LAUTERBACH
DEVELOPMENT TOOLS

General Commands Reference Guide I

MANUAL

General Commands Reference Guide I

General Commands Reference Guide I

Version 06-Jun-2024

History

12-Dec-2023 New command ITM.DataTraceCorrelatePusher.

10-Mar-2022 New command ITM.DataTraceCorrelateDistance.

I2C — I2C control

Intel® x86/x64

The **I2C** command group is used to read and write data over the I2C bus.

I2C.PIN — Set I2C pin to specified level

Intel® x86/x64

Format:	**I2C.PIN SCL** | **SDA** *<level>*
<level>:	**L** | **Z** | **0** | **1**

SCL, **SDA**	Pin names: serial clock line (SCL), serial data line (SDA)
L | **0**	Set specified I2C pin to low.
Z | **1**	Tristate specified I2C pin.

See also

- I2C.PIN()

Format:	**I2C.THreshold** *<level>*

The default threshold is 1.0V.

Currently regular I2C devices operate within a bus voltage range from 1.6V to 5V. With the default threshold of 1.0V the full bus voltage range from 1.6V to 5V should be supported and thus there should be no need to modify the I2C threshold.

I2C.TRANSFER Transfer bytes on I2C bus

Intel® x86/x64

Format 1:	**I2C.TRANSFER** *<device_id>* [{*<cmd_byte>*}] **WRite** {*<data_byte>*}
Format 2:	**I2C.TRANSFER** *<device_id>* [{*<cmd_byte>*}] **ReaD** | **ReaDNoNak** *<length>*

WRite

Example: EEPROM Write (with 8-bit EEPprom address) with I2C slave address 0xA0, 4 byte write (0x11,0x22,0x33,0x44) to address 0x30.

```
I2C.TRANSFER 0xA0 0x30 WRite 0x11 0x22 0x33 0x44
```

The following is generated on the I2C bus for a **WRite** transfer:

Start *<device_id_write>* *<cmd_bytes>* *<data_bytes>* Stop

ReaD, ReaDNoNack

Example: EEPROM Read (with 8-bit EEPprom address) with I2C slave address 0xA0, 4 bytes from address 0x30. The TRACE32 function **I2C.DATA()** allows to get the read data.

```
I2C.TRANSFER 0xA0 0x30 ReaD 4.

PRINT I2C.DATA(0.)
PRINT I2C.DATA(1.)
PRINT I2C.DATA(2.)
PRINT I2C.DATA(3.)
```

The following is generated on the I2C bus for a **ReaD** transfer:

Start *<device_id_write>* *<cmd_bytes>* Repeated-Start *<device_id_read>* *<data_bytes>* Stop

TRACE32 sends ACK for each byte read, it sends NACK for the last byte.

A **ReaDNoNaK** advises TRACE32 to ACK all bytes.

See also

- I2C.DATA()

I2C.TransferRAW — Transfer bytes on I2C bus

Intel® x86/x64

Format: **I2C.TransferRAW** *<32-bit word>* *<32-bit word>* …

Low level access mode to I2C controller. Each 32-bit word has the following meaning:

mask:	bit number:	
0x000000FF	Bit 7..0	DATA bits to send via I2C for this I2C Byte. A '1' bit means to **not** drive the SDA line. Sending a 0xFF might be used if you want to receive bits.
0x00000100	Bit 8	Send an I2C START condition before transferring the data bits. This might also be used for an I2C “Repeated Start”.
0x00000200	Bit 9	Send an I2C STOP condition after transferring the data bits and ACK bit.
0x00000400	Bit 10	Send an ACK bit after the data bits have been transferred. (i.e. pull SDA to low in the ACK bit phase).
0x00000800	Bit 11	Ignore ACK phase; do **not** abort if no ACK is received in the ACK bit phase. (Continue even if SDA line stays HIGH in the ACK bit phase).

For each 32-bit word specified, there will be a return value which can be read via the **I2C.DATA(<index>)** function. The return value means:

mask:	bit number:	
0x000000FF	Bit 7..0	DATA bits received via I2C for this I2C Byte. This reflects the bits received via the SDA line.

0x00000100	Bit 8	An asserted ACK was seen in the ACK phase of this byte (SDA line was pulled low).
0x00000200	Bit 9	If this bit is set, then an error occurred and no further bytes were transferred (the transfer was aborted here). When this bit is set, Bit 19..16 specify the reason for the abort.
0x000F0000	Bit 19..16	Abort reason. Please note that an abort also means that no explicit STOP condition might have been sent to the bus. So the I2C transaction might still be deemed ACTIVE. Error code in Bit 19..16: • 0x1: No ACK was received • 0x3: A timeout occurred (this means the SCL line was pulled LOW by something else.)

Integrator

Integrator logic analyzer

The trace method **Integrator** is available if a PowerIntegrator module is connected or if a saved PowerIntegrator recording is loaded to a TRACE32 Instruction Set Simulator.

The trace method **Integrator** provides the same functionality as common logic analyzers. It is mainly used to sample digital target signals such as port-signals, communication-interface signals or address/data-bus signals. All the recordings are automatically time-correlated with other TRACE32 trace units. By use of the embedded protocol analysis in TRACE32, the meaning of communication protocols can be visualized.

In addition, analog signals can be traced and triggered, which allows a detailed analysis of the target energy consumption.

Additional information can be found here:

- **"PowerIntegrator User's Guide"** (powerintegrator_user.pdf)
- **www.lauterbach.com/pi.html**
- **www.lauterbach.com/energyprofiling.html**

The chapter **"Integrator-specific Trace Commands"**, page 14 describes the Integrator-specific configuration commands. While the chapter **"Generic Integrator Trace Commands"**, page 22 lists the Integrator trace analysis and display commands, which are shared with other TRACE32 trace methods.

See also

- Trace.METHOD
- ▲ 'Generic Integrator Trace Commands' in 'General Commands Reference Guide I'

Integrator.ABCDEF Sampling configuration for probes ABCDEF

Format: **Integrator.ABCDEF** *<option>*

<option>:
2GHZ
500MHZ
Fixed500MHZ
250MHZ
State
StatePLL
StatePLLBoth

CLKA
CLKB
Falling
Rising
DDR
SAMPLE

2GHZ	Timing Mode 2 GHz for probes ABCDEF only, probes JKLMNO get lost.
500MHZ	Timing Mode 500 MHz for probes ABCDEF only, probes JKLMNO get lost.
Fixed500MHZ	Timing Mode fixed sampling with 500 MHz for probes ABCDEF only, probes JKLMNO get lost.
250MHZ	Timing Mode 250 MHz.
State	State Mode clocked by CLKA or CLKB.
StatePLL	State PLL Mode, clocked by CLKA or CLKB.
StatePLLBoth	State PLL Mode, clocked by CLKA or CLKB, for all probes.
CLKA	Clock A select for State-Mode or State-PLL-Mode.
CLKB	Clock B select for State-Mode or State-PLL-Mode.
Falling	sampling on falling edge of selected clock CLKA or CLKB.
Rising	sampling on rising edge of selected clock CLKA or CLKB.

DDR	sampling on rising and falling edge of selected clock CLKA or CLKB.
SAMPLE	sampling delay of selected clock CLKA or CLKB (-3 …+6 ns in steps of 250 ps), State-PLL-Mode only.

Integrator.Break — Stop trace

Format: **Integrator.Break**

The trace is stopped and the trace storage is ready for read-out.

Integrator.CSELect — Select signal for counter

Format: **Integrator.CSELect** *<channel>*

See also

- Trace

Integrator.DisConfig.LOAD — Load DLL for protocol analysis

Format: **Integrator.DisConfig.LOAD** *<dll-file> <integrator-signal-list>*

Loads a DLL that performs a protocol analysis for the bus trace.

See also

- <trace>.DisConfig

Format: **Integrator.JKLMNO** *<option>*

<option>: **250MHZ**
State
StatePLL

CLKJ
CLKK
Falling
Rising
SAMPLE

250MHZ	Timing Mode 250 MHz.
State	State Mode, clocked by CLKJ or CLKK.
StatePLL	State PLL Mode, clocked by CLKJ or CLKK.
CLKJ	Clock J select for State-Mode or State-PLL-Mode.
CLKK	Clock K select for State-Mode or State-PLL-Mode.
Falling	sampling on falling edge of selected clock CLKJ or CLKK.
Rising	sampling on rising edge of selected clock CLKJ or CLKK.
SAMPLE	sampling delay of selected clock CLKJ or CLKK (-3 … +6 ns in steps of 250 ps), State-PLL-Mode only.

Format:	**Integrator.Mode** [*<mode>*]
<mode>:	**Fifo** **Stack** **STREAM**

Selects the trace operation mode.

Fifo If the trace is full, new records will overwrite older records. The trace records always the last cycles before the break.

Stack If the trace is full recording will be stopped. The trace always records the first cycles after starting the trace.

STREAM The trace data is immediately conveyed to a file on the host after it was placed into the trace. This procedure extends the size of the trace memory to up to several T Frames.

STREAM mode can only be used if the average data rate at the trace port does not exceed the maximum transmission rate of the host interface in use. Peak loads at the trace port are intercepted by the trace memory, which can be considered to be operating as a large FIFO.

The streaming file is placed into the TRACE32 temp directory (**OS.PresentTemporaryDirectory()**) by default and is named *<trace32_instance_id>***streama.t32** (**OS.ID()**). If you explicitly want to specify a location for the streaming file use the command **Integrator.STREAMFILE** *<file>*.

Note that the contents of the streaming file are in a proprietary format and not intended for use in external applications. See **Trace.EXPORT** for details on how to export trace data for external applications.

Please be aware that the streaming file is deleted as soon as you de-select the STREAM mode or when you quit TRACE32.

See also

- <trace>.Mode

Format: **Integrator.Program** [*<file>*]

Opens an editor window, which is used for creating trigger programs. The program entry in the window is guided by softkeys and the online-help system. Writing the program is supported by softkeys and online help. To program the trace press the **Compile** button. The command **Integrator.ReProgram** can be used to load ready-to-run programs in the trigger unit. The commands in this manual refer to the trigger program, unless otherwise mentioned. Pleas refer for more information to **"PowerIntegrator Programming Guide"** (powerintegrator_prog.pdf).

See also

- SETUP.EDITOR
- ▲ 'Release Information' in 'Legacy Release History'

Integrator.ReProgram Program trigger unit

Format: **Integrator.ReProgram** [*<file>*]

Programs the trigger unit of the trace with an existing program. The program should be error free, when using this command. To write an error free program, use the command **Integrator.Program**.

Integrator.TOut Enable trigger output line

Format: **Integrator.TOut ON** | **OFF**

Enables or disables the trigger output line of the integrator.

Format:	**Integrator.TPreDelay** [*<time>* | *<percent>* | **DEFAULT**]
<percent>:	**0** … **1000%**
<cycles>:	**0** … **4000000000.**

Selects a delay time between the start of the recording and the release of the trigger system. This delay is useful if previous events before a trigger point are of interest.

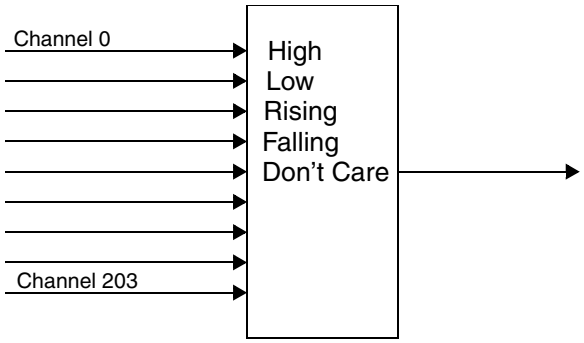
See also

- Trace

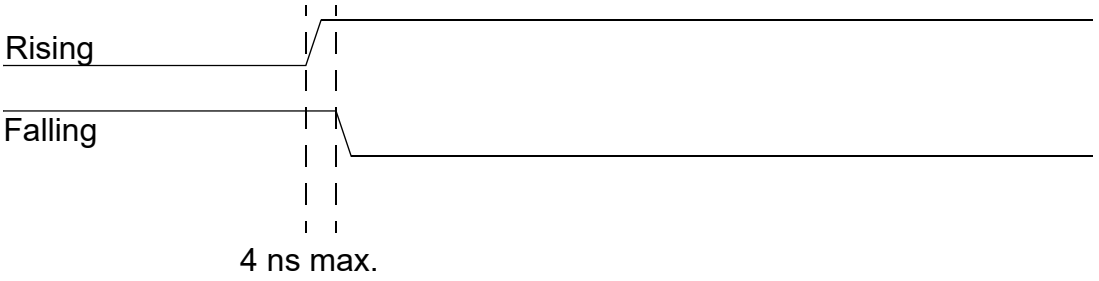
Format: **Integrator.TSYNC** [*<channel>*] [*<mode>*]

The command **Integrator.TSYNC.SIMPLE** resets the values of TPreDelay, TWidth, TCount and TDelay to defaults. Command **Integrator.TSYNC.SELect** doesn't modify these values.

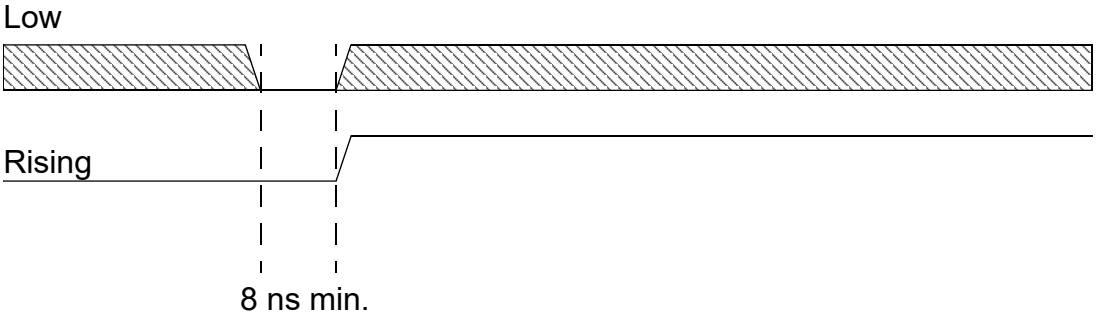
The trigger signal can be generated out of the 204 port channels. Each signal can be qualified as high, low, rising or falling edge.



More than 1 edge can be combined to a trigger word. To detect a valid combination of edges, the edges must have a max. skew of 4 ns.



Edges and state signals can be combined. The state signal must be stable 4 ns before the edge. The sampling of the state signal is guaranteed before the edge is detected.



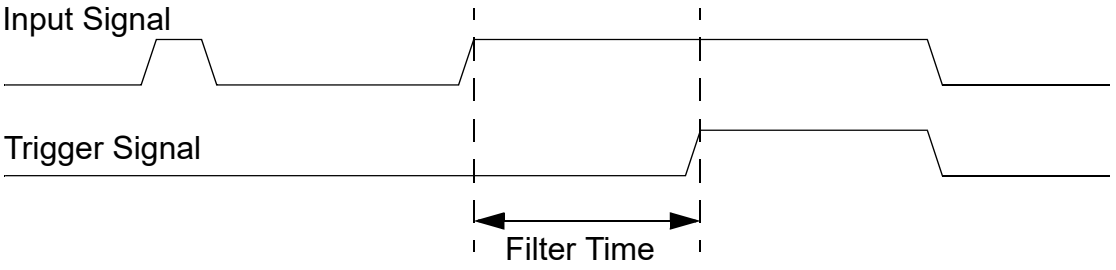
See also

■ Trace

▲ 'Release Information' in 'Legacy Release History'

Format:	**Integrator.TWidth** [*<value>*]
<value>:	**0** … **2.5 us**

The trigger filter time is defined. All trigger events which are shorter than this value are ignored



Integrator.ACCESS Define access path to program code for trace decoding

See command **<trace>.ACCESS** in 'General Commands Reference Guide T' (general_ref_t.pdf, page 131).

Integrator.Arm Arm the trace

See command **<trace>.Arm** in 'General Commands Reference Guide T' (general_ref_t.pdf, page 134).

Integrator.AutoArm Arm automatically

See command **<trace>.AutoArm** in 'General Commands Reference Guide T' (general_ref_t.pdf, page 135).

Integrator.AutoFocus Calibrate AUTOFOCUS preprocessor

See command **<trace>.AutoFocus** in 'General Commands Reference Guide T' (general_ref_t.pdf, page 135).

Integrator.AutoInit Automatic initialization

See command **<trace>.AutoInit** in 'General Commands Reference Guide T' (general_ref_t.pdf, page 140).

Integrator.BookMark Set a bookmark in trace listing

See command **<trace>.BookMark** in 'General Commands Reference Guide T' (general_ref_t.pdf, page 140).

Integrator.Chart — Display trace contents graphically

See command **<trace>.Chart** in 'General Commands Reference Guide T' (general_ref_t.pdf, page 144).

Integrator.ComPare — Compare trace contents

See command **<trace>.ComPare** in 'General Commands Reference Guide T' (general_ref_t.pdf, page 192).

Integrator.DISable — Disable the trace

See command **<trace>.DISable** in 'General Commands Reference Guide T' (general_ref_t.pdf, page 197).

Integrator.DisConfig — Trace disassembler configuration

See command **<trace>.DisConfig** in 'General Commands Reference Guide T' (general_ref_t.pdf, page 198).

Integrator.DisConfig.CYcle — Trace disassemble setting

See command **<trace>.DisConfig.CYcle** in 'General Commands Reference Guide T' (general_ref_t.pdf, page 198).

Integrator.DisConfig.FlowMode — Enable FlowTrace analysis

See command **<trace>.DisConfig.FlowMode** in 'General Commands Reference Guide T' (general_ref_t.pdf, page 200).

Integrator.DisConfig.RESet — Reset trace disassemble setting

See command **<trace>.DisConfig.RESet** in 'General Commands Reference Guide T' (general_ref_t.pdf, page 200).

Integrator.DRAW — Plot trace data against time

See command **<trace>.DRAW** in 'General Commands Reference Guide T' (general_ref_t.pdf, page 201).

Integrator.EXPORT — Export trace data for processing in other applications

See command **<trace>.EXPORT** in 'General Commands Reference Guide T' (general_ref_t.pdf, page 212).

Integrator.FILE — Load a file into the file trace buffer

See command **<trace>.FILE** in 'General Commands Reference Guide T' (general_ref_t.pdf, page 233).

Integrator.Find — Find specified entry in trace

See command **<trace>.Find** in 'General Commands Reference Guide T' (general_ref_t.pdf, page 235).

Integrator.FindAll — Find all specified entries in trace

See command **<trace>.FindAll** in 'General Commands Reference Guide T' (general_ref_t.pdf, page 237).

Integrator.FindChange — Search for changes in trace flow

See command **<trace>.FindChange** in 'General Commands Reference Guide T' (general_ref_t.pdf, page 238).

Integrator.Get — Display input level

See command **<trace>.Get** in 'General Commands Reference Guide T' (general_ref_t.pdf, page 242).

Integrator.GOTO — Move cursor to specified trace record

See command **<trace>.GOTO** in 'General Commands Reference Guide T' (general_ref_t.pdf, page 244).

Integrator.Init — Initialize trace

See command **<trace>.Init** in 'General Commands Reference Guide T' (general_ref_t.pdf, page 246).

Integrator.List — List trace contents

See command **<trace>.List** in 'General Commands Reference Guide T' (general_ref_t.pdf, page 248).

Integrator.ListNesting — Analyze function nesting

See command **<trace>.ListNesting** in 'General Commands Reference Guide T' (general_ref_t.pdf, page 263).

Integrator.ListVar — List variable recorded to trace

See command **<trace>.ListVar** in 'General Commands Reference Guide T' (general_ref_t.pdf, page 266).

Integrator.LOAD — Load trace file for offline processing

See command **<trace>.LOAD** in 'General Commands Reference Guide T' (general_ref_t.pdf, page 270).

Integrator.OFF — Switch off

See command **<trace>.OFF** in 'General Commands Reference Guide T' (general_ref_t.pdf, page 278).

Integrator.PROfileChart — Profile charts

See command **<trace>.PROfileChart** in 'General Commands Reference Guide T' (general_ref_t.pdf, page 283).

Integrator.PROTOcol — Protocol analysis

See command **<trace>.PROTOcol** in 'General Commands Reference Guide T' (general_ref_t.pdf, page 339).

Integrator.PROTOcol.Chart Graphic display for user-defined protocol

See command **<trace>.PROTOcol.Chart** in 'General Commands Reference Guide T' (general_ref_t.pdf, page 339).

Integrator.PROTOcol.Draw Graphic display for user-defined protocol

See command **<trace>.PROTOcol.Draw** in 'General Commands Reference Guide T' (general_ref_t.pdf, page 341).

Integrator.PROTOcol.EXPORT Export trace buffer for user-defined protocol

See command **<trace>.PROTOcol.EXPORT** in 'General Commands Reference Guide T' (general_ref_t.pdf, page 342).

Integrator.PROTOcol.Find Find in trace buffer for user-defined protocol

See command **<trace>.PROTOcol.Find** in 'General Commands Reference Guide T' (general_ref_t.pdf, page 343).

Integrator.PROTOcol.list Display trace buffer for user-defined protocol

See command **<trace>.PROTOcol.list** in 'General Commands Reference Guide T' (general_ref_t.pdf, page 344).

Integrator.PROTOcol.PROfileChart Profile chart for user-defined protocol

See command **<trace>.PROTOcol.PROfileChart** in 'General Commands Reference Guide T' (general_ref_t.pdf, page 347).

Integrator.PROTOcol.PROfileSTATistic Profile chart for user-defined protocol

See command **<trace>.PROTOcol.PROfileSTATistic** in 'General Commands Reference Guide T' (general_ref_t.pdf, page 348).

Integrator.PROTOcol.STATistic Display statistics for user-defined protocol

See command **<trace>.PROTOcol.STATistic** in 'General Commands Reference Guide T' (general_ref_t.pdf, page 350).

Integrator.REF Set reference point for time measurement

See command **<trace>.REF** in 'General Commands Reference Guide T' (general_ref_t.pdf, page 357).

Integrator.RESet Reset command

See command **<trace>.RESet** in 'General Commands Reference Guide T' (general_ref_t.pdf, page 357).

Integrator.SAVE Save trace for postprocessing in TRACE32

See command **<trace>.SAVE** in 'General Commands Reference Guide T' (general_ref_t.pdf, page 358).

Integrator.SelfArm Automatic restart of trace recording

See command **<trace>.SelfArm** in 'General Commands Reference Guide T' (general_ref_t.pdf, page 362).

Integrator.ShowFocus Display data eye for AUTOFOCUS preprocessor

See command **<trace>.ShowFocus** in 'General Commands Reference Guide T' (general_ref_t.pdf, page 365).

Integrator.SIZE Define buffer size

See command **<trace>.SIZE** in 'General Commands Reference Guide T' (general_ref_t.pdf, page 373).

Integrator.SnapShot — Restart trace capturing once

See command **<trace>.SnapShot** in 'General Commands Reference Guide T' (general_ref_t.pdf, page 373).

Integrator.SPY — Adaptive stream and analysis

See command **<trace>.SPY** in 'General Commands Reference Guide T' (general_ref_t.pdf, page 374).

Integrator.state — Display trace configuration window

See command **<trace>.state** in 'General Commands Reference Guide T' (general_ref_t.pdf, page 376).

Integrator.STATistic — Statistic analysis

See command **<trace>.STATistic** in 'General Commands Reference Guide T' (general_ref_t.pdf, page 378).

Integrator.STREAMCompression — Select compression mode for streaming

See command **<trace>.STREAMCompression** in 'General Commands Reference Guide T' (general_ref_t.pdf, page 485).

Integrator.STREAMFILE — Specify temporary streaming file path

See command **<trace>.STREAMFILE** in 'General Commands Reference Guide T' (general_ref_t.pdf, page 486).

Integrator.STREAMFileLimit — Set size limit for streaming file

See command **<trace>.STREAMFileLimit** in 'General Commands Reference Guide T' (general_ref_t.pdf, page 487).

Integrator.STREAMLOAD — Load streaming file from disk

See command **<trace>.STREAMLOAD** in 'General Commands Reference Guide T' (general_ref_t.pdf, page 488).

Integrator.STREAMSAVE — Save streaming file to disk

See command **<trace>.STREAMSAVE** in 'General Commands Reference Guide T' (general_ref_t.pdf, page 490).

Integrator.TCount — Set trigger counter

See command **<trace>.TCount** in 'General Commands Reference Guide T' (general_ref_t.pdf, page 490).

Integrator.TDelay — Trigger delay

See command **<trace>.TDelay** in 'General Commands Reference Guide T' (general_ref_t.pdf, page 491).

Integrator.TestFocus — Test trace port recording

See command **<trace>.TestFocus** in 'General Commands Reference Guide T' (general_ref_t.pdf, page 494).

Integrator.Timing — Waveform of trace buffer

See command **<trace>.Timing** in 'General Commands Reference Guide T' (general_ref_t.pdf, page 499).

Integrator.TRACK — Set tracking record

See command **<trace>.TRACK** in 'General Commands Reference Guide T' (general_ref_t.pdf, page 502).

Integrator.TRIGGER — Trigger the trace

See command **<trace>.TRIGGER** in 'General Commands Reference Guide T' (general_ref_t.pdf, page 502).

Integrator.TSELect — Select trigger source

See command **<trace>.TSELect** in 'General Commands Reference Guide T' (general_ref_t.pdf, page 503).

Integrator.View — Display single record

See command **<trace>.View** in 'General Commands Reference Guide T' (general_ref_t.pdf, page 504).

Integrator.ZERO — Align timestamps of trace and timing analyzers

See command **<trace>.ZERO** in 'General Commands Reference Guide T' (general_ref_t.pdf, page 505).

IProbe

IProbe logic analyzer

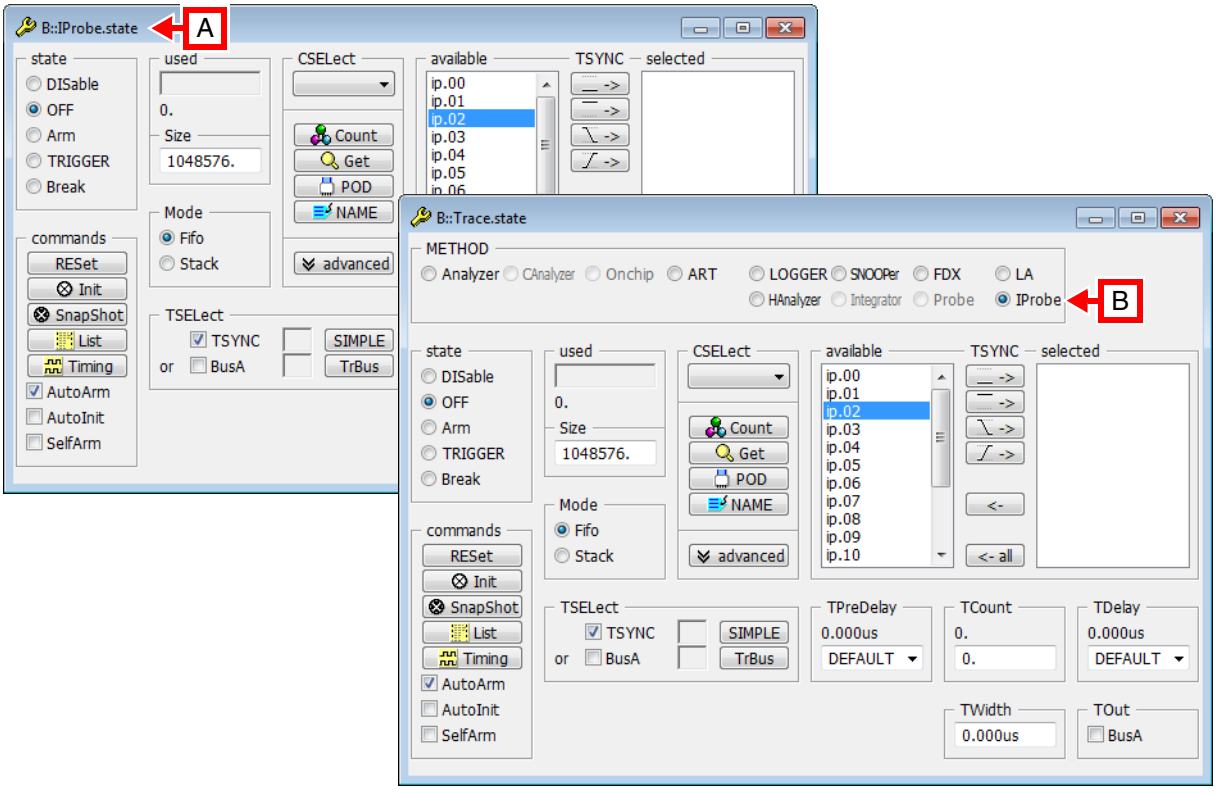
The **IProbe** commands describe the handling of the **IProbe logic analyzer** within the **PowerTrace II / PowerTrace III** and **PowerTrace Serial**. There are 3 different probes for different requirements:

- Standard digital probe
- Standard analog probe
- Differential probe

Each probe can just be connected alternatively. Depending on the used probe, there are commands that are common to all probes and probe-specific commands.

For selecting and configuring the trace method IProbe, use the TRACE32 command line or a PRACTICE script (*.cmm) or the **IProbe.state** window [**A**].

Alternatively, use the **Trace.state** window: Click the option **IProbe** or execute the command **Trace.METHOD IProbe** in order to select the **trace method IProbe** [**B**].



The trace method **IProbe** provides a reduced functionality in comparison to the trace method **Integrator** or common logic analyzers. The trace method **IProbe** is mainly used to sample digital target signals such as port-signals, communication-interface signals or address/data-bus signals. All the recordings are automatically time-correlated with other TRACE32 trace units. By use of the embedded protocol analysis in TRACE32, the meaning of communication protocols can be visualized.

In addition, analog signals can be traced and triggered, which allows a detailed analysis of the target energy consumption.

Refer for more information to **"IProbe User's Guide"** (iprobe_user.pdf).

The chapter **"Integrator-specific Trace Commands"**, page 14 describes the IProbe-specific configuration commands. While the chapter **"Generic Integrator Trace Commands"**, page 22 lists the IProbe trace analysis and display commands, which are shared with other TRACE32 trace methods.

See also

- Trace.METHOD

- 'IProbe Functions' in 'General Function Reference'
- 'Introduction' in 'IProbe User's Guide'

IProbe.ALOWerLIMit — Set lower trigger/filter comparator value

Format:	**IProbe.ALOWerLIMit** *<channel> <value>*
<channel>:	**V0** | **V1** | **V2** | **V3** | **I0** | **I1** | **I2**

Sets the lower limit for the trigger and filter logic of a physical ADC channel. The *<value>* must be given in Volts for voltage channels or Amperes for current channels.

The actual comparison performed depends on the **IProbe.ATrigMODE** setting.

See also

- IProbe.state

IProbe.ATrigEN — Enable/disable trigger contribution of a channel

Format:	**IProbe.ATrigEN** *<channel>* [**ON** | **OFF**]
<channel>:	**V0** | **V1** | **V2** | **V3** | **I0** | **I1** | **I2**

Enables or disables the contribution of a physical's channel comparator logic to the IProbe trigger. If this setting is enabled for multiple channels, a trigger is performed when any of the channels' trigger condition is satisfied.

f no [**ON** | **OFF**] argument is given, the current state of the setting is toggled.

NOTE:	Even if this setting is **OFF** for a given channel, the comparator may still be used for filtering. Refer to the **POD.ADC** command for details.

See also

- IProbe.state

Format:	**IProbe.ATrigMODE** *<channel> <mode>*
<channel>:	**V0** | **V1** | **V2** | **V3** | **I0** | **I1** | **I2**
<mode>:	**DISabled** | **GreaterUPPer** | **SmallerUPPer** | **GreaterLOWer** | **SmallerLOWer** | **INBound** | **BEYONDbound**

Sets the condition for a physical channel's comparator logic.

DISabled	No value matches.
GreaterUPPer	Value must be greater than upper limit. See **IProbe.AUPPerLIMit**.
SmallerUPPer	Value must be less than upper limit. See **CIProbe.AUPPerLIMit**.
GreaterLOWer	Value must be greater than lower limit. See **CIProbe.ALOWerLIMit**.
SmallerLOWer	Value must be less than lower limit. See **CIProbe.ALOWerLIMit**.
INBound	Value must be greater than lower limit and less than upper limit. See **CIProbe.ALOWerLIMit** and **CIProbe.AUPPerLIMit**.
BEYONDBound	Value must be less than lower limit or greater than upper limit. See **CIProbe.ALOWerLIMit** and **CIProbe.AUPPerLIMit**.

See also

- IProbe.state

Format: **IProbe.AUPPerLIMit** *<channel> <value>*

<channel>: **V0** | **V1** | **V2** | **V3** |
I0 | **I1** | **I2**

Sets the upper limit for the trigger and filter logic of a physical ADC channel. The *<value>* is in Volts for voltage channels and Amperes for current channels.

The actual comparison performed depends on the **IProbe.ATrigMODE** setting.

See also

- IProbe.state

IProbe.Break Manual IProbe break

Format: **IProbe.Break**

Forces IProbe trigger system to Break state.

See also

- IProbe.state

IProbe.CSELect Source select for system counter

Not available for Analog Probe!

One of the 17 digital input channels can be selected to be routed as a source of the System Frequency Counter. Refer to **Count**.

Either the default channel names will be displayed or the user-defined names of each channel.

See also

- IProbe.state

Format: **IProbe.EXPORT** *<file> <signal_list>*

Without *<signal_list>* all IProbe signals are saved.

See also

- IProbe.state

IProbe.Mode Set trace operation mode

Format: **IProbe.Mode Fifo | Stack**

Fifo If the trace is full, new records will overwrite older records. The trace records always the last cycles before the break.

Stack If the trace is full recording will be stopped. The trace always records the first cycles after starting the trace.

See also

- IProbe.state
- <trace>.Mode

IProbe.SELFTEST Iprobe self-test

Format: **IProbe.SELFTEST**

Not implemented.

See also

- IProbe.state

Format:	**IProbe.SIZE** *<buffersize>*
<buffersize>:	**1** … *<maxtrace>*
<maxtrace>:	**524288.** **10484976.**

Depending on the version of **PowerTrace II / PowerTrace III**, the trace buffer for the **IProbe logic analyzer** can be:

- 512 K frames (24 bit data and 48 bit timestamp) or
- 1 M frames (24 bit data and 48 bit timestamp)

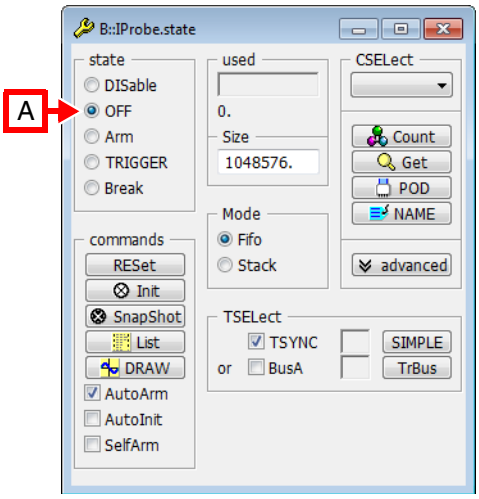
The trace buffer depth can be set between 1 and *<maxtrace>*. A smaller trace buffer speeds up analysis or searching.

See also

- IProbe.state

Format:	**IProbe.state**

Displays the configuration window for the **IProbe logic analyzer** with the **PowerTrace II / PowerTrace III**.



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

Exceptions:

- **Count** opens the **Count.state** window.
- **POD** opens the **POD IP** window.
- **NAME** opens the **NAME.list** window.
- **TrBus** opens the **TrBus.state** window.

See also

- <trace>.Arm
- <trace>.AutoArm
- <trace>.AutoInit
- <trace>.BookMark
- <trace>.Chart
- <trace>.ComPare
- <trace>.DISable
- <trace>.DisConfig
- <trace>.DRAW
- <trace>.FILE
- <trace>.Find
- <trace>.FindAll
- <trace>.FindChange
- <trace>.Get
- <trace>.GOTO
- <trace>.Init
- <trace>.List
- <trace>.ListNesting
- <trace>.ListVar
- <trace>.LOAD
- <trace>.OFF
- <trace>.PROfile
- <trace>.PROfileChart
- <trace>.PROTOcol
- <trace>.REF
- <trace>.RESet
- <trace>.SAVE
- <trace>.SelfArm
- <trace>.SnapShot
- <trace>.SPY
- <trace>.STATistic
- <trace>.STREAMCompression
- <trace>.STREAMFILE
- <trace>.STREAMFileLimit
- <trace>.TCount
- <trace>.TDelay
- <trace>.Timing
- <trace>.TRACK
- <trace>.View
- <trace>.ZERO
- IProbe.ALOWerLIMit
- IProbe.ATrigEN
- IProbe.ATrigMODE
- IProbe.AUPPerLIMit

■ IProbe.Break
■ IProbe.EXPORT
■ IProbe.SELFTEST
■ IProbe.TOut
■ IProbe.TRIGGER
■ IProbe.TSYNC
■ IProbe.CSELect
■ IProbe.Mode
■ IProbe.SIZE
■ IProbe.TPreDelay
■ IProbe.TSELect
■ IProbe.TWidth

▲ 'Release Information' in 'Legacy Release History'

IProbe.TOut Activates/deactivates the trigger output signal (BUSA)

Format: **IProbe.TOut** [**ON** | **OFF**]

See also

■ IProbe.state

IProbe.TPreDelay Define the trigger pre-delay counter

Format: **IProbe.TPreDelay**

See also

■ IProbe.state

IProbe.TRIGGER Ineffective command

Format: **IProbe.TRIGGER**

This command has no effect.
The IProbe logic analyzer changes to the state TRIGGER.

See also

■ IProbe.state

IProbe.TSELect — Select trigger input line

Format: **IProbe.TSELect** [**TSYNC** | **BusA**]

Selects TSYNC or Podbus trigger BUSA

See also

- IProbe.state

IProbe.TSYNC — Select trigger line and mode

Format: **IProbe.TSYNC**

See also

- IProbe.state

IProbe.TSYNC.SELect — Select trigger input pin and edge or state

Format: **IProbe.TSYNC.SELect**

Example:

```
IProbe.TSYNC.SELect IProbe.07 Falling

NAME.Group TEST IProbe.03 IProbe.07 IProbe.10
IProbe.TSYNC.SELect Group.TEST High
```

IProbe.TSYNC.SIMPLE — Select simple trigger

Format: **IProbe.TSYNC.SIMPLE**

Not available!

Format: **IProbe.TWidth**

See also

- IProbe.state

IProbe.Arm — Arm the trace

See command **<trace>.Arm** in 'General Commands Reference Guide T' (general_ref_t.pdf, page 134).

IProbe.AutoArm — Arm automatically

See command **<trace>.AutoArm** in 'General Commands Reference Guide T' (general_ref_t.pdf, page 135).

IProbe.AutoInit — Automatic initialization

See command **<trace>.AutoInit** in 'General Commands Reference Guide T' (general_ref_t.pdf, page 140).

IProbe.BookMark — Set a bookmark in trace listing

See command **<trace>.BookMark** in 'General Commands Reference Guide T' (general_ref_t.pdf, page 140).

IProbe.Chart — Display trace contents graphically

See command **<trace>.Chart** in 'General Commands Reference Guide T' (general_ref_t.pdf, page 144).

IProbe.ComPare — Compare trace contents

See command **<trace>.ComPare** in 'General Commands Reference Guide T' (general_ref_t.pdf, page 192).

IProbe.DISable — Disable the trace

See command **<trace>.DISable** in 'General Commands Reference Guide T' (general_ref_t.pdf, page 197).

IProbe.DisConfig — Trace disassembler configuration

See command **<trace>.DisConfig** in 'General Commands Reference Guide T' (general_ref_t.pdf, page 198).

IProbe.DRAW — Plot trace data against time

See command **<trace>.DRAW** in 'General Commands Reference Guide T' (general_ref_t.pdf, page 201).

IProbe.FILE — Load a file into the file trace buffer

See command **<trace>.FILE** in 'General Commands Reference Guide T' (general_ref_t.pdf, page 233).

IProbe.Find — Find specified entry in trace

See command **<trace>.Find** in 'General Commands Reference Guide T' (general_ref_t.pdf, page 235).

IProbe.FindAll — Find all specified entries in trace

See command **<trace>.FindAll** in 'General Commands Reference Guide T' (general_ref_t.pdf, page 237).

IProbe.FindChange — Search for changes in trace flow

See command **<trace>.FindChange** in 'General Commands Reference Guide T' (general_ref_t.pdf, page 238).

IProbe.Get — Display input level

See command **<trace>.Get** in 'General Commands Reference Guide T' (general_ref_t.pdf, page 242).

IProbe.GOTO — Move cursor to specified trace record

See command **<trace>.GOTO** in 'General Commands Reference Guide T' (general_ref_t.pdf, page 244).

IProbe.Init — Initialize trace

See command **<trace>.Init** in 'General Commands Reference Guide T' (general_ref_t.pdf, page 246).

IProbe.List — List trace contents

See command **<trace>.List** in 'General Commands Reference Guide T' (general_ref_t.pdf, page 248).

IProbe.ListNesting — Analyze function nesting

See command **<trace>.ListNesting** in 'General Commands Reference Guide T' (general_ref_t.pdf, page 263).

IProbe.ListVar — List variable recorded to trace

See command **<trace>.ListVar** in 'General Commands Reference Guide T' (general_ref_t.pdf, page 266).

IProbe.LOAD — Load trace file for offline processing

See command **<trace>.LOAD** in 'General Commands Reference Guide T' (general_ref_t.pdf, page 270).

IProbe.OFF — Switch off

See command **<trace>.OFF** in 'General Commands Reference Guide T' (general_ref_t.pdf, page 278).

IProbe.PROfile — Rolling live plots of trace data

See command **<trace>.PROfile** in 'General Commands Reference Guide T' (general_ref_t.pdf, page 282).

IProbe.PROfile.channel — Display profile of signal probe channels

See command **<trace>.PROfile.channel** in 'General Commands Reference Guide T' (general_ref_t.pdf, page 282).

IProbe.PROfileChart — Profile charts

See command **<trace>.PROfileChart** in 'General Commands Reference Guide T' (general_ref_t.pdf, page 283).

IProbe.PROTOcol — Protocol analysis

See command **<trace>.PROTOcol** in 'General Commands Reference Guide T' (general_ref_t.pdf, page 339).

IProbe.PROTOcol.Chart — Graphic display for user-defined protocol

See command **<trace>.PROTOcol.Chart** in 'General Commands Reference Guide T' (general_ref_t.pdf, page 339).

IProbe.PROTOcol.Draw — Graphic display for user-defined protocol

See command **<trace>.PROTOcol.Draw** in 'General Commands Reference Guide T' (general_ref_t.pdf, page 341).

IProbe.PROTOcol.EXPORT — Export trace buffer for user-defined protocol

See command **<trace>.PROTOcol.EXPORT** in 'General Commands Reference Guide T' (general_ref_t.pdf, page 342).

IProbe.PROTOcol.Find — Find in trace buffer for user-defined protocol

See command **<trace>.PROTOcol.Find** in 'General Commands Reference Guide T' (general_ref_t.pdf, page 343).

IProbe.PROTOcol.list Display trace buffer for user-defined protocol

See command **<trace>.PROTOcol.list** in 'General Commands Reference Guide T' (general_ref_t.pdf, page 344).

IProbe.PROTOcol.PROfileChart Profile chart for user-defined protocol

See command **<trace>.PROTOcol.PROfileChart** in 'General Commands Reference Guide T' (general_ref_t.pdf, page 347).

IProbe.PROTOcol.PROfileSTATistic Profile chart for user-defined protocol

See command **<trace>.PROTOcol.PROfileSTATistic** in 'General Commands Reference Guide T' (general_ref_t.pdf, page 348).

IProbe.PROTOcol.STATistic Display statistics for user-defined protocol

See command **<trace>.PROTOcol.STATistic** in 'General Commands Reference Guide T' (general_ref_t.pdf, page 350).

IProbe.REF Set reference point for time measurement

See command **<trace>.REF** in 'General Commands Reference Guide T' (general_ref_t.pdf, page 357).

IProbe.RESet Reset command

See command **<trace>.RESet** in 'General Commands Reference Guide T' (general_ref_t.pdf, page 357).

IProbe.SAVE Save trace for postprocessing in TRACE32

See command **<trace>.SAVE** in 'General Commands Reference Guide T' (general_ref_t.pdf, page 358).

IProbe.SelfArm — Automatic restart of trace recording

See command **<trace>.SelfArm** in 'General Commands Reference Guide T' (general_ref_t.pdf, page 362).

IProbe.SnapShot — Restart trace capturing once

See command **<trace>.SnapShot** in 'General Commands Reference Guide T' (general_ref_t.pdf, page 373).

IProbe.SPY — Adaptive stream and analysis

See command **<trace>.SPY** in 'General Commands Reference Guide T' (general_ref_t.pdf, page 374).

IProbe.STATistic — Statistic analysis

See command **<trace>.STATistic** in 'General Commands Reference Guide T' (general_ref_t.pdf, page 378).

IProbe.STREAMCompression — Select compression mode for streaming

See command **<trace>.STREAMCompression** in 'General Commands Reference Guide T' (general_ref_t.pdf, page 485).

IProbe.STREAMFILE — Specify temporary streaming file path

See command **<trace>.STREAMFILE** in 'General Commands Reference Guide T' (general_ref_t.pdf, page 486).

IProbe.STREAMFileLimit — Set size limit for streaming file

See command **<trace>.STREAMFileLimit** in 'General Commands Reference Guide T' (general_ref_t.pdf, page 487).

IProbe.TCount — Set trigger counter

See command **<trace>.TCount** in 'General Commands Reference Guide T' (general_ref_t.pdf, page 490).

IProbe.TDelay — Trigger delay

See command **<trace>.TDelay** in 'General Commands Reference Guide T' (general_ref_t.pdf, page 491).

IProbe.Timing — Waveform of trace buffer

See command **<trace>.Timing** in 'General Commands Reference Guide T' (general_ref_t.pdf, page 499).

IProbe.TRACK — Set tracking record

See command **<trace>.TRACK** in 'General Commands Reference Guide T' (general_ref_t.pdf, page 502).

IProbe.View — Display single record

See command **<trace>.View** in 'General Commands Reference Guide T' (general_ref_t.pdf, page 504).

IProbe.ZERO — Align timestamps of trace and timing analyzers

See command **<trace>.ZERO** in 'General Commands Reference Guide T' (general_ref_t.pdf, page 505).

ISTATistic

Instruction statistics

See also

- ISTATistic.ACCESS
- ISTATistic.ADD
- ISTATistic.Delete
- ISTATistic.EXPORT
- ISTATistic.Init
- ISTATistic.List
- ISTATistic.ListFunc
- ISTATistic.ListLine
- ISTATistic.ListModule
- ISTATistic.ListsYmbol
- ISTATistic.LOAD
- ISTATistic.METHOD
- ISTATistic.OFF
- ISTATistic.ON
- ISTATistic.RESet
- ISTATistic.SAVE
- ISTATistic.Set
- ISTATistic.state
- COVerage.EXPORT.JSON
- RTS

Overview ISTATistic

If the flat run-time analysis performed by the command **Trace.STATistic.sYmbol** shows that some functions are exceptionally time consuming, the command **ISTATistic.ListFunc** can be used to get more details on the run-time behavior of single instructions.

Additionally the command **ISTATistic.ListFunc** can be used to get a detailed code coverage analysis.

The information for the instruction statistics (ISTATistic) is kept in a database.

INCremental ISTATistic

[Example]

INCremental instruction statistics means: record a program section to the trace buffer first and then add the recorded program section to the instruction statistics database.

INCremental instruction statistics requires the following steps:

1. Configuration of the on-chip trace logic:

   - Switch the data trace off, if your on-chip trace logic supports data tracing.
   - Enable cycle-accurate tracing, if your on-chip trace logic supports cycle-accurate tracing. If cycle-accurate tracing is off, the run-time results on single instructions are less accurate since they need to be interpolated.

   Please be aware, that a big number of local **FIFOFULLs** might affect the result of the instruction statistic.

2. Trace configuration:

   - Advise TRACE32 PowerView to automatically clear the trace buffer (**Trace.AutoInit ON**) before the program execution is restarted. This setup avoids that the same trace contents is added twice to the instruction statistic database.
   - Inform TRACE32 about the core clock frequency with the command **Trace.CLOCK**.

- Set the trace to **Leash** mode. If the trace is working in **Leash** mode, the program execution is stopped as soon as the trace buffer is full.

3. Clear the instruction statistic database with **ISTATistic.RESet** command.
4. Start the program execution to sample the program flow into the trace buffer.
5. Execute the **ISTATistic.ADD** command to add the trace data to the database.
6. Display the results by using commands like **ISTATistic.ListFunc** or/and repeat step 4 - 5 until your done with your tests.

Example:

```
SYStem.CPU ARM1136JF                    ; core ARM1136JF

…                                       ; general setup for debugger

ETM.PortSize 16                         ; ETM setup
ETM.PortMode 1/2
ETM.DataTrace OFF
ETM.CycleAccurate ON

Trace.CLOCK 176.MHz                     ; inform TRACE32 about the core
                                        ; clock

Trace.Mode Leash                        ; switch trace to leash mode

Go                                      ; start program
                                        ; program stops automatically if
                                        ; trace is full

WAIT !STATE.RUN()

Trace.FLOWPROCESS                       ; transfer trace contents to host

PRINT %Decimal A.FLOW.FIFOFULL()        ; check the number of FIFOFULLs

ISTATistic.RESet                        ; clear the ISTAT database

ISTATistic.ADD                          ; add trace contents to ISTAT data
                                        ; base

ISTATistic.ListFunc                     ; display the results

…                                       ; continue with further
                                        ; measurements
```

[Example]

SPYing instruction statistics means: the trace information is streamed to a file on the host computer while recording. TRACE32 PowerView is able to display intermediate results.

SPYing instruction statistics requires the following steps:

1. Configuration of the on-chip trace logic:

   - Switch the data trace off, if your on-chip trace logic supports data tracing.
   - Enable cycle-accurate tracing, if your on-chip trace logic supports cycle-accurate tracing. If cycle-accurate tracing is off, the run-time results on single instructions are less accurate since they need to be interpolated.

   Please be aware that a big number of local **FIFOFULLs** might affect the result of the instruction statistic.

2. Trace configuration:

   - Inform TRACE32 about the core clock frequency with the command **Trace.CLOCK**.
   - Set the trace to **STREAM** mode. If the trace is working in **STREAM** mode, the trace information is streamed to a file on the host computer while recording.

3. Clear the instruction statistic database with **ISTATistic.RESet** command.

4. Intermediate results can be displayed if **ISTATistic.METHOD SPY** is selected.

5. Start the program execution. The results can be displayed while recording by using commands like **ISTATistic.ListFunc**.

Example:

```
SYStem.CPU ARM1136JF                    ; core ARM1136JF

…                                       ; general setup for debugger

ETM.PortSize 16                         ; ETM setup
ETM.PortMode 1/2
ETM.DataTrace OFF
ETM.CycleAccurate ON

Trace.CLOCK 176.MHz                     ; inform TRACE32 about the core
                                        ; clock

Trace.Mode STREAM                       ; switch trace to leash mode

ISTATistic.RESet

ISTATistic.METHOD SPY

ISTATistic.ListFunc                     ; display the results

Go                                      ; start program
                                        ; program stops automatically if
                                        ; trace is full

…
```

ISTATistic.ACCESS Define access path to program code for ISTAT

Format: **ISTATistic.ACCESS** *<path>*

<path>: **auto** | **VM** | **DualPort**

Default: auto.

auto	TRACE32 uses its own best practice procedure to read the program code.
VM	The program code is always read from the TRACE32 virtual memory.
DualPort	Advise TRACE32 to read the code via the run-time access to the target memory.

See also

- ISTATistic
- ISTATistic.state

[Examples]

Format:	**ISTATistic.ADD /***<option>* **ISTATistic.add** (deprecated) *<trace>***.ISTATistic.add** (deprecated)
<option>:	**FILE** | **NOTIME** | **NOFLOW** | **SAMPLES** | **BACK** | **FORE** **CORE** *<number>* | **MergeCORE** (SMP tracing only) **SplitCORE** | **MergeCORE** (SMP tracing only)

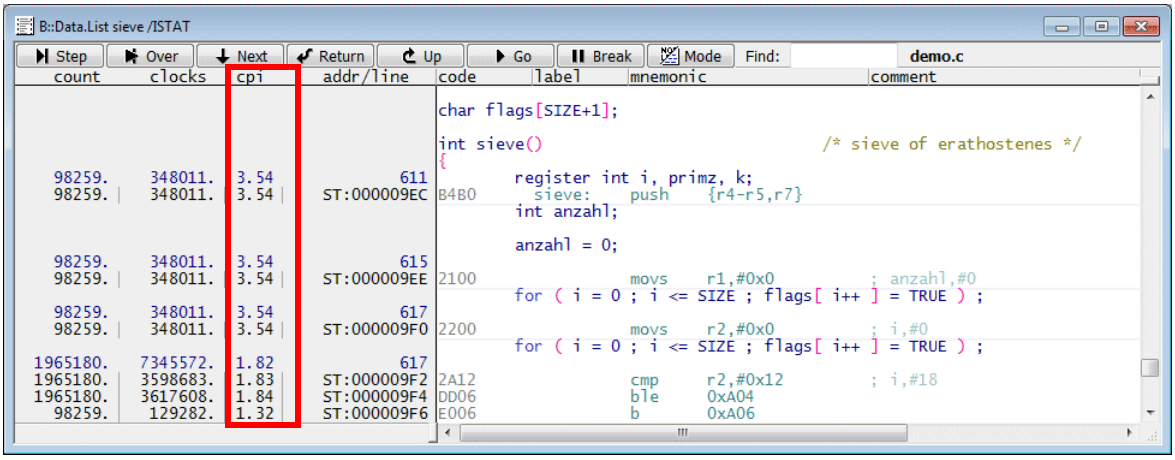
Adds the trace contents to the instruction statistic database.

FILE	Add the trace contents loaded by **Trace.FILE**.
NOTIME	Add the trace contents without timestamp information.
NOFLOW	Add the trace contents without the instruction flow.
SAMPLES	Add the recorded samples.
BACK	Use time distance to the previous record for the calculation of clocks and CPI (default: best fit).
FORE	Use time distance to the next record for the calculation of clocks and CPI (default: best fit).

<options> for SMP-specific tracing	
CORE *<number>*	Add the trace information generated by the specified core to the instruction statistic database.
MergeCORE	Add the trace information generated by all cores to the instruction statistic database (default).
SplitCORE	Same as MergeCORE.
JoinCORE	Same as MergeCORE.

Example for Cortex-M ETM trace

This example shows how to improve the precision of the CPI (Clock Per Instruction) analysis for the Cortex-M3/M4:



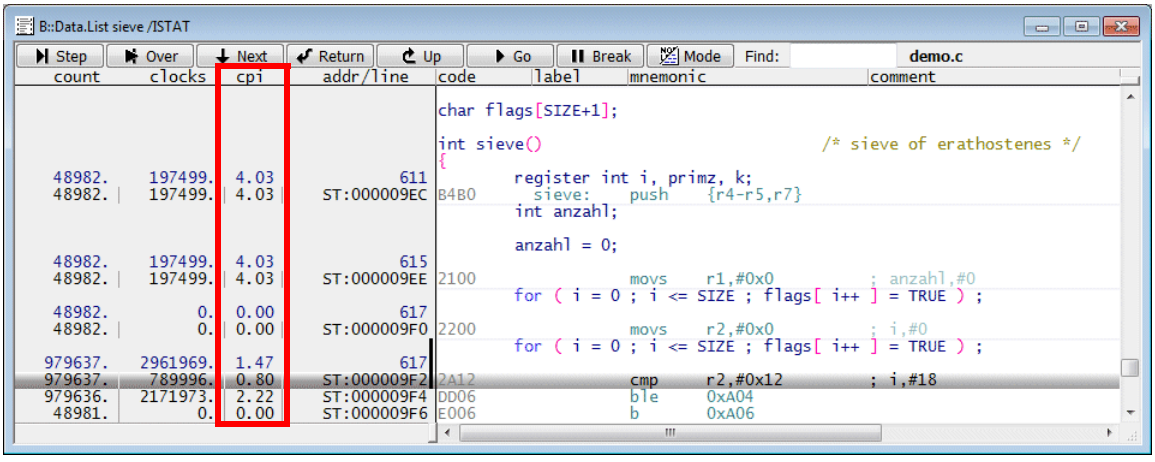
Since the ETM forwards the trace information to the TPIU as 32 byte blocks, the export of the trace information might be delayed especially when the core is not too busy. As a result the timestamp added by TRACE32 to the trace information at the time of recording might be inexact (see the screenshot above).

As an alternative a combination of the ETM instruction flow and the PC Sampler provided by the ITM can be used to get a more exact timestamp. To combine the ETM instruction flow and the PC Sampler proceed as follows:

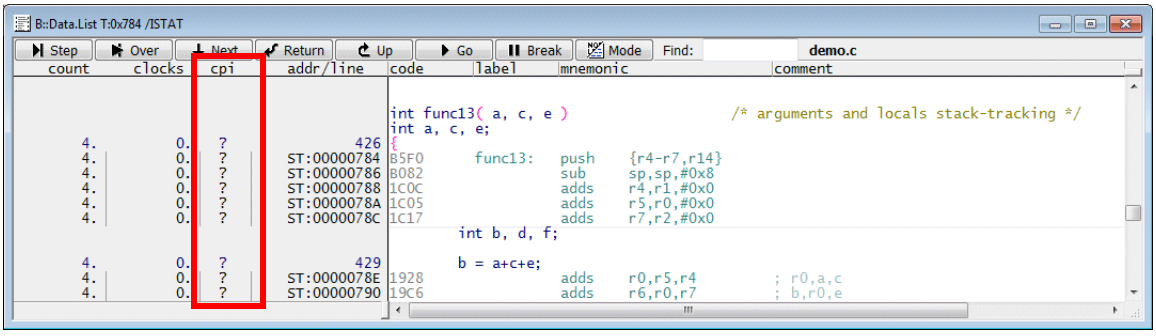
```
; configure the TRACE32 trace

; Trace is an alias for CAnalyzer (CombiProbe or µTrace (MicroTrace))
;Trace.METHOD CAnalyzer

; the program execution is stopped as soon as the trace is nearly full
Trace.Mode Leash

; inform TRACE32 about the target core clock
Trace.Clock 65.MHz

; configure the ETM

; enable the ETM to record the instruction flow
ETM.ON

; configure the ITM

; disable the ITM data trace
ITM.DataTrace OFF

; enable the ITM PC Sampler and specify that the current PC is exported
; all 128 clock cycles
ITM.PCSampler 1/128

; enable the ITM
ITM.ON
```

```
; fill the trace buffer

Go

Wait !RUN()

; evaluate the trace information

; add the instruction flow exported by the ETM without timestamp
; together with the PC information exported by the ITM to the instruction
; statistic database
ISTATistic.ADD

; display a detailed instruction statistic for the function sieve
Data.List sieve /ISTAT
```

The screenshot below shows the more accurate timestamps.



If the PC Sampler has not exported enough information for an instruction to calculate an exact result, a ? is displayed in the **CPI** column:



Example for combining ISTAT and SNOOPer

The instruction statistic database can be used to improve the timestamp information of an instruction execution (program flow) trace, if the processor architecture in use allows to read the PC while the program execution is running (PC snooping).

Please take care to ensure the following:

The same period of time has to be recorded into the Analyzer/Onchip trace (instruction execution) and into the SNOOPer trace (PC snooping). In most case it will be necessary to adjust the SNOOPer trace size. It is best to monitor both traces on the screen and stop the program execution as soon as one of the two traces is nearly full.

```
; reset the instruction statistic database
ISTATistic.RESet

; automatically erase the Analyzer trace memory before program execution
; is started
Analyzer.AutoInit ON

; configure the SNOOPer trace for PC snooping
SNOOPer.Mode PC

; configure the SNOOPer trace size
SNOOPer.SIZE Analyzer.SIZE()/1000.

; automatically erase the SNOOPer trace memory
; before program execution is started
SNOOPer.AutoInit ON

; open the Analyzer configuration window to monitor the trace recording
Analyzer.state

; open the SNOOPer trace configuration window to monitor the trace
; recording
SNOOPer.state

Go

; stop the trace recording when one of the traces is nearly full
Break

; add the instruction execution information from the Analyzer trace
; memory to the instruction statistic database, but ignore the time
; stamp
; add the PC and the timestamp information from the SNOOPer trace memory
; to the instruction statistic database
ISTATistic.ADD

; display a flat function runtime analysis based on the information in
; the instruction statistic database
ISTATistic.ListFunc

…
```

See also

■ ISTATistic ■ ISTATistic.state ■ COVerage.EXPORT.JSON

▲ 'Release Information' in 'Legacy Release History'

Format: **ISTATistic.Delete** *<range>* | *<address>*
<trace>.**ISTATistic.Delete** (deprecated)

Deletes the code coverage information for the specified addresses.

Example:

```
Data.List func17 /ISTAT COVerage                ; display code coverage
                                                ; details for func17

ISTATistic.Delete Var.RANGE(func17)             ; remove code coverage
                                                ; details for func17 from
                                                ; the ISTAT database

…                                               ; continue with your
                                                ; tests
```

See also

- ISTATistic
- ISTATistic.state

Using the **ISTATistic.EXPORT** commands, you can export instruction statistics for HLL functions, lines or modules to an XML file.

In addition, TRACE32 provides an XSL transformation template for formatting the XML file. The formatting is automatically applied to the XML file when it is opened in an external browser window. Prerequisite: The XSL file is placed in the same folder as the XML file.

For a demo script, see **ISTATistic.EXPORT.ListFunc**.

See also

- ISTATistic.EXPORT.CSV
- ISTATistic.EXPORT.ListFunc
- ISTATistic.EXPORT.ListLine
- ISTATistic.EXPORT.ListModule
- ISTATistic
- ISTATistic.state
- COVerage.EXPORT
- List.EXPORT

Format:	**ISTATistic.EXPORT.CSV** *<file>* [*<range>*] [**/***<option>*]
<option>:	**CORE** | **Sort** | **Track**

Exports instruction statistics information in CSV format.

<range>	Filter for exporting the specified address range or modules or program names.
<file>	Name of the CSV file that stores the instruction statistics. The file extension *.csv can be omitted.

For a description of the command line arguments and a PRACTICE script example, see **ISTATistic.EXPORT.ListFunc**.

See also

- ISTATistic.EXPORT
- ▲ 'Release Information' in 'Legacy Release History'

ISTATistic.EXPORT.ListFunc Export the HLL functions

[Example]

Format:	**ISTATistic.EXPORT.ListFunc** *<file>* *<range>* [**/Append**]

Exports instruction statistics for HLL functions to an XML file.

<range>	Filter for exporting the specified address range or modules or program names.
<file>	Name of the XML file that stores the instruction statistics. The file extension *.xml can be omitted.
Append	Appends the instruction statistics to an existing XML file - without overwriting the current file contents.

Example:

```
ISTATistic.ADD          ;update the instruction statistics database
ISTATistic.ListModule   ;display instruction statistics of all modules
ISTATistic.ListFunc     ;display instruction statistics of all functions
ISTATistic.ListLine     ;display instruction statistics of all
                        ;source lines

;export the instruction statistics for all modules of program "armle"
ISTATistic.EXPORT.ListModule  "~~/istatistic.xml"  \\armle

;export the instruction statistics for all HLL functions of the
;module "arm" and append to the existing file
ISTATistic.EXPORT.ListFunc    "~~/istatistic.xml"  \arm   /Append

;export the instruction statistics for all HLL lines of the
;function "sieve" and append to the existing file
ISTATistic.EXPORT.ListLine    "~~/istatistic.xml"  sieve  /Append

;for demo purposes: let's open the unformatted result in TRACE32
EDIT "~~/istatistic.xml"

;place the transformation template in the same folder as the XML file
COPY "~~/demo/coverage/single_file_report/t32transform.xsl" \
     "~~/t32transform.xsl"

;you can now open the formatted result in an external browser window
OS.Command start iexplore.exe "file:///C:/t32/istatistic.xml"
```

See also

- ISTATistic.EXPORT

ISTATistic.EXPORT.ListLine — Export the HLL lines

Format: **ISTATistic.EXPORT.ListLine** *<file> <range>* [**/Append**]

Exports instruction statistics for HLL lines to an XML file. For a description of the command line arguments and a PRACTICE script example, see **ISTATistic.EXPORT.ListFunc**.

See also

- ISTATistic.EXPORT

Format: **ISTATistic.EXPORT.ListModule** *<file>* *<range>* [**/Append**]

Exports instruction statistics for modules to an XML file. For a description of the command line arguments and a PRACTICE script example, see **ISTATistic.EXPORT.ListFunc**.

See also

- ISTATistic.EXPORT

ISTATistic.Init Initialize ISTAT database

Format: **ISTATistic.Init**
<trace>.**ISTATistic.Init** (deprecated)

Same functionality as **ISTATistic.RESet**.

See also

- ISTATistic
- ISTATistic.state

Format:	**ISTATistic.List** [*/<option>*]
<option>:	**CORE** *<number>* | **Sort** *<column>*
<column>:	**OFF** | **Address** | **Count** | **TIME** | **CLOCKS** | **RATIO** | **CPI**

Opens the **ISTATistic.List** window, displaying a run-time analysis for all or specified cores.

<column> For a description of the columns, see **ISTATistic.ListFunc**.

CORE *<number>* Displays a run-time analysis for the specified core. The analysis is based on the current contents in the ISTAT database.
The **CORE** option also affects the display of ISTAT information in the **Data.dump** and **List.*** windows.

Example 1:

```
ISTATistic.List /CORE 2.          ;display a run-time analysis for core 2

List.auto       /ISTAT            ;display a source listing and include
                                  ;ISTAT information for all cores
```

Example 2:

```
ISTATistic.List /CORE 2.          ;display a run-time analysis for core 2

List.auto       /CORE 2. /ISTAT   ;display a source listing and include
                                  ;ISTAT information only for the
                                  ;specified core 2
```

See also

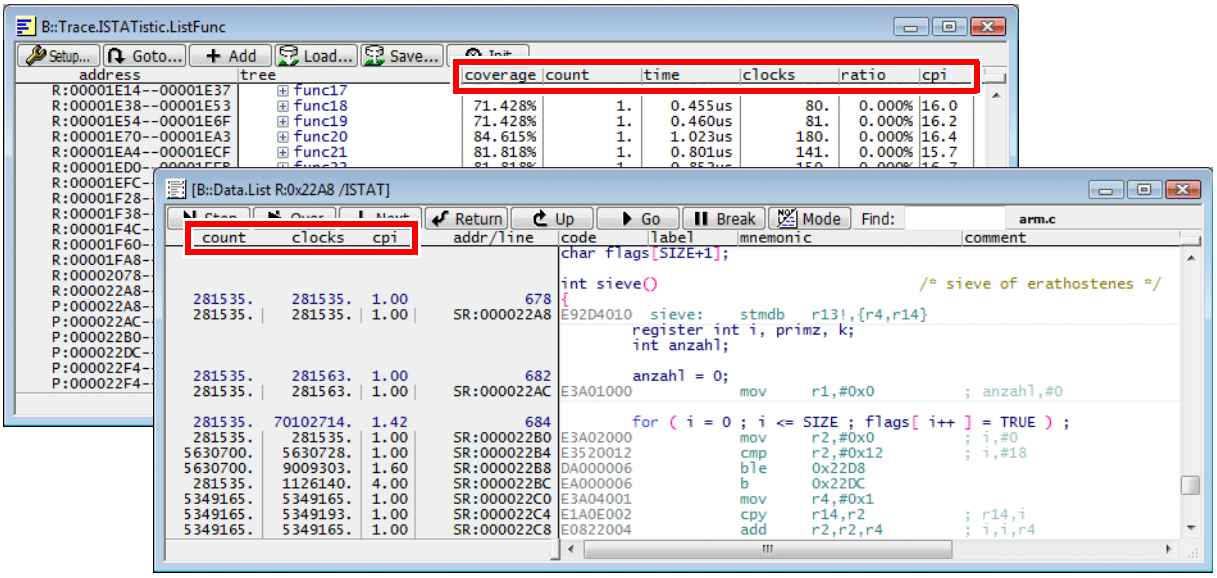
- ISTATistic
- ISTATistic.state

[Example]

Format: **ISTATistic.ListFunc** *<range>* | *<address>* [*/<option>*]
<trace>.**ISTATistic.ListFunc** (deprecated)

<option>: **CORE** *<number>*

Displays the function run-time details out of the ISTATistic database.



Run-time details on the single instructions of the function can be displayed by a double-click to the function or by the command **Data.List <address> /ISTAT**.

Description of Columns in the ISTATistic.ListFunc Window

coverage	Code coverage of the function (percentage)
count	Number of entries into the function range
time	Total time spent by the function
clocks	Total number of clocks spent in the function
ratio	Percentage of the total measurement time spent in the function
cpi	Average clocks per instruction (average value over all instructions of the function)

Example:

```
ISTATistic.ListFunc 0x2000

ISTATistic.ListFunc 0x1000--0x1fff
```

See also

- ISTATistic
- ISTATistic.state

ISTATistic.ListLine — List run-time analysis of HLL lines

Format: **ISTATistic.ListLine** *<range>* | *<address>* [**/***<option>*]
*<trace>***.ISTATistic.ListLine** (deprecated)

<option>: **CORE** *<number>*

Displays the high level line run-time details out of the ISTAT database.

See also

- ISTATistic
- ISTATistic.state

ISTATistic.ListModule — List module tree of ISTAT database

Format: **ISTATistic.ListModule** *<range>* | *<address>* [**/***<option>*]
*<trace>***.ISTATistic.ListModule** (deprecated)

<option>: **CORE** *<number>*

Displays the module run-time details out of the ISTAT database.

See also

- ISTATistic
- ISTATistic.state

Format: **ISTATistic.ListsYmbol** [*<range>* | *<address>*] [**/***<option>*]
*<trace>***.ISTATistic.ListsYmbol** (deprecated)

<option>: **CORE** *<number>*
Sort *<item>*
Track

Displays the flat run-time details for symbol regions out of the ISTAT database.

See also

- ISTATistic
- ISTATistic.state

ISTATistic.LOAD Load ISTAT database from file

Format: **ISTATistic.LOAD** *<file>*
*<trace>***.ISTATistic.LOAD** (deprecated)

Reloads the ISTAT database from the file to continue the analysis.

See also

- ISTATistic
- ISTATistic.state

Format: **ISTATistic.METHOD** *<option>*

<option>: **INCremental** | **SPY** | **RTS**

Sets the recording method for instruction statistics.

INCremental Incremental updated of instruction statistics database.
A program section is recorded to the trace buffer first and then added to the instruction statistics database (ISTAT database).

SPY Update of instruction database at regular intervals.
While trace data is being recorded, streaming to the host is interrupted at regular intervals in order to update the instruction statistics database (ISTAT database).
TRACE32 PowerView is able to display intermediate results.

RTS Real-time update of instruction database.
While trace data is being recorded, streaming to the host does NOT need to be interrupted to update the instruction statistics database (ISTAT database) with interim data.

See also

■ ISTATistic ■ ISTATistic.state

▲ 'Release Information' in 'Legacy Release History'

ISTATistic.OFF Deactivate the selected instruction statistics method

Format: **ISTATistic.OFF**

Instruction statistics data will not be recorded.

See also

■ ISTATistic ■ ISTATistic.state

Format: **ISTATistic.ON**

ISTATistic.ON switches the currently selected **ISTATistic.METHOD** on.

See also

- ISTATistic
- ISTATistic.state

ISTATistic.RESet Delete ISTAT database

Format: **ISTATistic.RESet**
<trace>.**ISTATistic.RESet** (deprecated)

Deletes the contents of the ISTAT database.

See also

- ISTATistic
- ISTATistic.state

ISTATistic.SAVE Save ISTAT database to file

Format: **ISTATistic.SAVE** *<file>*
<trace>.**ISTATistic.SAVE** (deprecated)

Saves the ISTAT database to a file. The file can be reloaded by the command **ISTATistic.LOAD** to continue with the analysis.

See also

- ISTATistic
- ISTATistic.state

Format: **ISTATistic.Set** *<range>* | *<address>*
<trace>.**ISTATistic.Set** (deprecated)

Marks the specified addresses as 1 time executed in the ISTAT database.

Example:

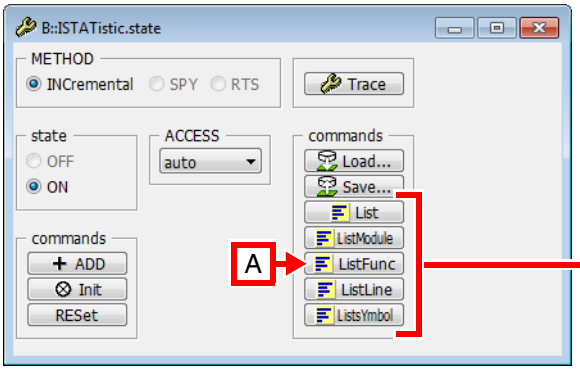
```
ISTATistic.RESet                          ; delete contents of ISTAT
                                          ; database

List.Mix func17 /ISTAT COVerage           ; display code coverage
                                          ; details for func17

ISTATistic.Set Var.RANGE(func17)          ; mark the instructions of
                                          ; func17 as 1 time executed
```

See also

- ISTATistic
- ISTATistic.state

Format:	**ISTATistic.state**

Opens the **ISTATistic.state** window, where you can configure the instruction analysis and display the resulting instruction statistics.



A For descriptions of the commands in the **ISTATistic.state** window, please refer to the **ISTATistic.*** commands in this chapter.
Example: For information about the **ListFunc** button, see **ISTATistic.ListFunc**.

See also

- ISTATistic
- ISTATistic.ACCESS
- ISTATistic.ADD
- ISTATistic.Delete
- ISTATistic.EXPORT
- ISTATistic.Init
- ISTATistic.List
- ISTATistic.ListFunc
- ISTATistic.ListLine
- ISTATistic.ListModule
- ISTATistic.ListsYmbol
- ISTATistic.LOAD
- ISTATistic.METHOD
- ISTATistic.OFF
- ISTATistic.ON
- ISTATistic.RESet
- ISTATistic.SAVE
- ISTATistic.Set
- RTS.OFF

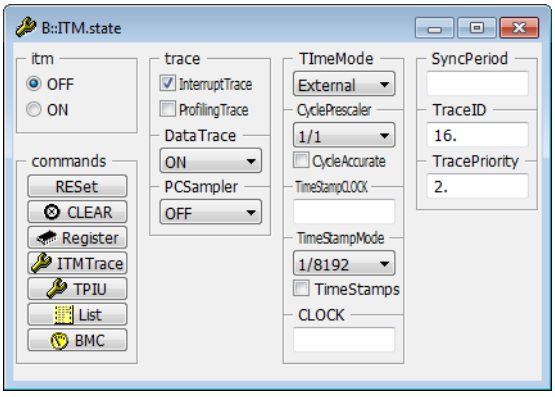
▲ 'Release Information' in 'Legacy Release History'

ITM CoreSight ITM (Instrumentation Trace Macrocell)

CoreSight ITM (Instrumentation Trace Macrocell) provides the following information:

- Address, data value and instruction address for selected data cycles
- Interrupt event information
- Program counter sampling for a pre-defined interval

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



To analyze the recorded trace data, use the **ITM<trace>** command groups.

See also

- ITM.CLEAR
- ITM.CLOCK
- ITM.CycleAccurate
- ITM.CycleMode
- ITM.CyclePrescaler
- ITM.DataTrace
- ITM.DataTraceCorrelateDistance
- ITM.DataTraceCorrelatePusher
- ITM.DWTADDRESS
- ITM.InterruptTrace
- ITM.OFF
- ITM.ON
- ITM.PCSampler
- ITM.PortFilter
- ITM.PortRoute
- ITM.PortSize
- ITM.ProfilingTrace
- ITM.Register
- ITM.RESet
- ITM.STALL
- ITM.state
- ITM.SyncPeriod
- ITM.TImeMode
- ITM.TimeStamp
- ITM.TimeStampCLOCK
- ITM.TimeStampMode
- ITM.TraceID
- ITM.TracePriority
- SystemTrace

▲ 'ITM<trace> - Trace Data Analysis' in 'General Commands Reference Guide I'
▲ 'Release Information' in 'Legacy Release History'

Format: **ITM.CLEAR**

Switches the ITM ON, clears the trace and resets the ITM registers to their default values.

See also

- ITM
- ITM.state

ITM.CLOCK Core clock frequency

Format: **ITM.CLOCK** *<core_clock>*

Used to calculate absolute timings out of cycle count packets (**ITM.CycleAccurate ON**).

See also

- ITM
- ITM.state

Format: **ITM.CycleAccurate** [**ON** | **OFF**]
ITM.CycleTrace (deprecated)

OFF	Trace information is time-stamped by TRACE32.
ON	Cycle count packets are inserted into the trace stream. Timestamps are calculated on base of • the cycle count information • and the core clock specified with the command • **ITM.CLOCK** *<core_clock>*.

Example:

```
ITM.Clock 50.MHz

ITM.CycleAccurate ON
```

See also

■ ITM ■ ITM.state

ITM.CycleMode Timestamp source

Format: **ITM.CycleMode** [**CORE** | **TPIU**]

Available only if SWV (Serial Wire Viewer) is present. Selects the clock source for the cycle count packets (**ITM.CycleAccurate ON**).

CORE (default)	Core clock.
TPIU	Clocked at TPIU baudrate speed.

See also

■ ITM ■ ITM.state

Format: **ITM.CyclePrescaler** [**1/1** | **1/4** | **1/16** | **1/64**]

Prescales the ATB clock which serves as an input to the ITM timestamp clock.

See also

- ITM
- ITM.state

Format:	**ITM.DataTrace ON** | **OFF** | *<data>*
<data>:	**Address** | **Data** **DataPC** | **OnlyPC** | **CorrelatedData**

The optional DWT (Data Watchpoint and Trace Unit) provides 4 data address comparators. The ITM can be configured to emit trace packets on data accesses that match one of these comparators.

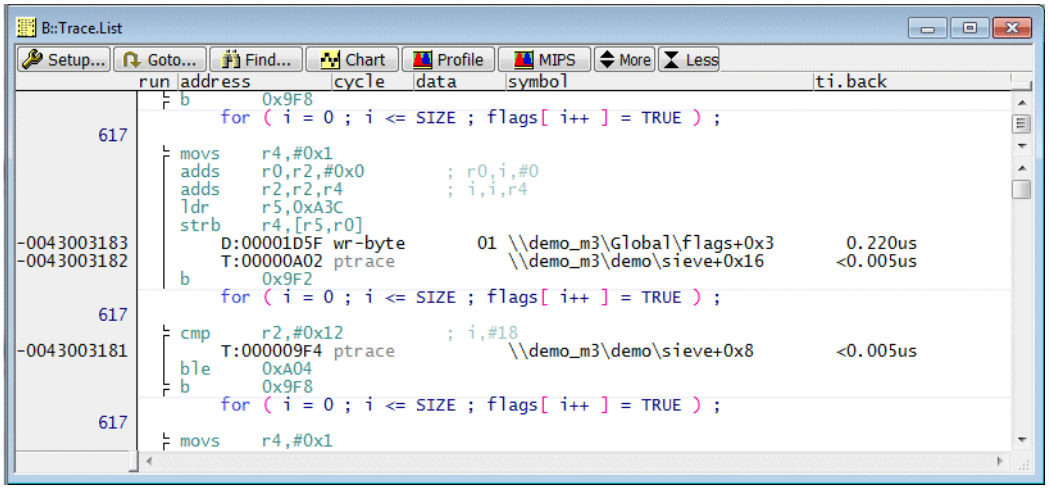
OFF	No information on accesses to data addresses that match a DTW comparator are emitted.
ON	If data address matches a DWT comparator: address and data value information on the data access are emitted by the ITM.
Address	If data address matches a DWT comparator: address information on the data access is emitted by the ITM.
Data	If data address matches a DWT comparator: data value information on the data access is emitted by the ITM.
DataPC	If data address matches a DWT comparator: address and data value information on the data access are emitted by the ITM. Additionally the address of the instruction that performed the data access is emitted.
OnlyPC	If data address matches a DWT comparator: address of the instruction that performed the data access is emitted by the ITM.
CorrelatedData	Emits the same information as DataPC, but if the command **Trace.List** is used, the information on the data access is merged into the ETM instruction flow display.

Examples:

```
Var.Break.Set flags[3] /TraceData     ; set a DWT data address comparator
                                      ; to flags[3]

ITM.DataTrace ON                      ; advise the ITM to emit address
                                      ; and data information on all
                                      ; accesses that match the
                                      ; comparator
```

```
Var.Break.Set flags[12] /TraceData     ; set a DWT data address comparator
                                       ; to flags[12]

ITM.DataTrace DataPC                   ; advise the ITM to emit address
                                       ; and data information on all
                                       ; accesses that match the
                                       ; comparator
                                       ; additionally the address of the
                                       ; instruction that performed the
                                       ; data access is emitted
```

```
Var.Break.Set flags[3] /TraceData      ; set a DWT data address comparator
                                       ; to flags[3]

ITM.DataTrace CorrelatedData           ; advise the ITM to emit address
                                       ; and data information on all
                                       ; accesses that match the
                                       ; comparator
                                       ; additionally the address of the
                                       ; instruction that performed the
                                       ; data access is emitted

Analyzer.List                          ; display ETM instruction flow

Trace.List                             ; combine the ETM instruction flow
                                       ; and the ITM data access
                                       ; information and display it
```



See also

- ITM
- ITM.state

[build 145315 - DVD 09/2022]

Format: **ITM.DataTraceCorrelateDistance** *<distance>*

Defines the maximum distance between packets from the ETM trace stream and related packets in the ITM trace stream.

The default value is 256. In some cases (usually when the trace port is running close to saturation), the distance can be larger.

Larger values affect the trace processing speed and make it more difficult to find unique matches to initially correlate the traces.

See also

- ITM
- ITM.state

ITM.DataTraceCorrelatePusher Push data cycles

[build 165344 - DVD 02/2024]

Format: **ITM.DataTraceCorrelatePusher** [**ON** | **OFF**]

When **ITM.DataTraceCorrelatePusher** is ON, it forces ITM packets to push data cycles when the correlated data trace is performed.

See also

- ITM
- ITM.state

ITM.DWTADDRESS Supply comparator values

Format: **ITM.DWTADDRESS** *<address1>* [*<address2>* *<address3>* *<address4>*]

Used for trace decoding only. This command tells the decoder which base address(es) to use for address offset packets when the ITM address comparators have not been configured with **ITM.DataTrace**.

See also

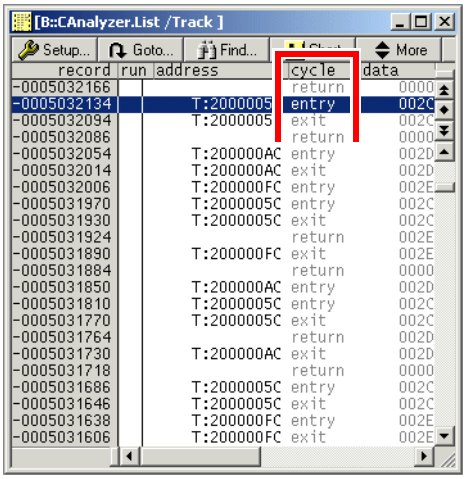
- ITM
- ITM.state

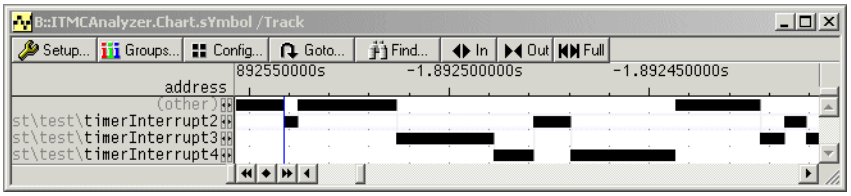
Format: **ITM.InterruptTrace** [**ON** | **OFF**]

Configures the ITM to emit interrupt event information.

Example:

```
ITM.InterruptTrace ON
Trace.METHOD CAnalyzer
Trace.List
Trace.Chart.sYmbol
```





Description of Values in the Cycle Column

entry	Interrupt entry
exit	Interrupt exit
return	Return to interrupted interrupt/function

See also

- ITM
- ITM.state

Format: **ITM.OFF**

Disables ITM functionality.

See also

- ITM
- ITM.state

ITM.ON Switch ITM on

Format: **ITM.ON**

Enables ITM functionality.

See also

- ITM
- ITM.state

ITM.PCSampler Emit PC at regular intervals

Format: **ITM.PCSampler OFF** | *<rate>*

<rate>: **1/64** | **1/128** | **1/256** | **1/1024** | **1/2048** | **1/4096** | **1/8192** | **1/16384** | **1/32768**

Configures the ITM to emit the program counter at regular intervals. The interval is specified in clock cycles.

```
ITM.PCSampler 1/256

ITMTrace.List

ITMTrace.STATistic.sYmbol BAR Ratio        ; all statistic commands that
                                           ; perform a flat analysis can
                                           ; be used
```

See also

- ITM
- ITM.state

Format:	**ITM.PortFilter** [*<value>* | *<mask>*]

Default: *<value>* = 0, *<maks>* = 0yxxxxxxxx

Displays output from the selected ITM channel only.

<value>	Display only ITM packets of channel #*<value>*
<mask>	Bitmask to display multiple channels. Format = '0ybbbbbbbb' with b = {0, cleared 1, set x}. don't care.

See also

- ITM
- ITM.state

ITM.PortRoute Selects the trace port

Format:	**ITM.PortRoute** *<port>*
<port>:	**AUTO** | **Analyzer** | **CAnalyzer**

Default: AUTO

Prepares the selected trace hardware for ITM trace capture.

AUTO	Automatic detection
Analyzer	PowerTrace (via TPIU)
CAnalyzer	Compact-Analyzer: CombiProbe or µTrace (MicroTrace)
Onchip	Onchip trace buffer (ETB, ETF or ETR)

See also

- ITM
- ITM.state

Format:	**ITM.PortSize SWV** | *<tpiu_portsize>*
<tpiu_portsize>:	**1** | **2** | **4** | **8** | **16** | **24** | **32**

Specifies the trace export interface and its size.

See also

- ITM
- ITM.state

ITM.ProfilingTrace Provide DWT counter information

Format:	**ITM.ProfilingTrace** [**ON** | **OFF**]

ON	Provides information about the selected DWT counter via the ITM trace.
OFF	Does not provide DTW counter information.

See also

- ITM
- ITM.state

Format: **ITM.Register** [*/<option>*]

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

Displays the ITM control registers.

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

See also

- ITM
- ITM.state

ITM.RESet Reset ITM settings

Format: **ITM.RESet**

Resets the setting in the **ITM.state** window to default and reset the ITM register.

See also

- ITM
- ITM.state

ITM.STALL Stall processor to prevent FIFO overflow

Format: **ITM.STALL** [**ON** | **OFF**]

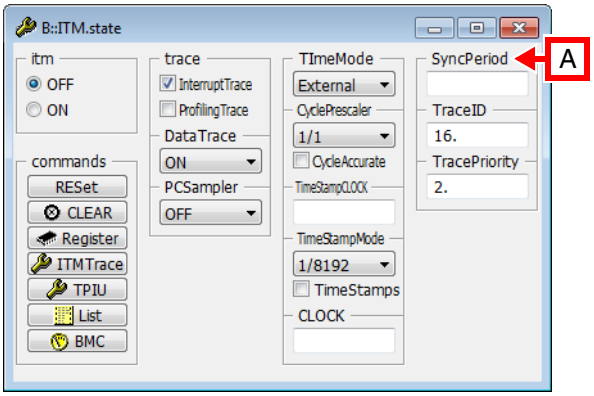
Allows the ITM to stall the processor to prevent a output FIFO overflow. If enabled, the trace will be no longer real time. This functionality is not supported by all Cortex-M cores.

See also

- ITM
- ITM.state

Format:	**ITM.state**

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



A For descriptions of the commands in the **ITM.state** window, please refer to the **ITM.*** commands in this chapter.
Example: For information about the **SyncPeriod** box, see **ITM.SyncPeriod**.

See also

- ITM
- ITM.CLOCK
- ITM.CycleMode
- ITM.DataTrace
- ITM.DataTraceCorrelatePusher
- ITM.InterruptTrace
- ITM.ON
- ITM.PortFilter
- ITM.PortSize
- ITM.Register
- ITM.STALL
- ITM.TImeMode
- ITM.TimeStampCLOCK
- ITM.TraceID
- ITM.CLEAR
- ITM.CycleAccurate
- ITM.CyclePrescaler
- ITM.DataTraceCorrelateDistance
- ITM.DWTADDRESS
- ITM.OFF
- ITM.PCSampler
- ITM.PortRoute
- ITM.ProfilingTrace
- ITM.RESet
- ITM.SyncPeriod
- ITM.TimeStamp
- ITM.TimeStampMode
- ITM.TracePriority

Format: **ITM.SyncPeriod** [*<packets>*]

Sets the number of regular ITM packets which will be output to the trace stream between two synchronization packets.

<packets> If omitted, then the default number of regular packets between synchronization packets is chosen by the debugger or the chip.

In this example, the number of regular ITM packets is 1024.

RP = regular ITM packet
SP = synchronization packet

See also

- ITM
- ITM.state

Format:	**ITM.TImeMode** *<mode>*
<mode>:	**OFF** **External** **AsyncTimeStamps** **CycleAccurate** **CycleAccurate+External** **PCSampler**

Defines which type of timing information will be generated for the current ITM trace data.

AsyncTimeStamps	Global timestamps from **ITM.TimeStamp** are used to interpolate a **CycleAccurate**-like trace.
CycleAccurate	Timestamps are provided by ITM. See **ITM.CycleAccurate**.
CycleAccurate+External	Timestamps are provided by ITM and trace hardware. The latter can be used to correlate ITM trace data to data of other trace sources in time.
External	Timestamps are provided by trace hardware.
OFF (default)	No timestamps.
PCSampler	The timestamps are based on PC sampling. The timestamp interval can be configured with **ITM.PCSampler**.

See also

- ITM
- ITM.state

Format: **ITM.TimeStamp** [**ON** | **1/128** | **1/8192** | **OFF**]

Only available on Cortex-M!

1/128	Generate global timestamp packets approximately every 128 cycles.
1/8192	Generate global timestamp packets approximately every 8192 cycles.
OFF (default)	Do not generate global timestamp packets.
ON	Generate global timestamp packet after every packet if output FIFO is empty.

See also

- ITM
- ITM.state

ITM.TimeStampCLOCK External clock frequency

Format: **ITM.TimeStampCLOCK** *<clock>*

Used to calculate absolute timings when timestamps are not derived from core clock (**ITM.CycleAccurate OFF** or **ITM.CycleMode TPIU**).

See also

- ITM
- ITM.state

Format: **ITM.TimeStampMode** [**ALL** | **1/128** | **1/8192**]

Only available on Cortex-M!

ALL (default)	Each ITM trace sample gets a timestamp.
1/128	The local timestamp counter is clocked approximately every 128 cycles.
1/8192	The local timestamp counter is clocked approximately every 8192 cycles.

See also

- ITM
- ITM.state

ITM.TraceID Set trace ID manually

Format: **ITM.TraceID** *<id>*

TRACE32 automatically assigns an appropriate source ID to the ITM (register ATBID). This command allows the user to specify a source ID for the ITM.

See also

- ITM
- ITM.state

ITM.TracePriority Set priority for the ITM manually

Format: **ITM.TracePriority** *<priority>*

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

See also

- ITM
- ITM.state

ITM<trace> Command groups for ITM<trace>

See also

- ITMAnalyzer
- ITMCAnalyzer
- ITMHAnalyzer
- ITMLA
- ITMOnchip
- ITMTrace

▲ 'ITM - Configuration of the Trace Source' in 'General Commands Reference Guide I'

Overview ITM<trace>

Using the **ITM<trace>** command group, you can configure the trace recording as well as analyze and display the recorded ITM trace data. The command groups consist of the name of the trace source, here **ITM**, plus the TRACE32 trace method you have chosen for recording the ITM 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 **"Related Trace Command Groups"** (general_ref_t.pdf).

Example:

```
ITMTrace.state            ;optional step: open the window in which the
                          ;trace recording is configured.
ITMTrace.METHOD Analyzer  ;select the trace method Analyzer for
;<configuration>           ;recording trace data.

ITM.state                 ;optional step: open the window in which
                          ;the trace source ITM is configured.
ITM.ON                    ;switch the trace source ITM on.
;<configuration>

;trace data is recorded using the commands Go, WAIT, Break

ITMAnalyzer.List          ;display the ITM trace data recorded with the
                          ;trace method Analyzer as a trace listing.

ITMTrace.List             ;this is the generic replacement for the above
                          ;ITMAnalyzer.List command.
```

Format:	**ITMAnalyzer.**<*sub_cmd*>

The **ITMAnalyzer** command group allows to display and analyze the information emitted by the CoreSight Instrumentation Trace Macrocell (ITM).

The ITM information emitted off-chip via the Trace Port Interface Unit (**TPIU**) is recorded by the TRACE32 PowerTrace.

<*sub_cmd*>	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 **ITMAnalyzer.List** refer to **<trace>.List**

See also

- ITM<trace>

ITMCAnalyzer Analyze ITM information recorded by TRACE32 CombiProbe

Format:	**ITMCAnalyzer.**<*sub_cmd*>

The **ITMCAnalyzer** command group allows to display and analyze the information emitted by the CoreSight Instrumentation Trace Macrocell (ITM).

The ITM information emitted off-chip via the Trace Port Interface Unit (**TPIU**) or via the Serial Wire Output (SWO) is recorded by the TRACE32 CombiProbe.

<*sub_cmd*>	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 **ITMCAnalyzer.List** refer to **<trace>.List**

See also

- ITM<trace>

Format: **ITMHAnalyzer.**<*sub_cmd*>

The **ITMHAnalyzer** command group allows to display and analyze the information emitted by the CoreSight Instrumentation Trace Macrocell (ITM). Trace data is transferred off-chip via the USB port and is recorded in the trace memory of the TRACE32 host analyzer.

<*sub_cmd*>	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 **ITMHAnalyzer.List** refer to **<trace>.List**

See also

- ITM<trace>

ITMLA Analyze ITM information from binary source

Format: **ITMLA.**<*sub_cmd*>

The **ITMLAnalyzer** command group allows to display and analyze the information emitted by the CoreSight Instrumentation Trace Macrocell (ITM). Trace data is collected from Lauterbach's Logic Analyzer or from a binary file.

<*sub_cmd*>	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 **ITMLAnalyzer.List** refer to **<trace>.List**

See also

- ITM<trace>

Format:	**ITMOnchip.**<*sub_cmd*>

The **ITMOnchip** command group allows to display and analyze the information emitted by the CoreSight Instrumentation Trace Macrocell (ITM).

The ITM information emitted to the target's onchip memory is read by TRACE32 via debug cable (JTAG).

<*sub_cmd*>	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 **ITMOnchip.List** refer to **<trace>.List**

NOTE:	If single-stepping through your code, make sure **TPIU.PortMode** is set to **Wrapped** or **Continuous.** Otherwise data may not be displayed correctly!

See also

- ITM<trace>

ITMTrace Method-independent analysis of ITM trace data

[Example]

Format:	**ITMTrace.**<*sub_cmd*>

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

<*sub_cmd*>	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 **ITMTrace.List** refer to **<trace>.List**

NOTE:	If **ITMTrace.METHOD** is **ONCHIP** and you are single-stepping through your code, make sure **TPIU.PortMode** is set to **Wrapped** or **Continuous**. Otherwise data may not be displayed correctly!

See also

- ITM<trace>