, if the device prompt is not active after starting the TRACE32 software.

```
B:
```

2. Select the CPU type to load the CPU specific settings.

```
SYStem.CPU MPC563
```

The default CPU is MPC563.

3. Inform the debugger where's FLASH/ROM on the target, this is necessary for the use of the on-chip breakpoints.

```
MAP.BOnchip 0x100000++0xffff
```

On-chip breakpoints are now used, if a program or spot breakpoint is set within the specified address range. A list of all available on-chip breakpoints for your architecture can be found under [On-chip Breakpoints](#).

4. Enter debug mode.

```
SYStem.Up
```

This command resets the CPU, enables the debug mode and stops the CPU at the first opfetch (reset vector). After this command is possible to access memory and registers.

5. Configure the IBUS.

```
SYStem.Option.IBUS IND ; Show cycles are generated for all  
; indirect changes in the program flow.  
; Recommended if a RISC Trace or  
; PowerTrace module is connected.
```

For proper Nexus Trace operation, use SYStem.Option.IBUS CHG. Refer to FAQ “Trace Impacts.Full Trace settings(MPC56x)”.

6. Set the special function registers to prepare your target memory for program loading.

```
Data.Set SPR:027E %Long 0x800
```

## 7. Load the program.

```
Data.LOAD.Elf diabp555.x      ; Load ELF file
```

The load command depends on the file format generated by your compiler. A full description of the **Data.Load** command is given in the [“General Commands Reference”](#).

The start-up sequence can be automated using the script language PRACTICE. 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-Debugger device
                                         ; prompt

WinCLEAR                            ; Delete all windows

MAP.BOnchip 0x100000++0xfffff      ; Specify where's FLASH/ROM

SYStem.CPU 0x563                   ; Select the processor type

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

Data.LOAD.Elf diabp563.x          ; Load the application

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

List.Mix                            ; Open a source listing      *)

Register.view /SpotLight           ; Open the register window   *)

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

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

PER.view                            ; Open a window for the special
                                         ; function registers

Break.Set sieve                     ; Set breakpoint to function sieve

Break.Set 0x1000 /Program          ; Set a software breakpoint to address
                                         ; 1000 (address 1000 is in RAM)

Break.Set 0x101000 /Program        ; Set an on-chip breakpoint to address
                                         ; 101000 (address 101000 is in FLASH)
```

\*) These commands open windows on the screen. The window position can be specified with the **WinPOS** command. Refer to the **PEDIT** command to write a script and to the **DO** command to start a script.

## General

---

- Locate the NEXUS connector as close as possible to the processor to minimize the capacitive influence of the line length and cross coupling of noise onto the NEXUS signals.

Ensure that the debugger signal (HRESET) is connected directly to the HRESET of the processor. This will provide the ability for the debugger to drive and sense the status of HRESET. The target design should only drive the HRESET with open collector, open drain. HRESET should not be tied to PORESET, because the debugger drives the HRESET and DSCK to enable BDM operation.

- Terminate MCKI, MSEI, MDIO:1 with 100 pF and 47  $\Omega$  in series, as close as possible to the corresponding CPU pin.
- Take care that the MDO6 line is connected to the MPC5xx pin **MBIO32B[6]/MPWM[4]/MDO[6]** and not to MPWM[18]/MDO[6].
- Pull up all inputs by 10 k $\Omega$  resistors to VREF, except RSTI/. (Refer to the Freescale Semiconductor recommendation AN2289/D)
- Connect all pins as recommended in AN2289/D.
- Do not use any cable extender.

## Correct Start-up Sequence for the NEXUS Debugger

---

1. Switch the power for the NEXUS debugger ON.
2. Start the TRACE32 software.
3. The **SYStem.Option.TriState** has to be OFF.
4. Switch ON the target.

If you don't care about the correct start-up sequence, the user program can start unintentionally before the debugger is activated. The result can be that the PLL is already set.

## Special Warning for MPC561 and MPC563

---

The MPC561 and MPC563 seem to be extremely sensitive about noise, overshots and wrong or missing termination of MCKI.

MCKI is the NEXUS clock from debugger to target.

## AXIOM Evaluation Board

---

The AXIOM EVB can have problems when using the MPC561 or MPC563. The reason is an open, not terminated line (DSCK\_MCKI) from J4 to one of the micor connectors at the base board.

To improve the behavior of the AXIOM EVB the following is recommended:

1. Cut the DSCK\_MCKI line on the base board close to J4.
2. Put a short cut jumper between pin 3 and pin 4 of the BDM connector.

## SYStem.Up Errors

---

The **SYStem.Up** command is the first command of a debug session where communication with the target is required. If you receive error messages while executing this command this may have the following reasons:

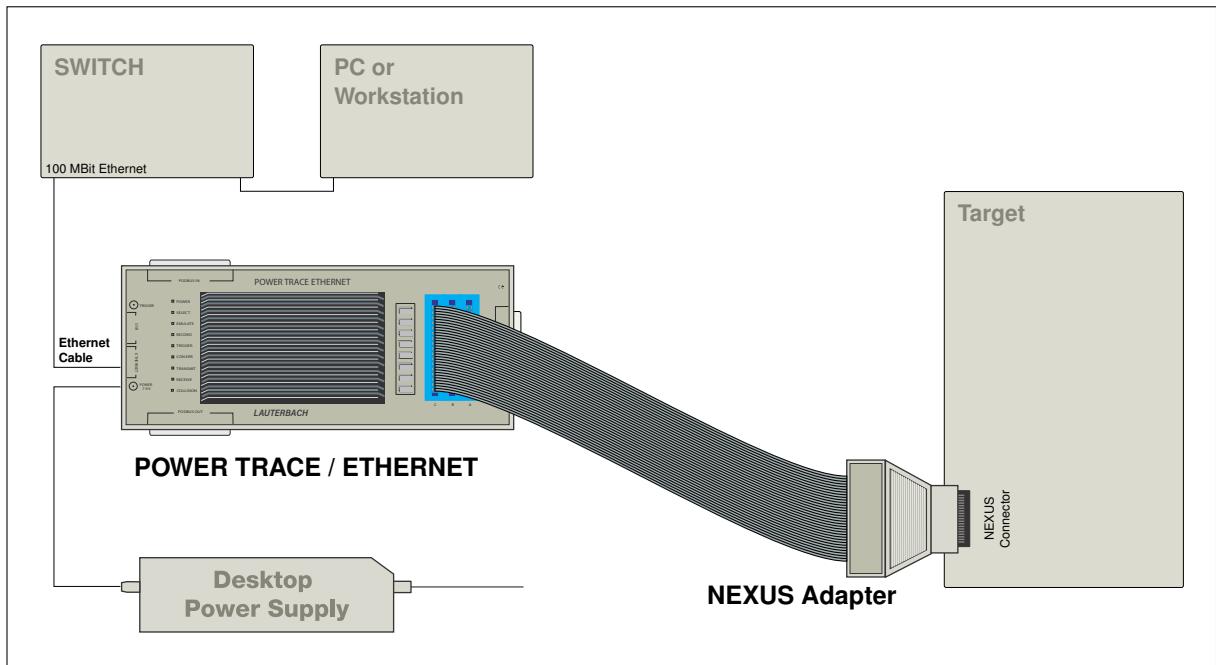
- The target has no power.
- The target is in reset: The debugger controls the processor reset and use the RESET line to reset the CPU on every SYStem.Up.

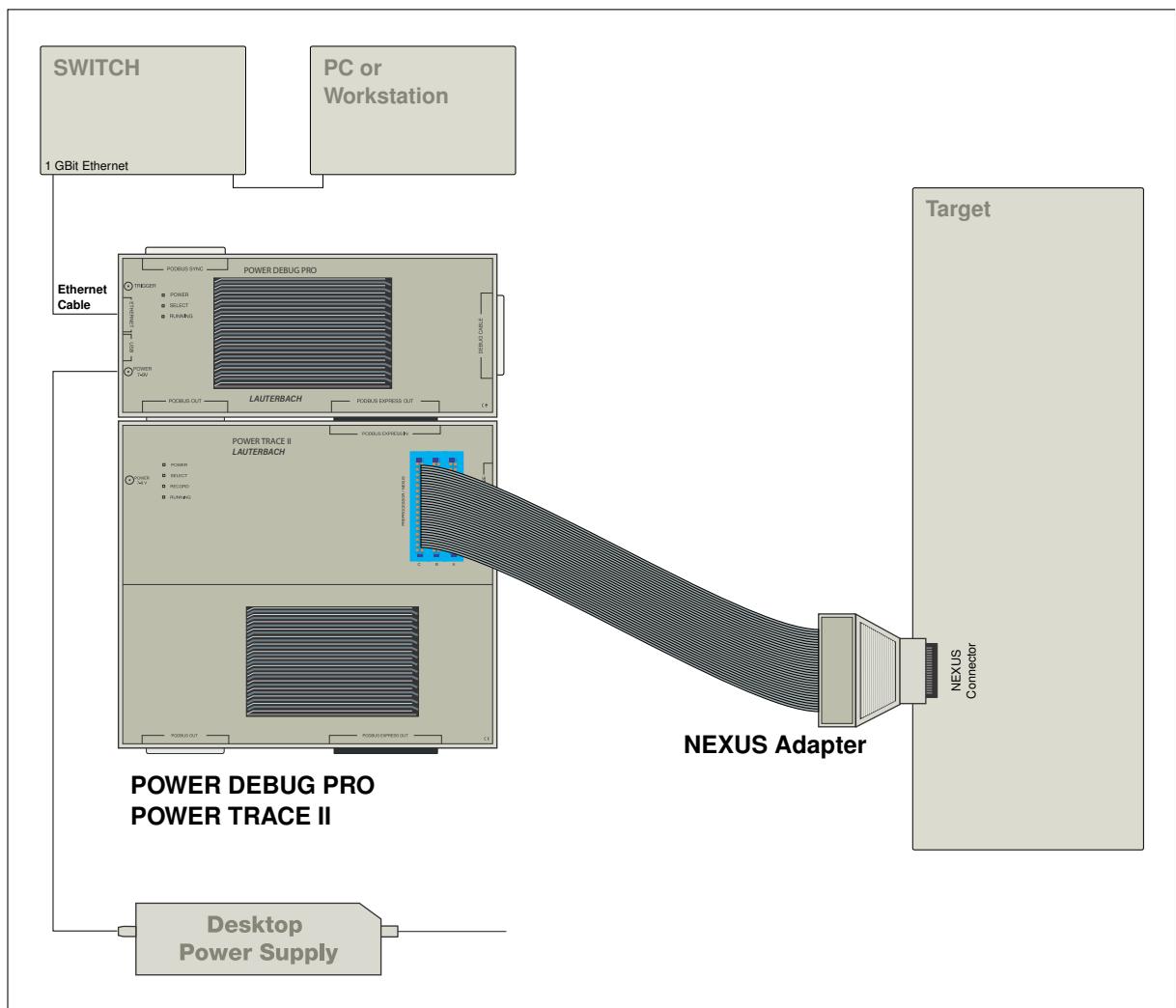
## FAQ

---

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

# Configuration





# Breakpoints

---

There are two types of breakpoints available: software breakpoints (SW-BP) and on-chip breakpoints (HW-BP).

## Software Breakpoints

---

Software breakpoints are the default breakpoints on instructions. Software breakpoints can be set to any instruction address in RAM and after some preparations also to instructions in FLASH. For more information, refer to the command **FLASH.AUTO**.

There is no restriction in the number of software breakpoints. Please consider that increasing the number of software breakpoints will reduce the debug speed.

## On-chip Breakpoints

---

The following list gives an overview of the usage of the on-chip breakpoints by TRACE32:

- **CPU family**
- **On-chip breakpoints:** Total amount of available on-chip breakpoints.
- **Instruction breakpoints:** Number of on-chip breakpoints that can be used for Program breakpoints.
- **Read/write breakpoints:** Number of on-chip breakpoints that can be used as Read or Write breakpoints.
- **Data breakpoints:** Number of on-chip data breakpoints that can be used to stop the program when a specific data value is written to an address or when a specific data value is read from an address.

CPU Family	On-chip Breakpoints	Instruction Breakpoints	Read/write Breakpoints	Data Breakpoints
MPC5xx	4 Instruction 2 Read/Write	4	2	2

# On-chip Breakpoints on InstructionsROM or FLASH

---

If a breakpoint is set to an instruction, a software breakpoint is used by default. If your code is in FLASH, ROM etc. you can advise TRACE32 to automatically use on-chip breakpoint for specific address ranges by using the command **MAP.BOnchip <range>**.

## On-chip Breakpoints on Read or Write Accesses

---

On-chip breakpoints are always used, if a Read or Write breakpoint is set. For the MPC5xx/8xx it is also possible to define a specific data value. Refer to the **Break.Set** command for more information.

## Example for Breakpoints

---

Assume you have a target with FLASH from 0 to 0xFFFF and RAM from 0x100000 to 0x11FFFF. The command to configure TRACE32 correctly for this configuration is:

```
Map.BOnchip 0x0--0x0FFFFF
```

The following breakpoint combinations are possible.

Software breakpoints:

```
Break.Set 0x100000 /Program ; Software Breakpoint 1
Break.Set 0x101000 /Program ; Software Breakpoint 2
Break.Set 0xx /Program ; Software Breakpoint 3
```

On-chip breakpoints:

```
Break.Set 0x100 /Program ; On-chip Breakpoint 1
Break.Set 0x0ff00 /Program ; On-chip Breakpoint 2
Break.Set flags /Write ; On-chip Breakpoint 3
Var.Break.Set \flags[3] /Write /DATA.Byte 0x1 ; On-chip Breakpoint 4
```

Simultaneous programming of the internal FLASH is currently not supported for the MPC56x.

## Programming Procedure

---

1. Load the application program into the virtual memory of TRACE32-ICD.

For the simultaneous FLASH programming the code can not directly be loaded from the host. The code has to be loaded into the virtual memory (VM) of TRACE32-ICD first.

TRACE32-PowerView can recognize empty 64-byte pages and skip them while programming. For this reason the virtual memory should be initialized with 0xff.

```
; initialize the virtual memory of TRACE32-ICD with 0xff
Data.Set VM:<start_address_internal_flash>++0x6ffff %Long
0xffffffff

; load the code for the internal FLASH into the virtual memory
Data.LOAD.Elf <file> <start_address_internal_flash>++0x6ffff /VM
```

2. Start the simultaneous programming.

```
FLASH.MultiProgram <start_address_internal_flash>++0x6ffff
```

If your application program also contains code for the external FLASH, this code has to be loaded separately.

# Memory Classes

---

The following memory classes are available:

Memory Class	Description
P	Program
D	Data
SPR	Special Purpose Register
NC	No Cache (only physically memory)
CP	Compressed Program

# General SYStem Commands

---

## SYStem.CONFIG

Configure debugger according to target topology

---

The **SYStem.CONFIG** command group is not supported for the MPC5xx.

## SYStem.CPU

Select CPU type

---

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

Selects the processor type.

## SYStem.LOCK

Lock and tristate the debug port

---

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

Default: OFF.

If the system is locked, no access to the debug port will be performed by the debugger. While locked, the debug connector of the debugger is tristated. The main intention of the **SYStem.LOCK** command is to give debug access to another tool. The command has no effect for the simulator.

## SYStem.MemAccess

Select run-time memory access method

---

Format: **SYStem.MemAccess NEXUS | Denied | StopAndGo**

Default: Denied.

### NEXUS

Memory access is done via the NEXUS interface.

<b>Denied</b>	Memory access during program execution to target is disabled.
<b>StopAndGo</b>	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.

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

Selects the target reset mode.

<b>Down</b>	Disables the debugger. Depending on the <b>SYStem.Option.TriState</b> The Nexus lines will be tristated or the CPU will be held in reset.
<b>StandBy</b>	This mode is used to start debugging from power-on. The debugger will wait until power-on is detected, then bring the CPU into debug mode, set all debug and trace registers and start the CPU. In order to halt the CPU at the first instruction, place an on-chip breakpoint to the reset address (Break.Set 0x100 /Onchip)
<b>Up</b>	Resets the CPU, enables the debug mode and stops the CPU at the first opfetch (reset vector). All registers are set to the default value.
<b>Go</b>	Resets the target with debug mode enabled and prepares the CPU for debug mode entry. After this command the CPU is in the system.up mode and running. Now, the processor can be stopped with the break command or until any break condition occurs.
<b>NoDebug</b>	Resets the target with debug mode disabled. In this mode no debugging is possible. The CPU state keeps in the state of NoDebug.
<b>Attach</b>	Not supported.

## Debugger functions for **StandBy**:

Target Power	HRESET/	Debugger State
not ok	xxx	<b>StandBy</b>
ok	active	<b>StandBy</b> (NEXUS initialized and waiting for HRESET/ no longer active)
ON	not active	<b>Up</b> (the NEXUS debugger enables the debug mode and starts the user program immediately. The program execution can be stopped manually or at an on-chip breakpoint)

The target power is checked by a voltage comparator. The voltage is defined by VREF.

If StandBy is selected, the NEXUS debugger will change into StandBy mode as soon as the target power fails or HRESET/ becomes active. To terminate this, select **SYStem.Mode Down** or **SYStem.Mode Up** explicitly.

# CPU specific SYStem Commands

---

## SYStem.LOADVOC

Load vocabulary for code compression

---

Format:	<b>SYStem.LOADVOC &lt;file&gt;</b>
---------	------------------------------------

Loads the vocabulary for code compression. This is usually not required, since the vocabulary is already in the ELF file.

## SYStem.Option.CCOMP

Enable code compression

---

Format:	<b>SYStem.Option.CCOMP [ON   OFF]</b>
---------	---------------------------------------

If the code compression unit of the MPC5xx is used, this option must be switched on before the program is loaded. Then correct disassembly is possible.

## SYStem.Option.CLEARBE

Clear MSR[BE] on step/go

---

Format:	<b>SYStem.Option.CLEARBE [ON   OFF]</b>
---------	-----------------------------------------

If the option CLEARBE is switched on, the BE bit of the MSR register will be cleared before every Go or Step.

## SYStem.Option.DCFREEZE

Freeze contents of cache while debugging

---

Format:	<b>SYStem.Option.DCFREEZE [ON   OFF]</b>
---------	------------------------------------------

If this feature is enabled the status of the data caches is preserved while debugging. This feature should be used in combination with **SYStem.Option.DCREAD** in order to read data as seen by the core. Otherwise all memory accesses are as for access class NC.

If disabled, the debugger might modify the caches contents with each data access e.g. a Data.dump window.

For caches that use hardware coherency (e.g. MESI protocol), the DCFREEZE feature is not supported. This respects multicore architectures that use non-shared caches.

## **SYSystem.Option.DUALPORT**

Run-time memory access for all windows

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

If **SYSystem.MemAccess NEXUS** is ON and **SYSystem.Option.DUALPORT** is ON, run-time memory access is automatically activated for each displayed memory location and variable.

## **SYSystem.Option.EXTVECTORS**

Workaround for revision C silicon

Format: **SYSystem.Option.EXTVECTORS <range>**

Workaround for revision C silicon.

## **SYSystem.Option.ETASFIX**

Workaround for MPC555 RiscTrace

Format: **SYSystem.Option.ETASFIX [ON | OFF]**

Enables Workaround for MPC555 RiscTrace.

## **SYSystem.Option.FAILSAVE**

Special error handling for debug port

Format: **SYSystem.Option.FAILSAVE [ON | OFF]**

The debug interface of the MPC8xx and MPC5xx returns the fatal error emulation debug port fail, when reading incorrect communication data from the debug port. With this option, it is possible to suppress this debug port fail, and recover the communication. This helps debugging in noisy environment.

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

Controls the internal CPU timer. If FREEZE is enabled, the timer will be stopped whenever the CPU enters the debug mode.

## **SYStem.Option.HighMemory**

Switch on high memory

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

If the upper addresses of the available address space are used to distinguish between the different chip select lines of the MPC56x, one must be aware that Nexus trace messages and dual ported memory access uses 25 address lines only (restricted by the Nexus aux. port protocol). A BDM-Debugger however, uses the full address space of 32 address lines. The only way to access full address space is to use the option **SYStem.Option.HighMemory ON**. Then all debugger accesses will be handled by BDM instructions in a Nexus message frame. In this case the CPU must be stopped in any case to access the memory. Bear in mind that the trace reconstruction can not work properly in this case, due to the fact that the address space in the trace messages can not be extended.

## **SYStem.Option.ICFLUSH**

Flush branch target cache before program start

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

Invalidates the instruction cache and flush the data cache before starting the target program (Step or Go). This is required when the CACHES are enabled and software breakpoints are set to a cached location.

## **SYStem.Option.IMASKASM**

Disable interrupts while single stepping

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.

## **SYStem.Option.LittleEnd**

Selection of little endian mode

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

With this option data is displayed little endian style.

Normally, the PowerPC debugger displays data big endian style.

## **SYStem.Option.NOTRAP**

Use alternative instruction to enter debug mode

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

Default: OFF.

By setting a software breakpoint the original code at the break location is patched by TRAP. If the TRAP command is already used by the application software for another purpose, an illegal instruction is patched instead of TRAP if the **SYStem.Option.NOTRAP** is ON.

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

Default: OFF.

<b>ON</b>	Activates the overlay extension and extends the address scheme of the debugger with a 16 bit virtual overlay ID. Addresses therefore have the format <code>&lt;overlay_id&gt;:&lt;address&gt;</code> . This enables the debugger to handle overlaid program memory.
<b>OFF</b>	Disables support for code overlays.
<b>WithOVS</b>	Like option <b>ON</b> , but also enables support for software breakpoints. This means that TRACE32 writes software breakpoint opcodes to both, the <i>execution area</i> (for active overlays) and the <i>storage area</i> . 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>
```

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

Normally, the PowerPC debugger displays data big endian style.

With this option data is displayed in PPC little endian style.

Available on: MPC5xx

Format: **SYStem.Option.SCRATCH <address> | AUTO**

Reading the FPU registers of the MPC5xx requires two memory words in target memory. This option defines which location is used. The content of the memory location will be restored after use.

## SYStem.Option.SLOWRESET

Activate SLOWRESET

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

After the debugger resets the CPU (e.g. via SYStem.Up), the debugger senses `HRESET` for 2 ... 3 s before an error message is displayed.

## SYStem.Option.TriState

Control for NEXUS lines

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

**ON** TriState NEXUS line in [SYStem.Down](#).

**OFF** Hold CPU in reset in [SYStem.Down](#).

## SYStem.Option.VECTORS

Define ranges for not-standard interrupt vectors

Format: **SYStem.Option.VECTORS <range> [<range> | <range> ...]**

Defines the address ranges for not-standard interrupt vectors for the disassembler. This is necessary if the interrupt vector table is relocated or if the enhanced interrupt control is used.

Format:

**SYStem.Option.WATCHDOG [ON | OFF]**

If this option is switched off, the watchdog timer of the CPU is disabled after the SYStem.Up.

Otherwise the watchdog will be periodically reset by the debugger. **Software Watchdog Timer (SWT)** — The SWT asserts a reset or non-maskable interrupt (as selected by the system protection control register) if the software fails to service the SWT for a designated period of time (e.g, because the software is trapped in a loop or lost). After a system reset, this function is enabled with a maximum time-out period and asserts a system reset if the time-out is reached. The SWT can be disabled or its time-out period can be changed in the SYPCR. Once the SYPCR is written, it cannot be written again until a system reset.



Software Watchdog Timer (SWT) — The SWT asserts a reset or non-maskable interrupt (as selected by the system protection control register) if the software fails to service the SWT for a designated period of time (e.g, because the software is trapped in a loop or lost). After a system reset, this function is enabled with a maximum time-out period and asserts a system reset if the time-out is reached. The SWT can be disabled or its time-out period can be changed in the SYPCR. Once the SYPCR is written, it cannot be written again until a system reset.

Format:

**SYStem.state**

Displays the **SYStem.state** window.

# CPU specific NEXUS Commands

---

## NEXUS.BTM

Program trace messaging enable

Format: **NEXUS.BTM [ON | OFF]**

Control for NEXUS program trace messaging.

**ON** (default) Program trace messaging enabled.

**OFF** Program trace messaging disabled.

## NEXUS.DTM

Enable data trace messaging

Format: **NEXUS.DTM <mode>**  
SYStem.Option.DTM [ON | OFF | Read | Write | ReadWrite] (deprecated)

<mode>: **ON | OFF | Read | Write | ReadWrite**

Controls the Data Trace Messaging method.

**OFF** Data trace messaging disabled (default)

**ON** (Same as ReadWrite)

**Read** Data trace messages for read accesses (load instructions)

**Write** Data trace messages for write accesses (store instructions)

**ReadWrite** Data trace messages for read and write accesses (load and store instructions)

Format: **NEXUS.OFF**

If the debugger is used stand-alone, the trace port is disabled by the debugger.

**NEXUS.ON**

Switch the NEXUS trace port on

Format: **NEXUS.ON**

The NEXUS trace port is switched on. All trace registers are configured by debugger.

**NEXUS.OTM**

Ownership trace messaging enable

Format: **NEXUS.OTM [ON | OFF]**  
SYStem.Option.OTM [ON | OFF] (deprecated)

Controls ownership trace messaging.

**OFF** Ownership trace messaging disabled (default)

**ON** Enable ownership trace messaging. An OTM is generated if the application writes to the PID register.

**NEXUS.PortSize**

Set trace port width

Format: **NEXUS.PortSize <port\_size>**  
SYStem.Option.Nexus <port\_size> (deprecated)

<port\_size>: **MDO8 | MDO2**

Sets the nexus port width to the number of used MDO pins. The setting can only be changed if no debug session is active ([SYStem.Down](#)).

Format: **NEXUS.PTM [ON | OFF]**

MPC565 and MPC566 only..

**ON** Enhanced program trace mode.

**OFF** Legacy program trace mode.

## **NEXUS.PTSM**

Enable program trace sync mode

Format: **NEXUS.PTSM [ON | OFF]**

If ON, Program trace messages contain the I-CNT packet. MPC565 and MPC566 only.

## **NEXUS.QFM**

Configure queue flush mode

Format: **NEXUS.QFM [ON | OFF]**

Configures Queue Flush Mode. MPC565 and MPC566 only.

**OFF** Information in the queue is removed.

**ON** Trace is stopped until the queue empties.

## **NEXUS.Register**

Display NEXUS trace control registers

Format: **NEXUS.Register**

This command opens a window which shows the NEXUS configuration and status registers of NPC, core and other trace clients.

Format: **NEXUS.RESet**

Resets NEXUS trace port settings to default settings.

**NEXUS.state**

Display NEXUS port configuration window

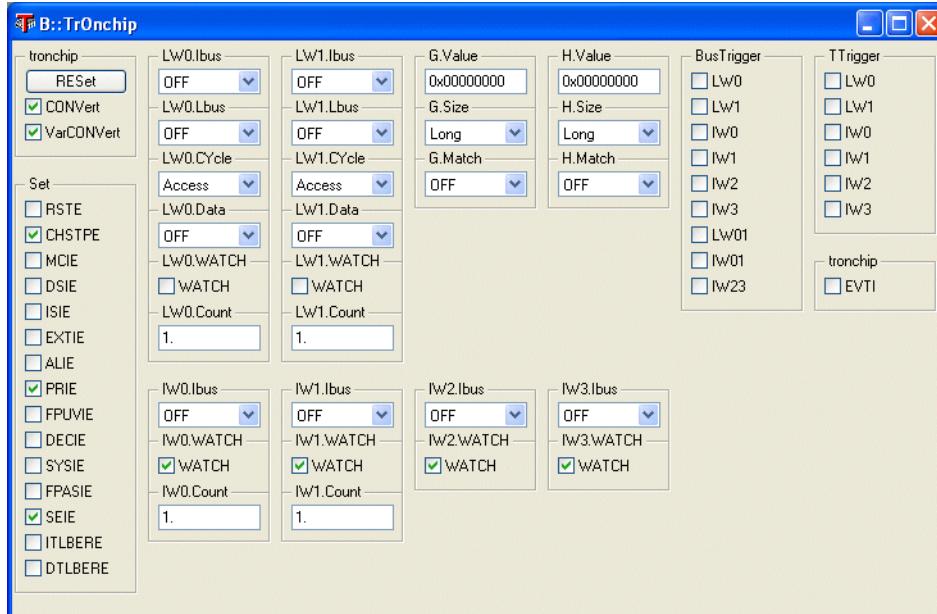
Format: **NEXUS.state**

Displays the NEXUS trace configuration window.

# CPU specific TrOnchip Commands

The following commands describe:

- the programming of the on-chip trigger resources for the MPC5xx
- the handling of the Debug Enable Register by TRACE32



Refer also to the User Manual of your processor for more information on both topics.

## TrOnchip.BusTrigger

Generate a trigger for the internal trigger bus

Format: **TrOnchip.BusTrigger <par>**

<par>: **All | LWO | LW1 | IW0 | IW1 | IW2 | IW3 | LW01 | IW01 | IW23**

Generates a trigger for the internal trigger bus, when the specified watchpoint is hit. The trigger is available on the TRIGGER connector of the POWERTRACE / ETHERNET.

**All** Generate a 100 ns trigger signal on any watchpoint hit.

**LW0 | LW1** Generate a 100 ns trigger signal when LW0/1 is hit.

**IW0 | IW1 | IW2 | IW3** Generate a 100 ns trigger signal when IW0/1/2/3 is hit.

**LW01**

Activate the trigger when LW0 is hit, deactivate the trigger when LW1 is hit.

**IW01 | IW23**

Activate the trigger when IW0 is hit, deactivate the trigger when IW1 is hit.

**Example:** Generate a 100 ns trigger pulse for the TRIGGER connector of the POWERTRACE / ETHERNET if the function sieve is entered.

```
; Program the on-chip trigger unit

Break.Set sieve /Alpha ; Set an Alpha breakpoint to the
; entry of sieve

TrOnchip.RESet ; Reset on-chip trigger unit

TrOnchip.IW0.Ibus Alpha ; The addresses marked with Alpha
; breakpoints define the I-Bus
; address

TrOnchip.IW0.WATCH ON ; Activate the watchpoint function

TrOnchip.BusTrigger IW0 ; Select IW0 as source for the
; trigger signal

Configure the internal trigger bus

TrBus.RESet ; Reset trigger bus settings

TrBus.Connect Out ; The TRIGGER connector works as
; output

TrBus.Mode Low ; The trigger signal is low active

TrBus.Out Break OFF
```

## TrOnchip.CONVert

Adjust range breakpoint in on-chip resource

Format:

**TrOnchip.CONVert [ON | OFF]**

For on-chip-breakpoints see the [corresponding chapter](#).

The MPC5xx family provides the following on-chip breakpoints:

<b>MPC5xx</b>	4 Instruction, 2 Read/Write	4 4 single I-bus breakpoints or 2 I-bus breakpoint ranges	2 2 single L-bus breakpoints or 1 L-bus breakpoint ranges	2
---------------	--------------------------------	-----------------------------------------------------------------------	--------------------------------------------------------------------------	---

**ON** (default) If all resources for the on-chip breakpoints are already used and if the user wants to set an additional on-chip breakpoint, TRACE32 converts an on-chip breakpoint set to a short address range (max. 4 bytes) to a single address breakpoint to free additional resources.

**OFF** If all resources for the on-chip breakpoints are already used and if the user wants to set an additional on-chip breakpoint, an error message is displayed.

#### Example:

```
TrOnchip.Convert ON

Break.Set 0x100++0x4 /Write      ; Set a write breakpoint to the
                                  ; address range 0x100++0x4

Break.Set 0x800 /Write           ; Set a write breakpoint to the address
                                  ; 0x800. The first set breakpoint is
                                  ; reduced to address 0x100
```

## TrOnchip.DISable

## Disable NEXUS trace register control

Format: **TrOnchip.DISable**

Disables NEXUS register control by the debugger. By executing this command, the debugger will not write or modify any registers of the NEXUS block. This option can be used to manually set up the NEXUS trace registers. The NEXUS memory access is not affected by this command. To re-enable NEXUS register control, use command **TrOnchip.ENABLE**. Per default, NEXUS register control is enabled.

Format: **TrOnchip.ENABLE**

Enables NEXUS register control by the debugger. By default, NEXUS register control is enabled. This command is only needed after disabling NEXUS register control using **TrOnchip.DISABLE**.

## TrOnchip.EVTI

Allow the EVT1 signal to stop the program execution

Format: **TrOnchip.EVTI [ON | OFF]**  
SYStem.Option.EVTIB (deprecated)

**ON** Allow the EVT1 signal to stop the program execution (faster).

**OFF** The program execution is stopped by sending a break sequence via NEXUS.

**Example:** Stop the program execution on the falling edge of the external signal on the TRIGGER connector of POWERTRACE / ETHERNET.

```
TrOnchip.EVTI ON ; Enable fast stop via an external
                   ; signal

; Configure the internal trigger bus

TrBus.RESet ; Reset trigger bus settings

TrBus.Connect In ; Configure TRIGGER as input

TrBus.Mode Falling ; The trigger is active on the falling edge
                   ; of the connected signal

TrBus.Set Break ON ; Define Break a trigger event

TrBus.Out Break OFF
```

Format: **TrOnchip.EVTO [ON | OFF]**

Default: OFF. If enabled, the debugger will use the EVTO for **Run-time** measurement and external watchdog control. This will improve the precision of run-time measurement and reduce external watchdog control delays.

## TrOnchip.EXternal

## Enable trace trigger input of NEXUS adapter

Format: **TrOnchip.EXternal <source>**

<source>: **OFF | IN0**

The NEXUS adapter provides an additional (active-high) trigger input to stop the trace recording. The input is labeled “IN” or “IN0” on LA-7610 and “IX0” on LA-7630 adapters. The input channel recognizes signals with a minimum pulse length of 20 ns.

The recorded value of the input channel can be observed in the Trigger.0 row of the **Trace.List** window.

```
; Show program flow and input channel
Trace.List DEFault Trigger.0
```

The Complex Trigger Unit (CTU) supports the input channel level as condition *IN*.

Format: **TrOnchip.G.Value** <hexmask> | <float>  
**TrOnchip.H.Value** <hexmask> | <float>  
**TrOnchip.G.Size** [Byte | Word | Long]  
**TrOnchip.H.Size** [Byte | Word | Long]  
**TrOnchip.G.Match** [OFF | EQ | NE | GT | LT | GE | LE]  
**TrOnchip.H.Match** [OFF | EQ | NE | GT | LT | GE | LE]

Defines the two data selectors of the MPC500/800 family.

<b>OFF</b>	Off
<b>EQ</b>	Equal
<b>NE</b>	Not equal
<b>LE</b>	Lower equal
<b>GE</b>	Greater equal
<b>LT</b>	Lower then
<b>GT</b>	Greater then
<b>ULE</b>	Unsigned lower equal
<b>UGE</b>	Unsigned greater equal
<b>ULT</b>	Unsigned lower then
<b>UGT</b>	Unsigned greater then

## TrOnchip.IWx

I-Bus watchpoint

### TrOnchip.IWx.Count

Event counter for I-Bus watchpoint

Format: **TrOnchip.IW0.Count** <count>  
**TrOnchip.IW1.Count** <count>

The occurrence of the specified I-Bus event can be counted.

Format: **TrOnchip.IW0.Ibus <selector>**  
**TrOnchip.IW1.Ibus <selector>**

<selector>: **OFF**  
**Alpha**  
**Beta**  
**Charly**  
**Delta**  
**Echo**

Defines the instruction for the I-Bus watchpoint.

## **TrOnchip.IWx.Watch**

Activate I-Bus watchpoint pin

Format: **TrOnchip.IW0.Watch [ON | OFF]**  
**TrOnchip.IW1.Watch [ON | OFF]**

**ON** A pulse is generated on IWP0/IWP1/IWP2/IWP3 if the I-Bus watchpoint is hit. The processor pins IWP0/IWP1/IWP2/IWP3 serve multiple functions. Please check your target hardware to find out which pin can be used for the trigger pulse. The smallest pulse length is one clock cycle.

**OFF** The program execution is stop on a hit of the L-Bus watchpoint.

## **TrOnchip.LWx**

L-Bus watchpoint

## **TrOnchip.LW0.Count**

Event counter for L-Bus watchpoint

Format: **TrOnchip.LW0.Count <count>**  
**TrOnchip.LW1.Count <count>**

The occurrence of the specified L-Bus event can be counted.

Format: **TrOnchip.LW0.CYcle <cycle>**  
**TrOnchip.LW1.CYcle <cycle>**

<cycle>: **Read**  
**Write**  
**Access**

Defines the cycle type for the L-Bus watchpoint.

Format: **TrOnchip.LW0.Data <selector>**  
**TrOnchip.LW1.Data <selector>**

<selector>: **OFF**  
**G**  
**H**  
**GANDH**  
**GORH**

Defines the data selector for the L-Bus watchpoint.

Format: **TrOnchip.LW0.Ibus <selector>**  
**TrOnchip.LW1.Ibus <selector>**

<selector>: **OFF**  
**Alpha**  
**Beta**  
**Charly**  
**Delta**  
**Echo**

Defines on which data address for the I-Bus watchpoint.

Format: **TrOnchip.LW0.Lbus <selector>**  
**TrOnchip.LW1.Lbus <selector>**

<selector>: **OFF**  
**Alpha**  
**Beta**  
**Charly**  
**Delta**  
**Echo**

Defines on which data address for the L-Bus watchpoint.

## **TrOnchip.LW0.Watch**

Activate L-Bus watchpoint pin

Format: **TrOnchip.LW0.Watch [ON | OFF]**  
**TrOnchip.LW1.Watch [ON | OFF]**

**ON** A pulse is generated on LWP0/LWP1 if the L-Bus watchpoint is hit. The processor pins LWP0/LWP1 serve multiple functions. Please check your target hardware to find out which pin can be used for the trigger pulse. The smallest pulse length is one clock cycle.

**OFF** The program execution is stop on a hit of the L-Bus watchpoint.

## **TrOnchip.RESet**

Reset on-chip trigger unit

Format: **TrOnchip.RESet**

Resets the on-chip trigger unit.

Format: **TrOnchip.Set <item> [ON | OFF]**

<item>: **CHSTPE ... SEIE**

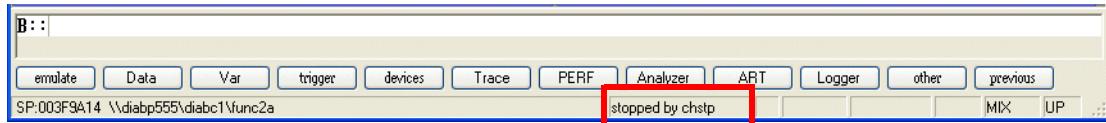
The program execution is stopped at the specified exception. For details and availability of a debug event on a specific processor, please refer to “Debug Enable Register (DER)” in the processor reference manual.

<b>ALIE</b>	Alignment Interrupt Enable.
<b>CHSTPE</b>	Checkstop Enable.
<b>DECIE</b>	Decrementer Interrupt Enable.
<b>DPIE</b>	Development Port Interrupt Enable.
<b>DSEE</b>	Data Storage Exception Enable.
<b>DTLBERE</b>	DTLB Error Interrupt Enable.
<b>DTLBMSE</b>	DTLB Miss Interrupt Enable.
<b>EBRKE</b>	External Breakpoint Interrupt Enable.
<b>EXTIE</b>	External Interrupt Enable.
<b>FPASEE</b>	Floating-point Assist Exception Enable.
<b>FPUVIE</b>	Floating-point Unavailable Interrupt Enable.
<b>IBRKE</b>	Instruction Breakpoint Interrupt Enable.
<b>ISEE</b>	Instruction Storage Exception Enable.
<b>ITLBERE</b>	ITLB Error Interrupt Enable.
<b>ITLBMSE</b>	ITLB Miss Interrupt Enable.
<b>LBRKE</b>	Load/store Breakpoint Enable.
<b>MCEE</b>	Machine Check Exception Enable.
<b>PRIE</b>	Program Interrupt Enable.
<b>TRE</b>	Trace Exception Enable.

<b>RSTE</b>	Reset Interrupt Enable.
<b>SEIE</b>	Software Emulation Interrupt En
<b>SYSIE</b>	System Interrupt Enable.

For details on the exceptions refer to the description of the Debug Enable Register in your processor manual.

If program execution is stopped by an exception, the name of the exception is shown in the command line of TRACE32. Refer to the description of the Exception Cause Register in your processor manual for details.



## TrOnchip.TEnable

## Set filter for the trace

Format: **TrOnchip.TEnable** <par> (deprecated)

Refer to the **Break.Set** command to set trace filters.

# TrOnchip.TOFF

Switch the sampling to the trace to OFF

Format: TrOnchip.TOFF (deprecated)

Refer to the **Break.Set** command to set trace filters.

TrOnchip.TON

Switch the sampling to the trace to ON

Format: TrOnchip.TON EXT | Break (deprecated)

Refer to the **Break.Set** command to set trace filters.

Format: **TrOnchip.TTrigger** <*par*> (deprecated)

Refer to the **Break.Set** command to set a trigger for the trace.

## TrOnchip.VarCONVert

Adjust HLL breakpoint in on-chip resource

Format: **TrOnchip.VarCONVert** [ON | OFF]

The MPC56x family provides the following on-chip breakpoints:

<b>MPC5xx</b>	4 Instruction, 2 Read/Write	4 4 single I-bus breakpoints or 2 I-bus breakpoint ranges	2 2 single L-bus breakpoints or 1 L-bus breakpoint ranges	2
---------------	--------------------------------	--------------------------------------------------------------------	--------------------------------------------------------------------	---

**ON** (default)

If all resources for the on-chip breakpoints are already used and if the user wants to set an additional on-chip breakpoint, TRACE32 converts an on-chip breakpoint set to a scalar variable to a single address breakpoint to free additional resources.

**OFF**

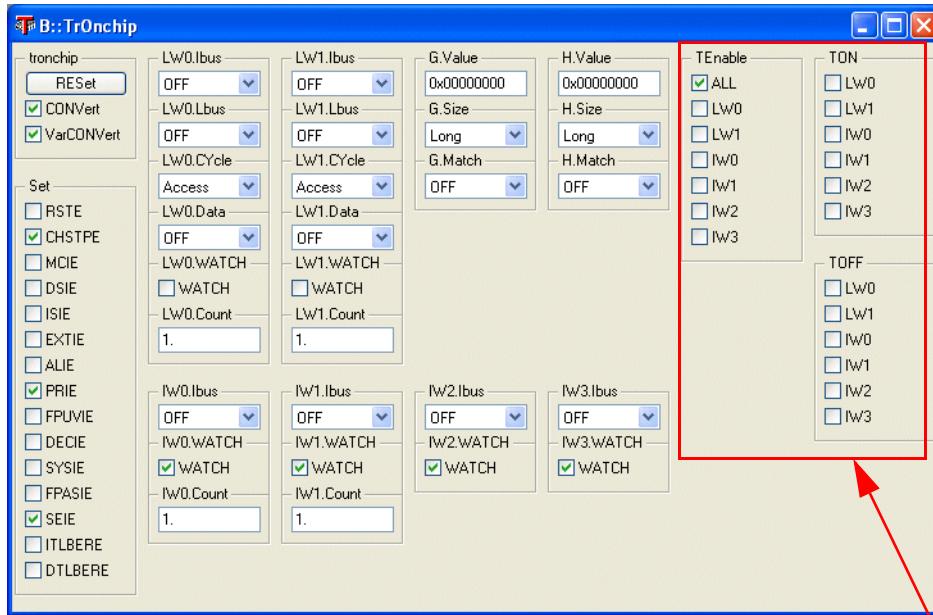
If all resources for the on-chip breakpoints are already used and if the user wants to set an additional on-chip breakpoint, an error message is displayed.

## TrOnchip.state

Display on-chip trigger window

Format: **TrOnchip.state**

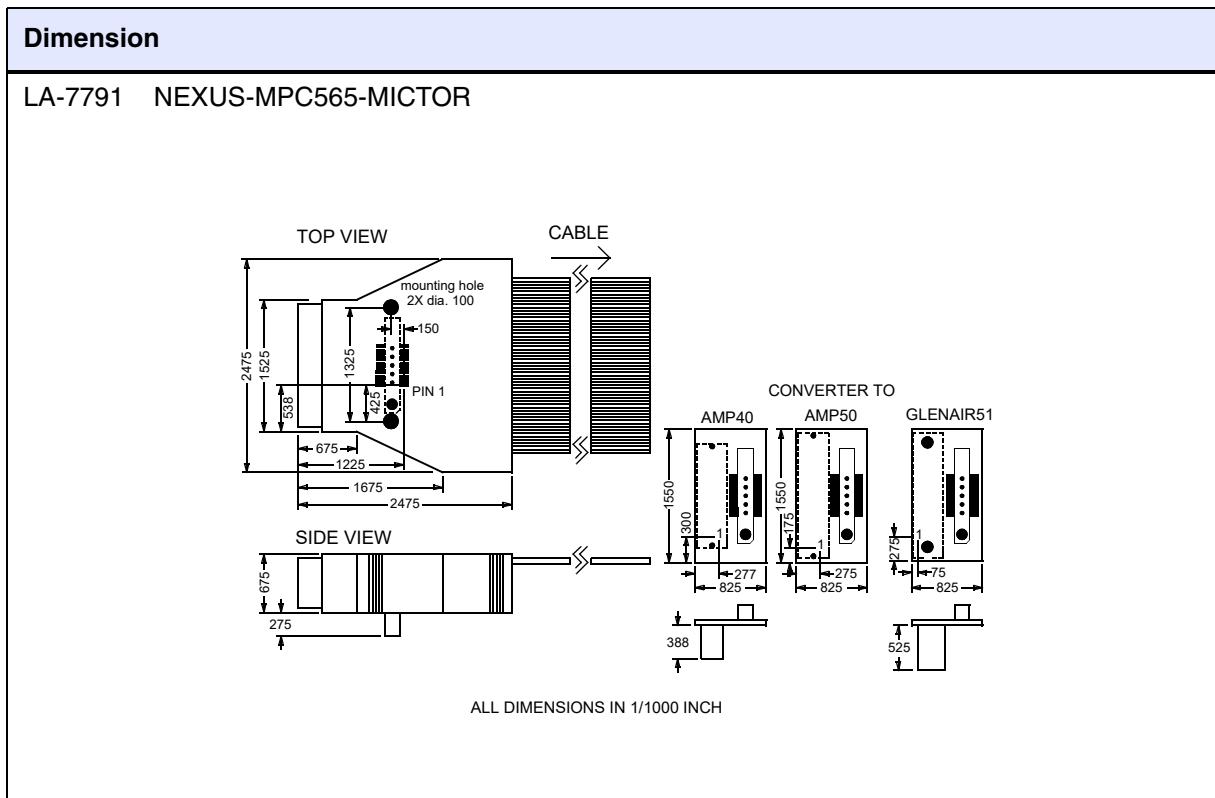
Opens the **TrOnchip.state** window.



Only available if Preprocessor for MPC500/800 is used

# Technical Data

## Mechanical Dimension



## Operation Voltage

Adapter	OrderNo	Voltage Range
Nexus Adapter for MPC56x family/Mictor38	LA-7791	2.3 .. 3.0 V