





[TRACE32 Online Help](#)

[TRACE32 Directory](#)

[TRACE32 Index](#)

TRACE32 Documents	
ICD In-Circuit Debugger	
Processor Architecture Manuals	
PQ/MPC500	
MPC5xx/8xx Debugger and Trace	1
Brief Overview of Documents for New Users	5
Warning	6
Quick Start	7
Target Design Requirement/Recommendations	10
General	10
RESET Configuration	11
BDM Termination	12
General Restrictions	13
Correct Start-up Sequence for the NEXUS Debugger	13
Special Warning for MPC561 and MPC563	14
AXIOM Evaluation Board	14
Troubleshooting	15
SYSem.Up Errors	15
FAQ	15
Configuration	16
Breakpoints	20
Software Breakpoints	20
On-chip Breakpoints	20
On-chip Breakpoints on InstructionsROM or FLASH	21
On-chip Breakpoints on Read or Write Accesses	21
Example for Breakpoints	21
Simultaneous FLASH Programming for MPC55556x	22
Memory Classes	23
Memory Coherency MPC8xx	23
Trace Extension	24
MPC555/MPC553 Pin Multiplexing	24

Troubleshooting MPC500/MPC800 RISC Trace	25
Used Options for RiscTrace	25
General SYStem Commands	26
SYStem.BdmClock	Define the BDM clock speed 26
SYStem.CONFIG	Configure debugger according to target topology 26
SYStem.CPU	Select CPU type 26
SYStem.MemAccess	Real-time memory access (non-intrusive) 27
SYStem.Mode	Establish the communication with the CPU 27
CPU specific SYStem Commands	29
SYStem.Option BASE	Set base address for on-chip peripherals 29
SYStem.LOADVOC	Load vocabulary for code compression 29
SYStem.Option BRKNOMSK	Allow program stop in a non-recoverable state 29
SYStem.Option CCOMP	Enable code compression 30
SYStem.Option CLEARBE	Clear MSR[BE] on step/go 30
SYStem.Option CLOCKX2	Select clock for real-time trace 30
SYStem.Option CSxxx	CS setting for program flow trace 31
SYStem.Option DCFREEZE	Freeze contents of cache while debugging 31
SYStem.Option DCREAD	Use DCACHE for data read 32
SYStem.Option DUALPORT	Run-time memory access for all windows 32
SYStem.Option FAILSAVE	Special error handling for debug port 32
SYStem.Option FREEZE	Stop timer in debug mode 33
SYStem.Option FreezePin	Use alternative signal on the BDM connector 33
SYStem.Option HighMemory	Switch on high memory 33
SYStem.Option IBUS	Configure the show cycles for the I-BUS 34
SYStem.Option ICFLUSH	Flush branch target cache before program start 34
SYStem.Option ICREAD	Use ICACHE for program read 35
SYStem.Option IMASKASM	Disable interrupts while single stepping 35
SYStem.Option IMASKHLL	Disable interrupts while HLL single stepping 35
SYStem.Option LittleEnd	Selection of little endian mode 36
SYStem.Option MMUSPACES	Enable space IDs 36
SYStem.Option NODATA	The external data bus is not connected to trace 37
SYStem.Option NOTRAP	Use alternative instruction to enter debug mode 37
SYStem.Option OVERLAY	Enable overlay support 38
SYStem.Option PPCLittleEnd	Control for PPC little endian 38
SYStem.Option SCRATCH	Scratch for FPU access 39
SYStem.Option SIUMCR	SIUMCR setting for the trace 39
SYStem.Option SLOWLOAD	Alternative data load algorithm 39
SYStem.Option SLOWRESET	Activate SLOWRESET 39
SYStem.Option STEPSOFT	Use alternative method for ASM single step 40
SYStem.Option VECTORS	Define ranges for not-standard interrupt vectors 40
SYStem.Option VFLS	Use VFLS pins for run/stop detection 40
SYStem.Option WATCHDOG	Enable software watchdog after SYStem.Up 40
SYStem.state	Display SYStem window 41

CPU specific MMU commands		42
MMU.DUMP	Page wise display of MMU translation table	42
MMU.List	Compact display of MMU translation table	43
MMU.SCAN	Load MMU table from CPU	44
MMU.TLB	Display MMU TLB entries	45
MMU.TLBSCAN	Load MMU TLB entries	45
MMU.Set	Set an MMU TLB entry	46
CPU specific TrOnchip Commands		47
TrOnchip.CONVert	Adjust range breakpoint in on-chip resource	47
TrOnchip.DISable	Disable NEXUS trace register control	47
TrOnchip.ENABLE	Enable NEXUS trace register control	47
TrOnchip.G/H	Define data selector	48
TrOnchip.IWx	I-Bus watchpoint	49
TrOnchip.IWx.Count	Event counter for I-Bus watchpoint	49
TrOnchip.IWx.Ibus	Instructions address for I-Bus watchpoint	49
TrOnchip.IWx.Watch	Activate I-Bus watchpoint pin	50
TrOnchip.LWx	L-Bus watchpoint	50
TrOnchip.LW0.Count	Event counter for L-Bus watchpoint	50
TrOnchip.LW0.CYcle	Cycle type for L-Bus watchpoint	50
TrOnchip.LW0.Data	Data selector for L-Bus watchpoint	51
TrOnchip.LW0.Ibus	Instructions address for L-Bus watchpoint	51
TrOnchip.LW0.Watch	Activate L-Bus watchpoint pin	52
TrOnchip.RESet	Reset on-chip trigger unit	52
TrOnchip.Set	Stop program execution at specified exception	52
TrOnchip.TCompress	Trace data compression	53
TrOnchip.TEnable	Set filter for the trace	53
TrOnchip.TOFF	Switch the sampling to the trace to OFF	54
TrOnchip.TON	Switch the sampling to the trace to ON	54
TrOnchip.TTrigger	Set a trigger for the trace	54
TrOnchip.VarCONVert	Adjust HLL breakpoint in on-chip resource	54
TrOnchip.state	Display on-chip trigger window	54
BDM Connector		56
10 pin BDM Connector MPC500/MPC800		56

The screenshot displays the Trace32 PowerPC debugger interface with the following components:

- Top Panel:** Menu bar (File, Edit, View, Var, Break, Run, CPU, Misc, Trace, Perf, Cov, MPC555, Window, Help) and a toolbar with various execution and debugging icons.
- B::Data List:** A table showing source code lines with addresses and line numbers.

addr/line	source
621	for (i = 0 ; i <= SIZE ; flags[i++] = TRUE) ;
623	for (i = 0 ; i <= SIZE ; i++)
625	{ if (flags[i])
627	{ primz = i + i + 3;
- B::Var. Frame /Locals /Caller:** A window showing the current stack frame for the `sieve()` function.


```

000|sieve()
    * i = 19
    * primz = -1048642
    * k = -1354776577
    * anzahl = 0
001|main()
    * j = 12345678
    * p = 0x003FAE48
    while ( TRUE )
    {
        sieve();
002|__init_main(asm)
003|start(asm)
-----end of frame
            
```
- B::PER "SIU":** A window showing the SIU System Configuration.


```

SIU System Configuration
IMMR 30310800 PARTNUM 00000030 MASKNUM 00000031 FLEN yes CLES no
SIUMCR 00C84000 EARB int EARP 0 DSHW yes DBGK BDM/full_trace DB
        GPC pflow/l-watch DLK no SC pflow_only RCTX rstconf
SYPCR FFFFFFF8 SWTC FFFF BMT FF BME ena SWF ena SWE dis SWRI in
SUIR FFFF
            
```
- B::Trace List:** A table showing the execution trace.

record	run	address	cycle	d.l	symbol	ti.back
-00000142		subi	r12,r12,0x50D0	; r12,r12,20688		
	P:003FA97C	execute			\\diabp555\diabc1\sieve+0x34	0.140us
-00000139		li	r11,0x1	; r11,1		
	P:003FA980	execute			\\diabp555\diabc1\sieve+0x38	0.160us
-00000136		stbx	r11,r12,r31	; r11,r12,i		
	P:003FA984	execute			\\diabp555\diabc1\sieve+0x3C	0.140us
-00000132		addi	r31,r31,0x1	; i,i,1		
	P:003FA988	execute			\\diabp555\diabc1\sieve+0x40	0.200us
- Bottom Panel:** A toolbar with buttons for emulate, Data, Var, trigger, devices, Trace, PERF, Analyzer, ART, Logger, other, and previous. Below the toolbar, the current state is shown as "SP:003FA98C \\diabp555\diabc1\sieve+0x44 stopped by ebrk HLL UP".

Architecture-independent information:

- **“Debugger Basics - Training”** (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.

WARNING:

To prevent debugger and target from damage it is recommended to connect or disconnect the debug cable only while the target power is OFF.

Recommendation for the software start:

1. Disconnect the debug cable from the target while the target power is off.
2. Connect the host system, the TRACE32 hardware and the debug cable.
3. Power ON the TRACE32 hardware.
4. Start the TRACE32 software to load the debugger firmware.
5. Connect the debug cable to the target.
6. Switch the target power ON.
7. Configure your debugger e.g. via a start-up script.

Power down:

1. Switch off the target power.
2. Disconnect the debug cable from the target.
3. Close the TRACE32 software.
4. Power OFF the TRACE32 hardware.

Starting up the BDM 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 MPC565.

The default CPU is the MPC860.

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++0x0fffff
```

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 NONE ; No show cycles are performed.
; Recommended for BDM debugger only.

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 SYS.O 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++0x0ffff ; Specify where's FLASH/ROM

SYSem.CPU 0x563 ; Select the processor type

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

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

- The TRACE32 internal buffer/level shifter will be supplied via the VCCS pin. Therefore it is necessary to reduce the VCCS pull-up on the target board to a value smaller 10 Ω .
- Terminate MCKI, MSEI, MDIO:1 with 100 pF and 47 Ω in series, as close a 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 Ω 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.

RESET Configuration

At HRESET the Hard Reset Configuration bits will be sampled. Depending on the $\overline{\text{RSTCONF}}$ pin the external or the internal configuration word is sampled.

RSTCONF	Configuration Word
0	DATA[0..31] pins
1	internal data default word (0x0000 0000)

The multifunction I/O pins (VFLS0/1) have to be configured correctly for the debugging. Drive actively the following pins:

MPC5xx DBGC(D9,D10) and DBPC(D11)

MPC8xx DBGC(D9,D10) and DBPC(D11,D12)

There are two signal schemes possible to indicate the processor status to the debugger. Option A is recommended but Option B is also supported for the BDM functionality.

Option B is used as an alternative to eliminate pin conflicts. Option B is typically used if:

- the internal watchpoints are used
- the amount of signals must be reduced to a minimum
- the target design uses PCMCIA Port B.

Option A: Using the VFLS pins

MPC800: (DBGC=[11]; DBPC=0; FRC=x)

MPC500: (DBGC=[00,10]; DBPC=0; GPC=x)

Comment	Signal Name	PIN	PIN	Signal Name	Comment
	IPB0/IWP0/VFLS0	1	2	/SRESET	
	GND	3	4	DSCK/TCK	
	GND	5	6	IP_BI/IWP1/VFLS1	
	HRESET	7	8	DSDI/TDI	
	VCCS	9	10	DSDO/TDO	

Option B: Using the FREEZE pin

MPC800: (DBGC=[11]; DBPC=0; FRC=0)

MPC500: (DBGC=[00,10]; DBPC=0; GPC=[10,11])

Comment	Signal Name	PIN	PIN	Signal Name	Comment
	FRZ/IRQ6	1	2	/SRESET	
	GND	3	4	DSCK/TCK	
	GND	5	6	FRZ/IRQ6	
	HRESET	7	8	DSDI/TDI	
	VCCS	9	10	DSDO/TDO	



If option B is used, the **SYSTEM.Option.FreezePin** must be switched on

When the PowerPC's development port (BDM) is used, the JTAG functionality is disabled.

BDM Termination

T32 PU/PD	Target PU/PD	Signal Name	PIN	PIN	Signal Name	Target PU/PD	T32 PU/PD
-	47kPU	FRZ/VFLS 0	1	2	/SRESET	10kPU	-
-	-	GND	3	4	DSCK	10kPD	4k7PD
-	-	GND	5	6	FRZ/VFLS 1	47kPU	-
10kPU	10kPU	HRESET	7	8	DSDI	10kPD	4k7PD
-	<10	VCCS	9	10	DSDO	>10k	-

General Restrictions

The CPU handles the debug mode similar to an exception.

SYStem.Option BRKNOMSK OFF: The program execution is not stopped as long as the processor is in a non-recoverable state (RI bit cleared in the Machine Status register).

SYStem.Option BRKNOMSK ON: The program execution can be stopped by a breakpoint even if the processor is in a non-recoverable state. Since the debug exception overwrites SRR0 and SRR1 it is not advisable to continue the debugging process.

MPC5xx	The CPU handles the debug mode similar to an exception. Therefore stopping during the non-recoverable state of the CPU will cause the SRR0/1 registers to be lost. Breakpoints should not be placed at the start and end of exception handlers to avoid this problem. Asynchronous breakpoints can be disabled when the CPU is in non-recoverable state (SYStem.Option BRKNOMSK command). Executing a GO command is not allowed when the CPU is in non-recoverable state. Single stepping on assembler level is allowed.
---------------	---

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 mictor 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:

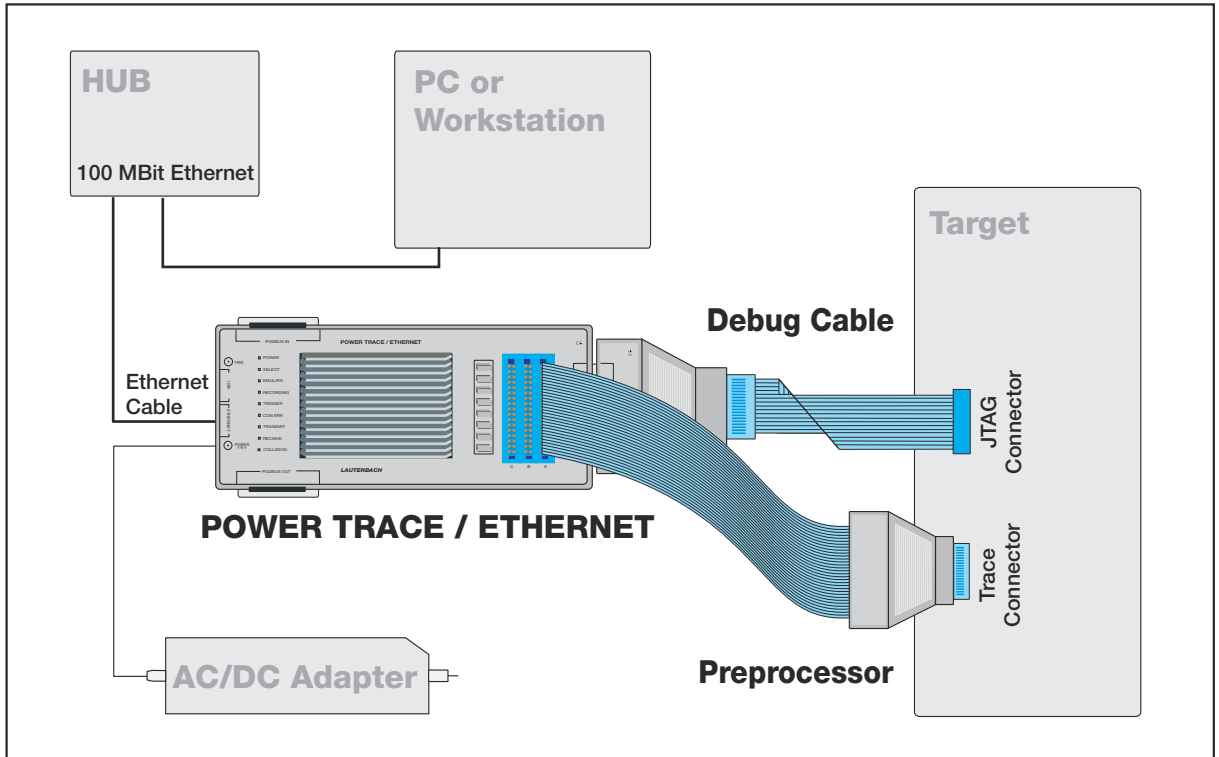
- The target has no power.
- The pull-up resistor between the JTAG/COP[VCCS] pin and the target VCC is too large.
- The target is in reset: The debugger controls the processor reset and use the RESET line to reset the CPU on every SYStem.Up.
- There is logic added to the JTAG/COP state machine: The debugger supports only one processor on one JTAG chain. Only the debugged processor has to be between TDI and TDO in the scan chain. No further devices or processors are allowed.
- There are additional loads or capacities on the JTAG lines.

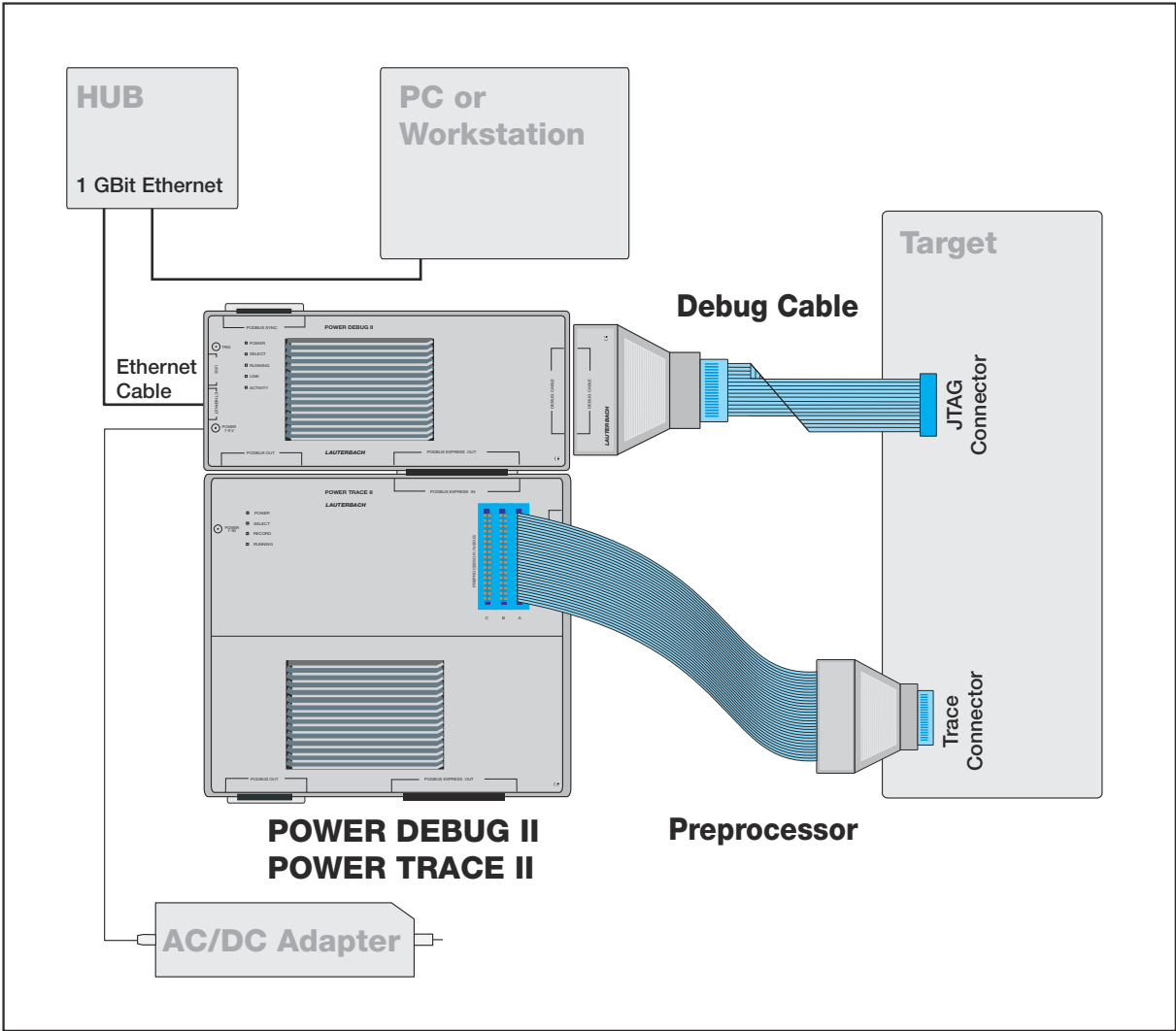
Target power fail	The target has no power.
Emulation debug port fail	HRESET/ is permanently active
NEXUS transmitter can't send data	Bad connection to the target. PowerOnRESET/ active or not connected

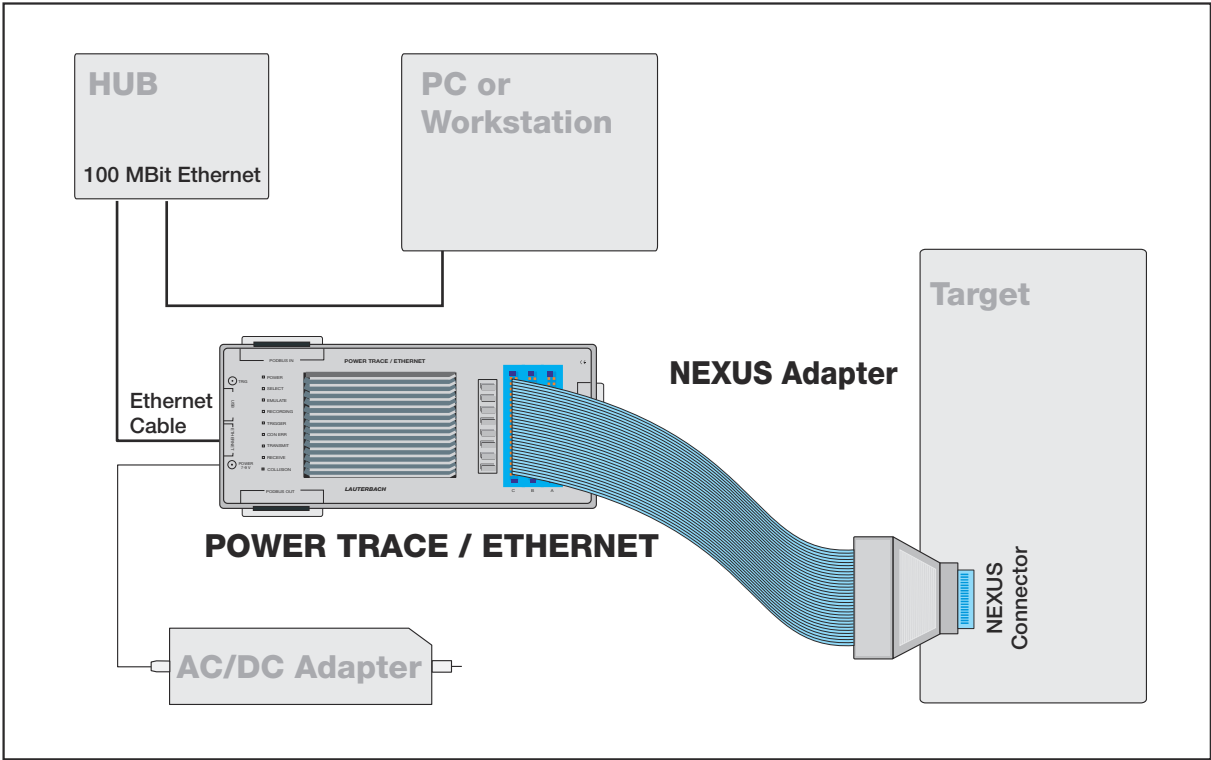
FAQ

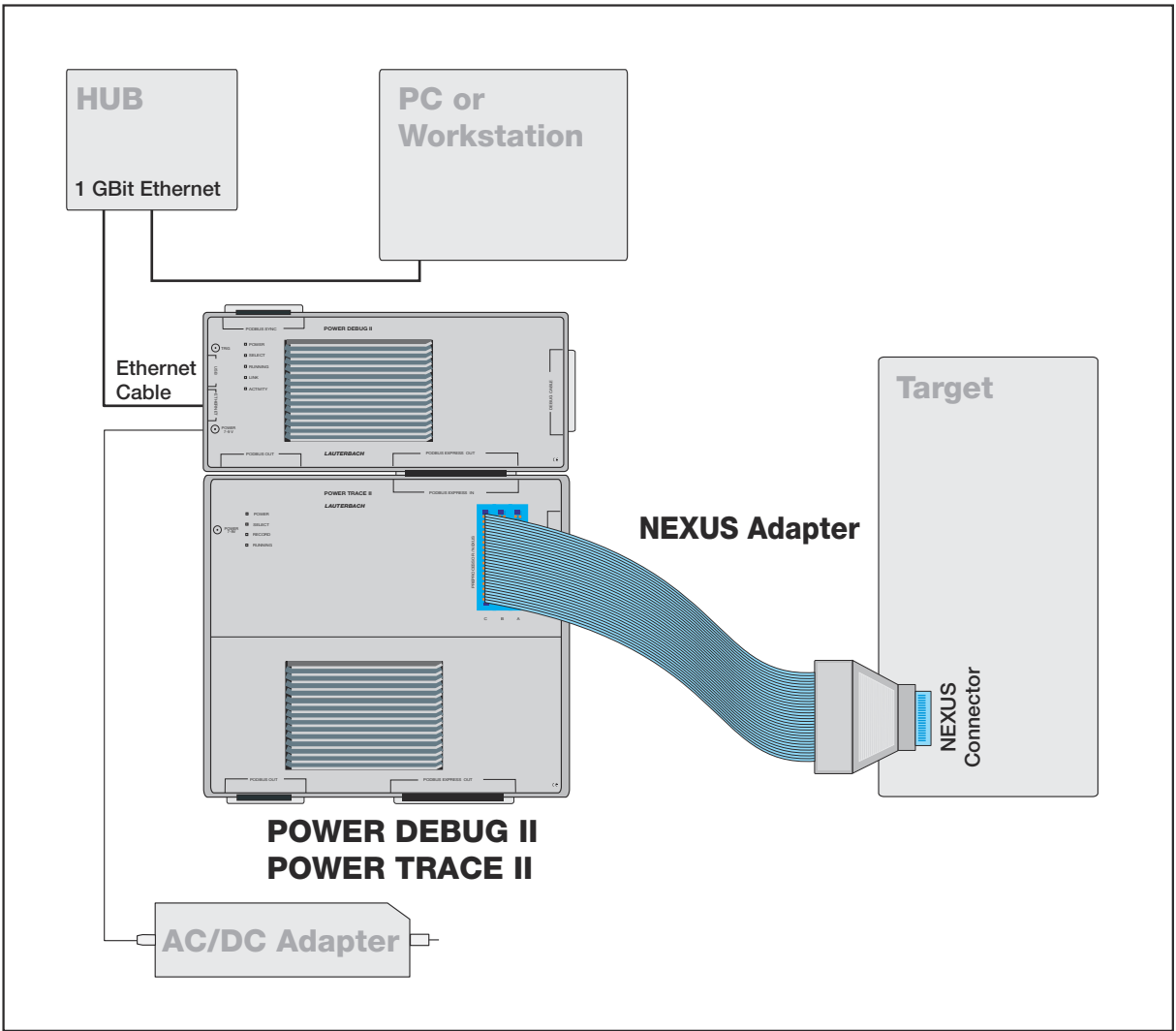
Please refer to our Frequently Asked Questions page on the Lauterbach website.

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
MPC500/800MPC5xx	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 0xFFFFF and RAM from 0x100000 to 0x11FFFF. The command to configure TRACE32 correctly for this configuration is:

```
Map.BOnchip 0x0--0xFFFFF
```

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 FLASH Programming for MPC5556x

Simultaneous programming of the internal FLASH is supported for the masks K1, K2, K3 and M of the MPC555.

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

The MPC555 supports simultaneous programming of all 14 flash modules.

- 8 64-byte pages in the 8 blocks of FLASH module A
- 6 64-byte pages in the 6 blocks of FLASH module B

Using simultaneous FLASH programming is up to 7 times faster!

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
IC	Instruction Cache (MPC8xx only)
DC	Data Cache (MPC8xx only)
NC	No Cache (only physically memory)
CP	Compressed Program

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.

Memory Coherency MPC8xx

Memory coherency on access to the following memory classes. If data will be set to DC, IC, NC, D or P the D-Cache, I-Cache or physical memory will be updated.

	D-Cache	I-Cache	Physical Memory
DC:	Yes	No	Yes
IC:	No	Yes	Yes
NC:	No	No	Yes
D:	Yes	Yes	Yes
P:	Yes	Yes	Yes

See also **SYSTEM.Option ICREAD** and **SYSTEM.Option DCREAD**.

MPC555/MPC553 Pin Multiplexing

CLKOUT	Always required.
A8..A29	Are always required.
D0..D11	Are required for tracing in compressed mode.
WR	Is required.
STS	Is not present when SIUMCR.DBGC== 00. In this case it is assumed that the program trace show cycle for indirect change of flow is appearing directly at the same clock where the indirect change of flow is shown. This should be always the case when running only with internal memories and having only indirect program show cycles active (no data cycles or data show cycles).
PTR	Is not present when SIUMCR.GPC !=00. In this case ALL program cycles are assumed to be program trace cycles. This is always the case when the program is running from internal memory and only indirect show cycles are enabled. When external program memory is used the trace may not be able to take the correct cycle as target for the indirect branch.
AT(2)	Is taken from the WE2/AT2 line when SIUMCR.ATWC==1 (AT0-3 lines enabled) or taken from the dedicated AT(2) line when SIUMCR.ATWC==0 (WE0-3 lines enabled) and SIUMCR.MLRC ==x1 (AT(2) function enabled). When non of the two variants is possible the debugger will assume that ALL cycles are program cycles (no data cycles). The program flow trace will not be affected by this, as long as the PTR line is available. When the AT(2) and PTR lines are both not available the trace will only work when the code is running from internal memory and only "indirect change of flow" show cycles are enabled.
VF0,VF1	Is taken from SIU when SIUMCR.DBGC==10, otherwise from the MIOS pins. MIOS must be configured when MIOS pins are used. If none of the pins are available then the program flow trace will not work. Direct cycle tracing in fully serialized mode with show cycles for all cycles will still work.
VFLS0,VFLS1	Is taken from SIU when SIUMCR.DBGC==x0, otherwise from the MIOS pins. MIOS must be configured when MIOS pins are used.
LWPx, IWPx	Optional lines. Only used when selective tracing features should be used.

Target is not running with trace attached

Some trace adapters use drivers with “Bus Hold” feature. This resistor (about 20 k Ω) can pull the lines connected to the trace to VCC or Ground. If the target is using high impedance resistors to select a specific level for the reset configuration it may not work. In this case make either the resistors on the target smaller or disable the external reset configuration. Pulling down the TS line may also cause such effects. Use a pull-up resistor (about 10 k Ω) in this case.

Nothing recorded (number of records in Analyzer.state window remains 0)

Check that CLKOUT is available on the trace probe. Check that VFLS0 and VFLS1 are correctly configured.

No cycle information displayed in Analyzer.List

Check the TS and STS signals.

Cycle type information in Analyzer.List is wrong

Check the RW and AT lines (CT lines for MPC50x).

Address information is wrong for DRAM accesses

Define DRAM areas with **MAP.DMUX** command.

Flowtrace (Analyzer.List /FT) gives no useful results

Make sure that indirect branch program trace cycles are enabled (**SYSTEM.Option ICTL IND**). Check that PTR signal is correctly recorded in trace. Check for presence of VF0, VF1 and VF2 signals. Make sure that program has executed an indirect branch while sampling data for the trace.

Used Options for RiscTrace

- **SYSTEM.OPTION NODATA** ON /OFF
- **SYSTEM.OPTION SIUMCR** ON /OFF
- SIUMCR Register [DBG, GPC] (Peripheral Window)

SYStem.BdmClock

Define the BDM clock speed

```
Format:          SYStem.BdmClock <rate>

<rate>:         EXT/4 | EXT/8 | EXT/16 | <fixed>

<fixed>:       1MHz ... 20MHz
```

Selects the frequency for the debug interface. A fixed frequency or a divided external clock can be used.

SYStem.CONFIG

Configure debugger according to target topology

There are no multicore capable CPUs available at the moment.

SYStem.CPU

Select CPU type

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

Selects the processor type.

Format: **SYSystem.MemAccess | Denied | StopAndGo**

Denied	No run-time memory access is possible for the MPC5xx/8xx family.
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.

SYSystem.Mode

Establish the communication with the CPU

Format: **SYSystem.Mode <mode>**

<mode>:
Down
StandBy
Up
Go
NoDebug

Selects the target reset mode.

Down	Disables the debugger.
StandBy	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)
Up	Resets the CPU, enables the debug mode and stops the CPU at the first opfetch (reset vector). All register are set to the default value.
Go	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.

NoDebug	Resets the target with debug mode disabled. In this mode no debugging is possible. The CPU state keeps in the state of NoDebug.
Attach	Not supported.

CPU specific SYStem Commands

SYStem.Option BASE

Set base address for on-chip peripherals

Format: **SYStem.Option Base** [AUTO | <value>]

Sets base address for on-chip peripherals. MPC800 only.

SYStem.LOADVOC

Load vocabulary for code compression

Format: **SYStem.LOADVOC** <file>

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

SYStem.Option BRKNOMSK

Allow program stop in a non-recoverable state

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

The CPU handles debug events similar to exceptions. When a debug event (normally a break) OR an exception occurs, the CPU copies the MSR (Machine Status Register) into SRR1 (Machine Status Save/Restore Register 1) and the IP (Instruction Pointer) into SRR0 (Machine Status Save/Restore Register 1). This means that after an exception occurred, the old values of IP and MSR are as backup in the SRR0 and SRR1 registers. If now a break happens, these values will be overwritten by the new MSR and IP values. So, it is possible to return to the exception routine and stop the processor, **but it's not possible to return to the main program and continue the user application!** The status after the start of the exception routine is called non recoverable state.

If one wants to break in a non recoverable state, you must switch the option BrkNoMsk to on.

ON

The program execution can be stopped by a breakpoint even if the processor is in a non-recoverable state. Since the debug exception overwrites SRR0 and SRR1 it is not advisable to continue the debugging process.

OFF

The program execution is not stopped as long as the processor is in a non-recoverable state (RI bit cleared in the Machine Status register).

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

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: **SYStem.Option CLEARBE [ON | OFF]**

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

SYStem.Option CLOCKX2

Select clock for real-time trace

Format: **SYStem.Option CLOCKX2 [ON | OFF]**

This option selects the clock for the Real-Time Trace. Option available for the TRACE32-ICD Risc Trace Module.

Format: **SYStem.Option CBTOR** [*<value>*]
 (MPC505, **SYStem.Option CSBTBAR** [*<value>*]
 MPC509) **SYStem.Option CSBTSBBAR** [*<value>*]
SYStem.Option CS0OR [*<value>*]
SYStem.Option CS1OR [*<value>*]
SYStem.Option CS2OR [*<value>*]
SYStem.Option CS3OR [*<value>*]
SYStem.Option CS4OR [*<value>*]
SYStem.Option CS5OR [*<value>*]
SYStem.Option CS6OR [*<value>*]
SYStem.Option CS7OR [*<value>*]
SYStem.Option CS8OR [*<value>*]
SYStem.Option CS9OR [*<value>*]
SYStem.Option CS10OR [*<value>*]
SYStem.Option CS11OR [*<value>*]
SYStem.Option CS1BAR [*<value>*]
SYStem.Option CS2BAR [*<value>*]
SYStem.Option CS3BAR [*<value>*]
SYStem.Option CS4BAR [*<value>*]
SYStem.Option CS5BAR [*<value>*]

Format: **SYStem.Option CS0BR** [*<value>*]
 (MPC850) **SYStem.Option CS1BR** [*<value>*]
SYStem.Option CS2BR [*<value>*]
SYStem.Option CS3BR [*<value>*]
SYStem.Option CS4BR [*<value>*]
SYStem.Option CS5BR [*<value>*]
SYStem.Option CS6BR [*<value>*]
SYStem.Option CS7BR [*<value>*]

For the flow trace functionality, it is necessary for the software to know the settings of the CS unit. The values of these options must be the same values as the register values of the chip.

SYStem.Option DCFREEZE

Freeze contents of cache while debugging

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

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.

SYStem.Option DCREAD

Use DCACHE for data read

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

Default: ON.

- ON** If data memory is displayed (memory class D:) the memory contents from the D-cache is displayed if the D-cache is valid. If D-cache is not valid the physical memory will be read. Typical command to display data memory are: **Data.dump**, **Var.Watch**, **Var.View**.
- OFF** If data memory is displayed (memory class D:) the memory contents from the physical memory is displayed.

SYStem.Option DUALPORT

Run-time memory access for all windows

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

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

SYStem.Option FAILSAVE

Special error handling for debug port

Format: **SYStem.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 FreezePin

Use alternative signal on the BDM connector

Available on: MPC8xx

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

As default, this option is off and the debugger set all necessary setting for the SIMCR register for the most frequently used **option A**. (VFLS0/1 pins are connected to BDM connector pin 1 and 6). The SYStem.Option.FreezePin can prevent the debugger for resetting/overwriting the SIMCR register to the default settings.

If **option B** is used (FREEZE pin is connected to the BDM connector) this SYStem.Option.FreezePin must be switched on.

NOTE: For the MPC5xx family all necessary configuration for the correct BDM pin setting have to be done in the RSTCONF word.

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**. Than 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.

Format: **SYStem.Option IBUS** [*<value>*]

With this option, you can set the instruction fetch show cycle and serialize control bits of the IBUS support control register.

SERALL	All fetch cycles are visible on the external bus. In this mode the processor is fetch serialized. Therefore the processor performance is much lower then working in regular mode.
SERCHG	All cycles that follow a change in the program flow are visible on the external bus. In this mode the processor is fetch serialized. Therefore the processor performance is much lower then working in regular mode.
SERIND	All cycles that follow an indirect change in the program flow are visible on the external bus. In this mode the processor is fetch serialized. Therefore the processor performance is much lower then working in regular mode.
SERNONE	In this mode the processor is fetch serialized. Therefore the processor performance is much lower then working in regular mode. No information about the program flow is visible on the external bus.
CHG	All cycles that follow a change in the program flow are visible on the external bus. The performance degradation is small here.
IND	All cycles that follow an indirect change in the program flow are visible on the external bus. The performance degradation is small here. This setting is recommended if a preprocessor for MPC500/800 is used.
NONE	No show cycles are performed. (Recommended when only a BDM debugger is used.)
RESERVED	Should not be used.

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.

MPC5xx: Flushes the Instruction Prefetch Queue before starting the program execution by Step or Go

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

Default: OFF.

ON If program memory is displayed (memory class P:) the memory contents from the I-cache is shown if the I-cache is valid. If I-cache is not valid the physical memory will be read. Typical command for program memory display are: Data.List, Data.dump.

OFF If program memory is displayed (memory class P:) the memory contents from the physical memory is displayed.

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.

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 MMUSPACES

Enable space IDs

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

Default: OFF.

Enables the usage of the MMU to support **multiple** address spaces. The command should not be used if only one translation table is used. Enabling the option will extend the address scheme of the debugger by a 16-bit memory space identifier (space ID).

This option is needed for operating systems that run several applications at the same virtual address space (e.g. Linux). The debugger uses this 16-bit memory space identifier to assign debug symbols to the memory space of the according process.

If a debug session requires space IDs, then you must enable the option before loading the debug symbols.

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

- ON** No external data bus is connected to the trace connector.
- OFF (default)** The external data bus is connected to the trace connector.

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.

- ON** With this setting the TRAP exception is no longer used for software breakpoints. UNDEF 0 is used instead. Use the command **TrOnchip.Set PRIE OFF**. With this setting the debug mode is no longer entered when a TRAP occurs. See also the Debug Enable Register in your processor manual. Now your application can handle the TRAP instruction.
- OFF** The TRAP exception is used for software breakpoints.

Format: **SYSystem.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:

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

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

Normally, the PowerPC debugger displays data big endian style.

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

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. If AUTO is used, two memory words of the on-chip RAM are used for reading the FPU registers.

SYStem.Option SIUMCR

SIUMCR setting for the trace

Format: **SYStem.Option SIUMCR** [<value>]

In order to trace the program and data flow, it is necessary for the TRACE32 software to know the settings of some peripheral pins. The value of this option must be the same value as the SIUMCR register of the chip.

SYStem.Option SLOWLOAD

Alternative data load algorithm

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

The debug interface of the MPC8xx and MPC5xx has a special mode for fast download of 32 bit data. For some older versions of the chips, it might be necessary to switch to a slower download mode to get proper results.

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 $\overline{\text{HRESET}}$ for 2 ... 3 s before an error message is displayed.

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

This method uses software breakpoints to perform an assembler single step instead of the processor's built-in single step feature. Works only for software in RAM. Do not turn ON unless advised by Lauterbach.

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 VFSL [ON | OFF]**

Uses VFSL pins for run/stop detection. Improves run-time measurement precision. See [RunTime](#) window.

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.

SYStem.state

Display SYStem window

Format: **SYStem.state**

Displays the **SYStem.state** window.

CPU specific MMU commands

This command is not necessary for the NEXUS debugger. It is only available to keep PRACTICE scripts compatible for both the BDM and the NEXUS debugger. 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.

MMU.DUMP

Page wise display of MMU translation table

Only available for MPC800 family.

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. <ul style="list-style-type: none">• 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 <code><task_magic> </code> <code><task_id> </code> <code><task_name> </code> <code><space_id>:0x0</code>	<p>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.</p> <ul style="list-style-type: none"> 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.
--	---

MMU.List

Compact display of MMU translation table

Format:	MMU.List <code><table> [<range> <address> <range> <root> <address> <root>]</code> MMU.<table>.List (deprecated)
<code><table></code> :	PageTable KernelPageTable TaskPageTable <code><task_magic> <task_id> <task_name> <space_id>:0x0</code>

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.

<code><root></code>	<p>The <code><root></code> 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.</p>
<code><range></code> <code><address></code>	<p>Limit the address range displayed to either an address range or to addresses larger or equal to <code><address></code>.</p> <p>For most table types, the arguments <code><range></code> or <code><address></code> can also be used to select the translation table of a specific process if a space ID is given.</p>
PageTable	<p>Lists the entries of an MMU translation table.</p> <ul style="list-style-type: none"> if <code><range></code> or <code><address></code> 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. <ul style="list-style-type: none"> 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.

MMU.SCAN

Load MMU table from CPU

Only available for MPC800 family.

Format:	MMU.SCAN <table> [<range> <address>] MMU.<table>.SCAN (deprecated)
<table>:	PageTable KernelPageTable TaskPageTable <task_magic> <task_id> <task_name> <space_id>:0x0 ALL <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, then 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. <ul style="list-style-type: none"> 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. <ul style="list-style-type: none"> 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 .

MMU.TLB

Display MMU TLB entries

Format:	MMU.TLB <tlb>
<tlb>:	IMMU DMMU

Displays a table of all MMU TLB entries of the specified TLB table.

MMU.TLBSCAN

Load MMU TLB entries

Format:	MMU.TLBSCAN MMU.TLBSCAN <tlb>
<tlb>:	IMMU DMMU

Loads the TLB table entries from the CPU to the debugger-internal MMU table. If no TLB table is specified, both are scanned.

Formats:

- MMU.Set TLB1** *<index>* *<mas1>* *<mas2>* *<mas3>*
- MMU.Set TLB2** *<index>* *<mas0>* *<mas1>* *<mas2>*
- MMU.<table>.SET** (deprecated)

Sets the specified MMU TLB table entry in the CPU. The parameter *<tlb>* is not available for CPUs with only one TLB table.

<i><index></i>	TLB entry index. From 0 to (number of TLB entries)-1 of the specified TLB table
<i><mas0></i>	Values corresponding to the values that would be written to the MAS registers in order to set a TLB (or MPU) entry. See the processor's reference manual for details on MAS registers. For processors with a core MPU (MPC57XX/SPC57X series), use TLB2 to generate an MPU entry).
<i><mas1></i>	
<i><mas2></i>	
<i><mas3></i>	

TrOnchip.CONVert

Adjust range breakpoint in on-chip resource

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

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

- | | |
|---------------------|--|
| 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. |

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.

TrOnchip.ENable

Enable NEXUS trace register control

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](#).

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.

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

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.

TrOnchip.IWx.Ibus

Instructions address for I-Bus watchpoint

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

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

Defines the instruction for the I-Bus watchpoint.

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.

TrOnchip.LW0.CYcle

Cycle type for L-Bus watchpoint

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 L-Bus watchpoint.

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.

TrOnchip.Set

Stop program execution at specified exception

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

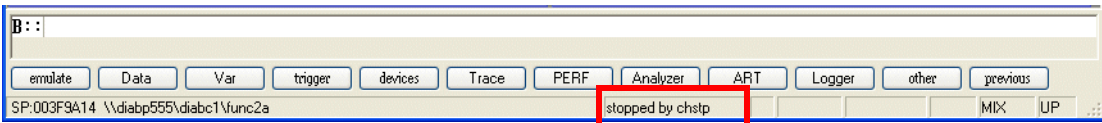
<item>: **CHSTPE ... SEIE**

The program execution is stopped at the specified exception.

- ALIE** Alignment Interrupt Enable
- CHSTPE** Checkstop Enable
- DECIE** Decrementer Interrupt Enable
- DTLBERE** DTLB Error Interrupt Enable

DTLBMSE	DTLB Miss Interrupt Enable
EXTIE	External Interrupt Enable
FPUVIE	Floating-point Unavailable Interrupt Enable
ITLBERE	ITLB Error Interrupt Enable
ITLBMSE	ITLB Miss Interrupt Enable
MCIE	Machine Check Interrupt
PRIE	Program Interrupt Enable
RSTE	Reset Interrupt Enable
SEIE	Software Emulation Interrupt Enable
SYSIE	System Interrupt Enable

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.TCompress

Trace data compression

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

Not implemented yet.

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.

TrOnchip.TTrigger

Set a trigger for the trace

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]

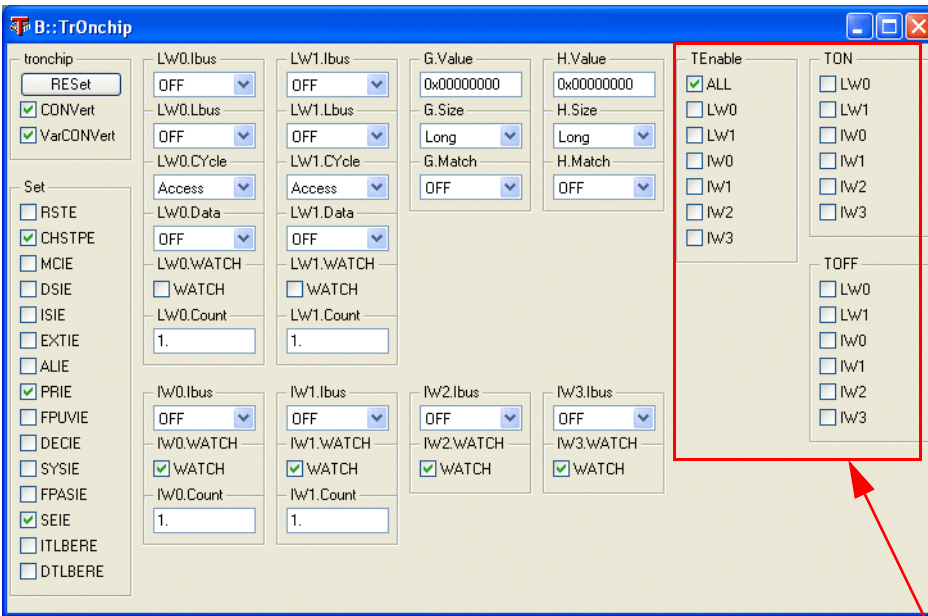
Command is of no relevance for the MPC5xx/8xx family.

TrOnchip.state

Display on-chip trigger window

Format: **TrOnchip.state**

Opens the **TrOnchip.state** window.



Only available if Preprocessor for MPC500/800 is used

10 pin BDM Connector MPC500/MPC800

Signal	Pin	Pin	Signal
VFLS0\FREEZE	1	2	SRESET\RESETIN-
GND	3	4	DSCK
GND	5	6	VFLS1\FREEZE
RESETOUT\HRESET-	7	8	DSDI
VDD	9	10	DSDO

The two signal names on pin 1, 2 and 6 have the same physical meaning. Only the use of the names differs between MPC500 and MPC800.