LAUTERBACH
DEVELOPMENT TOOLS

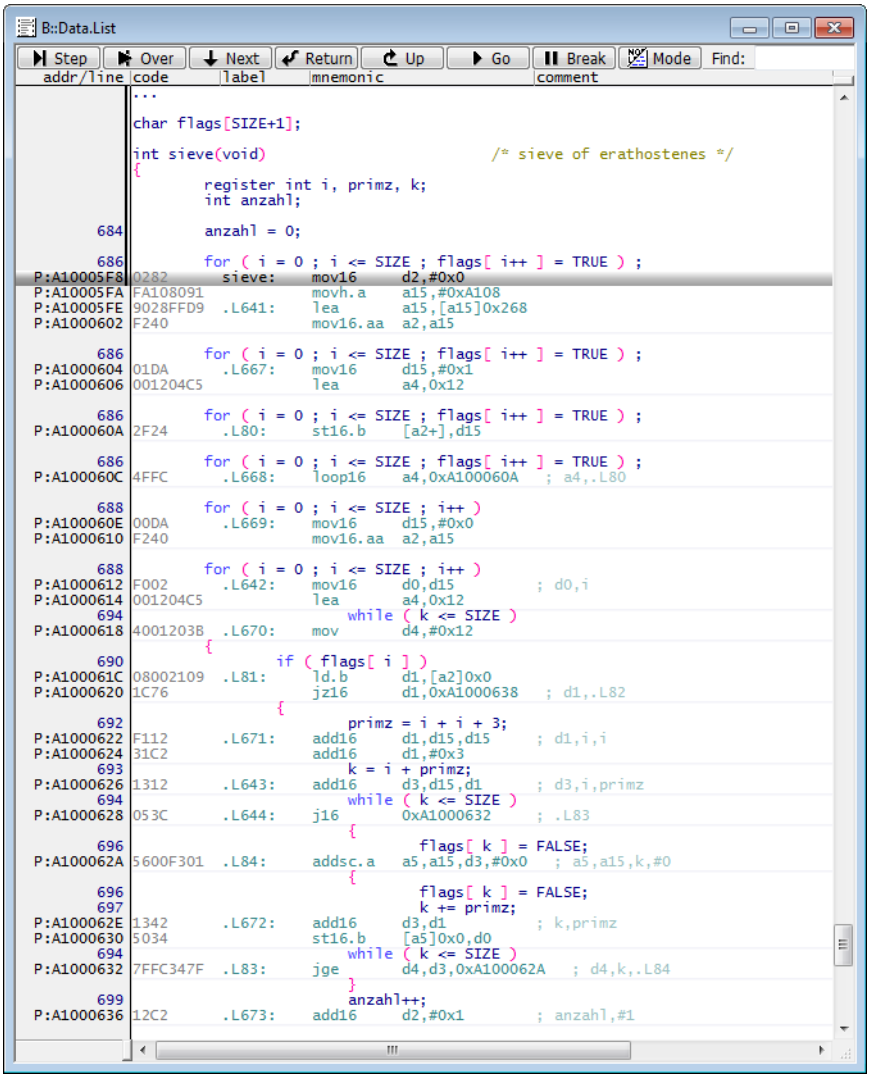
Simulator for TriCore

Release 09.2024

MANUAL

Simulator for TriCore

Version 05-Oct-2024



Introduction

This document describes the processor-specific settings and features for the TRACE32 Instruction Set Simulator for TriCore.

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

Simulator Features

The TRACE32 Simulator for TriCore covers the following:

- TriCore instruction set, starting from core version 1.2 up to the newest version.
- MultiCore simulation starting from core version 1.6.1 / TC2xx.
- Trap simulation.
- Interrupt simulation starting from core version 1.6.1 / TC2xx.
- Simple memory map simulation:
  - segment 0 is simulated as bus error,
  - starting from core version 1.6.1 / TC2xx: local / global adressing of scratchpad RAMs.
- MPU simulation starting from core version 1.6.1 / TC2xx.

Peripherals are not included but can be simulated by loading appropriate models.

TRACE32 Simulator License

[build 68859 - DVD 02/2016]

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

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

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.

Demo and Start-up Scripts

To search for PRACTICE scripts, do one of the following in TRACE32 PowerView:

- Type at the command line: **WELCOME.SCRIPTS**
- or choose **File** menu > **Search for Script**.

  You can now search the demo folder and its subdirectories for PRACTICE start-up scripts (*.cmm) and other demo software.

You can also manually navigate in the ~~/`demo/tricore/` subfolder of the system directory of TRACE32.

Quick Start of the Simulator

To start the simulator, proceed as follows:

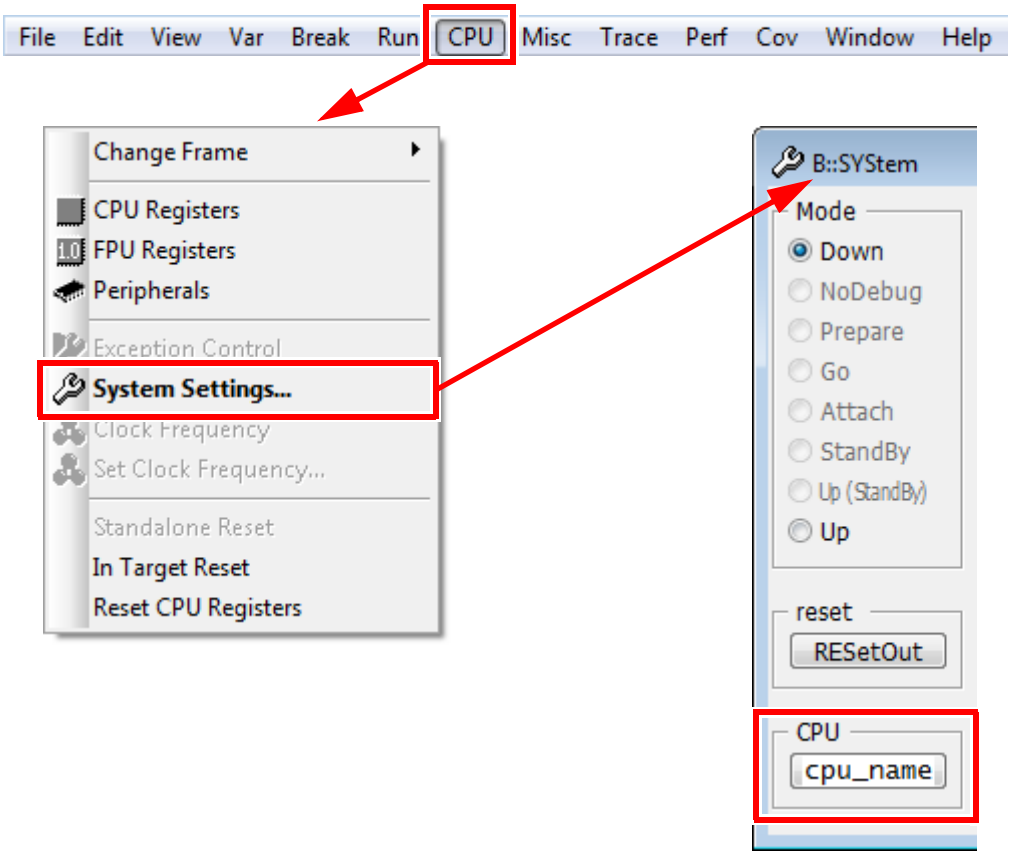
1. Select the device prompt for the Simulator and reset the system.

```
B::

RESet
```

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

2. Specify the CPU specific settings.




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

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

3. Enter debug mode.

```
SYStem.Up
```

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

4. Load the program.

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

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

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

5. Start-up example

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

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

WinCLEAR                            ; Clear all windows

SYStem.CPU <cpu_name>               ; Select CPU type

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

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

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

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

List.Mix                            ; Open source code window    *)

Register.view /SpotLight            ; Open register window       *)

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

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

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

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

In a TriCore system, multiple peripherals can generate interrupt requests to interrupt service providers e.g. CPUs.

The TriCore simulator supports interrupts starting from core version 1.6.1 / TC2xx. The Interrupt Control Unit (ICU) is to be implemented by a simulation model. The interrupt handling flow is as follows:

- The ICU (simulation model) arbitrates among the pending interrupts and writes the Pending Interrupt Priority Number (PIPN) of the winning service request to the Interrupt Control Register (ICR) of the CPU (the service provider). The simulation model signals the presence of a pending interrupt request to the simulator by setting the interrupt port of the corresponding CPU. The interrupt port number (p) is calculated from the service provider core number (n) as follows

  p= (-2) - (8 * n)

- If the simulated CPU decides to accept the requested interrupt, it updates the Current CPU Priority Number (CCPN) and clears the PIPN from the ICR register.
- The simulation model intercepts clearing the PIPN as an Interrupt acknowledge and clears down the requesting interrupt source. It must also determine the next pending interrupt or clear the interrupt port when all pending interrupts are served.

Changes to ICR from the simulator and the simulation model are visible via memory access callbacks to the corresponding CSFR memory address.

A demo simulation model for AURIX STM timers is available in the subdirectory
~~/demo/tricore/simulator/stm_aurix

Troubleshooting

No information available.

Memory Classes

The following memory classes are available:

Memory Class	Description
P	Program
D	Data
EEC	Emulation Memory on EEC Only available on TriCore Emulation Devices for accessing the Emulation Extension Chip

P: and D: display the same memory, the difference is in handling the symbols.

Prepending an E as attribute to the memory class will make memory accesses possible even when the CPU is running. See **SYStem.MemAccess** and **SYStem.CpuAccess** for more information.

In the Simulator, all memories are dual-port capable by default.

Breakpoints

There are two types of breakpoints available: Software breakpoints and On-chip breakpoints.

The simulator does not differ between software- and on-chip breakpoints.

Examples for Breakpoints

- Examples for instruction breakpoints:

```
Break.Set 0xD4001FD0 /Program          ; breakpoint on instruction
```

- Examples for breakpoints on data:

```
Break.Set 0xAFE10200 /Write            ; data write access breakpoint
```

Breakpoint on write access to `0xAFE10200`.

```
Break.Set 0xAFE10400 /Read             ; data read access breakpoint
```

Breakpoint on read access to 0xAFE10400.

Trace

The Simulator offers a complete instruction and data trace. Use **Trace.List** to display.

FAQ

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

SYStem.CONFIG Configure debugger according to target topology

<parameter>:	**DRPRE** *<bits>*
(JTAG):	**DRPOST** *<bits>*
	IRPRE *<bits>*
	IRPOST *<bits>*
	TAPState *<state>*
	TCKLevel *<level>*
	TriState [**ON** | **OFF**]
	Slave [**ON** | **OFF**]

The **SYStem.CONFIG** commands have no effect in Simulator. These commands describe the physical configuration at the JTAG port and the trace port of a multi-core hardware target. Since the simulator normally just simulates the instruction set, these commands will be ignored. Refer to the relevant **Processor Architecture Manual** in case you want to know the effect of these commands on a debugger.

SYStem.CPU Select CPU

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

Default: TC1797.

Selects the processor type.

<cpu> For a list of supported CPUs, use the command `SYStem.CPU *` or refer to the chip search on the Lauterbach website.

NOTE: In case your device is listed on the website but not listed in the `SYStem.CPU *` list, you may require a software update. Please contact your responsible Lauterbach representative.

Command has no effect on the TRACE32 Instruction Set Simulator.

Format: **SYStem.MemAccess** *<mode>*
SYStem.ACCESS (deprecated)

<mode>: **Enable** | **Denied** | **StopAndGo**

Default: Enable.

This option declares if and how a non-intrusive memory access can take place while the simulated CPU is executing code. Run-time memory access creates an additional load on the simulation. The MemAccess mode is printed in the state line.

The run-time memory access has to be activated for each window by using the memory class **E:** (e.g. Data.dump ED:0xA1000000) or by using the format option **%E** (e.g. **Var.View %E var1**). It is also possible to enable non-intrusive memory access for all memory areas displayed by setting **SYStem.Option.DUALPORT ON**.

Enable
CPU (deprecated)
: The dlebugger performs non-intrusive memory accesses.

Denied
: Non-intrusive memory access is disabled while the simulated CPU is executing code. Instead intrusive accesses can be configured with **SYStem.CpuAccess**.

StopAndGo
: Temporarily halts the simulated CPU to perform a memory access.

SYStem.ACCESS is an alias for this command.

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

Down The CPU is held in reset, debug mode is not active. Default state and state after fatal errors.

Up The CPU is not in reset but halted. Debug mode is active. In this mode the CPU can be started and stopped. This is the most typical way to activate debugging.

Initial Mode: Down.

The **SYStem.Option** command group provides architecture and CPU specific commands.

SYStem.Option.DCFREEZE Do not invalidate cache

Command has no effect on the TRACE32 Instruction Set Simulator.

SYStem.Option.DUALPORT Implicitly use run-time memory access

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

All TRACE32 windows that display memory are updated while the processor is executing code (e.g. **Data.dump**, **List.auto**, **PER.view**, **Var.View**). This setting has no effect if **SYStem.MemAccess** is disabled.

If only selected memory windows should update their content during runtime, leave **SYStem.Option.DUALPORT OFF** and use the access class prefix **E** or the format option **%E** for the specific windows.

SYStem.Option.OVERLAY Enable overlay support

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

Default: OFF.

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

OFF Disables support for code overlays.

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

Example:

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

SYStem.Option.ETK Debugging together with ETK from ETAS

Command has no effect on the TRACE32 Instruction Set Simulator.

SYStem.Option.HeartBeat Bug fix to avoid FPI bus conflict

Command has no effect on the TRACE32 Instruction Set Simulator.

SYStem.Option.ICFLUSH Flush instruction cache at “Go” or “Step”

Command has no effect on the TRACE32 Instruction Set Simulator.

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.PERSTOP Enable global peripheral suspend

Command has no effect on the TRACE32 Instruction Set Simulator.

SYStem.Option.SOFTLONG Set 32 bit software breakpoints

Command has no effect on the TRACE32 Instruction Set Simulator.

SYStem.RESetOut CPU reset command

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

Format:	**SYStem.state**

Opens the **SYStem.state** window with settings of CPU specific **SYStem** commands. Settings can also be changed here.

TrOnchip — Onchip triggers

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