TRACE32 Online Help
TRACE32 Directory
TRACE32 Index

TRACE32 Documents

System Trace

System Trace User's Guide

Introduction

Installation

Software Installation
Hardware Installation
  CombiProbe-ARM (LA-4502)
  PowerTrace II (LA-769x) + AutoFocus II Preprocessor for ARM (LA-7992)

Utilization of the STM

Start-up Script
Example STP Data to Onchip Buffer
Example STP Data to Parallel Trace Interface
Example Attach to onchip trace
Example Save/Load STP data embedded in CoreSight trace stream

SYStem.CONFIG.STM - Configuration of TRACE32

SYStem.CONFIG.STM Inform TRACE32 about trace source implementation
SYStem.CONFIG.STM.Mode STPv2 timestamp format
SYStem.CONFIG.STM.Type Configure STM type in TRACE32

STM - Configuration of the Trace Source

STM System trace configuration
STM.FilterMasters Display specified masters only
STM.FilterChannels Display specified channels only
STM.Init Initialize trace hardware
STM.OFF Switch STM off
STM.ON Switch STM on
STM.PortEndianness Select port endianness
STM.PortMode Select STM port modes
STM.PortRoute Select output of STP data
STM.PortSize Port size in bits
STM_PrintfTraceFormat Defines format of printftrace style messages
STM.Register Display STM register
STM.RESet Reset STM settings

©1989-2019 Lauterbach GmbH
STM.SetMaster
Set master ID manually 22
STM.state
Display STM settings 23
STM.SyncTime
Trace decoder resync time 24
STM.SyncPeriod
Add synchronization packets 24
STM.TimeStamps
Enables timestamps 24
STM.TimeStampCLOCK
Configure debugger for STM timestamp clock 25

STM - Configuration of the TI Specific Trace Source .................................................................................. 26
STM.HWMasters
Enable hardware masters for tracing 26
STM.IdleCount
Maximum idle packets 26
STM.IgnoreHeader
Ignore leading dword in printftrace message 27
STM.SWMasters
Enable software masters for tracing 27
STM.OCPAutoIdle
Set OCP idle clock behavior 28
STM.PATTERN
Enable test pattern generator 28
STM.ChannelRepeat
Period of channel packet insertion 29
STM.MasterRepeat
Period of master packet insertion 29

STM - Configuration of the CoreSight Specific Trace Source .................................................................................. 30
STM.DMArequests
DMA requests enable 30
STM.COMPression
Data compression enable 30
STM.EventMASK
Mask hardware event inputs 31
STM.PortMASK
Mask stimulus ports 31
STM.TraceID
Sets trace ID 31
STM.TracePriority
Set priority for STM manually 32

STM<trace> - Trace Data Analysis .................................................................................................................. 33
STM<trace> Command groups for STM<trace> 33
Overview STM<trace> 33
Displaying the Trace Results - Onchip Buffer 34
Displaying the Trace Results - Trace Port 34
STMAnalyzer
Analyze STM data recorded by TRACE32 PowerTrace 35
STMCAnalyzer
Analyze STM data recorded by TRACE32 CombiProbe 35
STMHAnalyzer
Analyze STM data recorded by the host 36
STMLA
Analyze STM data from binary file 37
STMOnchip
Analyze STM data stored on target memory 37
STMTrace
Method-independent analysis of STM trace data 38

PrintfTrace Decoder for STP-based software messages 39

CMI - For the Configuration of the Trace Source STM .................................................................................. 42
CMI
Clock management interface 42
CMI.EnableMessage
Enables event or activity message generation 42
CMI.Mode
Set event or activity mode 43
CMI.OFF
Switch CMI off 44
CMI.ON
Switch CMI on 44
CMI.Register
Display the CMI register 44

©1989-2019 Lauterbach GmbH
CMI.RESet 
Resets CMI settings to their defaults 45
CMI.SamplingWindow.CLocK 
Set sampling window ratio 45
CMI.SamplingWindow.Size 
Set sampling window size 45
CMI.state 
Display CMI settings 46
CMI Example 
47
CMI Module Configuration 
47

**CMI Trace - Trace Data Analysis**

CMITrace 
Analyze CMI trace data 48

**OCP_WP - For the Configuration of the Trace Source STM**

OCP 
OpenCoreProtocol WatchPoint 49
OCP.AutoIDLE 
OCP-WP clocking strategy 49
OCP.DebugPort 
Select target to be traced 50
OCP.OFF 
Switch OCP off 50
OCP.ON 
Switch OCP on 50
OCP.Register 
Display OCP registers 51
OCP.RESet 
Reset OCP settings to their defaults 51
OCP.state 
Display OCP settings 52
OCP.TraceFilter 
Set filter criteria 53
OCP.TraceFilter<x>.NAME 
Name a filter 53
OCP.TraceFilter<x>.MCmd 
Filters traffic by transaction type 53
OCP.TraceFilter<x>.Initiator 
Filters traffic by transaction initiator 54
OCP.TraceFilter<x>.REQinfo 
Filters traffic by transaction qualifier 54
OCP.TraceEnable 
Filter OCP traffic by address range 55
OCP.TraceOFF 
Stop tracing 55
OCP.TraceON 
Start tracing 56
OCP.TriggerOut<x> 
Generate trigger event 56
Displaying Trace Results 
57
OCP Module Configuration 
57

**PMI - For Configuration of the Trace Source STM**

PMI 
Power management interface 58
PMI.EnableMessage 
Enables event message generation 58
PMI.OFF 
Switch PMI off 59
PMI.ON 
Switch PMI on 59
PMI.Register 
Display the PMI registers 59
PMI.RESet 
Resets PMI settings to their defaults 60
PMI.SamplingWindow.CLocK 
Set sampling window clock 61
PMI.SamplingWindow.Size 
Set sampling window size 61
PMI.state 
Display PMI settings 61
Displaying Trace Results 
61
PMI Example 
62
PMI Module Configuration 
62

**PMITrace - Trace Data Analysis**

63
PMITrace

StatCol - For the Configuration of the Trace Source STM ............................................. 64

StatCol
Statistics collector

StatCol.RESet
Resets all statistics collector settings to their default

StatCol.state
Display statistics collector settings

StatCol.<probe>.OFF
Switch probe off

StatCol.<probe>.ON
Switch probe on

StatCol.<probe>.REQuestEVenT
Select event detector

StatCol.<probe>.ReSPonseEVenT
Select event detector

StatCol.<probe>.CollectTime
Set up collection period

StatCol.<probe>.Counter <counter> ADDRMAX
Filter max address

StatCol.<probe>.Counter <counter> ADDRMIN
Filter min address

StatCol.<probe>.Counter <counter> ADDRREN
Enable address filtering

StatCol.<probe>.Counter <counter> EventInfo
Select ‘EventInfo’ to count

StatCol.<probe>.Counter <counter> MAX
Set max threshold for events

StatCol.<probe>.Counter <counter> MIN
Set min threshold for events

StatCol.<probe>.Counter <counter> SELect
Set counter input

StatCol.<probe>.Counter <counter> Filter <filter> MUX
Input port

StatCol.<probe>.Counter <counter> Filter <filter> OFF
Switch filter off

StatCol.<probe>.Counter <counter> Filter <filter> ON
Switch filter on

StatCol.<probe>.Counter <counter> Filter
Set filter criteria

StatCol Macro Functions

AvgPayloadLength

THroughput

LnkOcc

ArbConf

TransUflow

IBusy

HistPayloadLen

HistPresDist

HistLatDist

AvgLatDist

Displaying Trace Results

StatCol Example

StatCol Module Configuration

FAQ ............................................................................................................. 78
Introduction

Generally speaking a system trace is a hardware module on a SoC which enables the developer to output predefined hardware or software messages without affecting the run-time behavior of the system.

This manual covers the following system trace implementations:

1. The System Debug Trace Interface (SDTI) by Texas Instruments used in OMAP34xx devices
2. The System Trace Module (STM) by Texas Instruments used in OMAP44xx devices
3. The System Trace Macrocell (STM) by ARM as a CoreSight component

Due to the various implementations some commands and setup routines apply to a certain type of system trace only. While setup routines and implementation specific commands will be handled in separate sections (TI specific or CoreSight specific), some common commands differ in the number of available arguments or in the meaning of the arguments. These differences will be marked as follows:

- SDTI (TI) for Texas Instruments’ SDTI implementation.
- STM (TI) for Texas Instruments’ STM implementation.
- STM (CS) for the CoreSight implementation.

Arguments not available for a specific implementation will be marked as 'n.a.' - not applicable.

Another difference between those implementations is the trace protocol: SDTI (TI) outputs data in XTIv2 format, STM (TI) in STPv1, and STM (CS) in STPv2.

To simplify matters the term “STP = System Trace Protocol” will be used in the following.

STM (TI) and STM (CS) in turn offer the opportunity to route trace data to an Embedded Trace Buffer (ETB, also a CoreSight component), while SDTI (TI) does not. Reading from the ETB only requires an ARM debugger, no trace hardware like CombiProbe or PowerTrace. All sections/commands referring to that ETB will contain the word 'onchip' in any way.

The second way of exporting STP data is a dedicated trace port. For STM (CS) this trace port is called 'Trace Port Interface Unit, TPIU' (again a CoreSight Component), for STM (TI) and SDTI (TI) this trace port is called 'Parallel Trace Interface, PTI'. In the following, the general term “trace port” will be used for both interfaces.
Preconditions

This manual assumes that the **In-Circuit Debugger is already installed**. You should be familiar with the features of the debugger. If you are not yet familiar with the debugger, refer to the “**Training Simulator and Demo Software**” (demo.pdf) and “**TRACE32 Installation Guide**” (installation.pdf).

Purpose of this Manual

The purpose of this manual is to get your trace running, to write a PRACTICE script (*.cmm) that does the necessary start-up procedure and to make you familiar with the main features of the trace. All list of all commands that are specific for the TRACE32-ICD trace for the C166 family can be found at the end of this manual.

Command Syntax

The TRACE32 commands are not case sensitive. In this tutorial, we use upper case letters for the characters that are necessary for the short form of the command entry. E.g. **Analyzer.List** can be shortened to **A.L**.

Where can I get more information?

TRACE32 provides a detailed online help offering the most current description of all debug features.

1. In TRACE32 choose **Help menu > Contents**.
2. See also **Online Help** for a brief overview of the online help.
Installation

Software Installation

The TRACE32-ICD software for the ARM debugger also includes the STM trace support. No extra software installation for the STM trace is required.

Hardware Installation

CombiProbe-ARM (LA-4502)

1. Simply attach the CombiProbe to your debugger.
2. Plug the header into the target's trace connector (or target adaption, if required).
1. Attach the debug cable to the debugger.

2. Connect the 'PODBUS EXPRESS OUT' port of the debugger to the "PODBUS EXPRESS IN" port of PowerTrace II.

3. Plug the preprocessor's flat cables into the according connectors of PowerTrace II: The shortest cable to the connector labelled 'A', the middle to connector 'B' and the longest to connector 'C'.

4. Connect the debug cable header to the target's JTAG port (or target adaption, if required).

5. Connect the preprocessor's MICTOR connector (labelled 'TRACE A') to the target's trace port (or target adaption, if required).
Utilization of the STM

Start-up Script

Example STP Data to Onchip Buffer

Target: OMAP4430

NOTE: This example applies to the STM by Texas Instruments only.

; Clock definition
Trace.CLOCK 100MHz ; Optional: If not defined, only raw timestamps will be displayed.

STM.RESet

; Route STP data to ETB
STM.PortRoute ONCHIP ; If ETM was on, it will be disabled here.

; !!!!!!!! Important !!!!!!!!
Onchip.AutoArm OFF ; If AutoArm remain on, several final trace bytes will be missing.

; Turn on STM module
STM.ON

; Manually enable the ETB
Onchip.Arm

... 

; Don’t forget to disable the ETB afterwards!
Onchip.OFF
Example STP Data to Parallel Trace Interface

Target: OMAP4430

NOTE: This example applies to the STM by Texas Instruments only.

; Define STP data format
SYStem.CONFIG.STM STP ; This will unlock the STM commands.

STM.RESet

; Route STP data to PTI
STM.PortRoute CAnalyzer

; Pad configuration ; Multiplex emu[0:4] signals to
; dpmemu[0:4] pads or multiplex
; emu[15:19] signals to dpmemu[15:19]
; pads. You can even do both.

; dpmemu[0:4]
Data.Set ahb:0x4a1001ac %long  Data.Long(ahb:0x4a1001ac)&0xffff
Data.Set ahb:0x4a1001b0 %long  0
Data.Set ahb:0x4a1001b4 %long  0

; dpmemu[15:19]
Data.Set ahb:0x4a1001cc %long  0
Data.Set ahb:0x4a1001d0 %long  0
Data.Set ahb:0x4a1001d4 %long  0

; Configure & init CombiProbe
CAnalyzer.THreshold 0.9
CAnalyzer.Init

; Turn on STM module
STM.ON
Example Attach to onchip trace

NOTE: This example applies to CoreSight compliant STMs only.

; Setup target
SYStem.CPU <cpu>
SYStem.CONFIG <config>
... ; Do not attach to the target yet!
SystemTrace.Method Onchip
Onchip.TraceConnect <buffer>
SYStem.Attach
Onchip.Disable
Onchip.Attach
SystemTrace.List

Example Save/Load STP data embedded in CoreSight trace stream

NOTE: This following examples assume that the target has been set up properly beforehand.

Save file:

Trace.Export.TracePort <file> Save in binary file format

Load file:

TPIU.PortMode WRAPPED
CoreSightTrace.METHOD LA
SystemTrace.METHOD LA
LA.IMPORT.TracePort <file>
LA.IMPORT.StartValid
SystemTrace.List
SYStem.CONFIG.STM - Configuration of TRACE32

SYStem.CONFIG.STM  Inform TRACE32 about trace source implementation

Format:  SYStem.CONFIG.STM[1 | 2] <parameter>

<parameter>:  Base <address> … | Name <identifier>
  Type <type>
  Mode <mode>
RESET

Default: Device specific.

Provides essential information about the STM (manufacturer, protocol, etc.) to TRACE32. Usually this step is already included in the CPU selection.

<table>
<thead>
<tr>
<th>STM</th>
<th>Single STM. If the chip contains more than one STM, the individual STM can be addressed by adding a number to the keyword STM, i.e. STM1 or STM2.</th>
</tr>
</thead>
<tbody>
<tr>
<td>STM1</td>
<td>Same as STM command. Used to differentiate between STM1 and STM2.</td>
</tr>
<tr>
<td>STM2</td>
<td>Used to configure a 2nd STM, if present.</td>
</tr>
<tr>
<td>Type</td>
<td>For details, see SYStem.CONFIG.STM.Type.</td>
</tr>
<tr>
<td>Mode</td>
<td>For details, see SYStem.CONFIG.STM.Mode.</td>
</tr>
<tr>
<td>RESET</td>
<td>Reset SYStem.CONFIG STM settings.</td>
</tr>
<tr>
<td>Name</td>
<td>User-defined name for STMs to distinguish between identical and different STMs having the same address on different buses.</td>
</tr>
<tr>
<td>Base</td>
<td>Base address of an STM.</td>
</tr>
</tbody>
</table>
**SYStem.CONFIG.STM.Mode**

**STPv2 timestamp format**

Format:  

```
SYStem.CONFIG.STM.Mode <mode>
```

*<mode>:*  

- STP: STP protocol (MIPI STPv1, D32 packets)
- STP64: STP64 protocol (MIPI STPv1, D64 packets)
- STPv2 [2 | 3 | 4]: STPv2 protocol (MIPI STPv2).

Default: Device specific.

Informs TRACE32 that the chip contains a System Trace Module. The TRACE32 command group **STM** will be enabled as a result.

- **STP**: STP protocol (MIPI STPv1, D32 packets)
- **STP64**: STP64 protocol (MIPI STPv1, D64 packets)
- **STPv2 [2 | 3 | 4]**: STPv2 protocol (MIPI STPv2).

STP version 2 (STPv2) offers the possibility to output timestamps in different formats. Usually the device specific format will be set up by TRACE32 automatically during CPU selection.

The STPv2 mode allows you to set up the timestamp format manually afterwards, if necessary.

```
2 NATDELTA
Natural binary delta timestamp; timestamp counter is reset after each timestamp packet.
```
**Configure STM type in TRACE32**

Format:  \texttt{SYStem.CONFIG.STM.Type <type>}

\texttt{<type>:} None \mid Generic \mid ARM \mid SDTI \mid TI

Configures the STM type in TRACE32.

- **None**: No STM type is configured in TRACE32.
- **Generic** (default: Generic) STM module is generic.
- **ARM**: System Trace Macrocell (STM) by ARM as a CoreSight component
- **SDTI**: System Debug Trace Interface (SDTI) by Texas Instruments
- **TI**: System Trace Module (STM) by Texas Instruments
STM - Configuration of the Trace Source

STM

System trace configuration

The **STM** command group is used to configure the trace source STM. To perform the configuration, you can use the TRACE32 command line, a PRACTICE script (*.cmm), or the **STM.state** window.

The commands available in the window differ depending on the selected CPU. Commands that are not available for a certain CPU are hidden.

![STM.state window](image)

To analyze the recorded trace data, use the **STM<trace>** command groups or one of the generic replacement command groups, e.g. **STMTrace** and **SystemTrace**.

**See also**
- STM.ChannelRepeat
- STM.FilterChannels
- STM.IgnoreHeader
- STM.OFF
- STM.PORTMASK
- STM.PrintTraceFormat
- STM.state
- STM.TimeStampCLOCK
- SystemTrace
- STM.COMPres
- STM.FilterMasters
- STM.Init
- STM.ON
- STM.PortMode
- STM.Register
- STM.SWMasters
- STM.TimeStamps
- STM.DMArequests
- STM.HWMasters
- STM.MasterRepeat
- STM.PATTERN
- STM.PortRoute
- STM.RESet
- STM.SynchPeriod
- STM.TraceID
- STM.EventMASK
- STM.IdleCount
- STM.OCPAutoIdle
- STM.PortEndianness
- STM.PortSize
- STM.SetMaster
- STM.SynchTime
- STM.TracePriority

▲ 'Release Information' in 'Release History'
▲ 'STM - Configuration of the CoreSight Specific Trace Source' in 'System Trace User's Guide'
▲ 'STM - Configuration of the TI Specific Trace Source' in 'System Trace User's Guide'
▲ 'STM<trace> - Trace Data Analysis' in 'System Trace User's Guide'
STM.FilterMasters
Display specified masters only

Format: STM.FilterMasters <id_1> <id_2> <id_3> <id_4>

Select up to four STM master IDs, which associated trace packets will be displayed in the trace results. All other STM packets will be masked out.

This command actually does not filter STM packets but only affects the display. After the filter has been reset, all STM packets will be shown. The filter is reset via STM.FilterMasters (without any ID specified).

See also
STM
STM.state

STM.FilterChannels
Display specified channels only

Format: STM.FilterChannels <id_1> <id_2> <id_3> <id_4>

Selects up to four STM channels, which will be displayed in the trace results. All other channels will be masked out.

This command actually does not filter STM packets but only affects the display. After the filter has been reset, all STM packets will be shown. The filter is reset via STM.FilterChannels (without any ID specified).

See also
STM
STM.state

STM.Init
Initialize trace hardware

Format: STM.Init

The trace hardware is initialized and set to its defaults.

See also
STM
STM.state
STM.OFF
Switch STM off

Format: STM.OFF

Disables the STM functionality.

See also
- STM
- STM.state

STM.ON
Switch STM on

Format: STM.ON

Enables the STM functionality.

See also
- STM
- STM.state

STM.PortEndianness
Select port endianness

Format: STM.PortEndianness [Big | Little]

Default: Big

If STM.PortSize is > 8, this command determines the byte order of the traceport.

- **Big**: MSB mapped to lower portbits.
- **Little**: MSB mapped to upper portbits.

See also
- STM
- STM.state
Format: `STM.PortMode <mode>`

- **<mode>:**
  - Continuous
  - Gated
  - Autoldle
  - Bypass
  - Continuous
  - HalfRate
  - FullRate
  - `1/<divisor>`

<table>
<thead>
<tr>
<th>SDTI (TI)</th>
<th>STM (TI)</th>
<th>STM (CS)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Continuous</td>
<td>Port clock remains active even if no STP data are available (default).</td>
<td>n.a.</td>
</tr>
<tr>
<td>Gated</td>
<td>Port clock is stopped if no STP data are available.</td>
<td>n.a.</td>
</tr>
<tr>
<td>Autoldle</td>
<td>Disables the Parallel Trace Interface (PTI) if no STP data are available (power saving).</td>
<td>n.a.</td>
</tr>
<tr>
<td>HalfRate</td>
<td>STP data are sampled on rising edge of port clock.</td>
<td>STP data are sampled on rising edge of port clock.</td>
</tr>
<tr>
<td>FullRate</td>
<td>STP data are sampled on rising and falling edge of port clock.</td>
<td>STP data are sampled on rising and falling edge of port clock.</td>
</tr>
<tr>
<td><code>1/&lt;divisor&gt;</code></td>
<td>The port clock rate is defined as ratio of OCP clock.</td>
<td>The port clock rate is defined as ratio of OCP clock.</td>
</tr>
<tr>
<td>Bypass</td>
<td>n.a.</td>
<td>n.a.</td>
</tr>
<tr>
<td>Wrapped</td>
<td>n.a.</td>
<td>n.a.</td>
</tr>
</tbody>
</table>

For STM (CS): `ETM.PortMode`.

**See also**
- STM
- STM.state
### STM.PortRoute

Select output of STP data

| Format: STM.PortRoute [AUTO | Analyzer | CAnalyzer | Onchip] |
|-----------------------------------------------|
| SDTI (TI) | STM (TI) | STM (CS) |
| AUTO | Data are directed to the Parallel Trace Interface (PTI) and recorded by the attached trace hardware (default). | Data are directed to the Parallel Trace Interface (PTI) and recorded by the attached trace hardware (default). | n.a. |
| Analyzer | Data are directed to the Parallel Trace Interface (PTI) and recorded by the PowerTrace II. | Data are directed to the Parallel Trace Interface (PTI) and recorded by the PowerTrace II. | n.a. |
| CAnalyzer | Data are directed to the Parallel Trace Interface (PTI) and recorded by the CombiProbe. | Data are directed to the Parallel Trace Interface (PTI) and recorded by the CombiProbe. | n.a. |
| Onchip | n.a. | Data are directed to the Embedded Trace Buffer. | n.a. |

For STM (CS): **ETM.PortRoute**.

### See also
- STM
- STM.state

©1989-2019 Lauterbach GmbH
STM.PortSize

Format:

| Port Size | STM.PortSize [1 | 1E | 1X | 2 | 2E | 2X | 4 | 4E | 4X | 8 | 12 | 16] |
|-----------|------------------------------------------------|

Default: 4

For SDTI (TI), STM (TI): Defines the number of parallel data pins of the trace port. Also the internal signal multiplexing of the Debug Resource Manager (DRM) is affected by this command. Please refer to the table below:

<table>
<thead>
<tr>
<th>Port</th>
<th>STM.PortSize</th>
<th>STM.PortSize</th>
<th>STM.PortSize</th>
<th>STM.PortSize</th>
<th>STM.PortSize</th>
</tr>
</thead>
<tbody>
<tr>
<td>no suffix</td>
<td>suffix ‘X’</td>
<td>suffix ‘E’</td>
<td>suffix ‘Z’</td>
<td>suffix ‘K’</td>
<td></td>
</tr>
<tr>
<td>(standard</td>
<td>(to be used</td>
<td>(to be used</td>
<td>(to be used</td>
<td>(to be used</td>
<td></td>
</tr>
<tr>
<td>configuration)</td>
<td>with LA-</td>
<td>with LA-</td>
<td>with LA-</td>
<td>with LA-</td>
<td></td>
</tr>
<tr>
<td></td>
<td>xxxx)</td>
<td>3812)</td>
<td>xxxx)</td>
<td>xxxx)</td>
<td></td>
</tr>
<tr>
<td>stm_clk</td>
<td>emu19</td>
<td>emu2</td>
<td>emu2</td>
<td>emu0</td>
<td>emu10</td>
</tr>
<tr>
<td>stm_data[0]</td>
<td>emu18</td>
<td>emu3</td>
<td>emu0</td>
<td>emu1</td>
<td>emu0</td>
</tr>
<tr>
<td>stm_data[1]</td>
<td>emu17</td>
<td>emu4</td>
<td>emu1</td>
<td>emu2</td>
<td>emu1</td>
</tr>
<tr>
<td>stm_data[2]</td>
<td>emu16</td>
<td>emu5</td>
<td>emu3</td>
<td>emu3</td>
<td>emu2</td>
</tr>
<tr>
<td>stm_data[3]</td>
<td>emu15</td>
<td>emu6</td>
<td>emu4</td>
<td>emu4</td>
<td>emu3</td>
</tr>
</tbody>
</table>

The trace signals are routed to emu signal lines only, not to the physical pads of the device! Refer to the example script of this manual of how to configure the pads!

For STM (CS): ETM.PortSize.

See also

- STM
- STM.state
STMPRINTFTraceFormat

Defines format of printftrace style messages

| Format: STMPRINTFTraceFormat | [Normal | Kernel] |

Default: Normal

**Normal**  
String messages as described in section Software Messages

**Kernel**  
Special string and FTRACE message format as described in Software Messages

See also
- STM
- STM.state

STMRegister

Display STM register

| Format: STMRegister | [<file>] [/<option>] |

| <option>: | SpotLight | DualPort | Track | CORE | <core_number> |

Displays the STM registers.

For a description of the options, see PER.view.

See also
- STM
- STM.state

▲ ‘STM - Configuration of the Trace Source’ in ‘System Trace User’s Guide’
STM.RESet

Reset STM settings

Format: STM.RESet

All STM settings are reset to their defaults.

See also
- STM
- STM.state

STM.SetMaster

Set master ID manually

Format: STM.SetMaster <master_id>

High-level STP messages from hardware modules (see CMI, PMI) or Software Messages must be preceded by an STP master packet in order to be decoded correctly in the according trace list window. If no master packet could be found the message will be marked as “unknown”.

However, by setting <master_id> manually the trace decoder assigns any unknown STP packets to the specified master until a valid STP master packet is found in the trace stream.

See also
- STM
- STM.state
STM.state

Opens the **STM.state** window, where you can configure the trace source STM.

The commands available in the window differ depending on the selected CPU. Commands that are not available for a certain CPU are hidden.

For descriptions of the commands in the **STM.state** window, please refer to the **STM.*** commands in this chapter.

**Example:** For information about **ON**, see **STM.ON**.

**Exceptions:**
- The **SystemTrace** button opens the **SystemTrace** control window, see `<trace>.state`.
- The **List** button opens the **SystemTrace.List** window, see `<trace>.List`.
- The **Printf** button opens the **PrintfTrace.List** window, see `<trace>.List`.
- The **TPIU** button opens the **TPIU** window, see **TPIU.state**.

**See also**
- **STM**
- **STM.EventMASK**
- **STM.IdleCount**
- **STM.OCPAutoIdle**
- **STM.PortEndianness**
- **STM.PortSize**
- **STM.SetMaster**
- **STM.TimeStampCLOCK**
- **STM.ChannelRepeat**
- **STM.FilterChannels**
- **STM.IgnoreHeader**
- **STM.OFF**
- **STM.PortMASK**
- **STM.PrintTraceFormat**
- **STM.SWMasters**
- **STM.TimeStamps**
- **STM.COMPression**
- **STM.FilterMasters**
- **STM.Init**
- **STM.ON**
- **STM.PortMode**
- **STM.Register**
- **STM.SyncPeriod**
- **STM.TraceID**
- **STM.DMArequests**
- **STM.HWMasters**
- **STM.MasterRepeat**
- **STM.PATTERN**
- **STM.PortRoute**
- **STM.RESet**
- **STM.SyncTime**
- **STM.TracePriority**
STM.SyncTime

Format: \texttt{STM.SyncTime \(<time>\)}

Time after which a resync is forced in the trace decoder.

See also
- STM
- STM.state

STM.SyncPeriod

Format: \texttt{STM.SyncPeriod \([<value>]\)}

Default: 0

Inserts synchronization packets (ASYNC + VERSION) periodically into the trace stream approximately each \(<value>\) bytes. If \(<value>\) is zero, no synchronization packets will be generated. This command is only applicable to STPv2 compliant System Trace implementations!

See also
- STM
- STM.state

STM.TimeStamps

Format: \texttt{STM.TimeStamps \([ON | OFF]\)}

Default: OFF

Enables or disables timestamp generation in the trace hardware.

See also
- STM
- STM.state
### STM.TimeStampCLOCK

Configure debugger for STM timestamp clock

<table>
<thead>
<tr>
<th>Format:</th>
<th>STM.TimeStampCLOCK &lt;freq&gt;</th>
</tr>
</thead>
</table>

Default: 0

Configures the debugger for the STM timestamp clock frequency of the target. The frequency is required to calculate timing information based on timestamp packets.

#### See also
- STM
- STM.state
STM.HWMasters

Enable hardware masters for tracing

Format:  STM.HWMasters <name> [ON | OFF]

Available <names> are device specific. If the corresponding hardware master is disabled, write accesses of the master to the STM will be ignored.

Default values

SDTI: N. a.
STM: All off

See also

STM
STM.state

STM.IdleCount

Maximum idle packets

Format:  STM.IdleCount <count>

If there are no STP packets to be sent, <count> number of idle packets are emitted by the PTI. Depending on the port mode (STM.PortMode Continuous or STM.PortMode Gated), the PTI then stops or continues emitting idle packets. If the same HW master or the same SW master + channel resumes sending STP messages, a leading master packet is generated by the STM.

See also

STM
STM.state
STM.IgnoreHeader

Ignore leading dword in printftrace message

Format: **STM.IgnoreHeader [ON | OFF]**

Default: OFF

Newer versions of the TI CToolsLib generate a leading 32-bit word in front of the printftrace message. If not ignored, this header will produce some strange characters at the beginning of the message in the **PrintfTrace.List** window.

See also
- STM
- STM.state

STM.SWMasters

Enable software masters for tracing

Format: **STM.SWMasters <name> [ON | OFF]**

Available *<names>* are device specific. If the corresponding software master is disabled, writes of that master to a stimulus port will have no effect.

Default values
- SDTI: CU1 = ON, CPU2 = ON, Debugger = OFF
- STM: Device dependent

See also
- STM
- STM.state
STM.OCPAutoIdle

Set OCP idle clock behavior

Format: STM.OCPAutoIdle [ON | OFF]

OFF
OCP clock is free running (default).

ON
OCP clock may be gated if interface is in idle mode.

See also
- STM
- STM.state

STM.PATTERN

Enable test pattern generator

Format: STM.PATTERN <pattern>

If <pattern> in nonzero, the selected test pattern is output instead of STP messages.

See also
- STM
- STM.state
STM.ChannelRepeat

Period of channel packet insertion

| Format: | STM.ChannelRepeat [OFF | <value>] |
|---------|----------------------------------|
| <value>: | 8 | 16 | 24 | 32 | 40 | 48 | 56 | 64 | 72 | 80 | 104 | 112 | 120 |

OFF
No extra channel packets are inserted into the STP data stream (default).

<value>
If <value> subsequent STP messages are written to the same software channel, an extra STP channel packet is inserted into the data stream. Due to the working load of the STM module it may happen that extra channel packets are inserted only every 2 * <value> packets from the same channel.

This option is only available if STP data are routed to the onchip buffer: STM.PortRoute.Onchip.

See also
- STM
- STM.state

STM.MasterRepeat

Period of master packet insertion

| Format: | STM.MasterRepeat [OFF | <value>] |
|---------|----------------------------------|
| <value>: | 8 | 16 | 24 | 32 | 40 | 48 | 56 | 64 | 72 | 80 | 104 | 112 | 120 |

OFF
No extra master packets are inserted into the STP data stream (default).

<value>
If <value> subsequent STP packets are generated by the same master, an extra STP master packet is inserted into the data stream. Due to the working load of the STM module it may happen that extra master packets are inserted only every 2 * <value> packets from the same master.

This option is only available if STP data are routed to the onchip buffer: STM.PortRoute.Onchip.

See also
- STM
- STM.state
STM - Configuration of the CoreSight Specific Trace Source

STM.DMArequests

DMA requests enable

| Format: | STM.DMArequests [OFF | 25% | 50% | 75% | 100%] |

Default: OFF

The STM can request the DMA to write to the stimulus ports. Requests in turn are only issued if the internal STM FIFO contains less data than the stated filling level. This command does not set up the DMA.

See also
- STM
- STM.state

STM.COMPression

Data compression enable

| Format: | STM.COMPression [ON | OFF] |

Default: OFF

Enables or disables the automatic data compression of the STM. E.g. with compression enabled a 32-bit packet (D32) will be converted into an 8-bit packet (D8) if the value written to a stimulus port is less than 256.

See also
- STM
- STM.state
STM.EventMASK

Mask hardware event inputs

Format:  

STM.EventMASK <mask>

Default: 0xFFFFFFFF

This 32-bit mask enables or disables hardware event inputs for packet generation. Thereby the LSB of the mask corresponds to hardware event input #0, the MSB corresponds to hardware event input #31.

See also

- STM
- STM.state

STM.PortMASK

Mask stimulus ports

Format:  

STM.PortMASK <mask>

Default: 0xFFFFFFFF

This 32-bit mask enables or disables stimulus ports for instrumentation; that is if a bit of the mask is cleared, writes accesses to the corresponding stimulus port will not result in the generation of STP packets. Thereby the LSB of the mask corresponds to stimulus port #0, the MSB corresponds to stimulus port #31.

See also

- STM
- STM.state

STM.TraceID

Sets trace ID

Format:  

STM.TraceID <id>

Default: 0x11

Sets the trace ID of the STM.

See also

- STM
- STM.state
STM.TracePriority

Set priority for STM manually

Format: STM.TracePriority <priority>

TRACE32 automatically assigns an appropriate priority to the STM. This command allows the user to change the priority for the STM trace information.

See also
- STM
- STM.state
STM<trace> - Trace Data Analysis

STM<trace> Command groups for STM<trace>

See also

- STMAnalyzer
- STMCAnalyzer
- STMHAnalyzer
- STMLA
- STMOnchip
- STMTrace

Overview STM<trace>

Using the STM<trace> command groups, you can configure the trace recording as well as analyze and display the recorded STM trace data. The command groups consist of the name of the trace source, here STM, plus the TRACE32 trace method you have chosen for recording the STM trace data.

For more information about the TRACE32 convention of combining <trace_source> and <trace_method> to a <trace> command group that is aimed at a specific trace source, see “Replacing <trace> with Trace Source and Trace Method - Examples” (general_ref_t.pdf).

Not any arbitrary combination of <trace_source> and <trace_method> is possible. For an overview of the available command groups “List of <trace> Command Groups consisting of <trace_source><trace_method>” (general_ref_t.pdf).

Example:

```plaintext
STMTrace.state ;optional step: open the window in which the trace recording is configured.
STMTrace.METHOD Analyzer ;select the trace method Analyzer for recording trace data.
STM.state ;optional step: open the window in which the trace source STM is configured.
STM.ON ;switch trace source STM on
STMTrace.List ;display a trace listing of the STM trace data recorded with the trace method Analyzer.
STMAnalyzer.List ;this is the equivalent and explicit command.
```
Displaying the Trace Results - Onchip Buffer

Trace results are displayed via STMOnchip.List command. The following channels can be added to the default list window:

- **STMTITS**: STM (TI) only: Displays raw timestamp information of DxxTS messages.
- **STMMASTER**: Displays the master ID of each message.
- **STMCHANNEL**: Displays the channel ID of each message.

STMMASTER and STMCHANNEL information can only be displayed if a master or channel message has been stored in the ETB prior to the current message. Otherwise the corresponding column will remain empty.

The recommended way to display STP onchip data is:

```
STMOnchip.List STMMASTER STMCHANNEL CYcle Data [STMTITS] TIme.Back List.NoDummy
```

Displaying the Trace Results - Trace Port

Trace results are displayed via STMCAnalyzer.List or STMAnalyzer.List command. Channels ‘STMMASTER’ and ‘STMCHANNEL’ may be added as in Onchip Buffer, whereas signal ‘STMTITS’ has no effect.

The recommended way to display STP data emitted via a trace port is:

```
STMCAnalyzer.List STMMASTER STMCHANNEL CYcle Data TIme.Back List.NoDummy
```
STMAnalyzer
Analyze STM data recorded by TRACE32 PowerTrace

The **STMAnalyzer** command group allows to display and analyze the information emitted by the system trace implementations listed in the “Introduction”, page 5.

The STM information is emitted off-chip via:

- The Trace Port Interface Unit (TPIU), which is configured with the **TPIU** command group.
- Or the Parallel Trace Interface (PTI), which is configured with the **STM** command group.

The emitted STM information is recorded by the TRACE32 PowerTrace.

<table>
<thead>
<tr>
<th>Format:</th>
<th>STMAnalyzer.&lt;sub_cmd&gt;</th>
</tr>
</thead>
</table>

For descriptions of the subcommands, please refer to the general `<trace>` command descriptions in “General Commands Reference Guide T” (general_ref_t.pdf).

**Example:** For a description of STMAnalyzer.List, refer to `<trace>`.List

See also
- STM<trace>

STMCAAnalyzer
Analyze STM data recorded by TRACE32 CombiProbe

The **STMCAAnalyzer** command group allows to display and analyze the information emitted by the system trace implementations listed in the “Introduction”, page 5.

The STM information is emitted off-chip via:

- The Trace Port Interface Unit (TPIU), which is configured with the **TPIU** command group.
- Or the Parallel Trace Interface (PTI), which is configured with the **STM** command group.
- Or via Serial Wire Output (SWO), which is also configured with the **TPIU** command group.
The emitted STM information is recorded by the TRACE32 CombiProbe.

<table>
<thead>
<tr>
<th>&lt;sub_cmd&gt;</th>
<th>For descriptions of the subcommands, please refer to the general &lt;trace&gt; command descriptions in “General Commands Reference Guide T” (general_ref_t.pdf).</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Example: For a description of STMAnalyzer.List, refer to &lt;trace&gt;.List</td>
</tr>
</tbody>
</table>

See also
- STM<trace>

STMAnalyzer

Analyze STM data recorded by the host

Format: STMAnalyzer.<sub_cmd>

The STMAnalyzer command group allows to display and analyze the information emitted by the system trace implementations listed in the “Introduction”, page 5.

Trace data is transferred off-chip via the USB port and recorded in the trace memory of the TRACE32 host analyzer.

<table>
<thead>
<tr>
<th>&lt;sub_cmd&gt;</th>
<th>For descriptions of the subcommands, please refer to the general &lt;trace&gt; command descriptions in “General Commands Reference Guide T” (general_ref_t.pdf).</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Example: For a description of STMAnalyzer.List, refer to &lt;trace&gt;.List</td>
</tr>
</tbody>
</table>

See also
- STM<trace>
### STMLA

Analyze STM data from binary file

**Format:**

```
STMLA.<sub_cmd>
```

<table>
<thead>
<tr>
<th>&lt;sub_cmd&gt;</th>
<th>For descriptions of the subcommands, please refer to the general <code>&lt;trace&gt;</code> command descriptions in “General Commands Reference Guide T” (general_ref_t.pdf).</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Example:</strong></td>
<td>For a description of STMLA.List, refer to <code>&lt;trace&gt;.List</code></td>
</tr>
</tbody>
</table>

---

### STMOnchip

Analyze STM data stored on target memory

**Format:**

```
STMOnchip.<sub_cmd>
STMOnchip2.<sub_cmd>
```

The STMOnchip command group allows to display and analyze the information emitted by the system trace implementations listed in the “Introduction”, page 5.

The STM trace is sent to the device-specific onchip trace memory.

<table>
<thead>
<tr>
<th>&lt;sub_cmd&gt;</th>
<th>For descriptions of the subcommands, please refer to the general <code>&lt;trace&gt;</code> command descriptions in “General Commands Reference Guide T” (general_ref_t.pdf).</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Example:</strong></td>
<td>For a description of STMOnchip.List, refer to <code>&lt;trace&gt;.List</code></td>
</tr>
</tbody>
</table>

---

See also

- STM<trace>
STMTrace

Method-independent analysis of STM trace data

[Example]

<table>
<thead>
<tr>
<th>Format:</th>
<th>STMTrace.&lt;sub_cmd&gt;</th>
</tr>
</thead>
</table>

The **STMTrace** command group can be used as a generic replacement for the above **STM<trace>** command groups.

<table>
<thead>
<tr>
<th>&lt;sub_cmd&gt;</th>
<th>For descriptions of the subcommands, please refer to the general &lt;trace&gt; command descriptions in “General Commands Reference Guide T” (general_ref_t.pdf).</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Example</strong>: For a description of <strong>STMTrace.List</strong>, refer to <strong>&lt;trace&gt;.List</strong></td>
<td></td>
</tr>
</tbody>
</table>

See also

- **STM<trace>**
Applications running on a CPU may use the System Trace to output 'printf'-style software messages. The trace output can be displayed or analyzed with the **PrintfTrace** command group. Three different message types are available:

- **String messages**
- **Kernel log messages**
- **Kernel FTRACE messages**

<table>
<thead>
<tr>
<th>&lt;sub_cmd&gt;</th>
<th>For descriptions of the subcommands, please refer to the general &lt;trace&gt; command descriptions in “General Commands Reference Guide T” (general_ref_t.pdf).</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Example:</strong> For a description of <strong>PrintfTrace.List</strong>, refer to &lt;trace&gt;.List</td>
<td></td>
</tr>
</tbody>
</table>

### String messages

String messages in general start with a data packet and are terminated by a time-stamped data packet or FLAG packet. Depending on the STP version being used, the PrintfTrace decoder decodes a STP software message as follows:

<table>
<thead>
<tr>
<th></th>
<th>STPv1</th>
<th>STPv2</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Start of message</strong></td>
<td>D8, D16, D32, D64</td>
<td>D4, D8, D16, D32, D64</td>
</tr>
<tr>
<td><strong>Message body</strong></td>
<td>D8, D16, D32, D64</td>
<td>D4, D8, D16, D32, D64</td>
</tr>
<tr>
<td><strong>End of message</strong></td>
<td>D8TS, D16TS, D32TS, D64TS</td>
<td>D4TS, D8TS, D16TS, D32TS, D64TS, FLAG</td>
</tr>
</tbody>
</table>
Kernel log messages

Similar format as string messages, except that messages are initiated by a timestamped packet and terminated by a FLAG packet:

<table>
<thead>
<tr>
<th></th>
<th>STPv1</th>
<th>STPv2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Start of message</td>
<td></td>
<td>D4TS, D8TS, D16TS, D32TS, D64TS</td>
</tr>
<tr>
<td>Message body</td>
<td></td>
<td>D4, D8, D16, D32, D64</td>
</tr>
<tr>
<td>End of message</td>
<td></td>
<td>FLAG</td>
</tr>
</tbody>
</table>

In order to differentiate between regular string and kernel messages, **STM.PrintfTraceFormat Kernel** must be used.

Kernel FTRACE messages

These messages resemble a simple flow trace based on function calls with a source and target address. They always start with a D32TS packet whose lower 16bit data must be 0x0001. The message body consists of 3 D32 packets, followed by a FLAG packet:

<table>
<thead>
<tr>
<th></th>
<th>STPv1</th>
<th>STPv2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Start of message</td>
<td>n.a.</td>
<td>D32TS (0x????0001)</td>
</tr>
<tr>
<td>Process ID</td>
<td>n.a.</td>
<td>D32</td>
</tr>
<tr>
<td>Target address</td>
<td>n.a.</td>
<td>D32</td>
</tr>
<tr>
<td>Source address</td>
<td>n.a.</td>
<td>D32</td>
</tr>
<tr>
<td>End of message</td>
<td>n.a.</td>
<td>FLAG</td>
</tr>
</tbody>
</table>

In order to differentiate between regular string and kernel FTRACE messages, **STM.PrintfTraceFormat Kernel** must be used.
The following signals are relevant for all three types of software messages:

<table>
<thead>
<tr>
<th>SIGNAL</th>
<th>DESCRIPTION</th>
</tr>
</thead>
</table>
| MESSAGE> | - Decoded normal string  
- FTRACE target function  
- Decoded kernel log       |
| sYmbol   | Fully translated FTRACE source function                                   |
| sYmbolN  | FTRACE source function                                                    |
| PID      | FTRACE process ID                                                         |
| STMMASTER| Master ID of software message.                                            |
| STMCHANNEL| Channel ID of software message.                                           |

**Example**

```
; Example of full trace listing for kernel messages

PrintfTrace.List STMMASTER STMCHANNEL pid sYmbolN MESSAGE
```
CMI - For the Configuration of the Trace Source STM

CMI

Clock management interface


See also

- CMI.EnableMessage
- CMI.Mode
- CMI.OFF
- CMI.ON
- CMI.Register
- CMI.RESet
- CMI.state

▲ ‘CMITrace - Trace Data Analysis’ in ‘System Trace User's Guide’

CMI.EnableMessage

Enables event or activity message generation

Format:

CMI.EnableMessage.<message> [ON | OFF]

<message>:

- ClockDomain (event message)
- ClockFrequency8 (event message)
- ClockFrequency4 (event message)
- ClockSource (event message)
- DPLLmask (event message)
- TargetActivity8 (activity message)
- TargetActivity4 (activity message)
- InitiatorActivity8 (activity message)
- InitiatorActivity4 (activity message)

Default: OFF

Event messages are emitted for all clock domains derived from the same Digital Phase-Locked Loop (DPLL). They are only emitted on state changes and if CMI.Mode EVENT has been selected.

Activity messages output the active cycles count of the target or initiator modules. They are emitted on a periodically basis, even if the debugger in a halted state. Activity monitoring must be enabled via CMI.Mode ACTIVITY in addition.

- **ClockDomain**
  Trace clock domain state changes (on / off).
- **ClockFrequency8**
  Trace clock frequency changes (8-bit divider ratio).
- **ClockFrequency4**
  Trace clock frequency changes (4-bit divider ratio).
**ClockSource**
Trace clock source selection changes (MUX input).

**DPLLmask**
Trace DPLL setting changes. Each of the 16 lower bits of DPLLmask represents one DPLL.

**TargetActivity8**
Count target activity cycles.  
(If CMI.SamplingWindow.Size \(\geq\) 16)

**TargetActivity4**
Count target activity cycles.  
(If CMI.SamplingWindow.Size \(<\) 16)

**InitiatorActivity8**
Count initiator activity cycles.  
(If CMI.SamplingWindow.Size \(\geq\) 16)

**InitiatorActivity4**
Count initiator activity cycles.  
(If CMI.SamplingWindow.Size \(<\) 16)

---

**See also**
- CMI
- CMI.state

---

### CMI.Mode

*Set event or activity mode*

**Format:**

```
CMI.Mode [EVENT | ACTivity]
```

**EVENT**
Selects event mode monitoring (default).

**ACTivity**
Selects activity mode monitoring.

---

**See also**
- CMI
- CMI.state

---
CMI.OFF

Switch CMI off

Format: CMI.OFF

Switches the CMI module off.

See also
- CMI
- CMI.state

CMI.ON

Switch CMI on

Format: CMI.ON

Switches the CMI module on.

See also
- CMI
- CMI.state

CMI.Register

Display the CMI register

Format: CMI.Register [/<option>]

<option>: SpotLight | DualPort | Track | CORE <core_number>

Displays the CMI registers.

<option> For a description of the options, see PER.view.

See also
- CMI
- CMI.state
CMI.RESet

Resets CMI settings to their defaults

Format: CMI.RESet

All CMI settings are reset to their defaults.

See also
- CMI
- CMI.state

CMI.SamplingWindow.CLocK

Set sampling window ratio

Format: CMI.SamplingWindow.CLocK <ratio>

Default: 1/1

<ratio> Divider ratio of the sampling window clock. It is derived from the CMI module's clock. Valid ratios range from 1/1 to 1/16.

CMI.SamplingWindow.Size

Set sampling window size

Format: CMI.SamplingWindow.Size <cycles>

Default: 1

<cycles> Size of the sampling window. Smaller windows allow for more accurate activity or event reports while bigger sampling windows reduce trace traffic. Valid sizes range from 1 to 256.
CMI.state

Format:  CMI.state

Shows CMI setup window.

See also

- CMI
- CMI.ON
- CMI.EnableMessage
- CMI.Register
- CMI.Mode
- CMI.RESet
- CMI.OFF
CMI Example

This example for an OMAP4430 which has got two CMI modules. In this case the modules are addressed as CMI1 or CMI2, respectively.

CMI Module Configuration

If CMI is not enabled for your specific device, the following commands allow for an afterward configuration. Both, the base address and the ID must be set in order to enable the CMI:

- **SYStem.CONFIG.TICMI1BASE**: Set the base address of the primary CMI module.
- **SYStem.CONFIG.TICMI2BASE**: Set the base address of a secondary CMI module.
- **SYStem.CONFIG.TICMI1ID**: Set the STM master ID of the primary CMI module.
- **SYStem.CONFIG.TICMI2ID**: Set the STM master ID of a secondary CMI module.

```plaintext
CAnalyzer.AutoArm OFF ; In activity mode CMI messages are output permanently, so we
CAnalyzer.ARM ; must arm the CAnalyzer before the CMI is activated.

CMI1.RESet
CMI1.SamplingWindow.Size 15
CMI1.Mode ACTivity
CMI1.ON
...
CMI1.OFF ; Turn off CMI ...
CAnalyzer.OFF ; ... before shutting down the CAnalyzer.

CMITrace.List CYcle cmita.<name1> cmita.<name2> List.NoDummy ; Display cycle activity of target <name1> and <name2>.
```
CMITrace - Trace Data Analysis

See also
- CMI

CMITrace
Analyze CMI trace data

Format:  

For descriptions of the subcommands, please refer to the general <trace> command descriptions in “General Commands Reference Guide T” (general_ref_t.pdf).

Example: For a description of CMITrace.List, refer to <trace>.List

In addition to the standard <trace> command the following signals related to the CMI are available:

<table>
<thead>
<tr>
<th>Signal</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CMICD.&lt;domain&gt;</td>
<td>Clock state of domain &lt;domain&gt;.</td>
</tr>
<tr>
<td>CMIDR.&lt;clock&gt;</td>
<td>Divider ratio of clock &lt;clock&gt;.</td>
</tr>
<tr>
<td>CMICS.&lt;clock&gt;</td>
<td>Source of clock &lt;clock&gt;.</td>
</tr>
<tr>
<td>CMIDPPLL.&lt;setting&gt;</td>
<td>DPLL setting &lt;setting&gt;</td>
</tr>
<tr>
<td>CMITA.&lt;target&gt;</td>
<td>Target &lt;target&gt; activity</td>
</tr>
<tr>
<td>CMIIA.&lt;initiator&gt;</td>
<td>Initiator &lt;initiator&gt; activity.</td>
</tr>
<tr>
<td>CMISTAT</td>
<td>Only applies to event messages: Error flag indicating event message loss(es) caused by an undersized sampling window.</td>
</tr>
<tr>
<td>CMILAT</td>
<td>Event messages: Export latency in multiples of the sampling window.</td>
</tr>
<tr>
<td></td>
<td>Activity messages: Export latency in multiples of target or initiator cycles.</td>
</tr>
</tbody>
</table>

Example:

; Display cycle activity of target <name1> and <name2>.  
CMITrace.List CYcle cmita.<name1> cmita.<name2> List.NoDummy
The OCP-WP monitors OCP requests directed to a selected target attached to the L3 interconnect of the OMAP4. Tracing the bus traffic is non-intrusive and enables the developer to capture all requests addressed to a target or only a subset of it defined by up to four different filters (see `OCP.TraceFilter<x>` commands).

See also
- `OCP.AutoIDLE`
- `OCP.Register`
- `OCP.TraceFilter`
- `OCP.DebugPort`
- `OCP.PRESET`
- `OCP.TraceOFF`
- `OCP.OFF`
- `OCP.state`
- `OCP.TraceON`
- `OCP.ON`
- `OCP.TraceEnable`
- `OCP.TriggerOut<x>`

### OCP.AutoIDLE

**OCP-WP clocking strategy**

Format: `OCP.AutoIDLE [ON | OFF]`

Default: OFF

If **ON**, the OCP-WP is gated whenever no activity can be observed on the OCP interface. If **OFF**, the OCP-WP is clocked permanently.

See also
- `OCP`
- `OCP.state`
OCP.DebugPort

Select target to be traced

Format:  \texttt{OCP.DebugPort [debug\_port] \mid number}  

Default: OFF

\begin{itemize}
\item \textit{debug\_port}\hspace{0.5cm} Trace transactions to the predefined L3 target \texttt{debug\_port}.
\item \textit{number}\hspace{0.5cm} Trace transactions to the user-defined L3 target with ID \texttt{number}.
\end{itemize}

\textbf{See also}
- \texttt{OCP}
- \texttt{OCP.state}

\section*{OCP.OFF}

Switch OCP off

Format:  \texttt{OCP.OFF}  

Switches the OCP module off.

\textbf{See also}
- \texttt{OCP}
- \texttt{OCP.state}

\section*{OCP.ON}

Switch OCP on

Format:  \texttt{OCP.ON}  

Switches the OCP module on.

\textbf{See also}
- \texttt{OCP}
- \texttt{OCP.state}
OCP.Register

Display OCP registers

Format: `OCP.Register [<-file>] [<-option>]`

`<option>`: `SpotLight` | `DualPort` | `Track` | `CORE <core_number>`

Displays the OCP registers.

For a description of the options, see `PER.view`.

See also
- OCP
- OCP.state

OCP.RESet

Reset OCP settings to their defaults

Format: `OCP.RESet`

All OCP settings are reset to their defaults.

See also
- OCP
- OCP.state
Shows OCP setup window.

A For descriptions of the commands in the OCP.state window, please refer to the OCP.* commands in this chapter. Example: For information about OFF [A], see OCP.OFF.

See also
- OCP
- OCP.ON
- OCP.TraceFilter
- OCP.AutoIDLE
- OCP.Register
- OCP.TraceOFF
- OCP.DebugPort
- OCP.RESet
- OCP.TraceON
- OCP.OFF
- OCP.TraceEnable
- OCP_TRIGGEROUT<xx>
**OCP.TraceFilter<x>.NAME**

Name a filter

Format: 

```
OCP.TraceFilter<x>.NAME <name>
```

Filters can be named in order to identify the filter a traced transaction has passed. The name of the filter can be displayed in the trace list window via `TraceOCP.List FilterName`.

**Example:**

```
OCP.TraceFilter0.NAME "Filter0"
```

**See also**

- `OCP.TraceFilter`

---

**OCP.TraceFilter<x>.MCmd**

Filters traffic by transaction type

Format: 

```
OCP.TraceFilter<x>.MCmd <command>
```

Default: ALL

Only transactions of type `<command>` will pass filter `<x>`.

**See also**

- `OCP.TraceFilter`
**OCP.TraceFilter<\textit{x}>.Initiator**

Filters traffic by transaction initiator

| Format: | OCP.TraceFilter\textit{x}.Initiator [ALL | <\textit{initiator}>] |
|---------|---------------------------------------------------|

**ALL**
Transactions from all initiators is traced.

**<\textit{initiator}>**
Only transactions from <\textit{initiator}> will pass filter <\textit{x}>.

Default: ALL

**See also**
- OCP.TraceFilter

---

**OCP.TraceFilter<\textit{x}>.REQinfo**

Filters traffic by transaction qualifier

| Format: | OCP.TraceFilter\textit{x}.REQinfo \textit{qualifier}.[<\textit{value}> | <\textit{mask}>] |
|---------|---------------------------------------------------|

Default: 0yXXX (Trace all)

**<\textit{value}>**
Only trace transactions if the <\textit{qualifier}> equals <\textit{value}>.

**<\textit{mask}>**
Alternative way to define the REQinfo filter criteria as bitmask; <\textit{mask}> must be of format ‘0ybbb’, whereas b = [0, Cleared 1, Set \textit{X}, Don’t Care

**Example:**

```plaintext
OCP.TraceFilter0.REQinfo MReqDomain.0y11X ;Trace transactions which have the two upper bits set ignore the state of the lowest bit.
```

**See also**
- OCP.TraceFilter

---

©1989-2019 Lauterbach GmbH
**OCP.TraceEnable**  
Filter OCP traffic by address range

### Format:

```
OCP.TraceEnable <range>
```

Default: 0x00000000-0xffffffff

OCP traffic is only captured if the address is within the specified `<range>`. The range must be specified as the offset from the base address of the selected debug port (OCP.DebugPort), not to the global address!  
**OCP.TraceEnable** and **OCP.TraceON** / **OCP.TraceOFF** cannot be applied at the same time!

**Example:**

```
; Debug port base address = 0xa0001000 
; Range to be monitored = 0xa0001000 to 0xa0001020
OCP.TraceEnable 0x00000000--0x00000020
```

**See also**

■ OCP  ■ OCP.state

---

**OCP.TraceOFF**  
Stop tracing

### Format:

```
OCP.TraceOFF [EMU1 | <address>]
```

Stops tracing if the trigger condition or address match occurs. Tracing will continue on an **OCP.TraceON** condition.

**OCP.TraceEnable** and **OCP.TraceOFF** cannot be applied at the same time!  
**OCP.TriggerOut<x>** and **OCP.TraceOFF EMU1** cannot be used at the same time!

Default: **OCP.TraceEnable**

- **EMU1**  
  Stops tracing upon a HIGH-TO-LOW transition of the EMU1 trigger input.

- **<address>**  
  Stops tracing upon an address match.

**See also**

■ OCP  ■ OCP.state

---

©1989-2019 Lauterbach GmbH
OCP.TraceON

Format: **OCP.TraceON** [EMU0 | <address>]

Starts tracing if the trigger condition or address match occurs. Tracing continues even if the trigger condition or address match no longer holds.

**OCP.TraceEnable** and **OCP.TraceON** cannot be applied at the same time!
**OCP.TriggerOut<x>** and **OCP.TraceON EMU0** cannot be used at the same time!

Default: **OCP.TraceEnable**

**EMU0**  Starts tracing upon a HIGH-TO-LOW transition of the EMU0 trigger input.

**<address>**  Starts tracing upon an address match.

---

**OCP.TriggerOut<x>**

Generate trigger event

Format: **OCP.TriggerOut** [<address | <range>]

**<address>**  Asserts trigger EMU<x> if the monitored address matches <address>.

**<range>**  Asserts trigger EMU<x> if the monitored address is within <range>.

**OCP.TriggerOut<x>** and **OCP.TraceON EMU0 / OCP.TraceOFF EMU1** cannot be used at the same time!

---

See also

- OCP
- OCP.state
Displaying Trace Results

The trace output can be displayed or analyzed with the **OCPTrace** command. Please refer to the general <trace> command description in “**General Commands Reference Guide T**” (general_ref_t.pdf).

In addition to the standard <trace> command, the following signals related to the OCP are available:

**FilterName**  
Name of the filter the OCP message has passed.

OCP Module Configuration

If OCP is not enabled for your specific device, the following commands allow for an afterward configuration. Both, the base address and the ID must be set in order to enable the OCP:

**SYStem.CONFIG.TIOCPBASE**  
Set the base address of the OCP module.

**SYStem.CONFIG.TIOCPID**  
Set the STM master ID of the OCP module.
The Power Management module monitors power domain state changes of other modules on the OMAP4. For more detailed information refer to the OMAP4 ETRM available from https://www-a.ti.com/extranet/programs/emulation/OMAP4_ETRM_2.0-Setup.exe.

See also
- PMI.EnableMessage
- PMI.OFF
- PMI.ON
- PMI.Register
- PMI.state

See also 'PMITrace - Trace Data Analysis' in 'System Trace User's Guide'

PMI.EnableMessage

Enables event message generation

Format: PMI.EnableMessage.<event_msg> [ON | OFF]

<event_msg>: LogicVoltage
              MemoryVoltage
              LogicPower
              MemoryPower

Default: OFF

Event messages are emitted in case a memory or logic block changes its voltage or power state.

LogicVoltage Voltage levels of logic blocks.
MemoryVoltage Voltage levels of memory blocks.
LogicPower Power FSM states of logic blocks.
MemoryPower Power state of memory blocks.

See also
- PMI
- PMI.state
PMI.OFF

Switch PMI off

Format: PMI.OFF

Switches the PMI module off.

See also
- PMI
- PMI.state

PMI.ON

Switch PMI on

Format: PMI.ON

Switches the PMI module on.

See also
- PMI
- PMI.state

PMI.Register

Display the PMI registers

Format: PMI.Register [/<option>]

<option>: SpotLight | DualPort | Track | CORE <core_number>

Displays the PMI registers.

For a description of the options, see PER.view.

See also
- PMI
- PMI.state

©1989-2019 Lauterbach GmbH
PMI.RESet

Resets PMI settings to their defaults

Format:

PMI.Reset

All PMI settings are reset to their defaults.

See also

- PMI
- PMI.state
PMI.SamplingWindow.CLock

Set sampling window clock

Format:

PMI.SamplingWindow.CLock <ratio>

Format:

PMI.SamplingWindow.Size <cycles>

PMI.state

Display PMI settings

Format:

PMI.state

Shows PMI setup window.

See also

- PMI
- PMI.EnableMessage
- PMI.OFF
- PMI.ON
- PMI.Register
- PMI.RESet

Displaying Trace Results

The trace output can be displayed or analyzed with the PMITrace command. Please refer to the general <trace> command description in "General Commands Reference Guide T" (general_ref_t.pdf).
In addition to the standard `<trace>` command, the following signals related to the PMI are available:

- **PMILV.<domain>** Voltage level of logic voltage domain `<domain>`.  
- **PMILVOFF** OFF mode voltage domain.  
- **PMIMV.<domain>** FSM state of memory voltage domain `<domain>`.  
- **PMILP.<domain>** Power state of logic power domain `<domain>`.  
- **PMIMP.<domain>** Power state of memory power domain `<domain>`.  
- **PMISTAT** Error flag indicating event message loss(es) caused by an undersized sampling window.  
- **PMILAT** Event messages: Export latency in multiples of the sampling window.

### PMI Example

```plaintext
PMI.RESet
PMI.SamplingWindow.Size 15
PMI.EnableMessage.LogicVoltage ON
PMI.ON
...
PMI.OFF

PMITrace.List CYcle pmilv.<domain> ; Display logic voltage domain
List.NoDummy ; <domain> voltage level.
```

### PMI Module Configuration

If PMI is not enabled for your specific device, the following commands allow for an afterward configuration. Both, the base address and the ID must be set in order to enable the PMI:

- **SYStem.CONFIG.TIPMIBASE** Set the base address of the PMI module.  
- **SYStem.CONFIG.TIPMIID** Set the STM master ID of the PMI module.
See also
PMI

PMITrace
Analyze PMI trace data

Format: `PMITrace.<sub_cmd>`

Using the **CMITrace** command group, you can analyze and display the recorded CMI trace data. The command group consists of the name of the trace source, here **CMI**, plus the keyword **Trace** of the `<trace>` command group.

<table>
<thead>
<tr>
<th><code>&lt;sub_cmd&gt;</code></th>
<th>For descriptions of the subcommands, please refer to the general <code>&lt;trace&gt;</code> command descriptions in “General Commands Reference Guide T” (general_ref_t.pdf).</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Example</strong>: For a description of <code>PMITrace.List</code>, refer to <code>&lt;trace&gt;.List</code></td>
<td></td>
</tr>
</tbody>
</table>

©1989-2019 Lauterbach GmbH
The NoC statistics collector provides information about the workload of an onchip bus system like throughput, latency, etc. For each bus system there is a separate implementation of the statistics collector (called 'probe'), hence all the commands listed in the following will affect the selected probe only, except for the StatCol.RESet command.

Configuring the statistics collector requires an in-depth knowledge of its structure and modes of operations. For those who do not have that knowledge or don’t need to make use of the full extent of the statistics collector’s features there are macro functions available. These set up most of the required configurations and are explained in chapter StatCol macro functions.

**See also**
- StatCol.<probe>.CollectTime
- StatCol.<probe>.ON
- StatCol.<probe>.RequestEvent
- StatCol.<probe>.ResponseEvent
- StatCol.state
- StatCol.<probe>.OFF
- StatCol.<probe>.RequestEvent
- StatCol.RESet

### StatCol.RESet
Resets all statistics collector settings to their default

**Format:**
```
StatCol.RESet
```

All statistics collector settings are reset to their defaults.

**See also**
- StatCol

### StatCol.state
Display statistics collector settings

**Format:**
```
StatCol.state
```

Shows statistics collector setup window.

**See also**
- StatCol
StatCol.<probe>.OFF

Switch probe off

Format: StatCol.<probe>.OFF

Switches the probe off.

See also
■ StatCol

StatCol.<probe>.ON

Switch probe on

Format: StatCol.<probe>.ON

Switches the probe on.

See also
■ StatCol

StatCol.<probe>.REQuestEVenT

Select event detector

Format: StatCol.<probe>.REQuestEVenT.<event>

<event>:
NONE
ANY
TRANSFER
WAIT
BUSY
PAKET
DATA
IDLES
LATENCY

Default: NONE
Selects the event detector for the probe’s request link.

- **NONE**: Do not detect any events.
- **ANY**: Detect all events.
- **TRANSFER**: Detect NTTP cell or OCP data/command transfers.
- **WAIT**: Detect WAIT cycles (NTTP only).
- **BUSY**: Detect BUSY cycles.
- **PAKET**: Detect packet headers or OCP commands.
- **DATA**: Detect payload transfers.
- **IDLES**: Detect idle cycles.
- **LATENCY**: Apply latency measurement.

**See also**
- StatCol

---

**StatCol.<probe>.ReSPonseEVenT**

Select event detector

Format: `StatCol.<probe>.REQuestEVeNt.<event>`

Default: NONE

Selects the event detector for the probe’s response link. See `StatCol.<probe>.REQuestEVeNt`.

**See also**
- StatCol

---

**StatCol.<probe>.CollectTime**

Set up collection period

Format: `StatCol.<probe>.CollectTime <cycles>`

Default: 255
Sets up the time interval in cycles after which the internal counters are reset and the result is sent to the STM.

See also

StatCol

---

**StatCol.<probe>.Counter <counter> ADDRMAX**  
Filter max address

Format:  
StatCol.<probe>.Counter <counter> ADDRessMAX.<value>

Default: 0

Sets the upper bound for address filtering. See StatCol.<probe>.Counter <counter> ADDRessEnable. This command is available for certain CPUs only.

**StatCol.<probe>.Counter <counter> ADDRMIN**  
Filter min address

Format:  
StatCol.<probe>.Counter <counter> ADDRessMIN.<value>

Default: 0

Sets the lower bound for address filtering. See StatCol.<probe>.Counter <counter> ADDRessEnable. This command is available for certain CPUs only.

**StatCol.<probe>.Counter <counter> ADDREN**  
Enable address filtering

Format:  
StatCol.<probe>.Counter <counter> ADDRessEnable [ON | OFF]

Default: Off

Only generates statistic data if address on bus is smaller than ADDRessMAX and greater than ADDRessMIN. This command is available for certain CPUs only.
Select ‘EventInfo’ to count

Format: \texttt{StatCol.<probe>.Counter <counter> EventInfo.<eventinfo>}

\texttt{<eventinfo>}: LENght
PRESsure
LATency

Default: LENght

Detects additional event information:

- LENght: Payload length.
- PRESsure: Link pressure.
- LATency: Transfer latency.

\textbf{StatCol.<probe>.Counter <counter> MAX}

Set max threshold for events

Format: \texttt{StatCol.<probe>.Counter <counter> MAX.<value>}

Default: 0

Increments \texttt{<counter>} if \texttt{StatCol.<probe>.Counter <counter> SESelect.MINMAX} is selected and the defined \texttt{EventInfo} is within Max.\texttt{.<value>} and Min.\texttt{.<value>}.

\textbf{StatCol.<probe>.Counter <counter> MIN}

Set min threshold for events

Format: \texttt{StatCol.<probe>.Counter <counter> MIN.<value>}

Default: 0

Increments \texttt{<counter>} if \texttt{StatCol.<probe>.Counter <counter> SESelect.MINMAX} is selected and the defined \texttt{EventInfo} is within Min.\texttt{.<value>} and Max.\texttt{.<value>}.
StatCol.<probe>.Counter <counter> SELECT

Format: \texttt{StatCol.<probe>.Counter <counter> SELECT.<input>}

\begin{itemize}
  \item \texttt{<input>}:
    \begin{itemize}
      \item HIT
      \item MINMAX
      \item ADD
      \item AND
      \item OR
      \item REQ
      \item RSP
      \item ALL
      \item EXT
    \end{itemize}
\end{itemize}

Default: HIT

Defines what kind of statistics the counter will count:

- **HIT**: Increment the counter by one each time an event has passed the counter's filter. (See \texttt{StatCol.<probe>.Counter <counter> Filter <filter>} commands).
- **MINMAX**: Increment the counter by one each time the selected EventInfo is within the range \texttt{Min.<value>} and \texttt{Max.<value>}.
- **ADD**: Add the selected EventInfo value to the counter if an event has passed the counter's filter. (See \texttt{StatCol.<probe>.Counter <counter> Filter <filter>} commands)
- **AND**: Increment the counter by one if an event has passed all filters of \texttt{<probe>}.
- **OR**: Increment the counter by one if an event has passed at least one of all filters of \texttt{<probe>}.
- **REQ**: Increment the counter by one each time a request message is detected on any port of \texttt{<probe>}.
- **RSP**: Increment the counter by one each time a response message is detected on any port of \texttt{<probe>}.
- **ALL**: Increment the counter by one each time a response or request message is detected on any port of \texttt{<probe>}.
- **EXT**: Increment the counter by one each time the external event input is sampled high.
StatCol.<probe>.Counter <counter> Filter <filter> MUX <input>

Format: StatCol.<probe>.Counter <counter> Filter <filter> MUX <input>

Selects one of the probe’s inputs as the input for the specified filter. Available inputs are depended on the probe.

StatCol.<probe>.Counter <counter> Filter <filter> OFF

Switch filter off

Format: StatCol.<probe>.Counter <counter> Filter <filter> OFF

Switches the filter off.

StatCol.<probe>.Counter <counter> Filter <filter> ON

Switch filter on

Format: StatCol.<probe>.Counter <counter> Filter <filter> ON

Switches the filter on.
StatCol.<probe>.Counter <counter> Filter

Set filter criteria

Format: StatCol.<probe>.Counter <counter> Filter <filter> <item>.[<value> | <mask>]

.GetItem: MaSTerADDRess
Read
Write
ERRor
REQuestUserInfo
ReSPonseUserInfo
SLaVeADDRess

Filters out packets which do not comply with the defined item bitmask or value.

MaSTerADDRess Master address (NTTP) or MConnId (OCP).
Read Read bit
Write Write bit
ERRor Error bit (NTTP only)
REQuestUserInfo RequestUserInfo bits (NTTP only)
ReSPonseUserInfo ResponseUserInfo bits (NTTP only)
SLaVeADDRess Slave address (NTTP only)

<value> (Hexa)decimal, octal or binary value that defines the required packet item.

<mask> Bitmask of format '0y.....': x = don’t care
1 = set
0 = cleared

(default: Don’t care)

Example:

MaSTerADDRess.0yxxxx11 ;Only packets with the lower two bits set of
;the master address will pass the filter
;element.
Macro functions set up the selected probe for common statistics and allow for only few (optional) additional configuration. Therefore they are best suited for users with only little knowledge of the statistics collector or for non-complex statistics tracing scenarios.

Of course macro functions do not make use of the entire feature set of the statistics collector probes. The following limitations apply when using macro functions only:

**Every counter can be assigned to exactly ONE macro function.** A counter cannot be used for multiple macro functions. That means that the number of available macro functions depends on the number of available counters of the selected probe. A counter is assigned to a macro function by the StatCol.<probe>.Counter <counter> FunCTioN <macro> command.

**Only the first available filter element of a filter will be used for filtering.** Any second (or third, ...) filter elements of a filter are disabled by default. Advanced users may enable and configure those additional filter elements to set up more complex filtering criteria. This may involve overwriting some configurations made by the macro functions, hence the recommended sequence is to first select the macro function and then to set up additional filtering criteria via the StatCol<probe>.Counter <counter> Filter <element> <item> commands.

### AvgPayloadLength

Average payload length: Outputs the average payload length in bytes of request transfers.

<table>
<thead>
<tr>
<th>Mandatory additional configuration</th>
<th>StatCol.&lt;probe&gt;.Counter &lt;counter&gt; MUX</th>
</tr>
</thead>
<tbody>
<tr>
<td>Optional additional configuration</td>
<td>StatCol.&lt;probe&gt;.Counter &lt;counter&gt; Filter &lt;element&gt; MaSTerADDResS</td>
</tr>
<tr>
<td></td>
<td>StatCol.&lt;probe&gt;.Counter &lt;counter&gt; Filter &lt;element&gt; SLaVeADDResS</td>
</tr>
<tr>
<td></td>
<td>StatCol.&lt;probe&gt;.Counter &lt;counter&gt; Filter &lt;element&gt; ReaD</td>
</tr>
<tr>
<td></td>
<td>StatCol.&lt;probe&gt;.Counter &lt;counter&gt; Filter &lt;element&gt; WRite</td>
</tr>
<tr>
<td></td>
<td>StatCol.&lt;probe&gt;.Counter &lt;counter&gt; Filter &lt;element&gt; REQUserInfo</td>
</tr>
</tbody>
</table>

### Throughput

Payload per cycle: Outputs the payload in bytes per cycle.

<table>
<thead>
<tr>
<th>Mandatory additional configuration</th>
<th>StatCol.&lt;probe&gt;.Counter &lt;counter&gt; MUX</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>StatCol.&lt;probe&gt;.CollectTime</td>
</tr>
</tbody>
</table>

©1989-2019 Lauterbach GmbH
<table>
<thead>
<tr>
<th>Optional additional configuration</th>
<th>StatCol.&lt;probe&gt;.Counter &lt;counter&gt; Filter &lt;element&gt; MasterADDRess</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>StatCol.&lt;probe&gt;.Counter &lt;counter&gt; Filter &lt;element&gt; SlaveADDRess</td>
</tr>
<tr>
<td></td>
<td>StatCol.&lt;probe&gt;.Counter &lt;counter&gt; Filter &lt;element&gt; Read</td>
</tr>
<tr>
<td></td>
<td>StatCol.&lt;probe&gt;.Counter &lt;counter&gt; Filter &lt;element&gt; Write</td>
</tr>
<tr>
<td></td>
<td>StatCol.&lt;probe&gt;.Counter &lt;counter&gt; Filter &lt;element&gt; REQUserInfo</td>
</tr>
</tbody>
</table>

**LnkOcc**

- Link occupancy: Percentage of non-idle cycles.

<table>
<thead>
<tr>
<th>Mandatory additional configuration</th>
<th>StatCol.&lt;probe&gt;.CollectTime</th>
</tr>
</thead>
<tbody>
<tr>
<td>Optional additional configuration</td>
<td>-</td>
</tr>
</tbody>
</table>

**ArbConf**

- Arbitration conflicts: Percentage of busy cycles caused by a target which cannot accept further write transactions from the initiator.

<table>
<thead>
<tr>
<th>Mandatory additional configuration</th>
<th>StatCol.&lt;probe&gt;.CollectTime</th>
</tr>
</thead>
<tbody>
<tr>
<td>Optional additional configuration</td>
<td>-</td>
</tr>
</tbody>
</table>

**TransUflow**

- Transaction underflow: Percentage of wait cycles (The initiator is not able to send as much data as requested by the target).

<table>
<thead>
<tr>
<th>Mandatory additional configuration</th>
<th>StatCol.&lt;probe&gt;.CollectTime</th>
</tr>
</thead>
<tbody>
<tr>
<td>Optional additional configuration</td>
<td>StatCol.&lt;probe&gt;.Counter &lt;counter&gt; Filter &lt;element&gt; MasterADDRess</td>
</tr>
<tr>
<td></td>
<td>StatCol.&lt;probe&gt;.Counter &lt;counter&gt; Filter &lt;element&gt; SlaveADDRess</td>
</tr>
</tbody>
</table>
IBusy

Initiator busy: Percentage of busy cycles caused by an initiator which cannot accept further read data from the target.

<table>
<thead>
<tr>
<th>Mandatory additional configuration</th>
<th>StatCol.&lt;probe&gt;.CollectTime</th>
</tr>
</thead>
<tbody>
<tr>
<td>Optional additional configuration</td>
<td>-</td>
</tr>
</tbody>
</table>

HistPayloadLen

Histogram of payload length: Filter packets by means of payload length. A histogram can be obtained by assigning the HPL macro to different counters with different min / max values.

<table>
<thead>
<tr>
<th>Mandatory additional configuration</th>
<th>StatCol.&lt;probe&gt;.Counter &lt;counter&gt; MIN (Minimum payload length in bytes)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>StatCol.&lt;probe&gt;.Counter &lt;counter&gt; MAX (Maximum payload length in bytes)</td>
</tr>
<tr>
<td>Optional additional configuration</td>
<td>StatCol.&lt;probe&gt;.Counter &lt;counter&gt; Filter &lt;element&gt; MasterADDRess</td>
</tr>
<tr>
<td></td>
<td>StatCol.&lt;probe&gt;.Counter &lt;counter&gt; Filter &lt;element&gt; SlaveADDRess</td>
</tr>
<tr>
<td></td>
<td>StatCol.&lt;probe&gt;.Counter &lt;counter&gt; Filter &lt;element&gt; Read</td>
</tr>
<tr>
<td></td>
<td>StatCol.&lt;probe&gt;.Counter &lt;counter&gt; Filter &lt;element&gt; Write</td>
</tr>
<tr>
<td></td>
<td>StatCol.&lt;probe&gt;.Counter &lt;counter&gt; Filter &lt;element&gt; REQUserInfo</td>
</tr>
</tbody>
</table>

HistPresDist

Histogram of pressure distribution: Filter packets by priority. A histogram can be obtained by assigning the HPD macro to different counters with different min / max values.

<table>
<thead>
<tr>
<th>Mandatory additional configuration</th>
<th>StatCol.&lt;probe&gt;.Counter &lt;counter&gt; MIN (Minimum pressure)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>StatCol.&lt;probe&gt;.Counter &lt;counter&gt; MAX (Maximum pressure)</td>
</tr>
</tbody>
</table>
HistLatDist

Histogram of latency distribution: Filter read packets by latency. A histogram can be obtained by assigning the HistLatDist macro to different counters with different min / max values.

<table>
<thead>
<tr>
<th>Optional additional configuration</th>
<th>StatCol.&lt;probe&gt;.Counter &lt;counter&gt; Filter &lt;element&gt; SLaVeADDResS</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>StatCol.&lt;probe&gt;.Counter &lt;counter&gt; Filter &lt;element&gt; ReaD</td>
</tr>
<tr>
<td></td>
<td>StatCol.&lt;probe&gt;.Counter &lt;counter&gt; Filter &lt;element&gt; WRite</td>
</tr>
<tr>
<td></td>
<td>StatCol.&lt;probe&gt;.Counter &lt;counter&gt; Filter &lt;element&gt; REQUserInfo</td>
</tr>
</tbody>
</table>

AvgLatDist

Average latency distribution: Output average latency of read transactions in latency / cycle.

<table>
<thead>
<tr>
<th>Mandatory additional configuration</th>
<th>StatCol.&lt;probe&gt;.Counter &lt;counter&gt; MIN (Minimum latency)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>StatCol.&lt;probe&gt;.Counter &lt;counter&gt; MAX (Maximum latency)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Optional additional configuration</th>
<th>StatCol.&lt;probe&gt;.Counter &lt;counter&gt; Filter &lt;element&gt; SLaVeADDResS</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>StatCol.&lt;probe&gt;.Counter &lt;counter&gt; Filter &lt;element&gt; MaSTerADDResS</td>
</tr>
<tr>
<td></td>
<td>StatCol.&lt;probe&gt;.Counter &lt;counter&gt; Filter &lt;element&gt; REQUserInfo</td>
</tr>
</tbody>
</table>

Displaying Trace Results

The trace output can be displayed or analyzed with the StatColTrace command. Please refer to the general <trace> command description in “General Commands Reference Guide T” (general_ref_t.pdf).
In addition to the standard `<trace>` command the following signals related to the statistics collector are available:

- **SCC0.<probe>** Value of statistics collector counter 0.
- **SCC1.<probe>** Value of statistics collector counter 1.
- **SCC2.<probe>** Value of statistics collector counter 2.
- **SCC3.<probe>** Value of statistics collector counter 3.
- **SCC4.<probe>** Value of statistics collector counter 4.
This example shows how to gather throughput statistics of the EMIF1 request port on an OMAP4. EMIF1 is monitored by the SDRAM probe on OMAP4.

```
StatCol.RESet
CAnalyzer.AutoArm OFF ; The statistics collector outputs data periodically so we must arm the
CAnalyzer.ARM ; CAnalyzer before the statistics collector is activated.

StatCol.SDRAM.Counter 0 FunCTion THRoughput
StatCol.SDRAM.Counter 0 Filter MUX.Emif1REQUEST
StatCol.SDRAM.CollectTime 255.
StatCol.SDRAM.ON
...
StatCol.SDRAM.OFF ; Turn off the statistics collector ...
CAnalyzer.OFF ; ... before shutting down the CAnalyzer.

StatColTrace.List CYcle SCC0 List.NoDummy ; Display value of counter 0
```

**StatCol Module Configuration**

If the statistics collector is not enabled for your specific device, the following commands allow for an afterward configuration. Both, the base address and the ID must be set in order to enable the statistics collector:

- **SYSTEM.CONFIG.TISCBASE** Set the base address of the statistics collector.
- **SYSTEM.CONFIG.TISCID** Set the STM master ID of the statistics collector.
FAQ

No information available