Debugger

Lauterbach TRACE32 Target Interface

Product Manual	Manual Date
DebuggerDebugger - Lauterbach Trace 32 Target Interface	02/99

Contents

Lauterbach TRACE32 Target Interface

Introduction

Another advanced feature of Debugger for the embedded systems development world is the ability to load different target components, which implements the interface with target systems. The TRACE32 Development System is introduced in this document. The TRACE32 is a product from **Lauterbach Datentechnik GmbH.**
Debugger supports the emulator (ICE) and the background debug mode (BDM, ICD) interface.
In this document, the specific features of the TRACE32 target component are described.
With this interface, an executable program from the Debugger environment can be downloaded to an external target system based on a Motorola MCU, which will execute it and give the feedback of the real target system behaviour to Debugger.
Debugger will fully supervise and monitor the MCU of the target system and control the CPU execution such as read and write in internal/external memory (even with the CPU running), single-step/run/stop processes in the CPU and the set of breakpoints in the code.

Note: **Unconcerned Components** *As the code is executed by an external MCU the memory statistics are not available with the TRACE 32 target component and Profiling, Coverage analysing and I/O Simulation will therefore not work with the TRACE 32 target component.*

General

In this chapter the specific features of the TRACE32 are described in detail. This includes a detailed description of the function and purpose of the Bus analyzer. The corresponding windows and dialog boxes are also explained together with the features they belong to.

TRACE32 The TRACE32 is an emulator system for MCUs, that provides emulation memory and a bus state analyzer.

Interfaces Before any communication between TRACE 32 and the Debugger is possible, the TRACE32 host driver program delivered from Lauterbach GmbH must be loaded.
For communication with the TRACE32 system, the Debugger uses the **Application Programming Interface** (**API**) delivered from Lauterbach GmbH. This interface (API) communicates via a socket interface with the TRACE32 host driver program. The TRACE32 host driver program just routes the API requests to the TRACE32 system. This can be done with an Ethernet, parallel or optical interface.

Interfacing Your System and the Target

First the TRACE32 host driver program from Lauterbach GmbH for the In Circuit Debugger or for the In Circuit Emulator must be installed. Be sure that the installation is correct for the used interface (Ethernet, parallel, optical...).

Preparing the TRACE32 driver The TRACE32 host driver program has to be configured for a usage with a remote control, such as the Debugger which uses itself the Application Programming Interface (API). To allow and set up a remote control, add the following lines to the `CONFIG.T32` file:

```
RCL=NETASSIST
PACKLEN=1024
PORT=20000
```

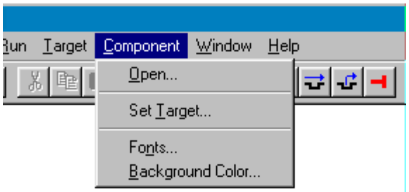
IMPORTANT: Make sure to have an empty line before “`RCL=NETASSIST`“.
The port number used here must correspond with the configuration of the UDP port in the Debugger.

Hardware Connection For the hardware connection, please refer to the ***TRACE32 Operation System Manual***.

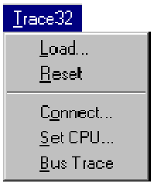
Loading the TRACE32 Target

Usually the target is set in the `PROJECT.INI` file, where `Target=Trace32`.

If the connection between the Debugger and the TRACE32 system is not already set, a *Load TRACE32 Driver* dialog pops up and you can set yourself the path and name of the TRACE32 host driver program and the UDP port. Please see the following section *Load TRACE32 Driver.*
If no target is set in the `PROJECT.INI` file or if a different target is set, you can load the TRACE32 target component selecting in the main menu *Component* | *Set Target...* as shown below and choose *Trace32* in the list of proposed targets.



After a successful target loading, the *Target* menu item is replaced by ***Trace32***.



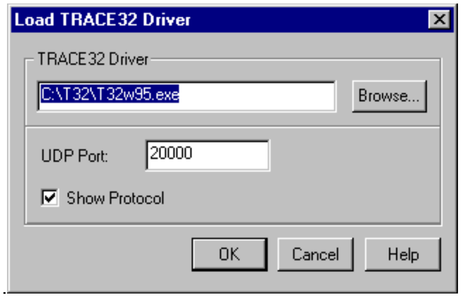
Load TRACE32 Driver program

While the Trace32 target component is loaded and the Trace32 Driver is not yet opened, the dialog shown below will pop up.

> ***Note:*** *This dialog can also be opened selecting **Trace32 | Connect...** when previous connection attempts with a TRACE32 failed or if the communication was lost.*

Make sure that the parameters on your host computer are correctly configured. Check that the used UDP port corresponds with the port specified in the `CONFIG.T32` file, otherwise no communication between the Debugger and the target is possible.

Load TRACE32 Driver If the TRACE32 Driver program is not yet opened and the Trace32 target is loaded, a dialog box *Load TRACE32 Driver* pops up in the Debugger as shown below.


Load TRACE32 Driver

TRACE32 Driver: C:\T32\T32w95.exe

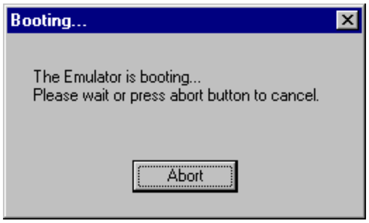
UDP Port: 20000

☑ Show Protocol

Enter the path and name of the TRACE32 Driver program in the *TRACE32 Driver* edit box or click on button *Browse* to find the program. The default path is `C:\T32\T32W95.EXE`.
Set the UDP port on the same value as specified in the `CONFIG.T32` file and click *OK*. The default UDP port is 20000.

> ***Note:*** *The name of the TRACE32 Driver and the UDP port which are saved through this dialog, override the environment variables* `DRIVER` *and* `PORT` *of the* `DEFAULT.ENV` *file.*

While the TRACE32 Driver is loading and the TRACE32 system is booting, the following dialog pops up. The boot process can last up to a minute and can be aborted by pressing the *Abort* button.

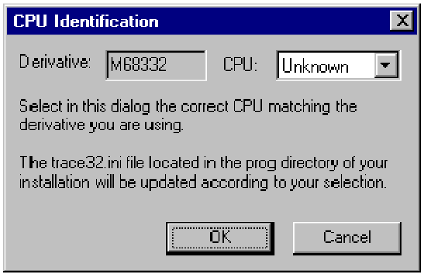


The Emulator is booting...
Please wait or press abort button to cancel.

The program name and the used UDP port are saved for later debug sessions. If no connection could be established, press *Abort* button, check if the used parameters are correct and try it again by choosing *Trace32 | Connect.*

Configuration of the CPU used by the debugger

The Lauterbach debugger returns a string which can be used to retrieve the CPU of the currently connected derivative. Usually this CPU identification is handled in the `TRACE32.INI` file which is located in the prog directory of your installation.
If the debugger can not retrieve the current CPU identification from the derivative string, the *CPU Identification* dialog is displayed.



Derivative: M68332 CPU: Unknown

Select in this dialog the correct CPU matching the derivative you are using.

The trace32.ini file located in the prog directory of your installation will be updated according to your selection.

This dialog allows to select the CPU corresponding to the string describing the derivative returned by the Lauterbach debugger. Use the *CPU* list box to select the correct CPU. When clicking *OK*, the selected CPU is set and the `TRACE32.INI` file is created/updated.
For more information regarding this dialog, please refers to section *Trace32 Menu Entries, CPU Identification* in this manual.

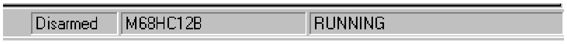
> ***Note:*** *This dialog can also be opened selecting* ***Trace32 | Set CPU...*** *if the currently set CPU does not match the current derivative or to update the Trace32.ini file.*

Configuration of the TRACE32 system

After the boot process, the TRACE32 system is configured by calling the `BOOT.CMD` file. The content of this file consists of commands from Lauterbach to correctly set the emulator. This file must be adapted depending on the MCU used. Examples of `BOOT.CMD` files are given in the *TRACE32 Boot Command File* section of this manual.

Note: *For details about the Configuration please see the TRACE32 manuals from Lauterbach GmbH.*

Thethe Debugger Debugger Status Bar for the TRACE32



When the TRACE32 Target Component has been loaded, specific information are given in the Debugger status bar (from left to right): the BUS analyzer mode, the MCU of the probe and the Debugger status.

Default Target Setup

As any other target, the TRACE32 target component can be loaded from the *Target* menu or can be set as a default target in the `PROJECT.INI` file which should be located in the current project directory.
Example of `PROJECT.INI` file.

```
[HI-WAVE]
Window0=Source      0   0  50  40
Window1=Assembly   50   0  50  40
Window2=Register   50  40  50  30
Window3=Memory     50  70  50  30
Window4=Data        0  40  50  25
Window5=Command      0  65  50  20
Window6=Module      0  85  50  15
Target=Trace32

[TRACE32]
PORT=20000
SHOWPROT=1
DRIVER=C:\T32\T32w95.exe (emulator driver)
or
DRIVER=C:\T32BDM\T32m12.exe (BDM, ICD driver)
```

Note: *For further information about the* `PROJECT.INI` *file please see the Debugger manual.*

TRACE32 Parameters

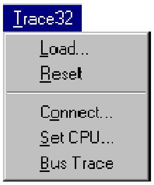
Under normal usage, these parameters are once interactively set at installation in the `PROJECT.INI` file and used in the future debugging sessions.

DRIVER This parameter specifies the path and the name of the TRACE32 Driver program. `C:\T32\T32W95.EXE` is the default driver program name for PCs.
Example: `DRIVER=C:\T32\T32w95.exe`

PORT The UDP Port used between the Debugger and TRACE32 Driver program. This port number must correspond with the port specified in the `CONFIG.T32` file.
Example: `PORT=20000`

SHOWPROT This variable is set to `0` or `1` (`1` to show the communication protocol, `0` to hide it) at the first communication attempt with the target. In the *Load TRACE32 Driver* dialog, you can check the *Show Protocol* checkbox to decide if you want to display the communication protocol in the Command Line component window. This option is saved under the `SHOWPROT` variable. Please see section *Load TRACE32 Driver program, Load TRACE32 Driver.*

Trace32 Menu Entries

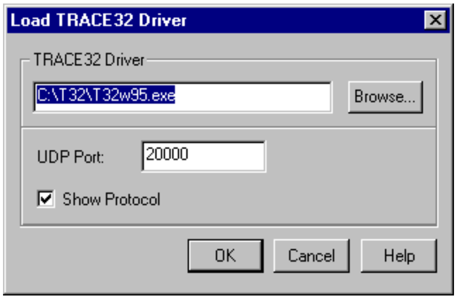


Reset

The *Trace32 | Reset* command resets the CPU and executes the `RESET.CMD` file, which must be located in the current project directory.

Connect to TRACE32

If *Trace32 | Connect* is selected, the TRACE32 system is initialized by calling the `BOOT.CMD` file. If there is no connection to the TRACE32 system, the following dialog is displayed.


Load TRACE32 Driver

TRACE32 Driver: C:\T32\T32w95.exe

UDP Port: 20000

☑ Show Protocol

TRACE32 Driver Enter the path and name of the TRACE32 Driver program in the *TRACE32 Driver* edit box or click on button *Browse* to find the program. The default path is `C:\T32\T32w95.exe`.

UDP Port Set the UDP port on the same value as specified in the `CONFIG.T32` file. The default UDP port is 20000.

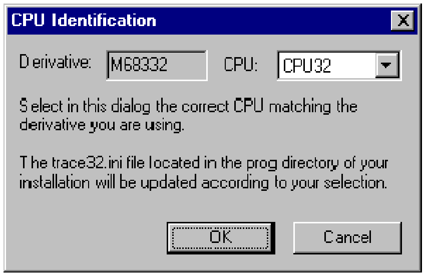
Show Protocol To report all the used TRACE32 API functions with there parameters in the command line window, check the check box *Show Protocol*.

> ***Note:*** *This feature is used by support personnel from Metrowerks.*

Please see section *Load TRACE32 Driver program, Load TRACE32 Driver.*

CPU Identification

The `TRACE32.INI` file is used to provide the information on the currently used CPU from a string describing the derivative returned by the Lauterbach Trace32 debugger.
If this information is corrupted or missing, select *TRACE32 | Set CPU...* to specify the CPU corresponding to the MCU you are using.
The *CPU Identification* dialog is displayed.



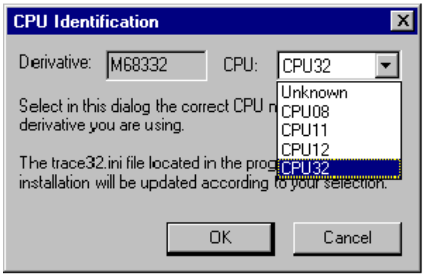
CPU Identification

Derivative: M68332 CPU: CPU32

Select in this dialog the correct CPU matching the derivative you are using.

The trace32.ini file located in the prog directory of your installation will be updated according to your selection.

This dialog allows to select the CPU corresponding to the string describing the derivative returned by the Lauterbach debugger.



Select from the CPU list box the CPU corresponding to the derivative you are using:

- `CPU08` for M68HC08 derivatives,
- `CPU11` for M68HC11derivatives,
- `CPU12` for M68HC12 derivatives,
- `CPU32` for M68K derivatives.

When clicking *OK*, the correct CPU is set in the debugger and the information is saved/updated in the `TRACE32.INI` file located in the prog directory of your installation.

Note: *This dialog is also opened if the trace32.ini file is missing or if the string returned from the Lauterbach device is no present in this file.*

Bus Tracing

Select *TRACE32 | Bus Trace* to run the Bus Analyser. For all details please see section *Bus Analyser* in this document.

Bus Analyzer

Introduction to the Bus Analyzer

Besides the emulation of the target system MCU, the most important feature that can be offered by a microcontroller development tool is an instrument to analyse program execution activities on the target MCU bus. This analyse allows the user to determine what is occurring in a system without actually affecting it.

> ***Note:*** *The Bus analyzer in the TRACE32 shows the logical state of the MCU bus. It does not show signal hold and setup times.*

Trace Modes If the bus analyzer has not been configured, the default settings are used. That means:

- The AutoArm option is enabled, i.e the analyzer will be armed automatically when the user program is started, and switched to off, after the program has been stopped.
- The FIFO operation mode is used, i.e the analyzer records the last cycles before stopping the recording.
- The SLAVE and PrePost mode are enabled, i.e the analyzer samples only cycles while the user program is running and the first and last cycle is always sampled, regardless of the trigger program.
- The size of the trace buffer is 32676 records.

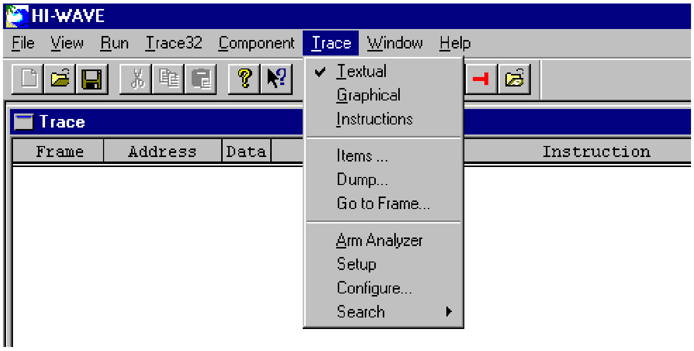
It is possible to set up the bus analyzer by using Command Line commands in the Command Line component window or in the `ANALYZER.CMD` file. This file is called when selecting *Trace* | *Setup*.

> ***Note:*** *For details about the Bus analyzer please see to the corresponding manuals of Lauterbach GmbH.*

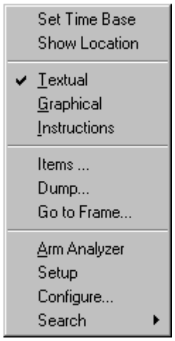
Textual or Graphical The Bus analyzer can display the data either textually or graphically. Use the horizontal and vertical scroll bars, as in other Windows applications, to move around the display of the trace buffer contents.

Using the Bus Analyzer

The Bus analyzer functions like any Debugger component, and has its own menu to control the features of the TRACE32 Bus analyzer hardware. This menu entry is called *Trace* in the Debugger main menu and the display window is also called *Trace*. Choose *Trace...* to find the menu shown below. This menu is also available as a pop-up menu in the Trace window when right-clicking.
From the user's perspective, using the Bus analyzer requires two steps. These steps are, collecting the desired bus data (running the program) and viewing the collected data.



The *Trace* window popup menu

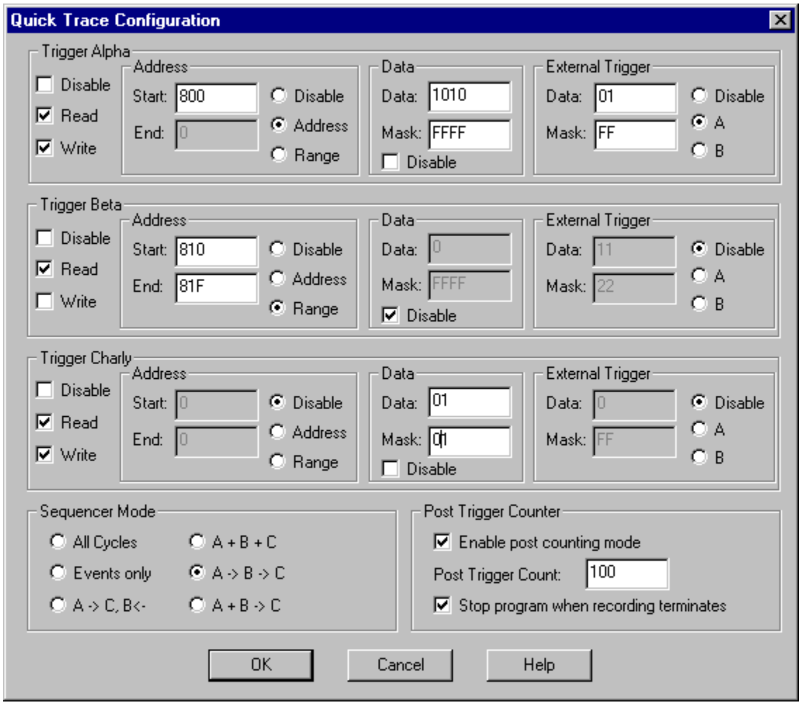


Configure the Bus Analyzer

The Trace32 Emulator allows to configure very complex triggers. This is done by writing a program which describes the trigger conditions and the behaviour of the trace logic. This program is then loaded into the Trace32.
Since this procedure needs some efforts to learn, the Trace32 target component contains a dialog which allows to configure the most common trigger functions interactively without having to know the full set of features. There are also several command line commands which will allow to configure all the items from the dialog also on the Debugger command line. For more complex trigger conditions, the TRACE32 host driver program can still be used in parallel to the Debugger.
However, if the Trace32 Bus Analyzer was not configured, all cycles are recorded.

Quick Trace Configuration

Select the *Trace* | *Configure...* Item to open the following dialog:



Quick Trace Configuration

Trigger Alpha
- [ ] Disable
- [✓] Read
- [✓] Write

Address: Start: 800 End: 0 — Disable [✓ Address] Range

Data: Data: 1010 Mask: FFFF — [ ] Disable

External Trigger: Data: 01 Mask: FF — Disable [✓ A] B

Trigger Beta
- [ ] Disable
- [✓] Read
- [ ] Write

Address: Start: 810 End: 81F — Disable Address [✓ Range]

Data: Data: 0 Mask: FFFF — [✓] Disable

External Trigger: Data: 11 Mask: 22 — [✓ Disable] A B

Trigger Charly
- [ ] Disable
- [✓] Read
- [✓] Write

Address: Start: 0 End: 0 — [✓ Disable] Address Range

Data: Data: 01 Mask: 01 — [ ] Disable

External Trigger: Data: 0 Mask: FF — [✓ Disable] A B

Sequencer Mode: All Cycles, Events only, A -> C, B<-, A + B + C, [✓ A -> B -> C], A + B -> C

Post Trigger Counter
- [✓] Enable post counting mode
- Post Trigger Count: 100
- [✓] Stop program when recording terminates

OK Cancel Help

Triggers There are three Trigger groups labelled *Trigger Alpha, Trigger Beta and Trigger Charly*. Each trigger can be disabled by checking the corresponding *Disable* checkbox. Read- and/or Write accesses of each Trigger can be specified by checking the *Read* and/or *Write* checkbox of the corresponding trigger group. The values entered in these trigger groups are interpreted as hexadecimal values.

Address Each trigger group contains an Address Item. There are three modes: If *Disable* is selected, the address Item is ignored in the trigger. When *Address* is selected, the address Item consists of a single address and if *Range* is selected, an address range is used as address Item in the corresponding trigger.

Data Each trigger group contains a Data Item. All bits of the Data field which corresponds to one in the Mask field are used in the corresponding trigger. If the *Disable* box is checked, the Data Item is ignored in the trigger.

External Trigger Each trigger group contains an External Trigger Item. An 8-bit value can be specified in the Data field that defines trigger input signals on the Trace32 for the trigger. Only the bits that corresponds to one in the Mask field are used in the trigger. *A* and *B* specifies the external trigger port used for the corresponding trigger. If *Disable* is selected, the external trigger Item is ignored in the trigger.

Sequencer Mode There are six different modes for the recording.
All cycles records all cycles continuously. Recording stops when the number of cycles specified in the *Post Trigger Counter* is reached, if enabled.
Events only records all cycles where a trigger triggers. Recording stops when the number of cycles specified in the *Post Trigger Counter* is reached, if enabled.
A -> C, <- B stops recording on cycles of two events in sequence, A then B, provided that third trigger B remains false. When the third trigger B occurs, the sequence starts again with the first trigger. This sequence can be used as two-event sequence by leaving trigger B disabled. If the *Post Trigger Counter* is enabled, recording stops after the number of cycles specified in the *Post Trigger Counter* is reached.
A + B + C stops recording after either trigger A or trigger B or trigger C. If the *Post Trigger Counter* is enabled, data storage ends after the number of cycles specified in the *Post Trigger Counter* is reached.
A -> B -> C stops recording after three events, A then B then C occurring in sequence. If the *Post Trigger Counter* is enabled, recording stops after the number of cycles specified in the *Post Trigger Counter* is reached.
A + B -> C stops recording on either of two events trigger A or trigger B then followed by trigger C. The sequencer can be simplified to involve fewer than three triggers by leaving trigger A or B disabled. If the *Post Trigger Counter* is enabled, data storage ends after the specified number of cycles in the *Post Trigger Counter.*

Post Trigger Counter If the *Enable post counting mode* box is checked, recording data ends after the value specified in *Post Trigger Count* is reached. The application is also stopped after recording if the *Stop program when recording terminated* box is checked.

Collecting Data

Arming the Analyzer If the AutoArm option is enabled (default option), the analyzer will be armed automatically when the user program is started.
In the other case, if the AutoArm option is disabled, the Bus analyzer has to be armed to collect data. Choose the entry *Trace | Arm Analyzer* to arm the Bus analyzer. After the Arm Analyzer entry was clicked, the entry changes to Disarm Analyzer.

Disarming the Analyzer If the AutoArm option is enabled (default option), the analyzer will be switched to off automatically, after the program has been stopped.
In the other case, if the AutoArm option is disabled, select the entry *Trace | Disarm Analyzer* to stop the analyzer. After the Disarm Analyzer entry was clicked, the entry changes to Arm Analyzer.
The status of the Bus analyzer is shown in the title bar of the analyzer window and in the status bar of the main window.

Start Emulation To begin emulation use the command *Run | Continue* in the Debugger or any other command which starts program execution. Emulation continues until it is stopped by either a breakpoint, a watchpoint or manually. When the emulation stops, the data in the Bus analyzer window is updated (see below).

Status Bar When the Bus analyzer is activated manually (if the AutoArm option is not used), the MBA status shown on the status bar at the bottom of the main window changes to ARMED, to indicate that the Bus analyzer is ready to begin to collect data. When emulation begins, the MCU status shown on the status bar changes to RUNNING. When the Bus analyzer collects data, the Bus analyzer state changes to ANALYZING.

Halt Data Collection To manually halt data collection, the user must select the submenu *Trace | Disarm Analyzer*.
Disarming the analyzer stops data collection without stopping emulation.

Halt Emulation if the AutoArm option is enabled (default option), data collection is halted. In the other case, data collection is halted, but the analyzer leaves armed. If the AutoArm- and SLAVE option is disabled, data collection is not halted.

Recording Bus Data There are two different modes to recording bus data. If the FIFO operation mode is used, the analyzer records the last cycles before stopping the recording. If the STACK operation mode is used, the analyzer stops recording, when the trace buffer is full.

Viewing Collected Data

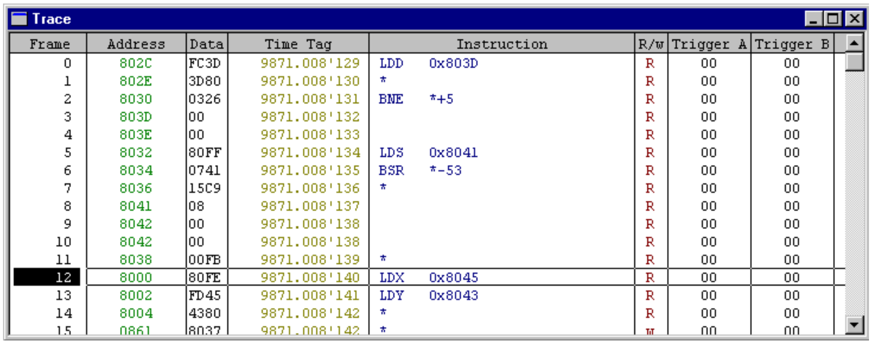
View Cycles

When the desired cycles have been collected, the Bus analyzer software provides a variety of methods to view those cycles. At this point, the trace buffer contains up to 32,767 stored frames. The highest numbered frames are usually the post-trigger frames. The lower-numbered frames are those frames stored before the trigger occurred, if any were stored.
When the Bus analyzer is deactivated (not in ANALYZING or ARMED state), the data is displayed in the *Trace* window.

Textual, Graphical or Instructions To specify select one in the *Trace* popup menu to set the display of the bus data.
The contents of the Bus analyzer can be displayed as text or as graphic. In the text representation, all frames or just the frames where an instruction starts, can be displayed. The user can also in the *Trace | Items*... dialog select the items which have to be displayed.

Textual Display

If the *Textual* format is chosen the software displays all the frames of the trace buffer contents in a textual form, as shown below. Use the scroll bar at the right to display other frames. Use the scroll bar at the bottom to display other signals.
The marker consists of two horizontal lines, which are used to mark a specific frame. The frame number of the marked frame is inverted. A line of the display corresponds to the data of a frame.
The Trace window contains the following basic items described below. It is possible to add or remove any item. Please see section *Adding / Removing Items in the Trace Window.*



Frame	Address	Data	Time Tag	Instruction	R/w	Trigger A	Trigger B
0	802C	FC3D	9871.008'129	LDD 0x803D	R	00	00
1	802E	3D80	9871.008'130	*	R	00	00
2	8030	0326	9871.008'131	BNE *+5	R	00	00
3	803D	00	9871.008'132		R	00	00
4	803E	00	9871.008'133		R	00	00
5	8032	80FF	9871.008'134	LDS 0x8041	R	00	00
6	8034	0741	9871.008'135	BSR *-53	R	00	00
7	8036	15C9	9871.008'136	*	R	00	00
8	8041	08	9871.008'137		R	00	00
9	8042	00	9871.008'138		R	00	00
10	8042	00	9871.008'138		R	00	00
11	8038	00FB	9871.008'139	*	R	00	00
12	8000	80FE	9871.008'140	LDX 0x8045	R	00	00
13	8002	FD45	9871.008'141	LDY 0x8043	R	00	00
14	8004	4380	9871.008'142	*	R	00	00
15	0861	8037	9871.008'142	*	W	00	00

Frame Is the cycle or the frame number 1 - 32768, that identifies the frame. The most recently stored frame is frame 32768 (or the highest-numbered frame that has been stored when fewer than 32768 frames have been stored).

Address Is the address bus value stored in the frame, displayed as hexadecimal digits. This is the address on the address bus when the frame is strobed into the trace buffer.

Data Is the data bus value stored in the frame, displayed as four hexadecimal digits. This is the value on the data bus when the frame is strobed into the trace buffer.

Time Tag Contains a representation of the time tag count stored when the frame is strobed into the trace buffer. The time tag is displayed as a number of seconds or fractions of seconds.

Instruction If the frame contains an instruction start, the instruction is displayed.

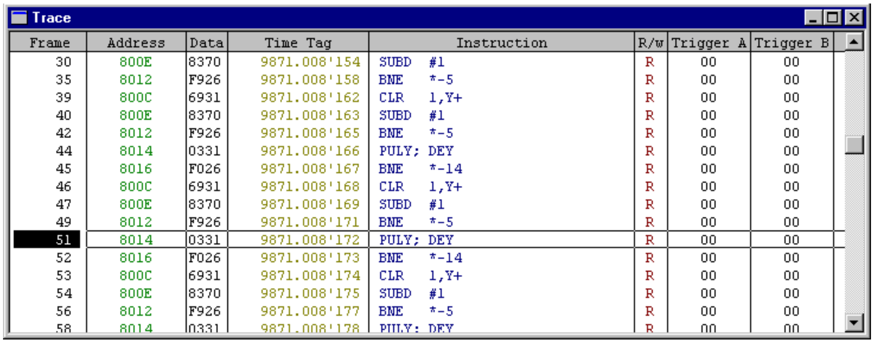
Trigger A/B Contains a representation of the external triggers A and B.

Time Measuring to reference a frame with a tag value of zero click on an entry and simultaneously type the key 'Z' on the keyboard. All other time tag values are now displayed relative to the bus cycle of this frame.

Control Signals The remaining fields in the *Trace* dialog box contain the values of the control signals.

Instructions Only Display

If the *Instructions only* format is chosen the software only displays those frames of the trace buffer contents in a text format where an instruction starts.



Trace

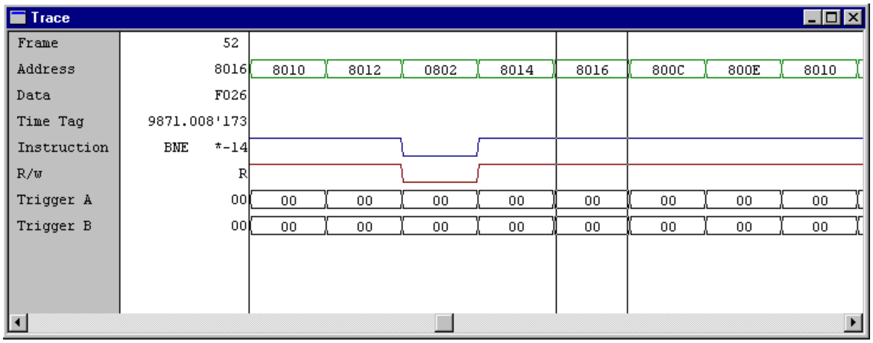
Frame	Address	Data	Time Tag	Instruction	R/w	Trigger A	Trigger B
30	800E	8370	9871.008'154	SUBD #1	R	00	00
35	8012	F926	9871.008'158	BNE *-5	R	00	00
39	800C	6931	9871.008'162	CLR 1,Y+	R	00	00
40	800E	8370	9871.008'163	SUBD #1	R	00	00
42	8012	F926	9871.008'165	BNE *-5	R	00	00
44	8014	0331	9871.008'166	PULY; DEY	R	00	00
45	8016	F026	9871.008'167	BNE *-14	R	00	00
46	800C	6931	9871.008'168	CLR 1,Y+	R	00	00
47	800E	8370	9871.008'169	SUBD #1	R	00	00
49	8012	F926	9871.008'171	BNE *-5	R	00	00
51	8014	0331	9871.008'172	PULY; DEY	R	00	00
52	8016	F026	9871.008'173	BNE *-14	R	00	00
53	800C	6931	9871.008'174	CLR 1,Y+	R	00	00
54	800E	8370	9871.008'175	SUBD #1	R	00	00
56	8012	F926	9871.008'177	BNE *-5	R	00	00
58	8014	0331	9871.008'178	PULY; DEY	R	00	00

Graphical Display

The figure below shows the graphical display of the Bus analyzer data. The graphical representation of the Bus analyzer data gives a better overview than the textual one. Switching between the different display formats can easily be done at any time by selecting the menu entries in the *Trace* menu. The leftmost sections of the Figures above and below display a textual description of the current frame with information about frame number, events, values on the data and address bus, time tag value, etc...

Zoom In or Out In the graphical display it is possible to 'zoom in' or to 'zoom out' either to see more details or to get a better general view. *Zoom in* / *Zoom out* are displayed in the *Trace* popup menu.

To zoom in select *Trace* | *Zoom In* or type 'I' on the keyboard.

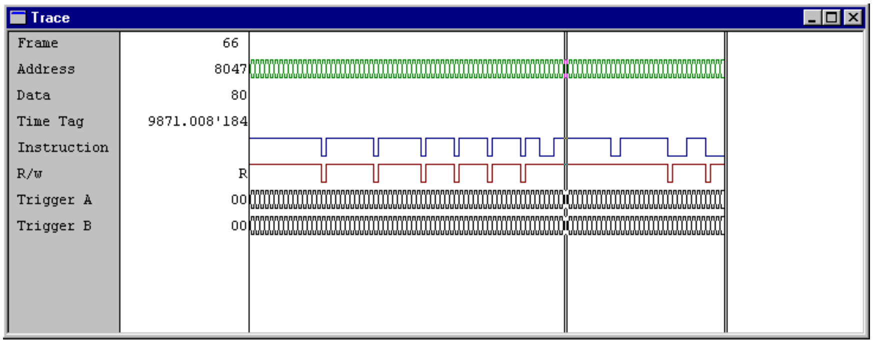


Trace

Frame	52
Address	8016
Data	F026
Time Tag	9871.008'173
Instruction	BNE *-14
R/w	R
Trigger A	00
Trigger B	00

Figure above shows the graphical display of the Bus analyzer data which is zoomed in and therefore shows more detail.

To zoom out select *Trace* | *Zoom Out* or type *O* on the keyboard.

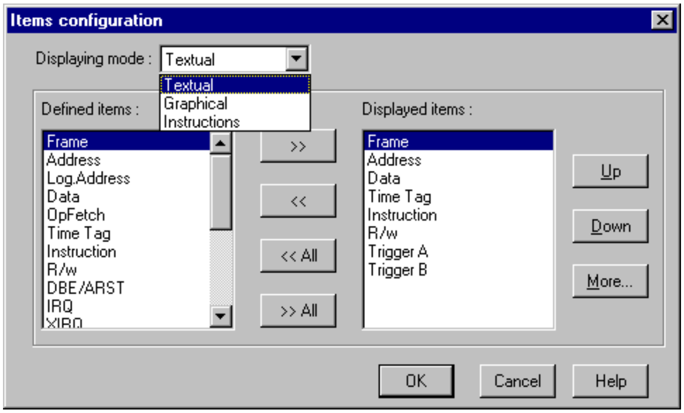


Dragging the marker over the display of the Bus analyzer data, may also generate updates in component windows, e.g. the source and assembly windows. Thus the Bus analyzer data can be viewed and thoroughly examined in the continually updated windows as the marker is moved with the help of the cursor over the graphic or text display of the Bus analyzer data. The vertical bar indicates the position of the marker and displays all the information of the current frame.

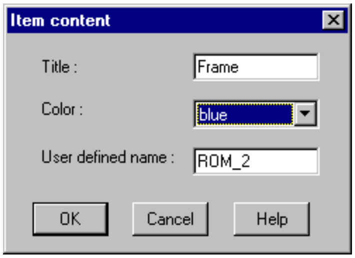
ShowLocation If the option *Trace* | *ShowLocation* check box is checked in the Bus Analyzer menu, the marker can be positioned with a left mouse click and the source and assembly windows are automatically updated as well.
If the option *Trace* | *ShowLocation* check box is *not* checked, the source and assembly windows are only updated, if the marker is dragged with a right mouse click.

Add / Remove Items in the *Trace* Window

The *Trace* window can be supplied with new items. In the default configuration, the following items are displayed in the window: *Frame*, *Events*, *Address*, *Data*, *Time Tag*, *Instruction* and *R/W*. It is possible to add new kinds of items (predefined and given in a list box in this dialog) and to remove them. It is possible to permute items in text or graphic mode simply dragging them into the window.



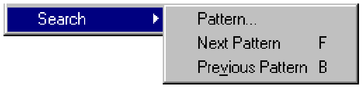
To edit an item, to set its colour or to give a specific name select *More*.



Title : Frame
Color : blue
User defined name : ROM_2

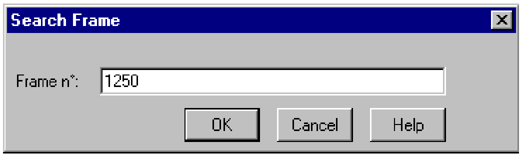
Scrolling the Display

Scrolling can be done in different ways. The scrollbars can be used as in Windows applications or the display can be scrolled to a specific trace buffer frame. It is also possible to search for a specific pattern (see below).



Search for a Frame

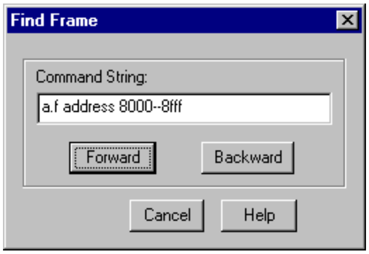
Select *Trace* | *Go to Frame*... to search for an occurrence of a frame and enter the desired frame number. If a frame with this number is found, the frame is selected. If the selected frame is not visible, the Bus analyzer window is scrolled to make the frame visible. If no frame can be found an error message is shown in a modal dialog box.



Frame n°: 1250

Search for a Pattern

To search for a frame with a specific pattern define the search pattern to use in order to find all stored bus cycles that match the pattern. The pattern can be defined as string which consists of a TRACE32 command, e.g *analyzer.find, address 100--200 data.b 55* will search for matching address range and data byte. Select *Trace* | *Search*|*Pattern*...



The Bus analyzer will look at specific collected cycles, matches the specific frame that shows the pattern and display this area in the window of the Debugger.

Cancel Button To remove the dialog box without any parameter being changed, select *Cancel*.

Forward Button To search forwards select *Forward* and a search is done to find the first frame which matches the specified pattern.

Backward Button To search backwards select *Backward* and a search is done to find the last frame which matches the specified pattern.

Next Pattern To search for the next occurrence of the pattern select the entry *Trace* |*Search* | *Next Pattern*. If the pattern is not found, an error message is shown in a modal dialog box.

Previous Pattern Select *Trace* |*Search* | *Previous Pattern* to search for the previous occurrence of the pattern. If the pattern is not found, an error message is shown in a modal dialog box.

Commands in the Run Menu in the Debugger

These menu commands control the program flow and are equivalent to the commands found in the procedure menu.

Continue Continues execution of a halted program until it reaches a breakpoint or until a run time error is detected. Execution also continues until a program is halted by the *Halt* command.

Halt This command stops the target processor and allows to examine the state of the application program.

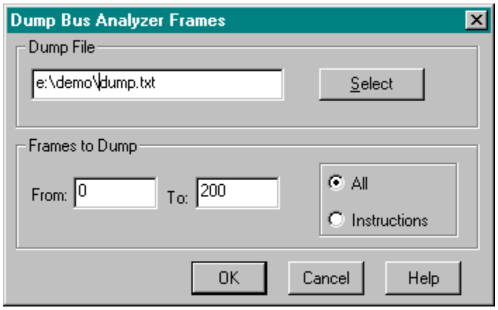
Step If the application is halted, this command performs a single step on source level, i.e. execution continues until the next source reference is reached.

Flat Is very similar to single stepping, but *no* stepping into called procedures.

Asm Step A halted program continues for one machine instruction from the point it was halted. This command is the same command as the single step command in the assembly window.

Dumping the Bus analyzer data to a file

To dump the Bus analyzer data to a file choose *Trace* | *Dump*...


Dump Bus Analyzer Frames

Dump File: e:\demo\dump.txt

Frames to Dump: From: 0 To: 200 [● All] Instructions

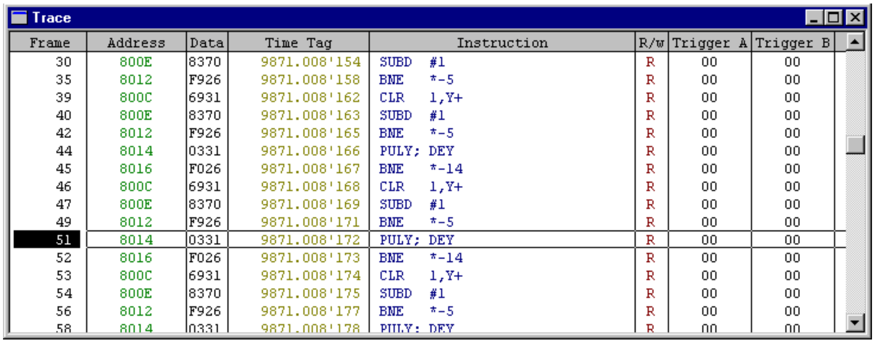
Dump File The Dump... command creates a modal dialog box which allows to select frames and dump them to a file. Only the items actually displayed in the Bus analyzer window are dumped. The group box *Dump File* contains an edit field where the name of the file to which the data has to be written, can be entered.

Select When the button *Select* is pressed, a standard file select box is opened and the selected file is than displayed in the edit field.

Frames to Dump The elements in the group box *Frames to Dump* allow to select the frames that have to be dumped to the file.
Just the frames within the range defined by the two edit fields *From:* and *To:* are dumped to the file.

All To dump all the frames within the selected range to the file, select *All*.

Instructions To dump just the frames where an instruction starts and this instruction is in the selected range, select *Instructions*.


Trace

Frame	Address	Data	Time Tag	Instruction	R/w	Trigger A	Trigger B
30	800E	8370	9871.008'154	SUBD #1	R	00	00
35	8012	F926	9871.008'158	BNE *-5	R	00	00
39	800C	6931	9871.008'162	CLR 1,Y+	R	00	00
40	800E	8370	9871.008'163	SUBD #1	R	00	00
42	8012	F926	9871.008'165	BNE *-5	R	00	00
44	8014	0331	9871.008'166	PULY; DEY	R	00	00
45	8016	F026	9871.008'167	BNE *-14	R	00	00
46	800C	6931	9871.008'168	CLR 1,Y+	R	00	00
47	800E	8370	9871.008'169	SUBD #1	R	00	00
49	8012	F926	9871.008'171	BNE *-5	R	00	00
51	8014	0331	9871.008'172	PULY; DEY	R	00	00
52	8016	F026	9871.008'173	BNE *-14	R	00	00
53	800C	6931	9871.008'174	CLR 1,Y+	R	00	00
54	800E	8370	9871.008'175	SUBD #1	R	00	00
56	8012	F926	9871.008'177	BNE *-5	R	00	00
58	8014	0331	9871.008'178	PULY; DEY	R	00	00

OK Button When the *OK* button is pressed, the dialog box is removed and the Bus analyzer data is dumped to the file.

Cancel Button When the *Cancel* button is pressed, the dialog box is removed without any data being dumped to a file.

Note: *For more details about the Bus Analyser, please refer to* ***TRACE32 State Analyzer User's Guide (version 96/01/08)*** *from* ***Lauterbach Datentechnik GmbH*** *.*

TRACE32 Boot Command File

The `BOOT.CMD` command file is needed to configure the TRACE32 system and is started while the boot process. The boot command file contains a basic configuration and can be adapted if necessary. The commands listed below might be present in this file. This is not an exhaustive list, other might also be present.

`system.up` or `system.mode aloneint`
Before any communication to the TRACE32 system is possible, the system must be powered up. The operating mode is set automatically according the available signals. `system.mode aloneint` should be used when using an emulator with only the probe.

`system.access cpu`
The access to the emulation probe must be enabled. The keyword used after *access* depends on the used emulation probe.

`map.def 0200--0ffff`
Through this command, memory is mapped from address `0x200...0xFFFF`. The slow mapper mode is selected and the memory is mapped intern. Do not map memory to the register map, this may cause problems with the IPRU.

`map.opfetch 08000--08fff`
The *opfetch* command defines program area. For the emulator it is necessary to know what are program fetches and what are data fetches. This command depends on the used emulation probe.

`map.ram 0200--0ffff`
Through this command, memory is mapped from address `0x200 to 0xFFFF`.

`map.intern`
Using this command, the whole address range is mapped to internal, the emulation memory is used.

`system.option mapgap on`
This command can be used if necessary to map the locations with no internal ram or rom while using the MCU embedded memory.

An example for a `BOOT.CMD` file used for the emulator with a HC08 probe (M68HC08AZ):

```
SYSTEM.Option MAPGAP ON
SYSTEM.Mode aloneint

;Disable Watchdog for the M68HC08AZ probe
Data.Set D:1F 1
```

An example for a `BOOT.CMD` file used for the emulator with a HC11 probe:

```
system.up
map.ram 0000--00fff
map.ram 01040--0ffff
map.intern
```

An example for a `BOOT.CMD` file used for the emulator with a HC12 probe:

```
system.up
system.access cpu
map.def 0200--0ffff
map.opfetch 08000--08fff
```

An example for a `BOOT.CMD` file used for the HC12 ICD/BDM:

```
system.cpu m68HC12B
system.up
```

An example for a `BOOT.CMD` file used for the emulator with a M68K probe (M68332):

```
; system up
system.mode aloneint
; map emulation memory
map.def 0--0ffff
map.opfetch 01000--03fff
map.ram 04000--043ff
; disable watchdog
data.set 0fffa21 40
; define pinassignment
data.set 0fffa44 %w 1557
data.set 0fffa46 %w  155
```

An example for a `BOOT.CMD` file used for the M68K ICD/BDM:

```
system.cpu M68332
system.up
```

> ***Note:*** *For details about the Configuration please see the TRACE32 manuals from Lauterbach GmbH.*

TRACE32 Target Startup File

The startup command file `STARTUP.CMD` is executed by the Debugger immediately after the TRACE32 target driver has been loaded. This file must be located in the current project directory. Any Debugger command can be used in this file and advantage taken of the wide set of commands introduced in the *Debugger* manual.
Example of a `startup.cmd` file content:

```
wb 0x0035   0x00
wb 0x0012   0x11
```

TRACE32 Reset Command File

To execute the TRACE32 reset command file `RESET.CMD` select *Reset* in the *`Trace32`* menu. This file must be located in the current project directory. Any Debugger command can be used in this file.

TRACE32 Analyzer Setup File

To setup the Bus analyzer, the `ANALYZER.CMD` file can be used. To execute the analyzer command file, select *Setup* in the *Trace* menu. This file must be located in the current project directory. Any Debugger command can be used in this file.

TRACE32 Commands

This section describes the TRACE32 specific commands.
TRACE32 specific commands are used like any other commands when typing them in the Command Line component or when inserting them into a command file.
For further details about commands, please see *the Debugger* manual section *Appendix*, *Debugger Commands* and also section *Command Line Component*.

PROTOCOL

If this command is used, all the commands and responses sent and received are reported in the Command Line component window.

Syntax: **`PROTOCOL`** `ON|OFF`

Example `PROTOCOL ON`

Note: *This feature is used by support personnel from Motorola or Metrowerks.*

Pass Through PT

This command executes a TRACE32 command within the Debugger. The TRACE32 command follows the **`PT`** command, and can be run in the Command Line component or in a command file. For the TRACE32 command, the short form is allowed.

Syntax: **`PT`** `<T32 Command>`

Example `PT a.f , address 100--200 data.b 55`

RESET

This command resets the CPU and executes the `reset.cmd` file.

Syntax: **`RESET`**

Example `RESET`

TRACE32 Commands

The TRACE32 commands listed below can be used directly. For the word before the dot, the short form is not allowed.

Syntax: `<T32 Command>.<parameters>`

Example `ANALYZER.f , address 100--200 data.b 55`

The following TRACE32 commands are available:

```
ANALYZER ,BREAK, COUNT ,DATA, EXCEPTION, FLAG, FLASH, FPU, GO, MAP,
MMU, MODE, NAME, ON, PER, PULSE, PULSE2, REFRESH, REGISTER, REST32,
RUNTIME, SETUP, SPOT, STEP, STORE, SYMBOL, SYNCH, SYSTEM, TASK,
TERM, TRADDRESS, TREVENT, TRIN, TRIGGER, VCO
```

> ***Note:*** *For more details about these commands, please refer to* ***TRACE32 Emulator Reference (version 96/01/08)*** *from* ***Lauterbach Datentechnik GmbH.***

Clear Trigger

The **CT** command clears the values of specified bus analyzer triggers (events), A, B and/or C. The command also disables the cleared triggers. The trigger are cleared in the trigger unit of the Trace32 after the Trigger Program (**TP**) command has been executed. The trigger values are set by the Set Trigger (**ST**) command, and the Quick Trace Configuration dialog windows. Triggers are enabled by the Trigger Enable (**TE**) command; Triggers are disabled by the Trigger Disable (**TD**) command. The Sequencer mode and the Post trigger counter are not concerned from the Clear Trigger command.

Syntax: `CT list | *`

list specifies a list of trigger identifiers.
***** specifies all triggers (A, B and C).

Example `CT A`

Set Trigger

The **ST** command sets the value of one of the three triggers. If no parameters are specified or if only the trigger id is entered, the command interpreter will display the Quick Trigger Configuration dialog, with which the user sets a trigger value interactively. The trigger is set in the trigger unit of the Trace32 after the Trigger Program (**TP**) command has been executed.

Syntax:
```
ST    [id [(address|addressRange|,)
      [(data|,)[clipsId clips)]]]
      [;R|;W|;RW][;D]]
```

id Specifies the trigger id of the analyzer trigger, A, B or C.

address: An address to which a trigger is set. The address is specified with an address constant, as follows:
addressRange: A range of addresses within which a trigger is set. The address range can be specified with a starting and ending address constant, or with a starting address and a length value, as follows:
`start-address[..end-address]`
or
`start-address[::length]`
, The comma indicates that the address, address range, data or clips has been omitted. The omitted item is ignored in the trigger.
data: A data value that defines the trigger. The value is specified as follows:
`value[:mask]`
When a mask is entered, only the bits of the value that correspond to ones in the mask are used in the comparison.
clipsId specifies the external trigger port A or B.
clips: A 8-bit value that defines logic clip signals on the Trace32 analyzer for the trigger. The value is specified as follows:
`clips[:mask]`
When a mask is entered, only the bits of the value that correspond to ones in the mask are used in the comparison.
;R: Trigger on a read bus cycle only.
;W: Trigger on a write bus cycle only.
;RW: Trigger on read and write bus cycles
;D: Disable trigger; i.e., set trigger value and disable trigger.

example: `STA 0x1000`
Sets analyzer trigger A to match accesses at address $1000.

example: `STB , 4`
Sets analyzer triggers B to match accesses with a data value of 4, at any address.

example: `STC ,,A 0x11:0xFF`
Sets analyzer triggers C to match external trigger A with data signals 0x11. The mask for the external trigger is set to 0xFF that means that all 8 bit are used in comparison.

Enable Trigger

The **TE** command enables specified triggers, whether these triggers have been set or disabled. Triggers are set by the Set Trigger (**ST**) command, and are disabled by the Trigger Disable (**TD**) command. Another related command, the Trigger Clear (**CT**) command, clears all triggers. The trigger is enabled in the trigger unit of the Trace32 after the Trigger Program (**TP**) command has been executed.

Syntax: `TE list | *`
list: A list of triggers to be enabled; each trigger is either A, B or C and is separated from any following trigger by a comma.

`*`: All triggers (A, B and C).

example: `TE A,B`
Enables triggers A and B.

example: `TE *`
Enables all triggers.

Disable Trigger

The **`TD`** command disables specified triggers, whether these triggers have been set or enabled. Triggers are set by the Set Trigger (**`ST`**) command, and are enabled by the Trigger Enable (**`TE`**) command. Another related command, the Trigger Clear (**`CT`**) command, clears all triggers. The trigger is disabled in the trigger unit of the Trace32 after the Trigger Program (**`TP`**) command has been executed.

Syntax: **`TD`** `list | *`
`list`: A list of triggers to be disabled; each trigger is either A, B or C and is separated from any following trigger by a comma.
`*`: All triggers (A, B and C).

example: `TD A,B`
Disables triggers A and B.

example: `TD *`
Disables all triggers.

Set Sequencer Mode

The **`SQ`** command sets the analyzer sequencer mode. If no parameters are specified, the command interpreter displays the Quick Trace Configuration dialog window with which the user can program the sequencer interactively. The sequencer mode is set in the trigger unit of the Trace32 after the Trigger Program (**`TP`**) command has been executed.

Syntax: **`SQ`** `[mode [count[;S]]]`

`mode`: Specifies the sequencer mode. The following modes are available:

`ALL`	Records all bus cycles.
`EVENT`	Records events only.
`SEQ0`	Sequential recording mode: A -> C, B <-
`SEQ1`	Sequential recording mode: A + B + C
`SEQ2`	Sequential recording mode: A -> B -> C
`SEQ3`	Sequential recording mode: A + B -> C

For details about the sequential recording mode please see in chapter *Quick Trace Configuration*.
`count`: Sets the post trigger counter. The recording stops after the specified value is reached. If the post trigger counter is specified it is enabled automatically.

;S: If this option is set, the program is also stopped after the recording is terminated.

example: `SQ EVENT 100;S`

With this settings, only the events are recorded. After 100 events the recording and the application is stopped.

Set Event

The **SE** command sets the mode of the different input events (address, data or clips) for the specified Trigger. If no parameters are specified, the command interpreter displays the Quick Trace Configuration dialog window with which the user can program the sequencer interactively. The mode is set automatic if the Trigger is set with the **ST** Command. However, through the **SE** Command it is possible to enable and disable single input events of each Trigger. The events are set in the trigger unit of the Trace32 after the Trigger Program (**TP**) command has been executed.

Syntax: `SE[id[(addressMode|,)[(dataMode|,)[clipsMode]]]]`

id: Specifies the trigger id of the analyzer trigger, A, B or C.

, The comma indicates that the addressMode or dataMode, has been omitted. The omitted item is not modified.

addressMode: Specifies the address mode. The following modes are available:

`dis`	disable address event in the trigger
`address`	a single address is used in the trigger
`range`	an address range is used in the trigger

dataMode: Specifies the data mode. The following modes are available:

`en`	enables data event
`dis`	disables data event

clipsMode: Specifies the clips mode. The following modes are available:

dis	disables external trigger event
A	use external trigger port A
B	use external trigger port B

example: `SEA address,A`

With this settings, a single address and the external trigger port A is used as trigger events in the trigger A. The data mode is not modified.

Trigger Program

The **TP** loads the settings done through the commands **TE**, **TD** ,**ST** ,**CT**, **SQ** and **SE** in the trigger unit of the Trace32.

Syntax: `TP`

example: `TP`

Loads the current settings into the trigger unit.

Index