

AutoFocus User's Guide

MANUAL

AutoFocus User's Guide

TRACE32 Online Help

TRACE32 Directory

TRACE32 Index

TRACE32 Documents	
AutoFocus User's Guide	1
Introduction	4
Intended Audience	4
Prerequisites	4
Contacting Support	5
Installation	7
Hardware Installation	7
Preprocessor AutoFocus I (LA-7991)	9
External Termination PCB (shipped before 2006)	11
Termination PCB (shipped after 2006)	11
Preprocessor AutoFocus II (LA-7992)	12
Preprocessor AutoFocus MIPI II (LA-7993)	14
Software Installation	15
Recommendation for the Software Start	15
Recommendation for Power Down	15
Utilization	16
Preprocessor	16
Automatic Setup	16
Manual Setup	18
Trace Port	20
Start-up Script	21
Example Arm ETMv1	21
Loading and Storing the Settings of Sampling Points	23
FAQ	24
Diagnosis	25
Displaying Error Messages	25
Searching for Errors	26
Types of Trace Decoder Errors	28
Harderror	28
Flowerror	28
Fifofull	29
Trace Test Failure Messages	29
Access the Diagnosis Tool	30

Diagnosis Check List	31
Basic Checks	31
Advanced Checks	37
Protocol Check	43
ETMv1.x	43
ETMv3.x	46
Timing Requirements	49
Choose the Best Termination PCB Type	51
How to understand A.ShowFocusEye and A.ShowFocusClockEye	52
Recommendations for Target Board Design	55
Pin Remapping	57
Functional Description - Pin Remapping	57
Pinouts	59
AutoFocus II (LA-7992)	59
AutoFocus II MIPI (LA-7993)	60
Technical Data	61
AutoFocus Preprocessor Hardware Versions	61
Electrical Specification for AutoFocus II Preprocessors	64
Electrical Characteristics	64
Absolute Maximum Ratings	64
Switching Characteristics	64
Absolute Maximum Ratings	64
Timing Characteristics	65
Absolute Minimum Ratings	65
Average time resolution	65
Maximum channel delay	65
Additional Data	65
PCB	65
Components	65
Simulation Model	66
Autofocus V2	66
Autofocus Mipi	66
Dimensions	67
Connector Layout for Arm	76
ETMv1/2	76
ETMv1/2 with Multiplexed Mode	76
ETMv1/2 with 4 bit Demultiplexed Mode	77
ETMv1/2 with 8/16 bit Demultiplexed Mode	77
Dual ETMv1/2 - Not supported	79
ETMv3 32 bit	80
20 pin JTAG Connector	81

Introduction

Sampling high-speed parallel buses requires special concepts to compensate for the effects of wave reflections, component tolerances, different trace lengths, limited pad driver capabilities, signal coupling etc. All these effects might add up to a reduced signal amplitude, jitter and channel-to-channel skew.

Trace port analyzers (TPAs) that cannot compensate for these issues might not be able to trace high-speed targets, even if they support high data rates in theory.

The real challenge, however, is not to process high-speed data within a digital design, but to get the trace signals from the target into the TPA hardware without any information loss.

Trace probes with AutoFocus technology meet that challenge!

Intended Audience

This manual categorizes users into the following groups:

- *AutoFocus average users*: In addition to reading chapter [Installation](#), they should focus on the sections [Automatic Setup](#) and [Basic Checks](#).
- *AutoFocus advanced users*: As they are familiar with the above chapters and sections, advanced users should also read the sections [Advanced Checks](#) and [Manual Setup](#).
- *AutoFocus expert users*: As they are familiar with the above chapters and sections, they should also read the section [Protocol Check](#).

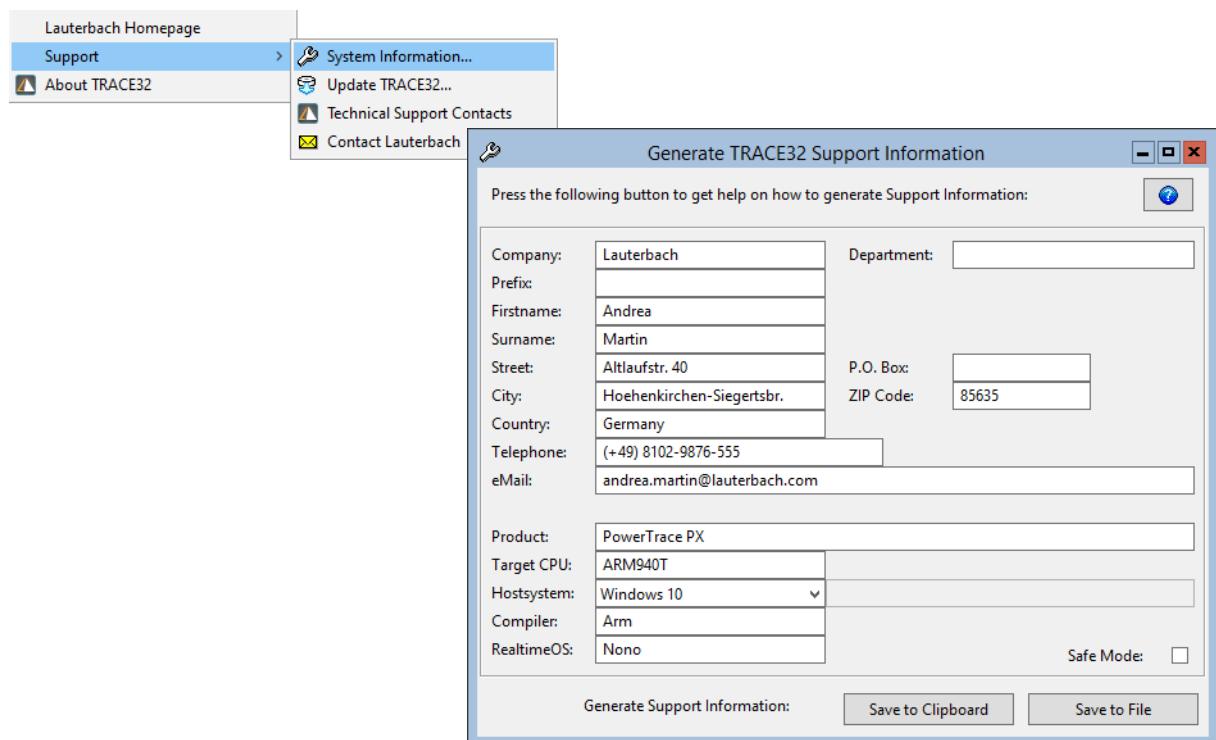
Prerequisites

- This document assumes that you already have the PowerTrace I or PowerTrace II or PowerTrace III hardware and a fully functional TRACE32 software installation.
- For greater than 16-bit trace ports, you need a second flex extension cable. This prerequisites does not apply AutoFocus II MIPI-based preprocessors.

Contacting Support

Be sure to include detailed system information about your TRACE32 configuration.

1. To generate a system information report, choose **TRACE32 > Help > Support > Systeminfo**.



NOTE:

Please help to speed up processing of your support request. By filling out the system information form completely and with correct data, you minimize the number of additional questions and clarification request e-mails we need to resolve your problem.

2. Preferred: click **Save to File**, and send the system information as an attachment to your e-mail.
3. Click **Save to Clipboard**, and then paste the system information into your e-mail.

Support Address

In addition to the system information report, please send the following information and files to support@lauterbach.com:

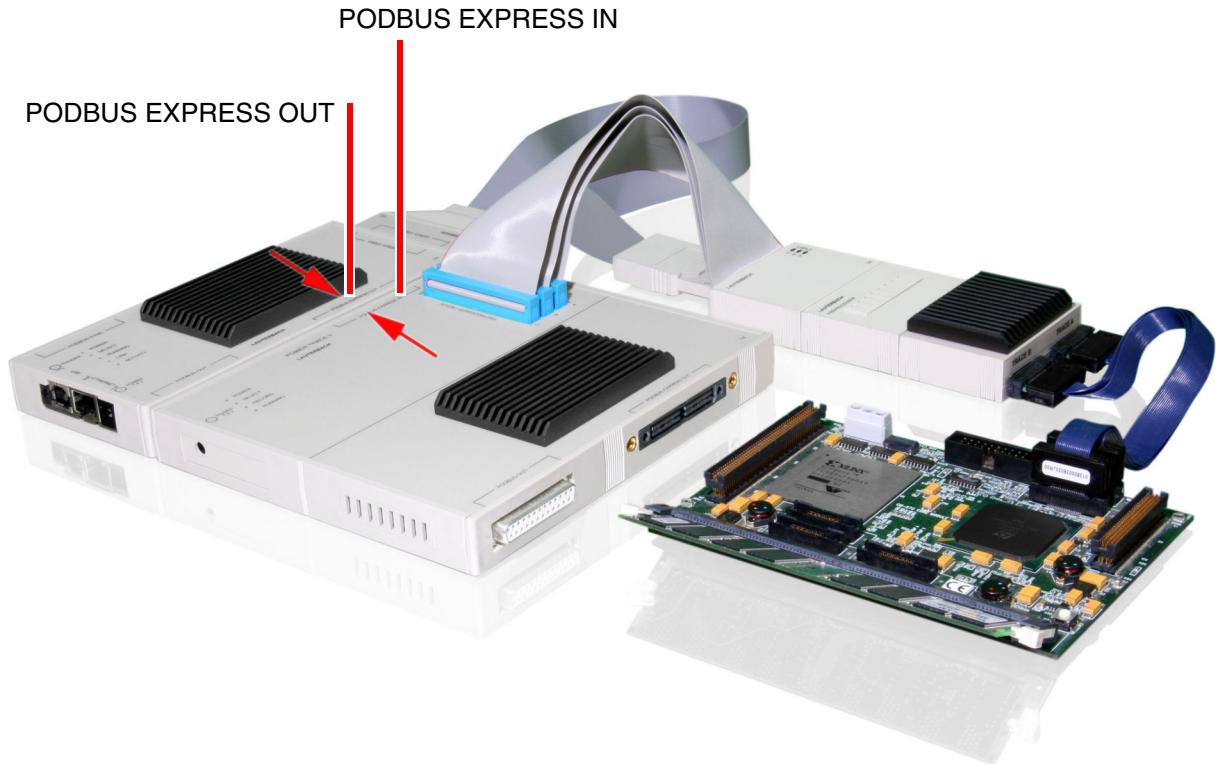
- Used start-up/configuration scripts
- The complete text of the error messages you get ([AREA.view](#))
- A screenshot of **VERSION.SOFTWARE**
- A screenshot of **VERSION.HARDWARE**
- A screenshot of **VERSION.HARDWARE2**
- A screenshot of the trace configuration window (e.g. **ETM** or AET window)

- JTAG voltage level
- JTAG frequency
- Trace port I/O voltage level
- CPU core frequency
- Trace connector pinout (board schematics and layout)
- Is the problem frequency dependent?
- Does another trace hardware behave differently?
- A screenshot of **Trace.ShowFocusClockEye** (set threshold to VCC/2 and press SCAN)
- A screenshot of **Trace.ShowFocusEye** (press SCAN before)
- A screenshot of **Trace.ShowFocus** (press SCAN before)

- [Hardware Installation](#)
- [Software Installation](#)

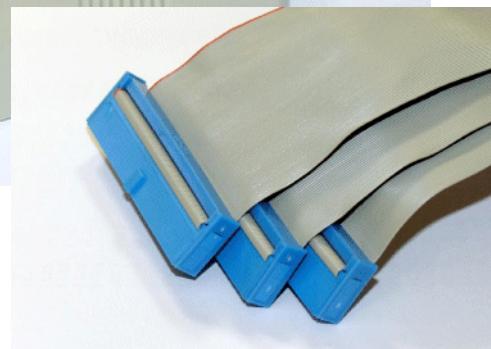
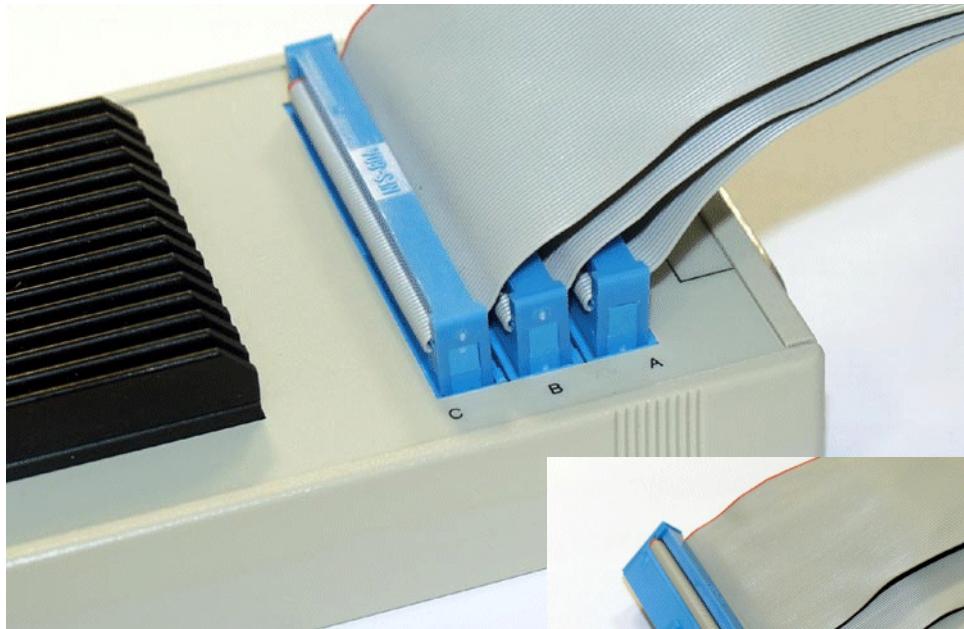
Hardware Installation

1. For PowerTrace II / PowerTrace III modules, connect the PODBUS EXPRESS IN connector to the PODBUS EXPRESS OUT connector of the PowerDebug II interface.



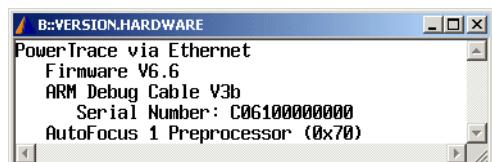
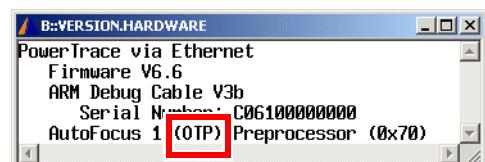
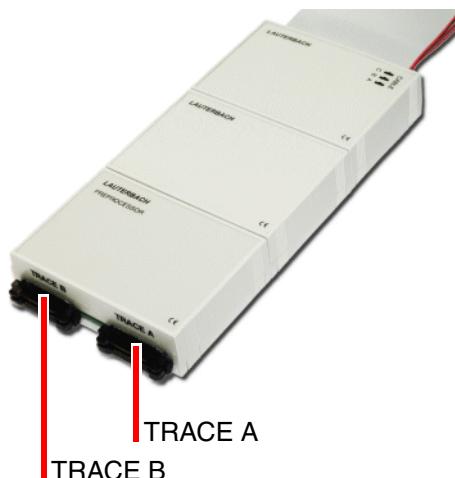
2. Connect the AutoFocus preprocessor to the **POWER TRACE** or **POWER TRACE II / POWER TRACE III**.

- The three flat cables have different lengths and need to be connected without crossing.
- The shortest cable needs to be connected to slot A, the middle to slot B and the longest to slot C.



There are two types of LA-7991, which can be distinguished with **VERSION.HARDWARE**.

- The LA-7991 OTP is based on a one-time-programmable FPGA that became obsolete in 2005. In the **VERSION.HARDWARE** window it is marked with '(OTP)'.
- The LA-7991 OTP is succeeded by a re-programmable version.



Both types have a similar performance, but there is a difference in the time resolution when it comes to adjustment of sampling points. You might notice this in the **Trace>ShowFocus** window. However this should not impact the actual trace result.

Before 2006 both the OTP as well as its re-programmable successor were shipped. Starting 2006 only the re-programmable Preprocessor with integrated termination is shipped.

	Preprocessors shipped before 2006 might be marked "(OTP)" in the VERSION.HARDWARE window, indicating that they are one-time-programmable. They support only ETM v1-3 pinouts (Arm7/9/10/11). Some CTOOLs pinouts do not follow the ETM v1-3 specification (e.g. OMAP1030). As a consequence, only simple tracing without trace compression is possible, or a new preprocessor must be acquired (e.g. LA-7992).
--	---

To connect the preprocessor for tracing the target hardware has to be equipped with one or two 38-pin MICTOR connectors. For dimensions and target connector pinout of the preprocessor refer to the chapter **Technical Data**. All trace signals are connected after plugging the preprocessor (TRACE A) into the target's trace connector.

Use a Mictor Flex Extension to connect the preprocessor with the trace target connector. Let the second connector (TRACE B) unused, if the target does not support >16bit trace ports.

AutoFocus I



AutoFocus II



There are two different ways to connect the debug cable:

- using the JTAG connector of your target platform
- using the JTAG connector of the preprocessor

The JTAG connector of the preprocessor is a 20-pin connector. The connector is located under the blue flat cable connectors and conforms to Arm's 20-in debug connector. If you are using a 14-pin debug cable you need to use a **JTAG Arm Converter 14-20 (LA-7747)**. Other architectures require an converter or must use a separate connector on the target.



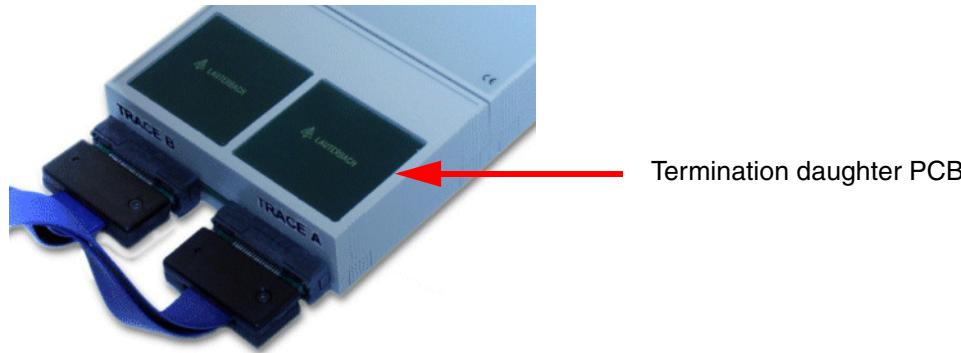
For Arm architectures mostly the debug connector of the preprocessor is the better choice, if both connectors are available on the target hardware.
Bad JTAG communication using the targets debug connector is caused by extending the JTAG lines by the preprocessor.



Don't connect any JTAG signals on the trace connector if a separate debug connector is used. Electrical reflections will impact the JTAG communication. The preprocessor itself does not use any JTAG-signal, they are simply routed to the back end connector.

External Termination PCB (shipped before 2006)

Most AutoFocus preprocessors shipped in 2005 came with two pairs of termination daughter PCBs. One pair labeled "1.5 ... 5 V", the other labeled "1.5 ... 3.3 V" or "1.5 ... 2.5 V" for early versions of the preprocessor.



How to choose the proper termination PCB:

- Complete range version (1.5 ... 5 V)

The complete range version cuts the signal amplitude roughly in half. Hence it is safe to use, even for 5 V targets, but it might not be optimal for target voltages below 2.5 V.

- Low voltage version (1.5 ... 3.3 V or 1.5 ... 2.5 V)

The low voltage version does not affect the signal amplitude significantly. This module usually shows better results in terms of data eye width, especially for target voltages below 2.5 V. For early versions of this termination module, the signal amplitude after the termination PCB was conservatively specified for a maximum of 2.5 V which is why these modules were labeled "1.5 ... 2.5 V". As more data became known, the maximum voltage could be increased to 3.3 V, so this module is now labeled "1.5 ... 3.3 V".

You must not use the low-voltage termination for target voltages above 3.3 V!

Termination PCB (shipped after 2006)

- Integrated low-voltage termination (1.5 ... 3.3 V)

The low-voltage termination is now integrated in the main PCB. For target voltages greater 3.3 V a voltage converter (LA-7922) has to be used. However, this voltage converter might reduce the maximum trace frequency. You should always contact support@lauterbach.com to discuss solutions for target voltages outside the specified range of 1.8 ... 3.3 V or if you require a customized termination module.

Not all termination PCBs are compatible with all AutoFocus preprocessors, so it is best to only use the termination PCBs that were shipped together with your preprocessor. Please refer to the table below to find out PCB ID combinations that are compatible. There is an ongoing effort to optimize the termination module for even higher frequencies, especially for the low voltage targets. In the table below, **bold** termination PCB IDs indicate that the termination PCB contains the latest improvements.

If you are unable to trace your target application at its maximum operating frequency and you do not have the latest available termination module, contact sales@lauterbach.com for delivery arrangements. Use the PowerTrace-AF Diagnosis tool to find out your preprocessors PCB IDs. (See [Access the Diagnosis Tool](#).)

LA-7991 PCB ID	Termination PCB ID for 1.8 ... 3.3 V	Termination PCB ID for 1.8 ... 5 V
0x0 (OTP)	0x4	0x1
0x7 (OTP)	0x4	0x1
0xE (OTP)	0x6, 0xE, 0xF	0x2
0x0	0xF	0x2
0x1	0xF (integrated)	not applicable

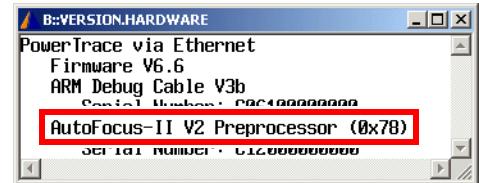
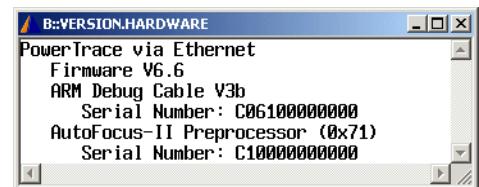
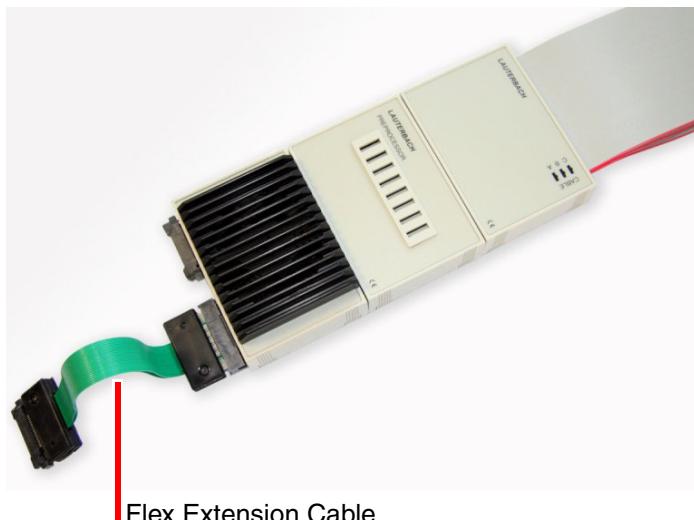


Not all termination PCBs are compatible with all AutoFocus preprocessors, so it is best to only use the termination PCBs that were shipped together with your preprocessor.
You must not use the low-voltage termination for target voltages above 3.3 V

Preprocessor AutoFocus II (LA-7992)

The AutoFocus preprocessor II is the next generation of AutoFocus preprocessors. Its handling is similar to [Arm-ETM AutoFocus \(LA-7991\)](#).

There are two types of LA-7992, which can be distinguished with **VERSION.HARDWARE**.



The preprocessor IDs are given in parentheses in the **VERSION.HARDWARE** windows, here: (0x71), (0x78).

There is a gain in performance from version 1 to version 2. In addition, the crosstalk behavior has improved and the all-over bandwidth was increased.



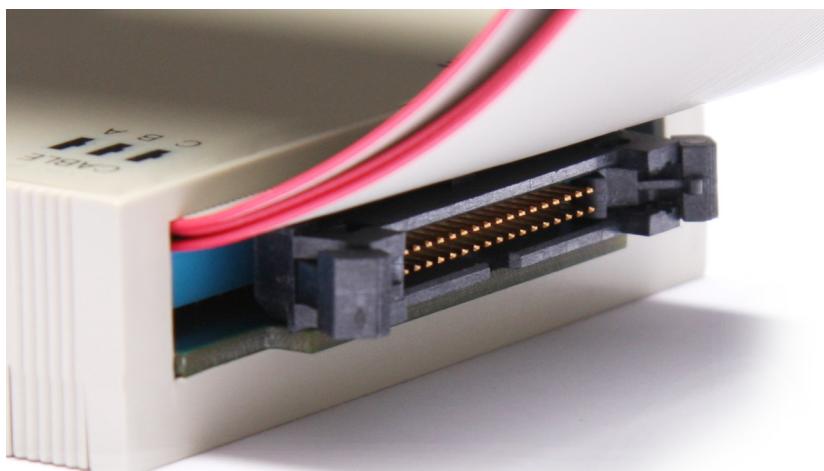
The termination circuit is integrated for all AutoFocus II based devices. Changeable termination modules are not available for these devices.

The AutoFocus MIPI preprocessor is based on AutoFocus II V2 technology. The main difference is the connectorization, which is conform to the MIPI specification for debug and trace signals. Its handling is similar to [Arm-ETM AutoFocus \(LA-7991\)](#).



The preprocessor can also be determined by the ID shown in [VERSION.HARDWARE](#).

The JTAG connector of the preprocessor is a 34-pin connector. The connector is located under the blue flat cable connectors. A converter is required, if you are using a debug cable with a non-MIPI connector (e.g. an [Arm Converter Arm-20 to MIPI-34 \(LA-3770\)](#)).



TRACE32 PowerView includes debug and trace tool support. An extra software installation is not required, but valid license keys are required.

Recommendation for the Software Start

1. Disconnect the debug cable from the target while the target power is off.
2. Start the TRACE32 software.
3. Connect the debug cable directly to the target.
 - If there is no appropriate jack on your target, you can also connect the debug cable to the preprocessor (for Arm/Mipi conform devices only).
 - Alternatively, connect the debug cable to an additional converter PCB, which splits debug and trace signals. The converter PCB needs to be ordered separately at Lauterbach.
4. Connect the preprocessor to your target's trace port by using the flex extension cable delivered with your preprocessor. For port sizes greater 16 bit, you need to connect port TRACE B as well, using a second Mictor flex extension (not for LA-7993).

NOTE: The second flex extension has to be ordered additionally.

5. Switch the target power on.
6. Run your start-up script.

Recommendation for Power Down

1. Switch off the target power.
2. Disconnect the debug cable and preprocessor (flex extension cable) from the target.
3. Shut down the TRACE32 PowerView GUI.
4. Switch off the TRACE32 hardware.

- [Preprocessor](#)
- [Trace Port](#)

Preprocessor

You can set up the preprocessor automatically or manually.

The automatic setup uses a pattern generator or a short test program to test the trace port. The availability of the pattern generator is target dependent. TRACE32 will do the detection for you. Anyway, the way to force test-program-based testing is still open for the user.

The short test program will be copied to the target memory (RAM) where the program pointer or the stack pointer must point to. If both pointers does not point to RAM and no pattern generator is available, an error message will be shown.



The automatic setup is adequate in the most cases with free running clock signal. In case of heavy electrical issues, large port sizes, gated clocking or high frequency systems (>450MBaud) the manual setup may be the better choice.

Automatic Setup

The automatic setup is applicable for the following cases:

- Constant core/trace frequency
- High signal integrity
- Low signal skew

Prerequisite:

- You have just run your start-up script in TRACE32 to set up your target.

To do an automatic setup of the preprocessor - option 1:

1. Make sure that the final core/trace frequency is set ([Data.Set](#)).
2. Make sure that the target application is stopped ([Break](#)).
3. Set program counter to target RAM address space ([Register.Set PC](#)).

NOTE:	Steps 1 to 3 or 1 to 4 should be part of your start-up script.
--------------	--

4. In the **Trace.state** window, click the **AutoFocus** button or execute [Analyzer.AutoFocus](#).
5. If you receive a success message, you are ready to use the trace feature of TRACE32.
6. If you receive an error message, take these steps:
 - Open an **AREA** window ([AREA.view](#)).
 - Continue at chapter “[Diagnosis](#)”, page 24.

The test pattern allows very good detection of bit errors, but it is quite synthetic. A real target application might produce additional electrical action on the PCB, which leads to a different signal integrity of the trace signals. A second way to do the automatic setup might help. The main difference is that the trace data of a real application are used.

Prerequisite:

- You have just run your start-up script in TRACE32.

To do an automatic setup - option 2:

1. Make sure that the final core/trace frequency is used ([Data.Set](#)).
2. Load your target application to the TRACE32 virtual memory ([Data.Load.auto ... /VM](#)).
3. Run the target application ([Go](#)).
4. In the **Trace.state** window, click the **AutoFocus** button or execute [Analyzer.AutoFocus](#).
5. If you receive the message, you are ready to use the trace feature of TRACE32.
6. If you receive an error message, take these steps:
 - Open an **AREA** window ([AREA.view](#)).
 - Continue at chapter “[Diagnosis](#)”, page 24.

This section describes a manual setup of all sampling parameters for the preprocessor. The manual setup is applicable for the following cases:

- Varying core/trace frequency
- Gated trace clock
- Low signal integrity or asymmetric clock waveforms
- High signal skew
- AutoFocus fails

To manually set up the sampling parameters for the preprocessor:

1. Enable termination ([Analyzer.TERMination ON](#))
2. Set the threshold for clock and data to VCC/2 ([Analyzer.THreshold VCC](#))
3. Scan the clock eye ([Analyzer.TestFocusClockEye](#))
Accumulate multiple scans in case of varying clock frequency
([Analyzer.TestFocusEye /Accumulate](#))
4. Look for best threshold level ([Analyzer.ShowFocusClockEye](#))
5. Set the threshold for clock ([Analyzer.THreshold <clk>](#))
6. Scan the data eye ([Analyzer.TestFocusEye](#))
Accumulate multiple scans in case of varying clock frequency
([Analyzer.TestFocusEye /Accumulate](#))
7. Look for best threshold level ([Analyzer.ShowFocusEye](#))
8. Set the threshold for data ([Analyzer.THreshold <clk> <dat>](#))
9. Scan the digital data eyes ([Analyzer.TestFocus](#))
Accumulate multiple scans in case of varying clock frequency.
([Analyzer.TestFocus /Accumulate](#))
10. Define potential sampling points ([Analyzer.ShowFocus](#))
11. Set the sampling points - one for each trace signal ([Analyzer.SAMPLE](#)).
Alternatively, double-click in the [Analyzer.ShowFocus](#) window to set the sampling points.
12. Test the trace port ([Analyzer.TestFocus](#))

	More than one data eye could be shown in the Analyzer.ShowFocus window at higher trace port speeds. It can be essential to select the correct data eye in case of gated clocking. You should set all sampling points to the next visible data eye and test again, if the trace port test fails at #12.
---	--

If you want to go deeper into the subject please see the information given in chapter [Advanced Check](#).

Alternative:

1. Load your target application to the target board and to the TRACE32 virtual memory ([Data.Load.auto ... /PlusVM](#)). The option **PlusVM** allows you to do both steps simultaneously.
2. On the TRACE32 toolbar, click **Go** to start your target application.
3. Perform steps 1 to 12 in the step-by-step procedure above.

The AutoFocus preprocessor can be used to trace many different architectures. Each architecture requires its own configuration on the target side to configure the CPU to send trace information.

A common trace setup consists of the following parts:

TRACE32 start-up

- Connection of system modules
- Establishing host connection
- Starting TRACE32 PowerView

TRACE32 setup

- Debug and trace port connection
- CPU (selection, system mode,...)
- Peripheral (external buffer enable,...)
- GPIO/trace port (PLL, pin manager,...)
- Trace logic (base address, filter,...)
- Application (frequency, I/O setup,...)

TRACE32 preprocessor setup

- Trace port (width,...)
- Sampling parameters (threshold, sampling point, termination,...)

Trace port test

- Check for correct data transmission and recording

TRACE32 offers a powerful feature called PRACTICE. The Lauterbach script language PRACTICE is used for automating tests, configuring the TRACE32 PowerView GUI and your debug environment.

The following example shows how to configure a trace port (ETM) and how to save the configuration.

Example Arm ETMv1

(A) The following ETM settings are required:

- Define the width of the trace port with the command **ETM.PortSize**.
- Define the mode of the trace port with the command **ETM.PortMode**.
- Define if the ETM works in Halfrate mode or not with the command **ETM.HalfRate**.
- Turn on the ETM with the command **ETM.ON**.

(B) The target must be configured:

- Set up the I/O-ports. To find the required information, refer to your CPU manual.
- Set up the board (buffers, jumpers, etc.). To find the required information, refer to your target user guide.
- Configure the operating frequency. The frequency depends on your own application.

(C) Finally, the AutoFocus preprocessor needs to be set up correctly:

- Set up the AutoFocus preprocessor with the command **Analyzer.AutoFocus**.
- Check the trace channel with **Analyzer.TestFocus** (included in Analyzer.AutoFocus).

This example is made for an Arm9 target (e.g. CM966E-S by Arm):

```
; JTAG DEBUGGER SETUP
SYStem.RESet ; Initialize system
SYStem.JtagClock RTCK ; Select JTAG clock
SYStem.CPU Arm966E ; Select CPU type
SYStem.Up ; Start debugger

; TARGET SETUP
Data.Set SD:0x10000014 %LE %L 0a05f ; Unlock target registers
Data.Set SD:0x10000008 %LE %LONG 20 ; Set target frequency
SYStem.Option.BigEndian OFF ; Set endianism

; PROGRAM SETUP
Data.LOAD.ELF armle.axf /SPATH /LPATH ; Load example PRACTICE script
Register.Set PC main ; Set program counter to program
; start
Register.Set R13 0x1000 ; Initialize stack pointer

; (A) ETM SETUP
ETM.PortSize 16 ; Set the trace port width to 16

ETM.PortMode Normal ; Set the trace mode to Normal
; mode

ETM.HalfRate OFF ; Set full rate mode for ETM

ETM.DataTrace Both ; Trace Address and Data
ETM.ON ; Turn ETM on

; (B) not applicable in this example

; (C) Configure AutoFocus preprocessor
Analyzer.THreshold VCC ; set threshold to 50% of the
; voltage level of pin12 of the
; target connector
Analyzer.TERMination ON ; connect termination voltage
; during trace
Analyzer.AutoFocus ; Set threshold and sampling
; points automatically

; Test the trace channel
Analyzer.TestFocus ; Load, execute and trace test
; program and report errors

ENDDO ; End of script
```

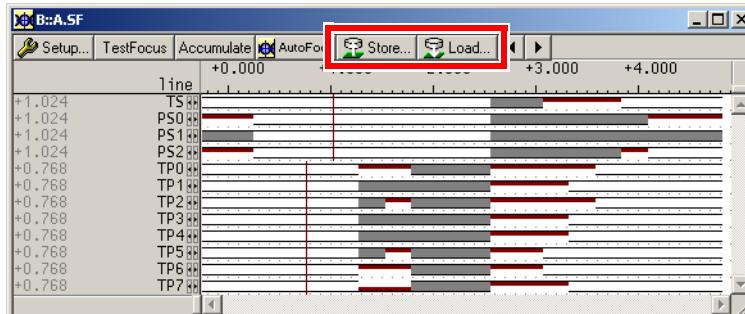
NOTE:

Don't forget to check the ETM port with [Analyzer.TestFocus](#). The check must always finish with success.

For AutoFocus preprocessors, you can store the settings of the current session and re-load these settings for the next session. Alternatively, you can copy and paste the stored settings into your start-up script. The settings are then loaded automatically when you start TRACE32 with that start-up script.

Current session - To store the settings of the sampling points:

1. In the **Trace.ShowFocus** window, click **Store**.



Trace.ShowFocus as it appears for a re-programmable LA-7991

2. Enter a file name, and then click **Save**.

Clicking the **Store** button calls **STOre <file> AnalyzerFocus** and generates a PRACTICE script similar to this:

```
IF ANALYZER()
(
    ANALYZER.TERMINATION ON          ; connect termination voltage
                                    ; during trace
    ANALYZER.THRESHOLD 1.19 0.99    ; clock reference voltage
                                    ; = 1.19 V
                                    ; data reference voltage
                                    ; = 0.99 V

    ANALYZER.SAMPLE    TS    -0.219    ; Store trace channel sampling
    ANALYZER.SAMPLE    PS0   -0.219    ; points
    ANALYZER.SAMPLE    PS1    +0.365
    ANALYZER.SAMPLE    PS2    +0.365
    ANALYZER.SAMPLE    TP0   -1.387
    ANALYZER.SAMPLE    TP1   -1.387
    ANALYZER.SAMPLE    TP2   -1.387
    ANALYZER.SAMPLE    TP3   -1.387
    ANALYZER.SAMPLE    TP4   -0.803
    ANALYZER.SAMPLE    TP5   -1.387
    ANALYZER.SAMPLE    TP6   -1.387
    ANALYZER.SAMPLE    TP7   -1.387
)
```

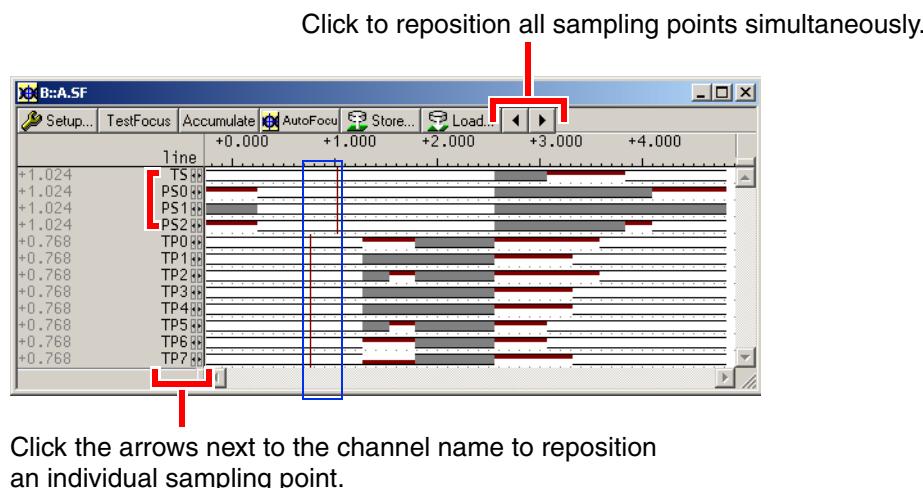
New session - To re-load the stored sampling point settings:

1. In a new **Trace.ShowFocus** window, click **Load**.
2. Select the file containing the stored sampling point settings, and then click **Open**. The sampling point settings are displayed in the **Trace.ShowFocus** window.

It is not recommended to manually edit the sampling point settings in the stored PRACTICE script even if this is possible. Instead, edit the settings in the **Trace>ShowFocus** window:

- Use the left / right arrows to adjust the group delay (all sampling points will be moved).
- You can reposition individual channel sampling points (the blue rectangle in the picture below) to the left or right by double-clicking a new position.
 - Alternatively, click the left or right arrows next to the signal name to reposition the corresponding channel sampling point.

In the example below, you could move TS and PS[2:0] to the right and TP[7:0] to the left to sample closer to the middle of the data eyes.



FAQ

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

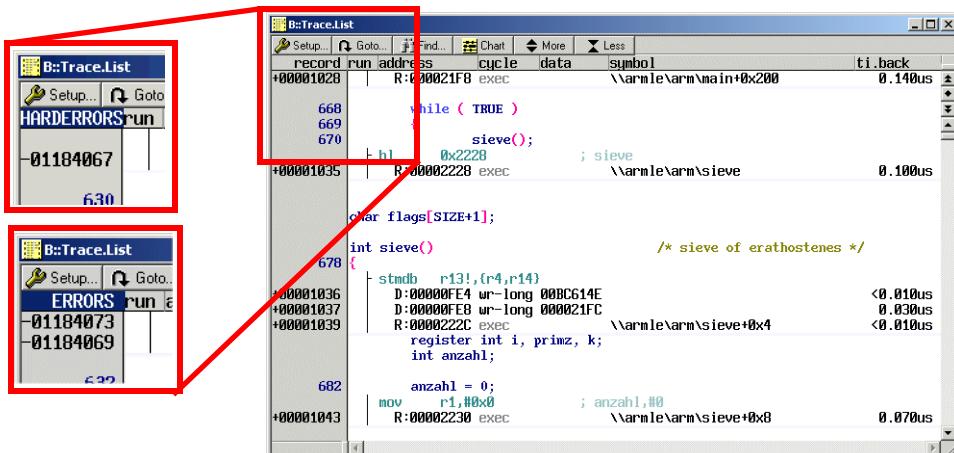
Displaying Error Messages



Advanced trace analysis commands like **Trace.STATistic.Func**, **Trace.STATistic.TASK** or **PERF.List** display only accurate results if the trace recording works error free.

Error messages are displayed:

- In the upper left corner of the **Trace.List** window:



- In the message line:



- In the **Area.view** window:



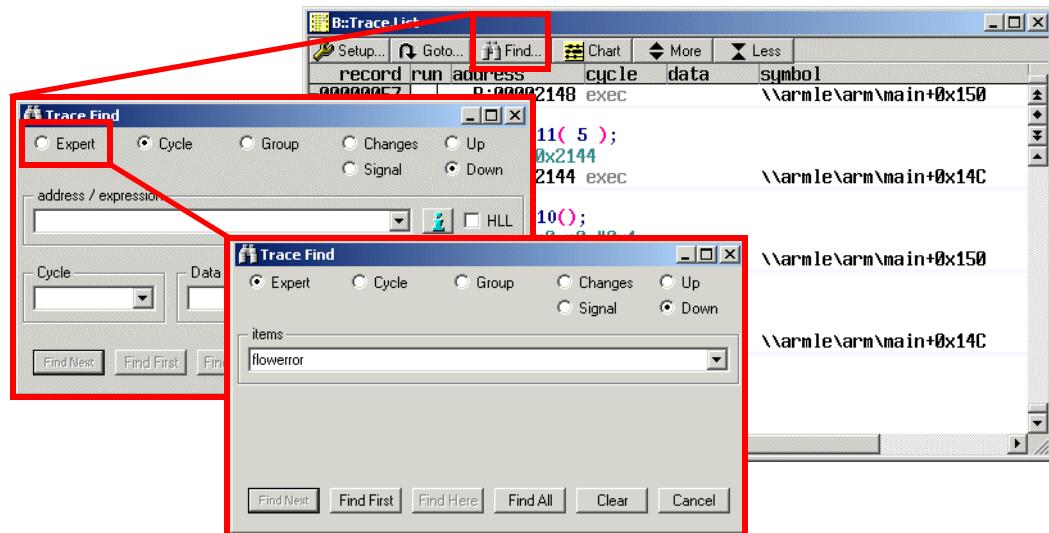
Searching for Errors

TRACE32 uploads only the requested trace information to the host to provide a quick display of the trace information. Consequently, errors that are out of the uploaded range are not immediately visible.

There are several ways to search for errors within the trace, all of them will force TRACE32 to upload the complete trace information to the host:

1. The Trace Find dialog:

- In the **Trace.List** window, click **Find** to open a dialog with special search options:



- Select **Expert** and enter “flowerror” in the **items** field. The item entry is not case sensitive.
- Use the **Find First** and **Find Next** buttons to jump to the next flowerror within the trace.
- Click **Find All** to open a window listing all flowerrors.

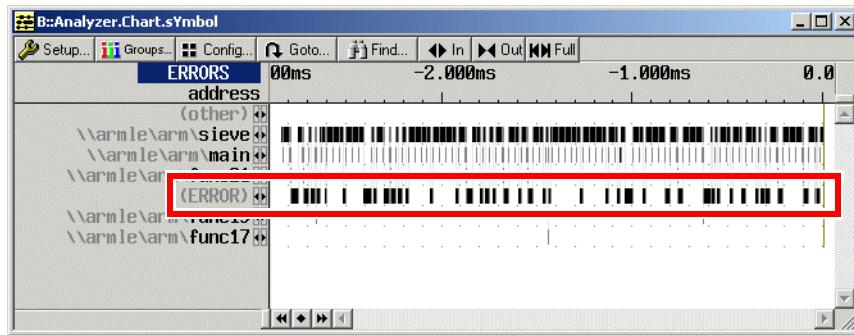
2. The command **Trace.FindAll , FLOWERROR**

This command searches for **errors** within the entire trace buffer. The records are listed in a window. The command corresponds to **Find All** described above.

43	run	address	cycle	data	symbol	ti.back
00203067		R:00001E34	exec		\var{mle}\arm\func21+0x10	
00198311		R:00001E34	exec		\var{mle}\arm\func21+0x10	65.500us
00195443		R:0000226C	exec		\var{mle}\arm\sieve+0x44	39.530us
00193559		R:00001DE4	exec		\var{mle}\arm\func19+0x10	25.970us
00189917		R:0000227C	exec		\var{mle}\arm\sieve+0x54	49.930us
00183125		R:00002254	exec		\var{mle}\arm\sieve+0x2C	93.570us
00175149		R:0000227C	exec		\var{mle}\arm\sieve+0x54	109.800us
00173609		R:00002254	exec		\var{mle}\arm\sieve+0x2C	21.200us
00171469		R:0000226C	exec		\var{mle}\arm\sieve+0x44	29.500us
00167741		R:0000227C	exec		\var{mle}\arm\sieve+0x54	51.360us
00165011		R:00001E38	exec		\var{mle}\arm\func21+0x14	37.640us
00162243		R:00002254	exec		\var{mle}\arm\sieve+0x2C	38.230us
00159333		R:00002254	exec		\var{mle}\arm\sieve+0x2C	39.770us
00149817		R:00002254	exec		\var{mle}\arm\sieve+0x2C	131.000us
00140301		R:00002254	exec		\var{mle}\arm\sieve+0x2C	131.000us

3. The command **Trace.Chart.sYmbol**

This command starts a statistical analysis. An additional symbol (ERROR) is shown if errors were found.



The search could take a long time depending on the used memory size of the trace module and the type of host interface. Check the status to estimate the time.

Types of Trace Decoder Errors

One of the following errors may occur:

- [Harderror](#)
- [Flowerror](#)
- [Fifofull](#)

Harderror

There is no valid trace data. Possible reasons are:

- Trace port multiplexed with other I/O functions (no valid trace data)
- Trace signal capturing failed (setup/hold time violations)
- Wrong version of PowerTrace module
- Target frequency too high

Please see [Diagnosis Check List](#).

Flowerror

The traced data is not consistent with the code in the target memories. Possible reasons are:

- Memory contents have changed (e.g. self modifying code).
- Wrong trace data (as result of [Harderror](#))

Please see [Diagnosis Check List](#).

The trace output Fifo has overflowed. The amount of trace data generated by the trace logic was greater than the trace port band width. To reduce the risk of a Fifo overflow:

- Increase the port size if possible.
- Restrict the **DataTrace to read cycles** (write accesses can be reconstructed via **CTS**).
- Restrict the **DataTrace to write cycles**, a Fifo overflow becomes less likely.
- Reduce amount of trace data by using filters: use the filter **TraceEnable** or **TraceData**
- STALL the CPU if a Fifo overflow is likely to happen, if supported.
- Suppress the output of the data flow information if a Fifo overflow is likely to happen, if supported (**ETM.DataSuppress**).



Trace macrocells are not always able to prevent overflows of the internal Fifo. Even when STALL is enabled overflows may occur.

Trace Test Failure Messages

Trace.TestFocus supports a built-in trace test. This command loads a short test program to the target memory (RAM) and traces its execution. Afterwards the recorded program flow and data pattern will be checked for any errors.

“Analyzer data capture o.k.” will be shown if the test was successful.

Test failures might be caused by a variety of reasons, usually error messages such as:

“**Trace test failed: not enough samples in the trace**” will give you a clue as to what might have caused the failure.

Access the Diagnosis Tool

Lauterbach provides a diagnosis tool for the **Preprocessor with AUTOFOCUS**. It is recommended that you add the **AF Diagnosis** button to the TRACE32 toolbar for quick access to the diagnosis tool.



To access the diagnosis tool via a toolbar button:

1. Add this script line to the PRACTICE script file system-settings.cmm in your TRACE32 system directory:

```
DO ~/demo/etc/diagnosis/autofocus/add_afdiag_button.cmm
```

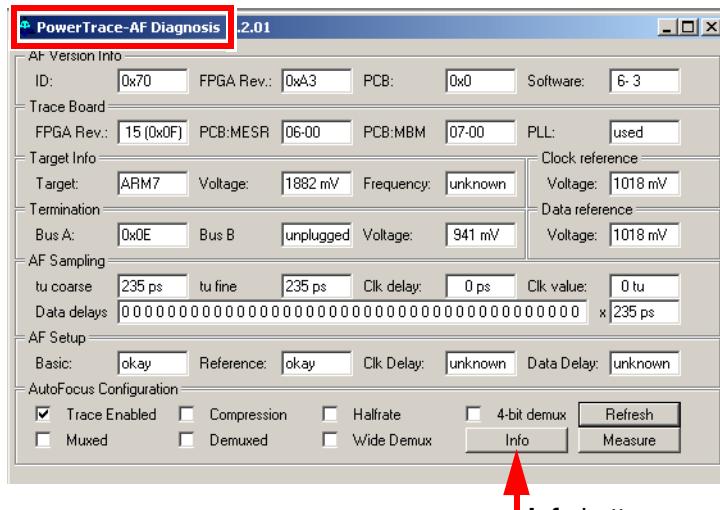
2. Re-start TRACE32.

The **AF Diagnosis** button is now available on the TRACE32 toolbar.

To just start the diagnosis tool, execute the following command:

```
DO ~/demo/etc/diagnosis/autofocus/afdiagnosis.cmm
```

If there is no AutoFocus preprocessor attached, executing this script will generate an error message. Otherwise, it displays some diagnosis results in the following window:



Do not feel discouraged if some of the diagnosis results displayed in the window are meaningless to you. Instead follow the steps in section [Diagnosis Check List](#).

In this section:

- [Basic Checks](#)
- [Advanced Checks](#)
- [Protocol Check](#)

Basic Checks

1. Check

Did your debugger remain in control over the target at all times when attempting to capture a trace? Error messages such as “**emulation debug port fail**” indicate that the debugger lost control over the target. In case your debugger lost control over the target:

- Is there a separate JTAG connector on your target?
If available, connect the debug cable directly to this connector.
- Are all supply voltage levels stable when the trace port is active?
Double check your targets supply voltage.

2. Check

Some CPU types do not have dedicated trace port pins. Instead trace signals are multiplexed with other signals. A special port pin setup may be required to get trace functionality. Example:

```
PER.Set SD:0x111D640 %Word 0x9AA0 ; Enable ETM functionality on
PER.Set SD:0x111D6A4 %Word 0x2901 ; GPIOs
PER.Set 0x111600D %Long %LE 0x01E ; Enable CLK
```

Check your CPU manual for the correct port pin configuration.



Advanced target applications might use more than one initialization procedure or the setup might change during run time again. Make sure that the ETM port is actually enabled when attempting to trace.

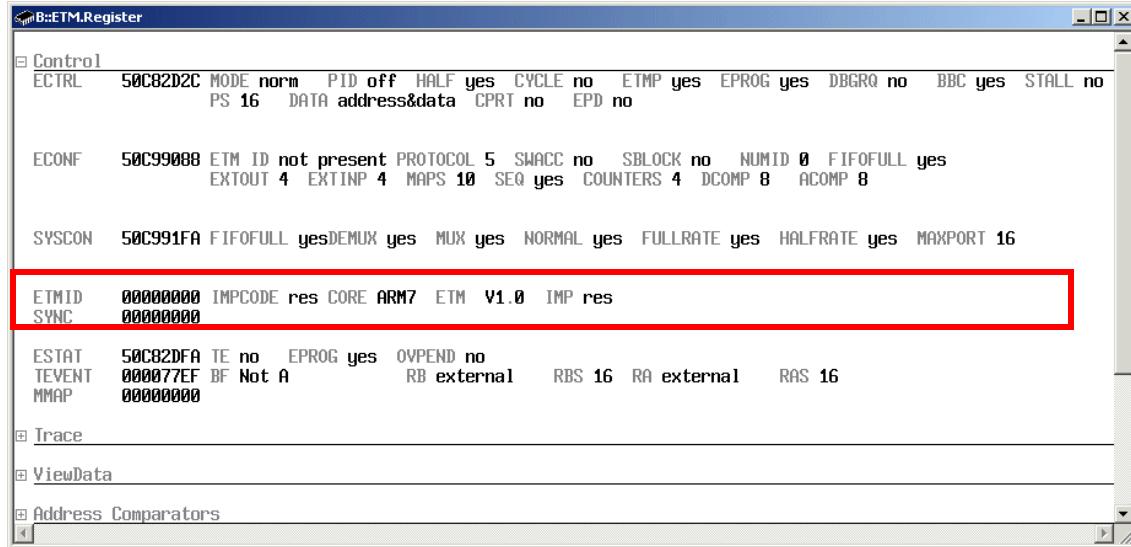
3. Check

In case of shared trace pins, additional buffers may be used on the target hardware. Make sure that these buffers are enabled.

4. Check

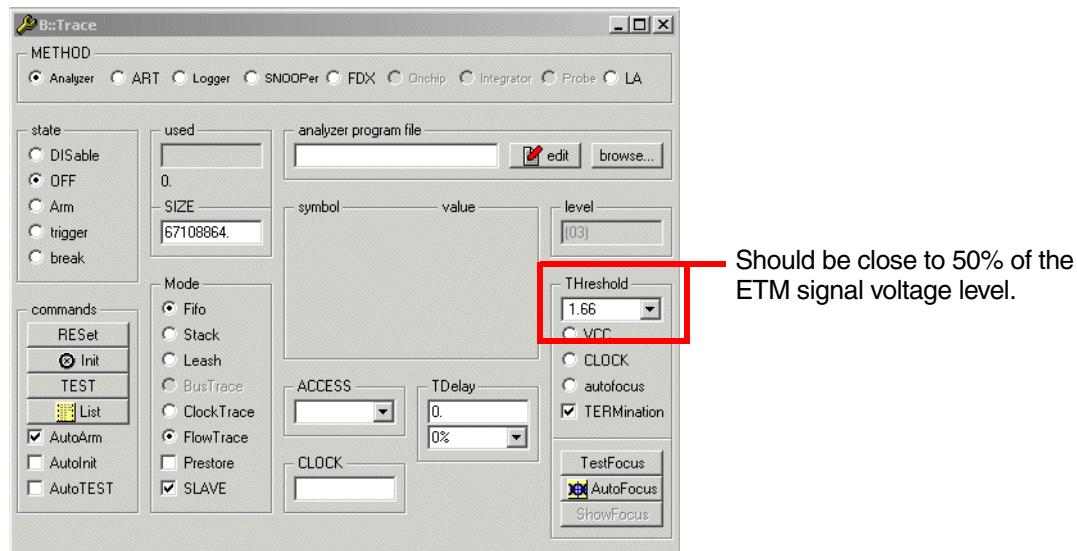
Check your CPU manual for the correct trace settings such as port size, port mode and clocking mode. Wrong settings result in errors.

The trace configuration registers may offer additional help. These are architecture-specific and not always available. For ETM execute [ETM.Register](#):



5. Check

Most of the trace hardware is set up automatically. Special care has to be taken when setting the threshold level ([Trace configuration](#)):



The threshold values for clock and/or data signals is/are set automatically at software start, depending on the voltage level of pin 12 of the target connector. However, if your target is turned off or not connected during software start, you may need to execute [Trace.THreshold VCC](#) or manually set it to the proper value (e.g. 0.9 V, 1.25 V or 1.65 V).

6. Check

Execute **Trace.AutoFocus** and both an automatic hardware configuration as well as the trace test described above will be executed. For frequency-scaling targets, multiple scans may be necessary to check the correct sampling points.

```
; Example: extended trace test
; a) Hardware configuration and trace test
Trace.AutoFocus

; b) Extended trace test
; Accumulate data eye information (10 runs)
RePeat 10.
(
  Trace.TestFocus /Accumulate
)
; c) Show results
Trace.ShowFocus
```

7. Check

Check the trace port data rate. Depending on the trace macrocell, the data rate of the trace port could be coupled on the core clock frequency or not. E.g.:

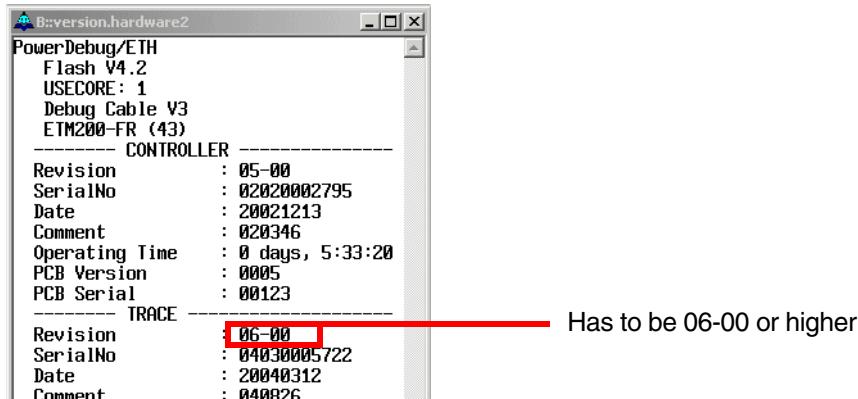
ETMv1/2: Bw=fcore * 1bit/ch

ETMv3: Bw=fport * 2bit/ch

Product Number	Datarate (Bw)
PowerTrace LA-7690, LA-7707	<320Mbaud/s
PowerTrace II LA-7692, LA-7693, LA-7694	800Mbaud/s

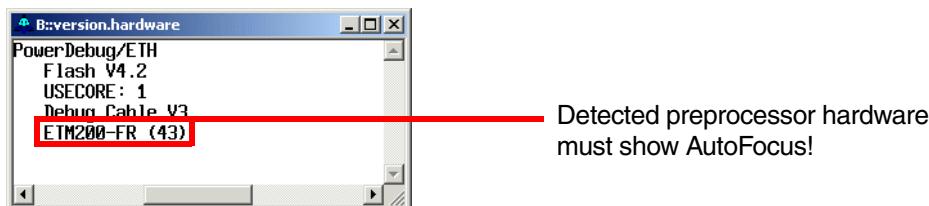
8. Check

Check the version of your PowerTrace I module. Version 6 is required for targets running faster than 200 MHz ([VERSION.HARDWARE2](#)).



9. Check

Check the preprocessor seen by the TRACE32 software. [VERSION.HARDWARE](#) shows all detected hardware.



10. Check

In case of preprocessors with more than one Mictor connector, double-check that the Mictor connectors are properly connected to your target. For trace port sizes \leq 16 bit, TRACE B remains unconnected. For further information on the correct connection of TRACE A / TRACE B, refer to [Dimensions](#).

11. Check

Check the trace port pinout.

Did your target board work with other PowerTrace/Preprocessor combinations? If your target worked with other PowerTrace/preprocessor combinations, your trace port pinout can be assumed to be correct. This is not always true for TPAs from other vendors. Lauterbach tools assume that the trace port pinout follows exactly the trace specification of the used architecture (please refer to [Connector Layout](#)).

Did the PowerTrace/preprocessor combination work for other targets? If yes, what has changed on your new target board? Often messages such as **Trace test failed: not enough samples in the trace** or **Trace test failed: pin connection error** might indicate the source of the error.

Error message	Possible reason
Trace test failed: not enough samples in the trace	<ul style="list-style-type: none">• Trace port not enabled• Buffers not enabled• Threshold out of signal range
Trace test failed: pin connection error	<ul style="list-style-type: none">• Short-circuit between pins• Wrong connector pin out• Unsupported ETM mode• Trace port not enabled

Also check the voltage level of the reference voltage on pin 12. It is used as a reference for all trace signals. It should correspond to the amplitude of your trace signals. For some targets this might differ from the JTAG signals. Pin 14 is used as reference for all JTAG signals in case the JTAG debugger is connected via the trace preprocessor probe. One way of testing the voltage on pin 12 is by executing **Trace.THreshold VCC** and double-checking that the voltage displayed in **Trace.CONFIGURATION** window is approximately 50% of your target voltage.

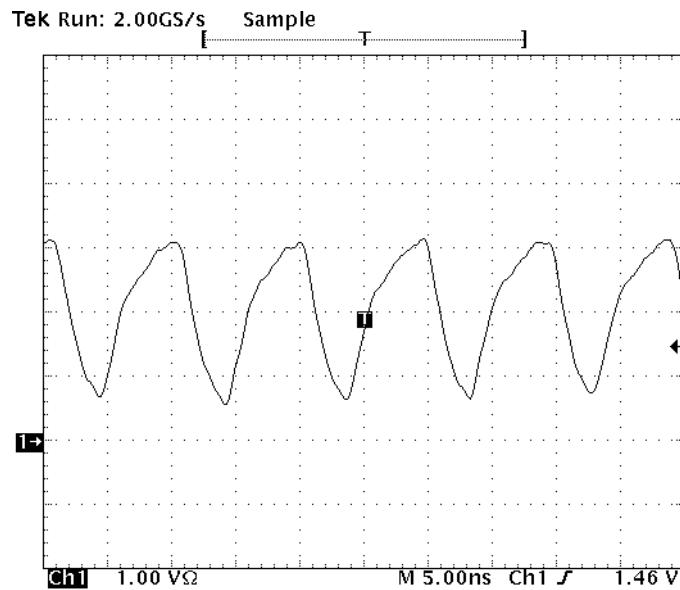
12. Check

Try both settings for the termination voltage:

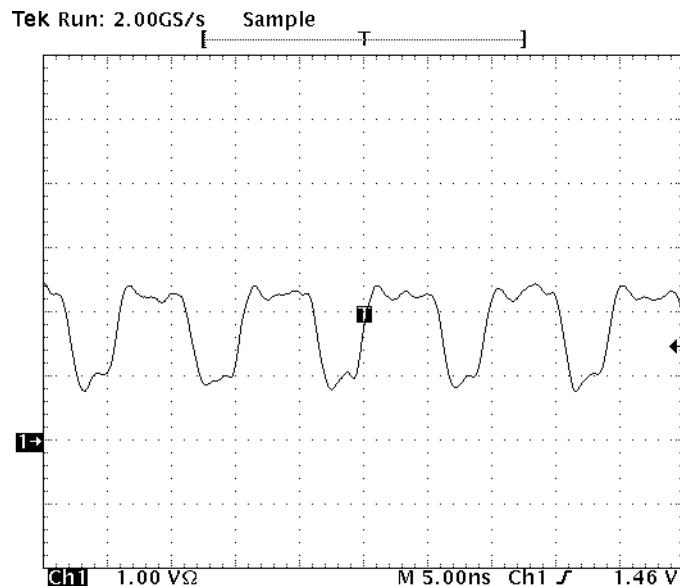
```
Trace.TERMination ON           ; Connect termination voltage during
                                ; trace capture
Trace.TERMination OFF          ; Always disconnect termination voltage
```

In most cases it is best to have the termination voltage connected during trace capture. However, for lower frequencies and/or weak output drivers, it can be helpful to disable the termination. This feature is supported by all preprocessors with AutoFocus technology.

Usually a not terminated signal will result in slower rise and fall times and it might have over and undershoots. For AutoFocus preprocessors, you will notice the slower rise and fall times by a reduced data eye size (white areas in the **Trace>ShowFocus** window):



The terminated signal, however, will have a reduced amplitude, “gravitating” towards 50% of the reference voltage on pin 12 of the target connector (see [Connector Layout](#)), as well as slightly faster rise and fall times:



13. Check

Is your TRACE32 software up to date?

VERSION . SOFTWARE

Check www.lauterbach.com/download_trace32.html or ask your local support team for an update or contact support@lauterbach.com.

1. Check

Make sure that you have completed the [basic check](#) list.

Especially do not forget to execute [Trace.AutoFocus](#) after setting up your trace port with the frequency you wish to trace and before attempting to trace signals.

Do you get any error or warning messages? For very high frequencies you may need to repeat this command until the hardware configuration is successful.

2. Check

What is your target voltage? For AutoFocus preprocessors with pluggable termination module: is the proper termination PCB plugged in?

For further information, see [External Termination PCB \(shipped before 2006\)](#).

3. Check

Use the PowerTrace-AF Diagnosis tool for preprocessors with AutoFocus technology to read in the current state of the driver software / preprocessor. (See [Access the Diagnosis Tool](#).)

After executing commands that might change the state of the preprocessor (e.g. [Trace.AutoFocus](#)), use the **Refresh** button to load current values from the low-level driver into the diagnosis window.

Press the **Measure** button if you want to repeat measurements like the target voltage, frequency etc.

The **Info** button will display even more detailed information and also generate a log file.



The trace clock frequency does not necessarily equal the CPU clock frequency. For example, a trace port operating with double data rate results in a trace clock frequency that is half the CPU clock frequency, another trace port operating with quad data rate results in a trace frequency that is a quarter of the CPU clock frequency. The frequency measurement might not be very accurate for frequencies below 20 MHz.

Clock and data reference voltage should normally be somewhere close to 50% of the target voltage.

Termination - Bus A / Bus B (unplugged indicates that no termination is connected).

Check the stability of your target voltage by pressing the **Measure** button a couple of times. Check the minimum and maximum values that were measured for that session by pressing the **Info** button:

Target Voltage	:	1882 mV
Minimum measured	:	1882 mV (minimum of 3 measurements)
Maximum measured	:	1882 mV (maximum of 3 measurements)

4. Check

TRACE32 differentiates between two views of data eyes:

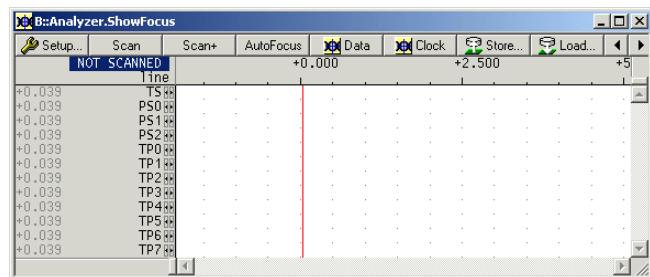
- Analog view of data eyes (check 5)
- Digital view of data eyes: Check 4 is only about the digital view of data eyes.

Trace data is sampled at clock edges, on rising only or rising and falling. They must be stable around the clock edge over a short time. This time window of stable data is reflected in so-called data eyes.

NOTE:	The data seen by the AutoFocus preprocessor depends highly from the clock threshold setup. Depending on the signal integrity, the data eyes are more or less wide open.
--------------	---

TRACE32 offers a powerful feature called **Analyzer.ShowFocus** to analyze the signal integrity of the trace port.

Analyzer.ShowFocus opens the digital **ShowFocus** window. **Analyzer.ShowFocus** is a kind of digital view of the data eyes.



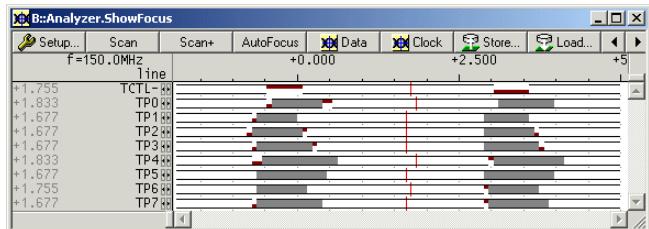
The horizontal axis reflects time line in nanoseconds. On the left, the current delay is shown for each trace signal. The red line shows the sampling point. It can be different for each signal. Data lines are delayed if values smaller than zero are set or not all sampling points are equal. In case of values larger than zero, the clock line is delayed.

NOTE:	The data delay is used to eliminate data skew. Clock delay is used to correct setup and hold times. Both delays types can be used at the same time.
--------------	---

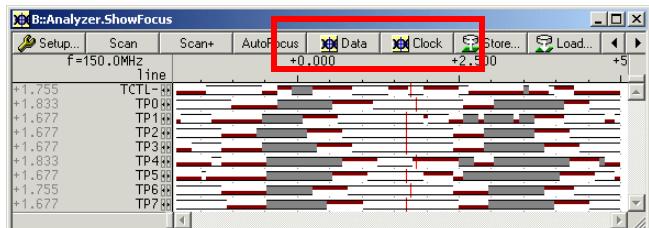
The horizontal axis shows all enabled signals listed. That means the number of signals depends on the setup of the trace port. The window needs to be re-opened after the trace port setup has changed.

5. Check

Pressing the **SCAN** button will execute **Analyzer.TestFocus** to update the window. In best case it should look like the following:



In this case no timing issues are expected. The sampling point should be placed in the middle of the white areas. In comparison to the following scan results, where more investigations are required:

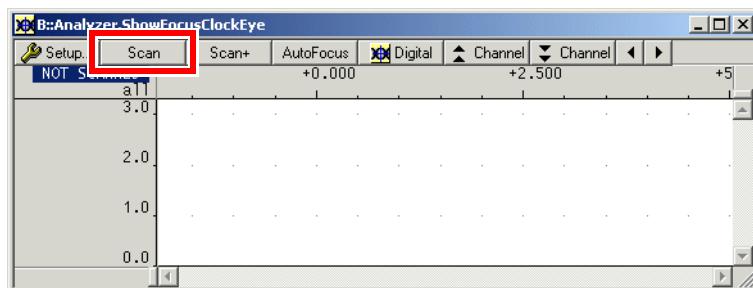
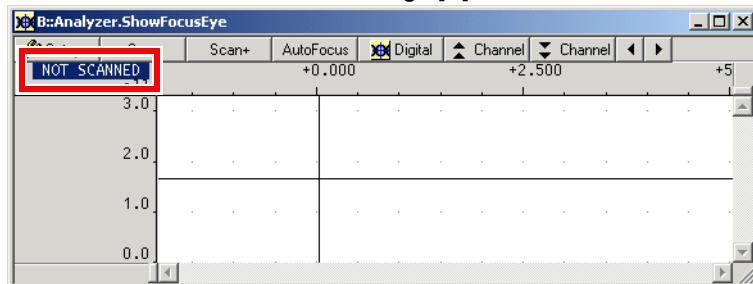


Color Legend:

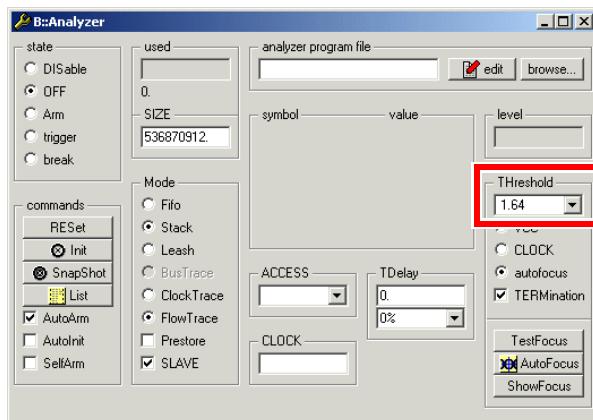
- **White** color indicates stable areas.
- **Red** horizontal lines mark setup violations on clock edges:
 - Red upper line: rising clock edge
 - Red lower line: falling clock edge
- **Gray** areas mean unstable data (setup violations) on rising and falling clock edges.

Click the buttons **Data** (**Eye** in newer versions) and **Clock** (**Clock Eye** in newer versions) to open [Analyzer.ShowFocusEye](#) and [Analyzer.ShowFocusClockEye](#).

The horizontal axis shows the voltage [V] now, the vertical axis is still the time [ns] axis:



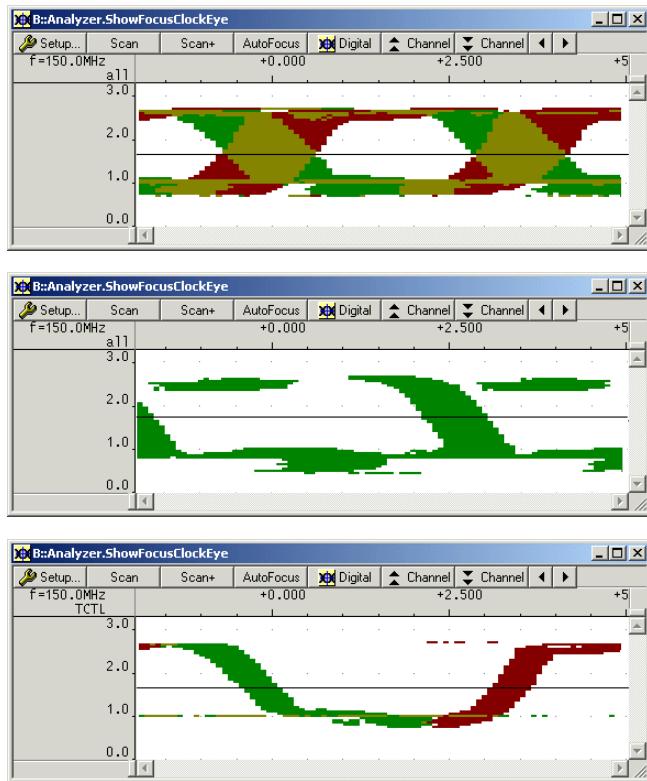
At the beginning, the message NOT SCANNED is shown. Before you click the **Scan** button, the data threshold should be set to the middle of the trace port I/O voltage:



If there is only one value, it applies to clock and data signals.

If there are two values, the 1st applies to the clock signal, the 2nd applies to the data signals.

First, you should take a look at the **Analyzer.The ShowFocusClockEye** window. The integrity of the clock signal is important for any further data eye analysis, because it is used for sampling. The result depends on the used test data (test pattern, target application) and may look like one of the following figures:

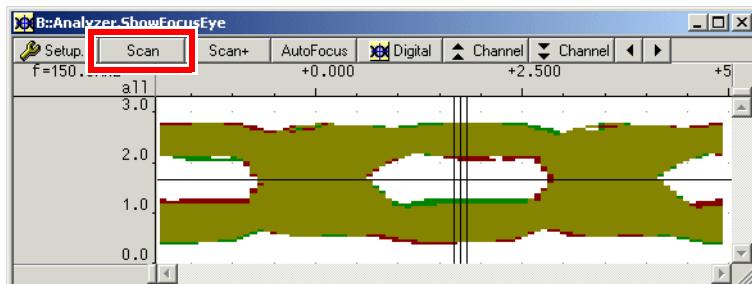


The graph does NOT show any data eyes, but can be interpreted in a similar way ([How to understand](#)).



The colored parts should define a white area, which can be enclosed or open on the upper/lower side. The clock threshold should be set to the middle of this area (as shown above).

Let's now take a look at the **Analyzer.ShowFocusEye** window. After the clock threshold is set, press the **Scan** button of the **Analyzer.ShowFocusEye** window. The result may look like the following:



The data threshold level should be set to the middle of the data eyes.



The graph shown in the **Analyzer.ShowFocusEye** window is similar to a scope measurement. It reflects the trace signal seen by the preprocessor. The values cannot be guaranteed, but give an suitable impression of the signal integrity.

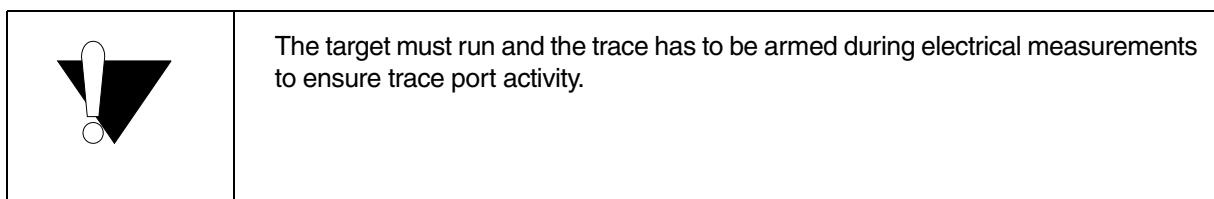
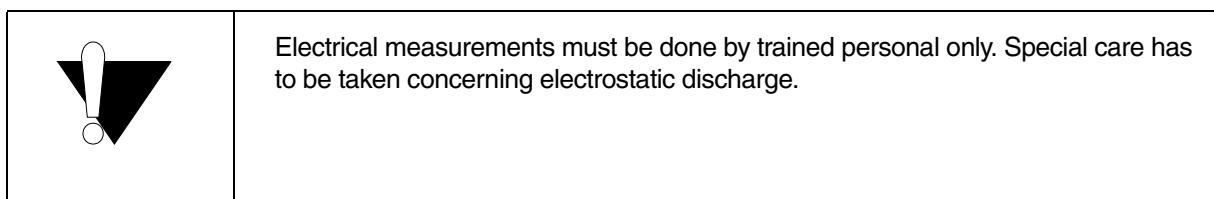
Protocol Check

Detailed trace data analysis is possible, if detailed information about the trace protocol is available. The following are examples for the ETM.

- [ETMv1.x](#)
- [ETMv3.x](#)

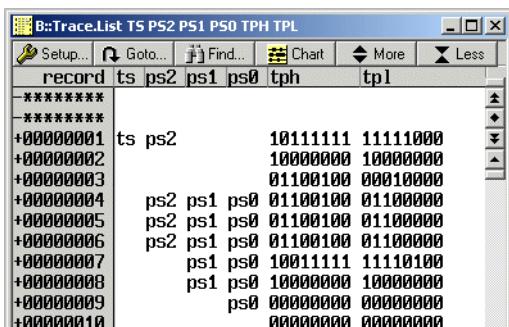
ETMv1.x

If the basic check was completed and the trace results are still not error-free, the following procedure might help you determine the source of the error. By manually disassembling the captured trace data (for a single step), you might notice swapped pins or data channels that might have timing issues.



To check the ETM trace signals - ETMv1.x:

1. Run the command [Trace.List TS PS2 PS1 PS0 TPH TPL](#) to display the captured trace data in a **Trace.List** window:



record	ts	ps2	ps1	ps0	tph	tpl
-*****						
-*****						
+00000001	ts	ps2			10111111 11111000	
+00000002					10000000 10000000	
+00000003					01100100 00010000	
+00000004			ps2	ps1	ps0 01100100 01100000	
+00000005				ps2	ps1 01100100 01100000	
+00000006					ps2 01100100 01100000	
+00000007					ps1 01011111 11110100	
+00000008					ps1 01000000 10000000	
+00000009					ps0 00000000 00000000	
+00000010					00000000 00000000	

Example: 16bit ETMv1.x port

ETMv1 signal	Description
TS	Trace sync signals are synchronization points for the TRACE32 software. Invalid states will cause Harderrors and Flerrors

ETMv1 signal	Description
PS2, PS1, PS0	Trace status signals correspond to pipeline states at execution time. Invalid states will cause Harderrors and could freeze TRACE32 for a while.
TP (TPH, TPL)	Trace packet signals hold information about data, address and program counter. TPL correspond to TP[7..0] and TPH correspond to TP[15..8]. Invalid values will cause Flowerrors

When using the option “ETM” ([Trace.List ETM](#)), all trace signals independent of the ETM version are shown. Although the format is slightly different to what we have seen previously.

This what our 16bit ETMv1.x port would look like, if we use [Trace.List ETM](#):

record	ts	ps0	ps1	ps2	tp0	tp1	tp2	tp3	tp4	tp5	tp6	tp7	tp8	tp9	tp10	tp11	tp12	tp13	tp14	tp15
+01098039		ps0	ps1	ps2			tp2				tp6	tp7								
+01098040		ps0	ps1	ps2			tp2				tp6	tp7								
+01098041	ts			ps2			tp2		tp4	tp5	tp6	tp7			tp10			tp14	tp15	
+01098042					tp0	tp1	tp2	tp3	tp4	tp5	tp6	tp7			tp10			tp14	tp15	
+01098043							tp2				tp6	tp7							tp15	
+01098044		ps0	ps1				tp2	tp3			tp6	tp7							tp15	
+01098045		ps0	ps1									tp7							tp15	
+01098046		ps0	ps1					tp3				tp7							tp15	
+01098047		ps0	ps1									tp7							tp14	
+01098048		ps0	ps1						tp3			tp7							tp14	
+01098049		ps0	ps1								tp6			tp10				tp14	tp15	
+01098050		ps0	ps1				tp2				tp6			tp10				tp14	tp15	

2. Enable the **AutoInit** mode ([Analyzer.AutoInit ON](#)) and do one single step to get a trace capture. The [Trace.List](#) window ([Trace.List TS PS2 PS1 PS0 TPH TPL](#)) for ETMv1.x will look similar to this:

record	ts	ps2	ps1	ps0	tp0	tp1

+00000001		ps2	ps1	ps0	01101110	10100000
+00000002		ps2	ps1	ps0	01101110	10100000
+00000003		ps2	ps1	ps0	01101110	10100000
+00000004	ts	ps2			11000100	11010000
+00000005					10000000	00000000
+00000006					11011101	00010000
+00000007					ps0	11011101
+00000008					ps1	ps0
+00000009					ps1	ps0
+00000010					ps2	ps1
+00000011					ps2	ps1
+00000012					ps2	ps1
+00000013					ps2	ps1
+00000014					ps2	ps1
+00000015					ps2	ps1

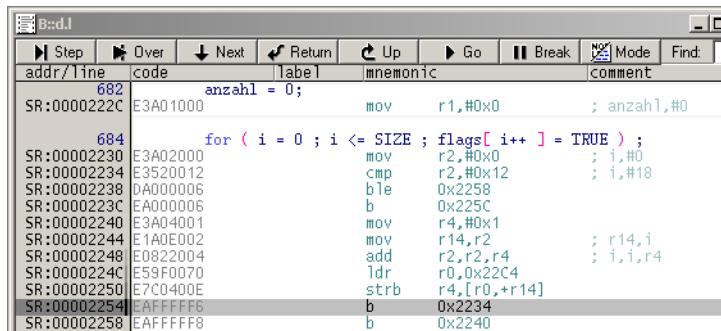
This trace synchronisation (ts) is similar for all single steps.

When looking for sources of error (e.g. pin swapping in your trace port pinout or timing issues), you may want to disassemble the captured data manually. This requires some knowledge of the trace protocol (for more information refer to the ETM architecture specification).

Below is an example of how to reconstruct the execution address from the traced data without knowing too much about the trace protocol. Note that 5 bytes are needed to transmit a 32-bit address on the trace port:

1 1000100 1 1010000 1 0000000 1 0000000 0 0010000	Trace packet bits can be structured as shown to the left. Each eighth bit is set to one if a packet follows and is cleared for the last packet.
1 1010000 1 1000100 1 0000000 1 0000000 0 0010000	Trace packet bits can be sorted as shown to the left.
101.0000 10.0010.0 0.0000.00 0000.000 001.0000	Dividing the row into groups of four bits (nibble) gives the address in binary format. Since the address is transmitted LSB first, this is done from right to left and top to bottom.
0101.0000 0010.0010 0000.0000 0000.0000 001	Now we write two nibbles per line again from right to left and top to bottom. Bits[6:4] of the fifth packet of a full branch address contain a reason code (here: 0x1 = "tracing has been enabled").
5 0 2 2 0 0 0 0 001	Finally the conversion to the hexadecimal format gives the full address 0x00002250.

3. Now compare this address to the one you would expect by looking at the **Data.List** window. Since you executed a single step you would expect the address before the current PC location to be the one transmitted on the trace port:



The screenshot shows the Data.List window with the following assembly code:

```

682    anzahl = 0;      mov    r1,#0x0    ; anzahl,#0
SR:0000222C E3A01000

684    for ( i = 0 ; i <= SIZE ; flags[ i++ ] = TRUE ) ;
SR:00002230 E3A02000    mov    r2,#0x0    ; i,#0
SR:00002234 E3520012    cmp    r2,#0x12   ; i,#18
SR:00002238 D4000006    ble    r2,0x2258
SR:0000223C EA000006    b     0x225C
SR:00002240 E3A04001    mov    r4,#0x1
SR:00002244 E1A0E002    mov    r14,r2   ; r14,i
SR:00002248 E0822004    add    r2,r2,r4   ; i,i,r4
SR:0000224C E59F0070    ldr    r0,0x22C4
SR:00002250 E7C0400E    strb   r4,[r0,+r14]
SR:00002254 EAFFFFF6    b     0x2234
SR:00002258 EAFFFFF8    b     0x2240

```

A red arrow points to the last instruction, `b 0x2240`, with the label "Program counter (PC)".

4. You may need to perform a couple of single steps to see:

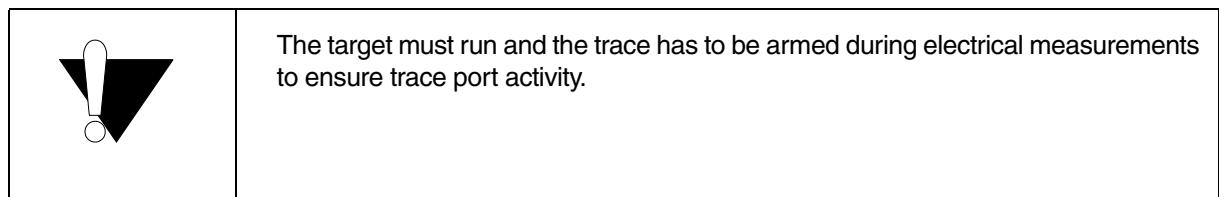
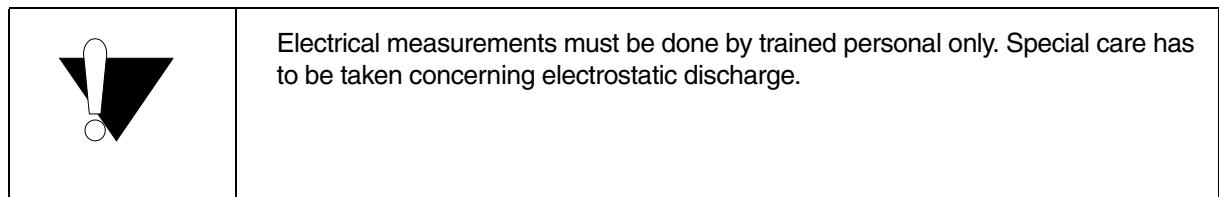
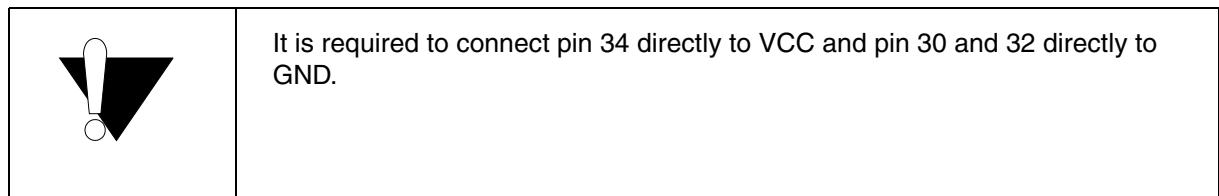
- (a) if there is a logical error (pairs of trace signals are swapped) indicating an erroneous pinout or
- (b) if the errors seem to be related to certain trace signals being wrong every now and then. For (b), timing violations could be the problem.

All AutoFocus preprocessors can adjust the sampling points only if their **timing requirements** are satisfied. In addition, the maximum operating frequency has to be considered, see tables in **“AutoFocus Preprocessor Hardware Versions”**, page 60.

ETMv3.x

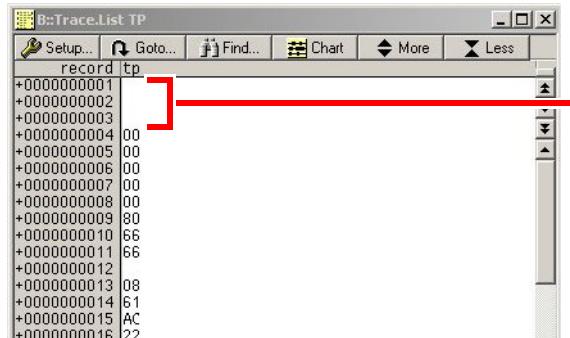
If the basic check was completed and the trace results are still not error free, the target connector pinout should be checked again (**ETMv3**). The following procedure might help you determine the source of the error. By manually disassembling the captured trace data (for a single step), you might notice swapped pins or data channels that might have timing issues.

Pin 34, 36, and 38 of the connector are handled by the trace hardware as normal data signals. That means these signals will also be affected by the termination circuitry.



To check the ETM trace signals - ETMv3.x:

1. Run the command **Trace.List TP** to display the captured trace data in a **Trace.List** window:

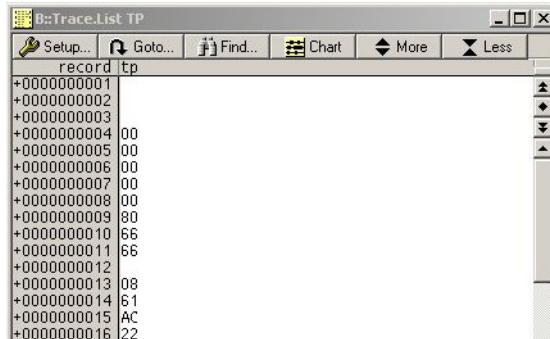


Gaps indicate that there is no valid trace data.
(TCTL signal = 1)

Example: 8bit ETMv3.x port

ETMv3 signal	Description
TCTL	Trace control signal defines valid TP packets. The signal is low active. Invalid states will cause Harderrors and Flowerrors.
TP	Trace packet signals hold information about data, address and program counter. Invalid values will cause Flowerrors.

2. Enable the **AutoInit** mode (**Analyzer.AutoInit ON**), and then do one single step to get a trace capture. The **Trace.List** window will look similar to this:



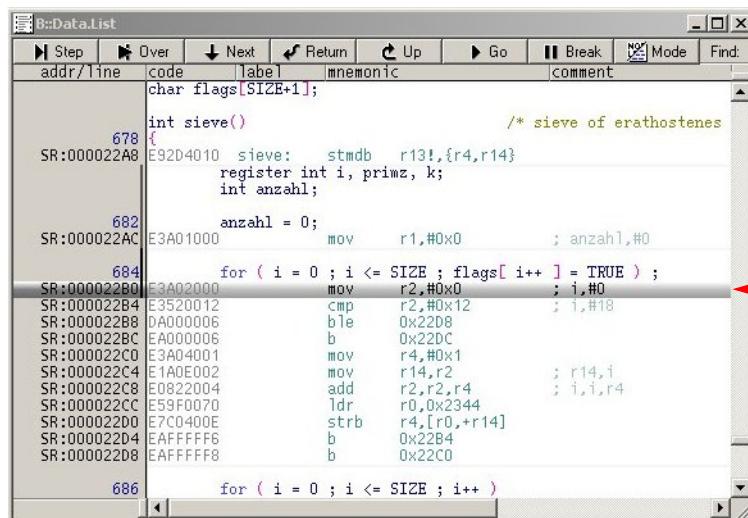
When looking for sources of error (e.g. pin swapping in your trace port pinout or timing issues), you may want to disassemble the captured data manually. This requires some knowledge of the trace protocol (for more information refer to the ETM architecture specification).

Below please find an example of how to reconstruct the execution address from the traced data without knowing too much about the trace protocol:

00 00 00 00 00 80	synchronisation sequence (ISYNC)
66 66	idle sequence (IDLE)
08 61	address synchronisation sequence (ASYNC)
AC 22 00 00	address (0x000022AC)
84 D9 A2 80 80 B8	program execution information sequence (PHEADER)

3. Now compare this address (in our example: 0x000022AC) to the one you would expect by looking at the [Data.List](#) window.

Since you executed a single step you would expect the address before the current program counter location (PC) to be the one transmitted on the trace port:



The screenshot shows the B:Data.List window with the following content:

addr/line	code	label	mnemonic	comment
678	char flags[SIZE+1];			
SR:000022A8	int sieve()			/* sieve of erathostenes
682	{			
SR:000022AC	stmdb r13!,{r4,r14}			
	register int i, primz, k;			
	int anzahl;			
684	anzahl = 0;			
SR:000022B0	mov r1,#0x0			; anzahl,#0
	for (i = 0 ; i <= SIZE ; flags[i++] = TRUE) ;			
686	E3A02000			
SR:000022B4	mov r2,#0x0			; i,#0
SR:000022B8	E3520012			cmp r2,#0x12 ; i,#18
SR:000022BC	DA000006			ble 0x22D8
SR:000022C0	EA000006			b 0x22DC
SR:000022C4	E3A04001			mov r4,#0x1
SR:000022C8	E1AOE002			mov r14,r2 ; r14,i
SR:000022CC	E0822004			add r2,r2,r4 ; i,i,r4
SR:000022D0	E59F0070			ldr r0,0x2344
SR:000022D4	E7C0400E			strb r4,[r0,+,r14]
SR:000022D8	EAFFFFF6			b 0x22B4
	EAFFFFF8			b 0x22C0
686	for (i = 0 ; i <= SIZE ; i++)			

4. You may need to perform a couple of single steps to see:

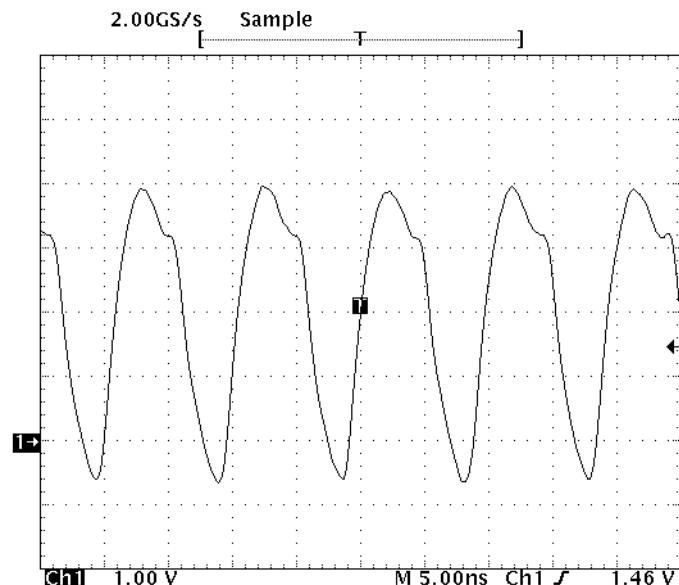
- (a) if there is a logical error (pairs of trace signals are swapped) indicating an erroneous pinout or
- (b) if the errors seem to be related to certain trace signals being wrong every now and then. For (b), timing violations could be the problem.

All AutoFocus preprocessors can adjust the sampling points only if their [timing requirements](#) are satisfied. In addition, the maximum operating frequency has to be considered, see tables in [“AutoFocus Preprocessor Hardware Versions”](#), page 60.

Timing Requirements

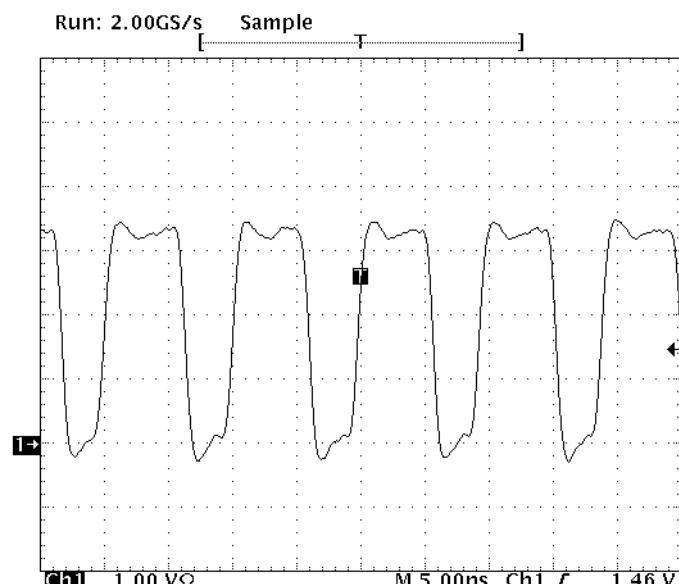
If you suspect timing issues to be the source of the error, then take a closer look at the setup and hold times of your trace port channels. An oscilloscope (2 channels, bandwidth >1000 MHz, >10 GS/s, probe <5pF) is required for the following measurements.

Make sure that you use short ground connections. The following two screenshots show the influence of probe and ground connection length on the clock wave form:



The graph shows a typical clock wave form.

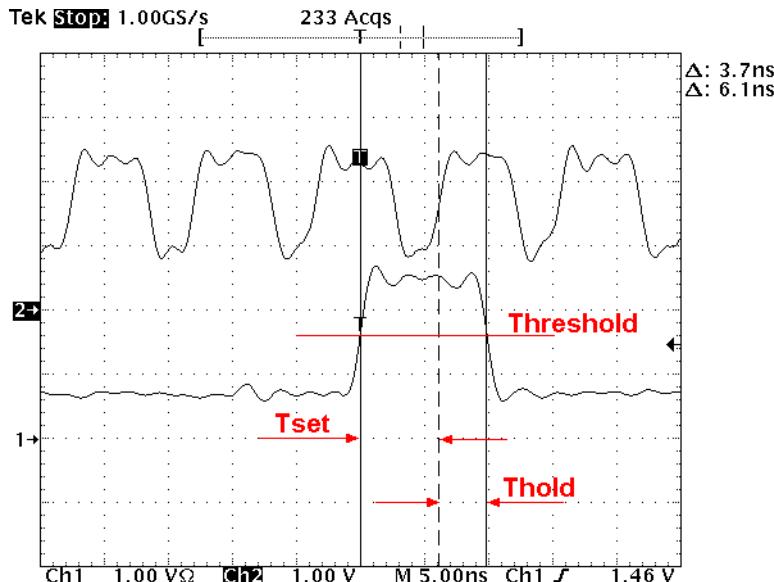
Long GND connection, Probe: 500 MHz, 8 pF, 10 MΩ, x10



The graph shows the same clock wave form.

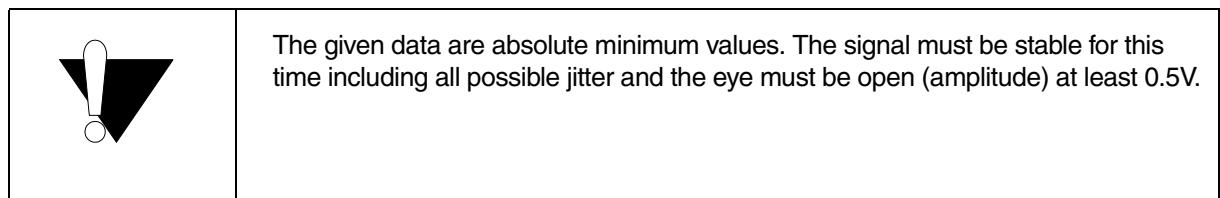
Short GND connection, Probe: 750 MHz, < 2 pF, 1 MΩ, x10

Measure the setup and hold times for all data channels, which will also give you the maximum channel-to-channel skew. The AutoFocus preprocessor can handle data skew, but a minimum setup and hold time slot has to be fulfilled for all channels. The following picture shows how to measure setup/hold times.



The threshold level should be set to the middle of the trace signal. The following table shows setup and hold time requirements for different AutoFocus preprocessors.

AUTOFOCUS	$T_s + T_h \geq 1.2 \text{ ns}$
AUTOFOCUS II	$T_s + T_h \geq 0.6 \text{ ns}$



The AutoFocus II technology is very powerful, if it comes to configuring itself for different target trace ports. Usually you do not need to worry about setup and hold times for data rates as high as 200 Mbit/s, although it is always a good idea to keep the channel-to-channel skew low and the clock duty cycle close to 50/50. Depending on your trace port data rate the maximum channel-to-channel skew has to stay below:

$$\text{ch2ch_skew} \leq t_{\text{period}} - (T_s + T_h)$$

For instance, a trace port with a data rate of 300 Mbaud/s should have no more than $(1/300 \text{ MHz} - 0.6 \text{ ns}) = 2.73 \text{ ns}$ channel-to-channel skew. In addition the maximum operation frequency has to be considered, see tables in "[AutoFocus Preprocessor Hardware Versions](#)", page 60.

Choose the Best Termination PCB Type

If your AutoFocus preprocessors has termination daughter PCBs, you may wish to find out, which of the two termination PCB types suits your needs best. To help you choose the best termination PCB type, TRACE32 can print some data eye statistics to the **AREA** window.

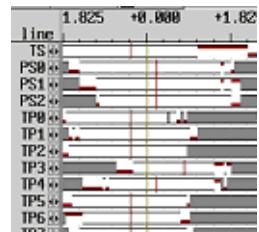
To choose the best termination PCB type:

1. Execute the [Analyzer.AutoFocus](#) command.
2. On the TRACE32 toolbar, click the **AF Diagnosis** button. (See [Access the Diagnosis Tool.](#))
3. Click the **Info** button of the **PowerTrace-AF Diagnosis** tool.

The diagnosis is displayed in the **AREA** window. Here is an example for a 1.8 V target.

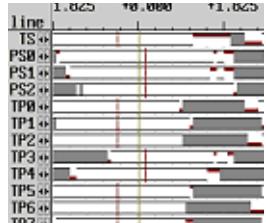
- Complete range version (1.5 ... 5 V)

```
Active channels : 12
Max. possible sampling points: 1812
Possible setup violations :
BUS0 (rising edge) : 674 sampling points
BUS0 (falling edge) : 658 sampling points
36 percent of implemented time window has possible setup violations
```



- Low-voltage version (1.5 ... 2.5 V targets)

```
Active channels : 12
Max. possible sampling points: 1812
Possible setup violations :
BUS0 (rising edge) : 573 sampling points
BUS0 (falling edge) : 542 sampling points
30 percent of implemented time window has possible setup violations
```



The low-voltage version has less setup violations, so the data eyes are broader and easier to sample. Hence it is expected to be able to handle higher frequencies than the complete range version for that particular target.

How to understand A.ShowFocusEye and A.ShowFocusClockEye

The **Analyzer.ShowFocusClockEye** window is frequently confused with the **Analyzer.ShowFocusEye** window. Both windows show similar graphs, but contain absolutely different information.

NOTE:

The dimensions of both, the data eyes and the clock waveform, do highly depend on the clock threshold level.

The following figures show the resulting sample clock signal in dependency of the clock threshold level setup:

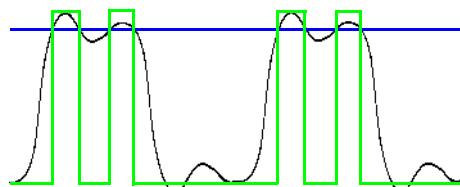


Fig.A

Target clock
Clock threshold
Resulting sample clock

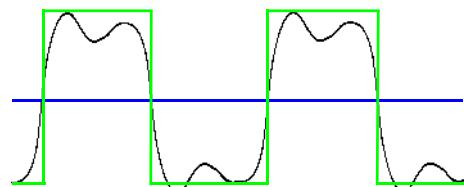
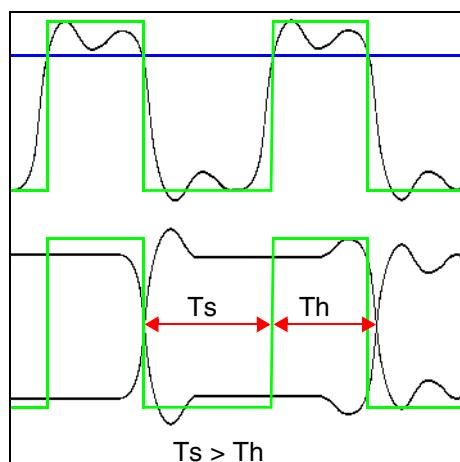


Fig.B

The threshold can be set manually using the command **Analyzer.ThresHold <value>** or automatically with the command **Analyzer.ThresHold Clock**. The algorithm attempts to find a threshold level close to a duty cycle of 50/50 as seen in the figure B.

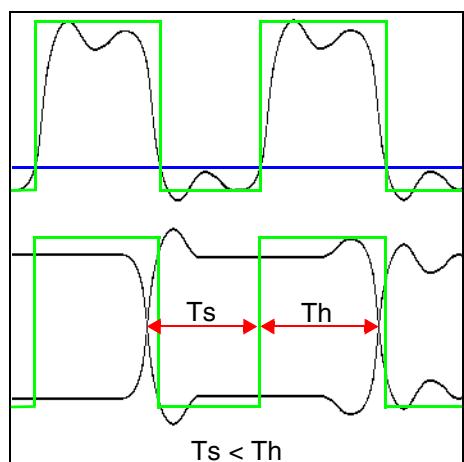
Important to know: The clock threshold level influences the data setup (Ts) and hold (Th) times in dependency of the clock waveform. The higher the trace port data rates are, the more attention you must pay to this effect.



Target clock
Clock threshold
Sample clock

Data
Timing
Sample clock

$Ts > Th$

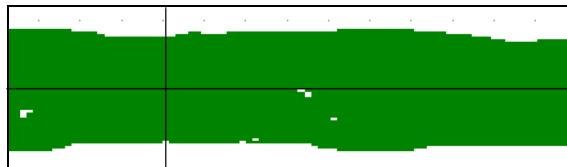


$Ts < Th$

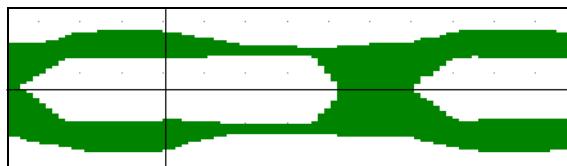
NOTE:

It is possible to change the data setup and hold times through the clock threshold level. But this is not the right way to compensate for data or clock skew. However, this effect sometimes is the last option to get a suitable trace listing.

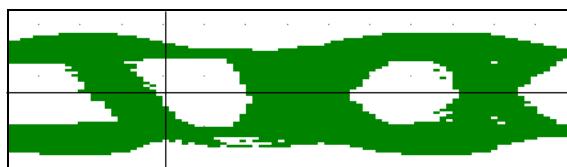
With the background information provided above, it is easy to understand why data eyes change with the clock threshold level. The following scans were made with different clock threshold levels. The **Analyzer.ShowFocusEye** graphs show the same data signal.



Clock threshold at 0.8V

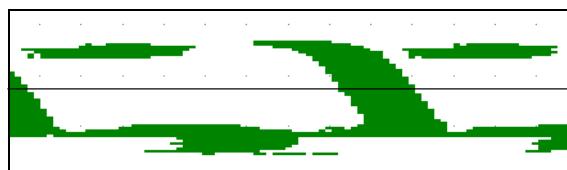


Clock threshold at 1.7V
This setup would give suitable trace results.



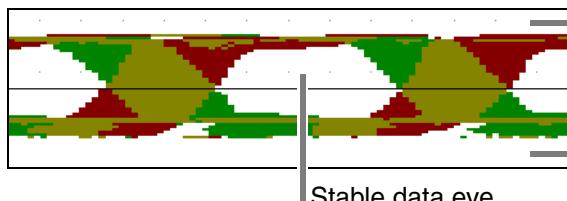
Clock threshold at 2.6V

The corresponding **Analyzer.ShowFocusClockEye** window looks like the following:



Sampling on rising clock edges

On the other targets, e.g. ETMv3, the graph may look like this:

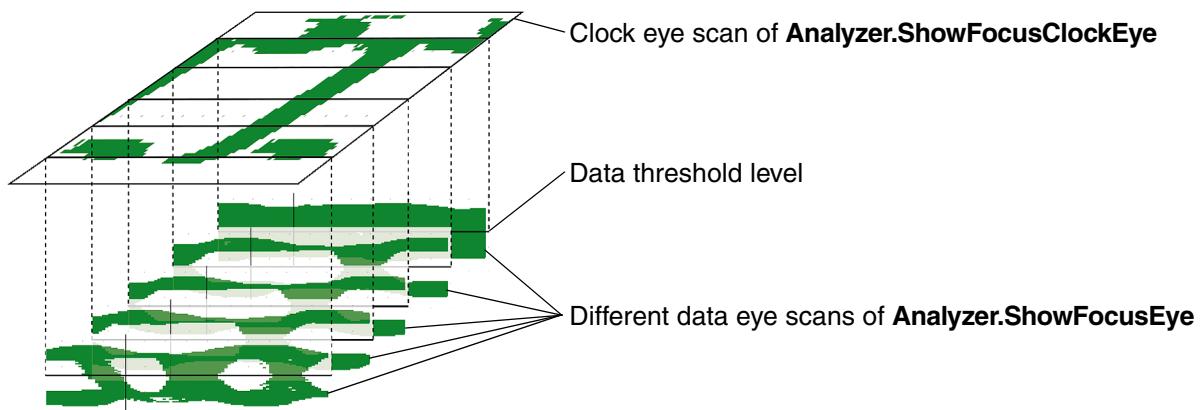


Sampling on both clock edges

Color Legend:

- **Green** and **red** - Data is instable because of setup-time violations and/or hold time violations of the input register:
 - Green color means sampled on rising clock edge.
 - Red color means sampled on falling clock edge.
- **White** color indicates stable areas.
- **White** areas enclosed by green and/or red color are referred to as stable data eyes.
- **Olive green** areas mean instable data (setup violations) on rising and falling clock edges.

The correlation between **Analyzer.ShowFocusEye** and **Analyzer.ShowFocusClockEye** is shown with the following 3D-figure:



On the lower part of the figure, different data eye scans are cascaded. The data threshold level and the sampling point are marked by black lines. The data threshold line also defines the level where the clock eye picture would be normally placed within this graph.

Summary:

The clock eye scan shows a third dimension of multiple data eye scans.

- The scan of **Analyzer.ShowFocusEye** keeps the clock threshold constant while the data threshold is changed.
- The scan of **Analyzer.ShowFocusClockEye** keeps the data threshold constant while the clock threshold is changed.

Recommendations for Target Board Design

- Place the trace connector close to the target processor. Make sure that the ground pins are connected to your target's ground plane (MICTOR or Samtec).
AMP part numbers for MICTOR connectors (designed for $50\ \Omega$ systems) can be found under: www.lauterbach.com/adetmcon.html
For information on MICTOR connectors please refer to: www.amp.com
Samtecs part numbers for QSH connectors (designed for $50\ \Omega$ systems) can be found on: www.lauterbach.com/adetmmipi60.html
For information on Samtec connectors please refer to: www.Samtec.com
Additional information on flex extension cables can be obtained by contacting support@lauterbach.com.
- Consider setup and hold time requirements. AutoFocus preprocessors are able to de-skew data and clock; however, it is important to meet their minimum **timing requirements**.
- Match your traces to reduce channel-to-channel skew: same length, same PCB layer and same amount of vias for all traces. Avoid stubs. Ideally traces should have a $50\ \Omega$ impedance, but values between $50\ \Omega$ and $100\ \Omega$ are suitable.
- Avoid close parallel routing of the trace bus. The closer the tracks the more crosstalk will influence the signal integrity. Special routing (meandering - used for length equalization) should be used to prevent large areas of parallel track placement.
Even at higher data rates this is an important point.
- An adequate number of bypass capacitors are crucial to keep the supply voltage stable when the trace port is driven by your application. This is especially important for port sizes greater 8 bit and high frequencies. If your supply voltages are not stable, the target processor might assume illegal JTAG tap states and the trace port analyzer (TPA) might lose control over the target (typically this might result in messages such as "**emulation debug port fail**").
To save cost, just allocate additional footprints for bypass capacitors in your PCB layout. That way you can mount additional bypass capacitors for the development PCBs that have to drive TPAs and omit them for the production PCBs.
- Capacitor vias should never be shared, each capacitor requires its own vias close to the pads.
- A series termination on the target is usually not required. All trace channels are terminated on the Preprocessor to 50% of the voltage level of pin 12 of the trace connector (Mictor and Samtec).
If desirable, footprints for a series termination on the target can be implemented and mounted with $0\ \Omega$. If necessary, advanced users can implement a series-parallel combination.
- Trace reference voltage (Vref-Trace) of the trace port connector has to be on the same voltage level as the I/O level of the trace port's clock and data channels. JTAG reference voltage (Vref-Debug) has to be on the same voltage level as the I/O level of the JTAG signals (usually the same as Vref-Trace). Again: The termination voltage of the TPA will be 50% of VrefTrace.

- The target voltage has to be within the specified range, see “[AutoFocus Preprocessor Hardware Versions](#)”, page 60.
For other voltage levels, contact support@lauterbach.com.

NOTE: For very high data rates (> 300 Mbit/s) 1.8 V signals are usually easier to support by the target than higher voltage levels.

- Recommendations for output drivers:

If possible, keep output drive strength and slew rate programmable (e.g. 8, 12, 16 mA for both low and high slew).

NOTE: For very high data rates (> 300 Mbit/s) 12mA drivers should be used to get sharp edges and sufficient data eye opening.

If you cannot support high driver strength for all trace signals, then at least make an investment in the clock signal. Usually the clock signal is the bottleneck, so the clock signal should have at least 6 mA, better 8 ... 12 mA, drive strength. Whenever possible, you should also use double data rate clocking to divide the clock frequency by two.

Keep in mind, that the drive strength of the output buffers has to be supported by proper PCB layout and adequate number of bypass capacitors for all frequency ranges!

Your trace port reference voltage should not show any significant ripple, even if signals are traced by the TPA.

Functional Description - Pin Remapping

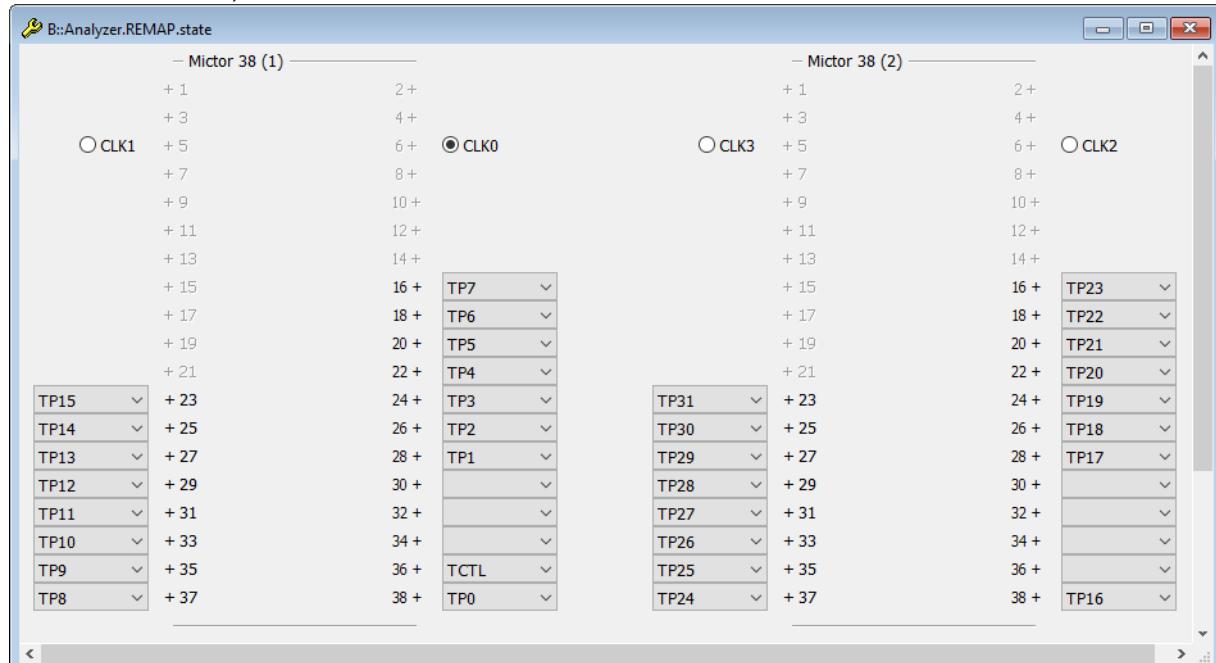
NOTE:

The examples in this chapter refer to the pin remapping capability of the AutoFocus II (trace ID 0x71, 0x78) and AutoFocus II MIPI (trace ID 0x72) preprocessors.

Any other preprocessors do not support pin remapping!

The actual pinout of the preprocessors is determined by several parameters, e.g. the CPU selection, the port type or port size.

In order to display the current trace port configuration, just type **Analyzer.REMAP.state** into the TRACE32 command line. The following window will open if the AutoFocus II is attached (a similar window will open for the AutoFocus MIPI):



Example: AutoFocus II pinout for 32-bit ETMv3

Each physical pin of the preprocessor's connector is represented by the corresponding pin number in the diagram.

In case your target board does not comply with one of the preconfigured pinout configurations, you are usually required to buy or create an adaption board. But in case only the data and control pins (excluding the clock signal) need to be remapped, you can simply reconfigure the preprocessor's pinout by selecting the adequate channel from the drop-down list.

Example 1:

Move channel 'TP1' from pin #28 to pin #30.

Select 'TP1' from the drop-down list next to pin #30. The updated window will look like this:

TP15	+	23	24 +	TP4
TP14	+	25	26 +	TP3
TP13	+	27	28 +	TP2
TP12	+	29	30 +	TP1
TP11	+	31	32 +	
TP10	+	33	34 +	
TP9	+	35	36 +	TCTL
TP8	+	37	38 +	TP0

Example 2:

Move channel 'TP1' from pin #28 to pin #26.

Select 'TP1' from the drop-down list next to pin #26, which is already assigned to channel 'TP2'. In this case 'TP2' will be mapped to the previous location of 'TP1' automatically (that is 'TP1' and 'TP2' will be swapped):

TP15	+	23	24 +	TP4
TP14	+	25	26 +	TP3
TP13	+	27	28 +	TP1
TP12	+	29	30 +	TP2
TP11	+	31	32 +	
TP10	+	33	34 +	
TP9	+	35	36 +	TCTL
TP8	+	37	38 +	TP0

The remapping of channels can be embedded in a PRACTICE script using the [Analyzer.REMAP.PIN](#) command. The commands to reproduce the results from example 1 and example 2 are:

```
Analyzer.REMAP.PIN TP1 30. ; Example 1
Analyzer.REMAP.PIN TP1 26. ; Example 2
```

NOTE:

The names of the channels are not static but may vary according to the CPU selection and trace port type. In case a target board or chip has got two physical trace ports, switching on or off a trace port may also influence the preprocessor's pinout and channel configuration.

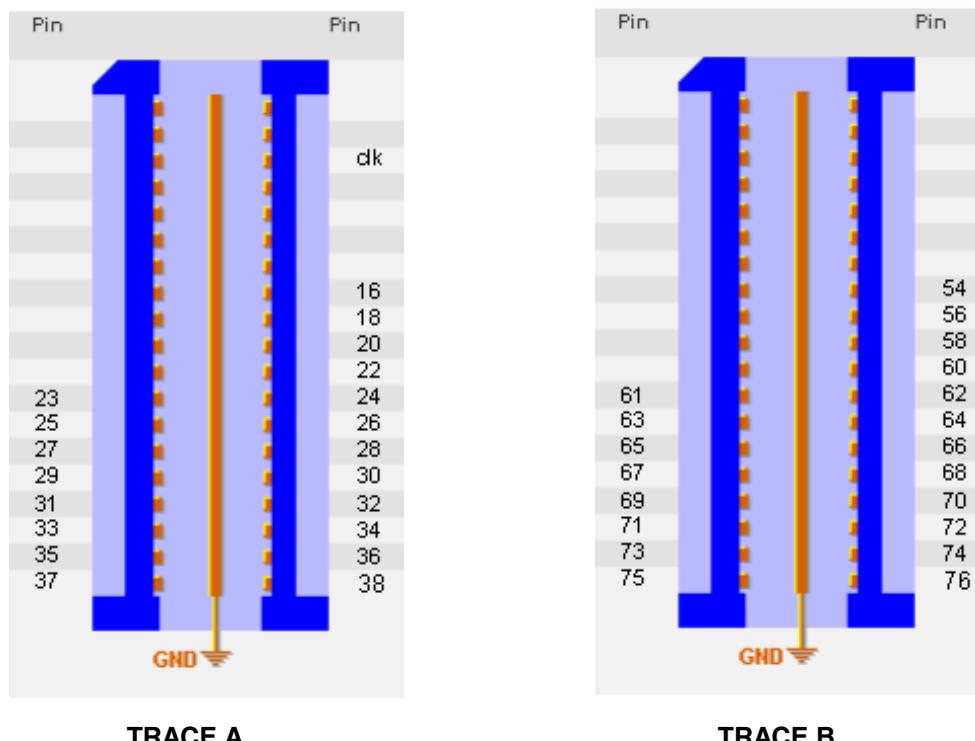
Useful list of commands which have an influence on the trace port configuration:

SYStem.CPU	Defines the trace port's version (e.g. ETMv1 or ETMv3).
ETM.PortSize	Number of data pins (also applicable for other trace ports like STM or RTP).
ETM.ON/OFF	Switches to (or from) a different trace port configuration if there is more than one physical trace port (also applicable for other trace ports like STM or RTP).
Analyzer.PortType	Defines or overrides a trace port configuration.
Analyzer.REMAP.PIN	Remaps pins.

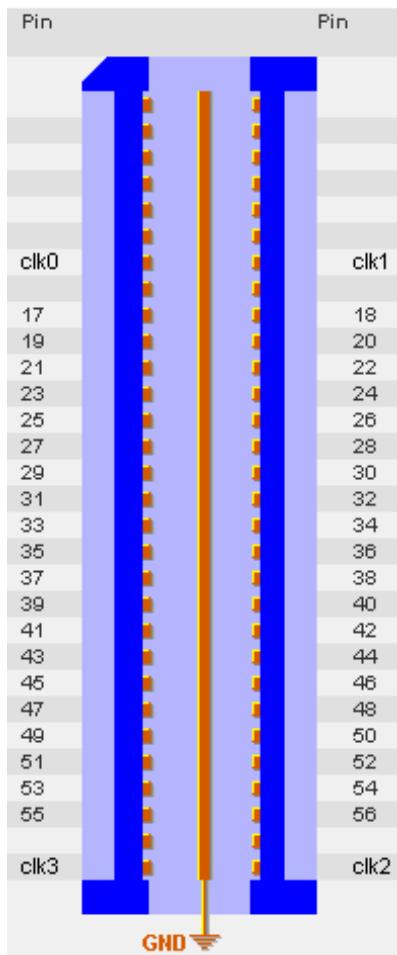
Pinouts

AutoFocus II (LA-7992)

Layout of the two Mictor38 connectors and declaration of the remappable pin numbers to be used with the **Analyzer.REMAP.PIN** command:



Layout of the Samtec MIPI 60 connector and declaration of the remappable pin numbers to be used with the [Analyzer.REMAP.PIN](#) command:



- [AutoFocus Preprocessor Hardware Versions](#)
- [Electrical Specification for AutoFocus II Preprocessors](#)

AutoFocus Preprocessor Hardware Versions

You can identify the preprocessor version by typing **VERSION.HARDWARE** into the TRACE32 command line, or compare your preprocessor with the pictures in these sections:

- [“Preprocessor AutoFocus I \(LA-7991\)”, page 8](#)
- [“Preprocessor AutoFocus II \(LA-7992\)”, page 11](#)
- [“Preprocessor AutoFocus MIPI II \(LA-7993\)”, page 13](#)

Preprocessor versions and a description of the main differences are described in the following tables:

- [“LA-7991”, page 62](#)
- [“LA-7992 and LA-7993”, page 63](#)

Product number	LA-7991	LA-7991	LA-7991
TRACE32 name	AutoFocus I (OTP)	AutoFocus I	AutoFocus I
TRACE32 ID	70 (OTP)	70	70
Delivery year	11/2004-12/2005	06-09/2005	since 08/2005
Serial number	yes	yes	yes
Supported target voltage range	1.8 ... 5 V	1.8 ... 5 V	1.8 ... 3.3 V
Housing	yes	yes	yes
Number of ribbon cables	3	3	3
Maximum data rate	300 MBaud/s	350 MBaud/s	350 MBaud/s
Max. operating frequency	300MHz	350MHz	350MHz
Input delay resolution	-	-	480 ps
Termination	pluggable	pluggable	47 Ω Thevenin
Threshold level	0..3.3 V programmable	0..3.3 V programmable	0..3.3 V programmable
Connectorization	MICTOR, 2*38pin	MICTOR, 2*38pin	MICTOR, 2*38pin
Data eye height of input signals	> 0.5 V	>0.5V	>0.5 V
Data eye width of input signals	>1.2ns	>1.2ns	>1.2ns

Product number	LA-7992	LA-7992	LA-7993
TRACE32 name	AutoFocus II	AutoFocus II V2	AutoFocus II MIPI
TRACE32 ID	71	78	72
Delivery year	since 2006	since 11/2012	since 2010
Serial number	yes	yes	yes
Supported target voltage range	1.8 ... 3.3 V	1.8 ... 3.3 V	1.8 ... 3.3 V
Housing	yes (with fan)	yes (with fan)	yes (with fan)
Number of ribbon cables	3	3	3
Maximum data rate	500 MBaud/s	>600 MBaud/s	>600 MBaud/s
Max. operating frequency	300MHz	350MHz	350MHz
Input delay resolution	78ps	78ps	78ps
Termination	47 Ω Thevenin	100 Ω Thevenin	47 Ω Thevenin
Threshold level	0..3.3 V programmable	0..3.3 V programmable	0..3.3 V programmable
Connectorization	MICTOR, 2*38pin	MICTOR, 2*38pin	QTH, 60pin
Data eye height of input signals	> 0.5 V	> 0.5 V	> 0.5 V
Data eye width of input signals	>0.3ns	>0.3ns	>0.3ns

Electrical Specification for AutoFocus II Preprocessors

Electrical Characteristics

Programmable Threshold	0.01-3.3V
Vih	Threshold+0.25V
Vil	Threshold-0.25V
Required Data Eye Opening	>0.5V (also with termination enabled)
Clock/Data	1.2-3.3V
VTref	1.2-3.3V
Switchable Active Termination	100 Ohms to VTref/2 (Data) 100 Ohms to VTref/2 (Clock)

Absolute Maximum Ratings

Clock/Data	0-3.6V
VTref	0-5V

Switching Characteristics

Clock frequency	0..300MHz
Data rate	600Mbaud (<=16bit) 300Mbaud (17..32bit)

Absolute Maximum Ratings

Maximum clock frequency	350MHz
Data rate	700Mbaud (<=16bit) 350Mbaud (17..32bit)

Timing Characteristics

Data setup time	0.3ns
Data hold time	0.3ns

Absolute Minimum Ratings

Data setup time + Data hold time	$\geq 0.3\text{ns}$
---	---------------------

Average time resolution

Clock channel	78 ps
Data channels	78 ps

Maximum channel delay

Clock channel	6.4 ns
Data channels	3.3 ns

Additional Data

PCB

Track width	178um
Track thickness	17um CU
Pad thickness	17um CU + 10um NiAu
Track clearance	>178um
Er	4.7
Material	FR4

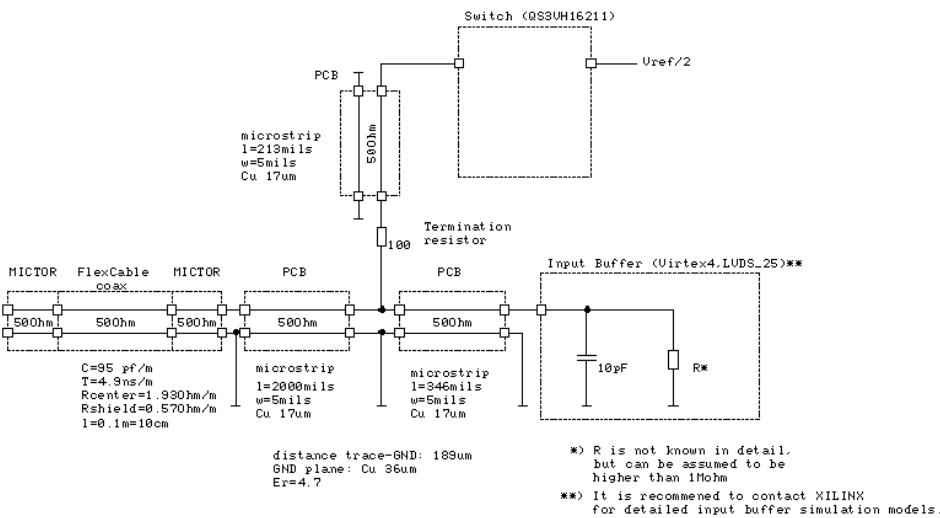
Components

AMP MICTOR connector	767006-1
Samtec	QTH 60pin

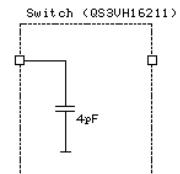
Simulation Model

Autofocus V2

TERMINATION ON

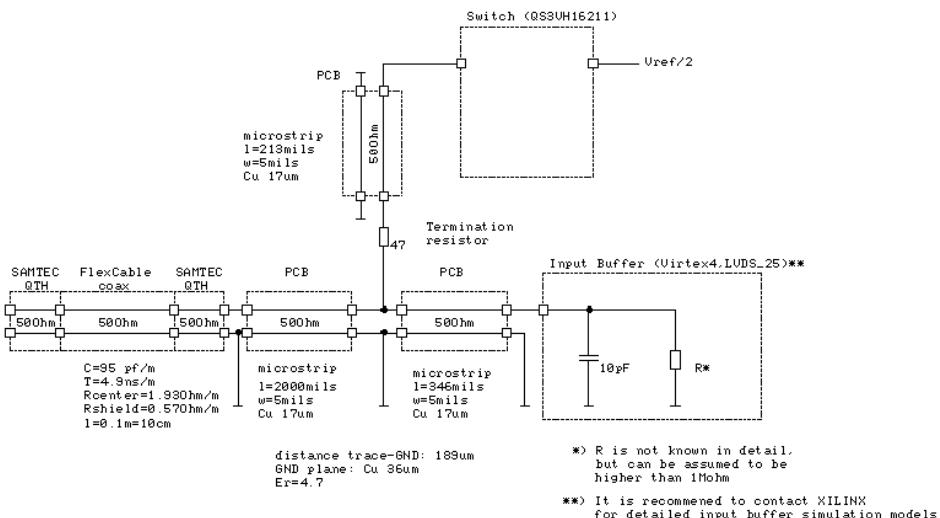


TERMINATION OFF

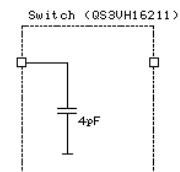


Autofocus Mipi

TERMINATION ON



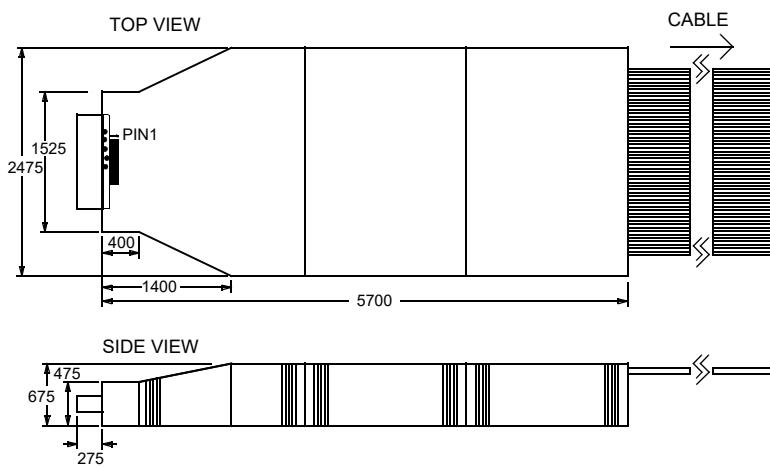
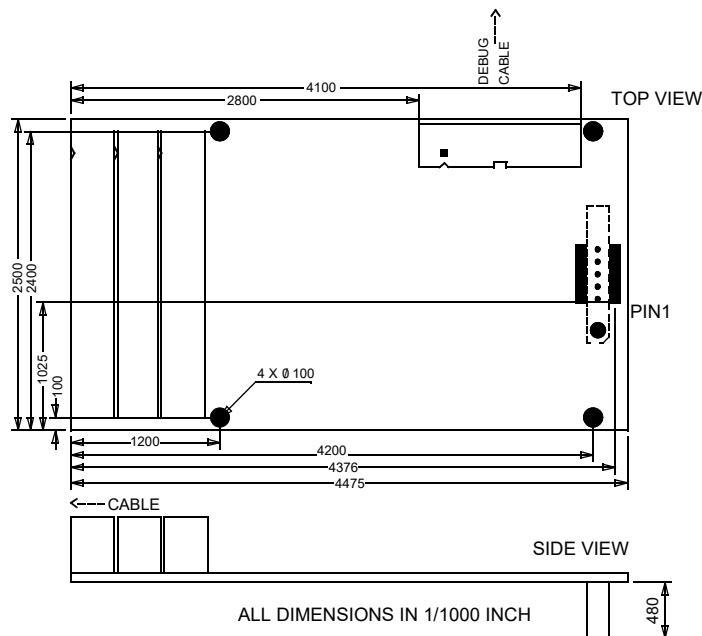
TERMINATION OFF



Dimensions

Dimension

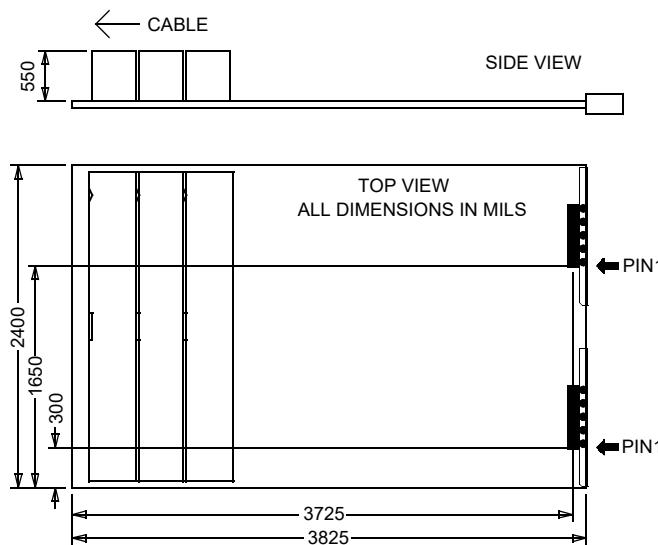
LA-7921 PP-ARM-ETM/200



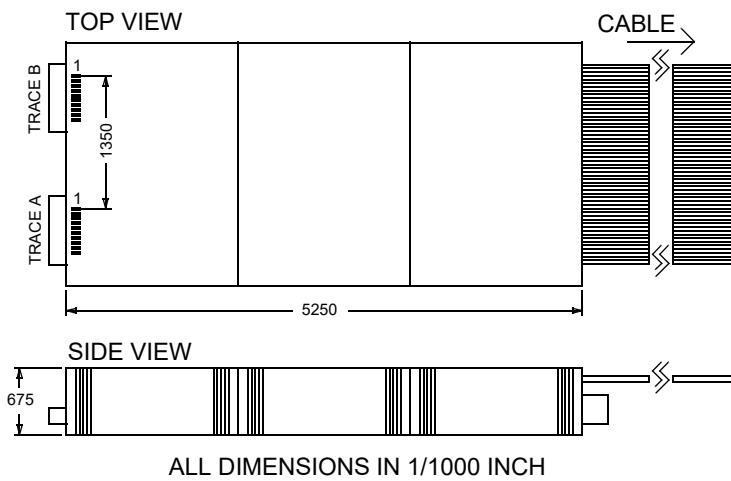
ALL DIMENSIONS IN 1/1000 INCH
new case, delivery from December 2003

Dimension

LA-7923 PP-ARM-ETM-TWO

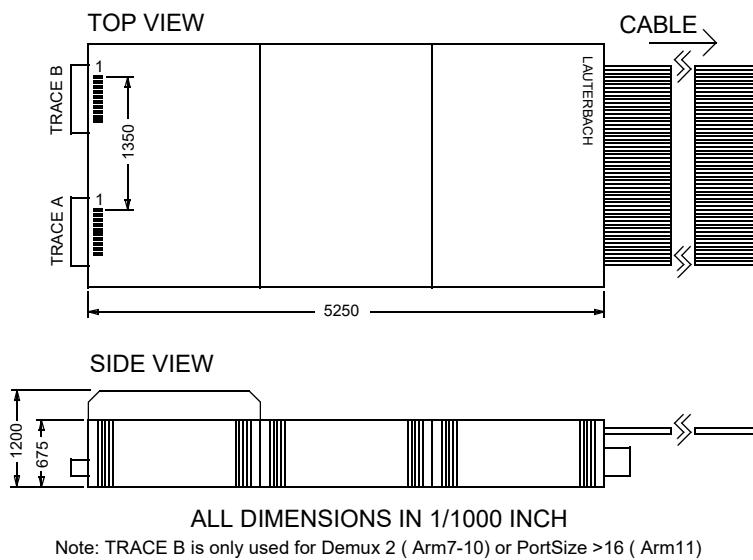


LA-7991 PP-ARM-ETM-AF

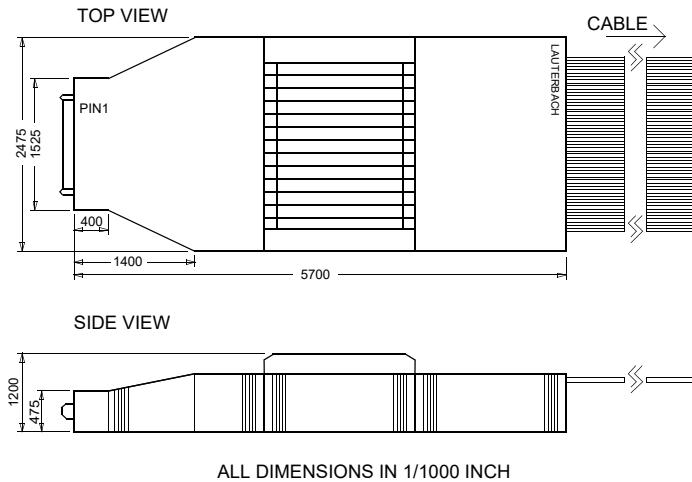


Dimension

LA-7992 PP-ARM-ETM-AF-2

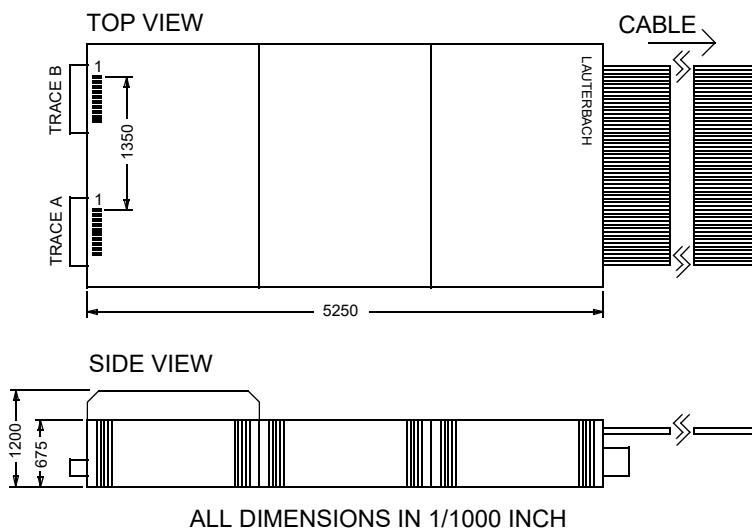


LA-7993 PP-ARM-ETM-AF-MIPI

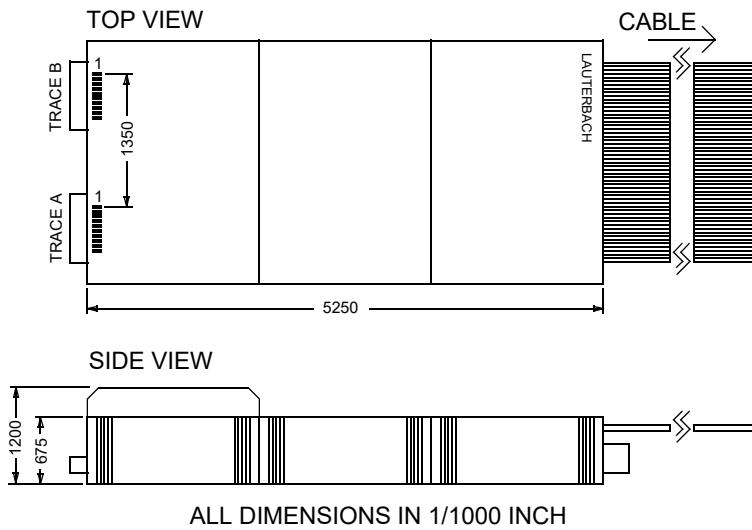


Dimension

LA-7995 PP-C55X-AF-2

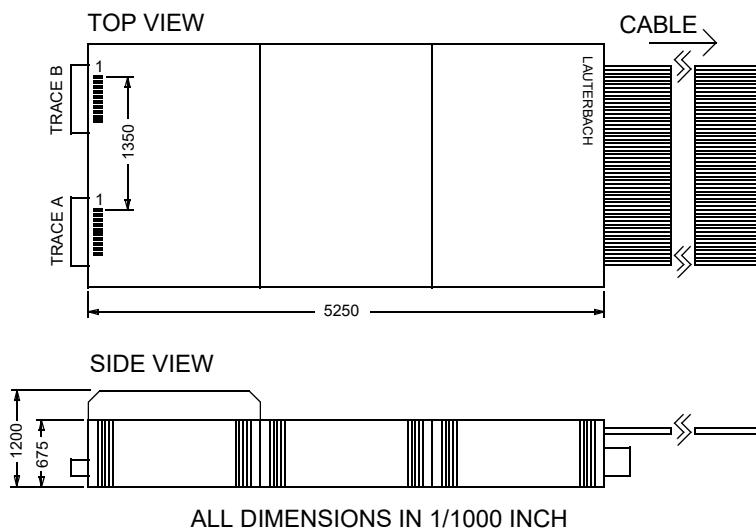


LA-7996 PP-CEVA-AF-2



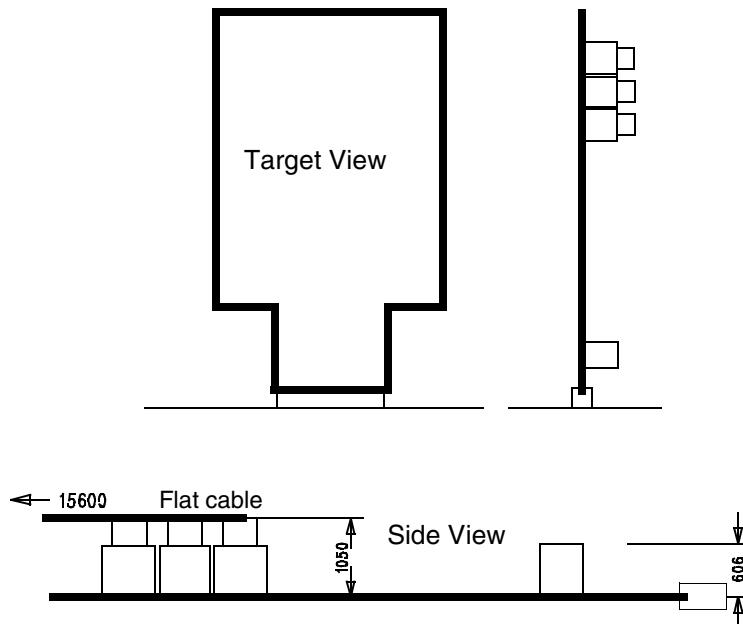
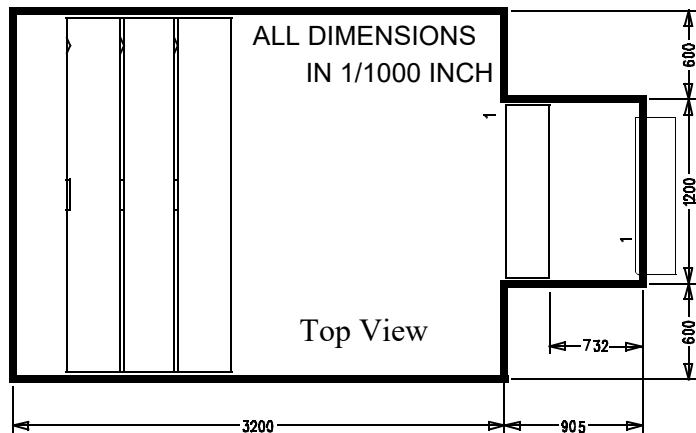
Dimension

LA-7999 PP-STARCORE-AF-2



Dimension

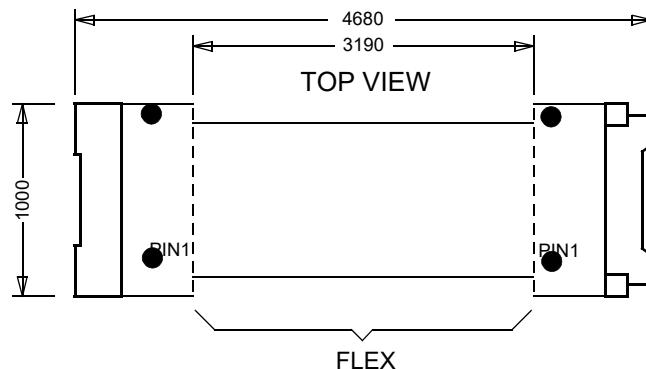
LA-7889 PP-ARM-ETM/120



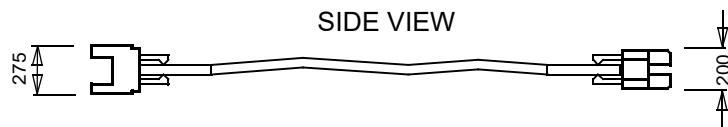
Dimension

LA-1370 MICTOR-FLEXEXT

MICTOR-EXTENDER
HORIZONTAL-HORIZONTAL

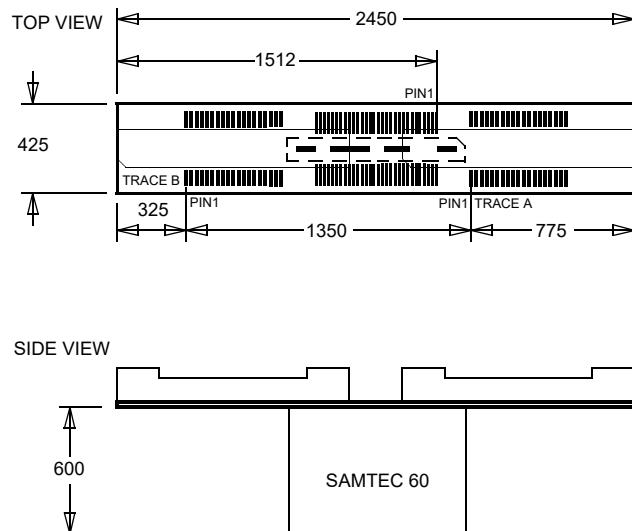


SIDE VIEW



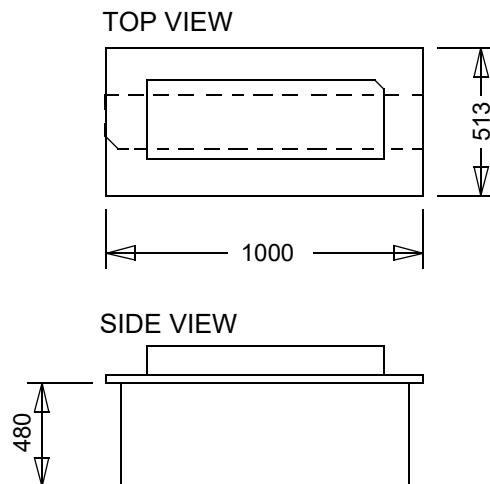
Dimension

LA-3816 CON-2XMICTOR-MIPI60



ALL DIMENSIONS IN 1/1000 INCH

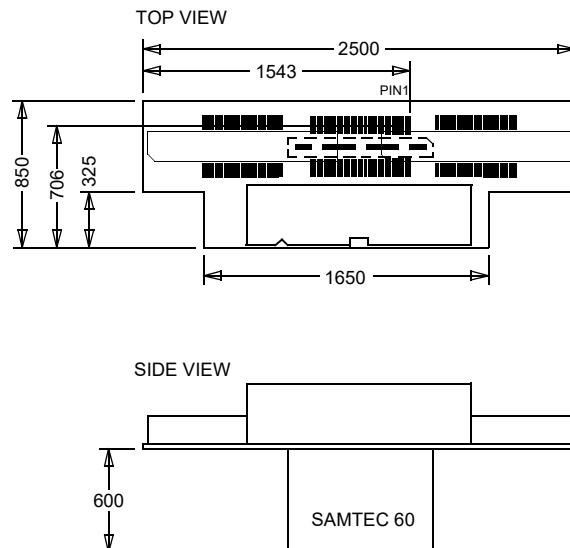
LA-3817 CON-MIPI60-MICTOR38



ALL DIMENSIONS IN 1/1000 INCH

Dimension

LA-3818 CONV-MIC+JTAG-MIPI60



ALL DIMENSIONS IN 1/1000 INCH

Connector Layout for Arm

ETMv1/2

Signal	Pin	Pin	Signal
N/C	1	2	N/C
N/C	3	4	N/C
N/C	5	6	TRACECLK
DBGRQ	7	8	DBGACK
RESET-	9	10	EXTRIG
TDO	11	12	VREF-TRACE
RTCK	13	14	VREF-DEBUG
TCK	15	16	TRACEPKT7
TMS	17	18	TRACEPKT6
TDI	19	20	TRACEPKT5
TRST-	21	22	TRACEPKT4
TRACEPKT15	23	24	TRACEPKT3
TRACEPKT14	25	26	TRACEPKT2
TRACEPKT13	27	28	TRACEPKT1
TRACEPKT12	29	30	TRACEPKT0
TRACEPKT11	31	32	TRACESYNC
TRACEPKT10	33	34	PIPESTAT2
TRACEPKT9	35	36	PIPESTAT1
TRACEPKT8	37	38	PIPESTAT0

ETMv1/2 with Multiplexed Mode

Signal	Pin	Pin	Signal
N/C	1	2	N/C
N/C	3	4	N/C
N/C	5	6	TRACECLK
DBGRQ	7	8	DBGACK
SRST-	9	10	EXTRIG
TDO	11	12	VTREF
RTCK	13	14	VCC
TCK	15	16	N/C
TMS	17	18	N/C
TDI	19	20	TRACEPKT1415
TRST-	21	22	TRACEPKT1213
N/C	23	24	TRACEPKT1011
N/C	25	26	TRACEPKT0809
N/C	27	28	TRACEPKT0607
N/C	29	30	TRACEPKT0405
N/C	31	32	TRACEPKT0003
N/C	33	34	PS02TPKT02
N/C	35	36	PS01TPKT01
N/C	37	38	PS00TSYNC

ETMv1/2 with 4 bit Demultiplexed Mode

Signal	Pin	Pin	Signal
N/C	1	2	N/C
N/C	3	4	N/C
N/C	5	6	TRACECLKA
DBGREQ	7	8	DBGACK
SRST-	9	10	EXTRIG
TDO	11	12	VTREF
RTCK	13	14	VCC
TCK	15	16	N/C
TMS	17	18	N/C
TDI	19	20	N/C
TRST-	21	22	N/C
TRACEPKTB3	23	24	TRACEPKTA3
TRACEPKTB2	25	26	TRACEPKTA2
TRACEPKTB1	27	28	TRACEPKTA1
TRACEPKTB0	29	30	TRACEPKTA0
TRACESYNCB	31	32	TRACESYNCA
PIPESTATB2	33	34	PIPESTATA2
PIPESTATB1	35	36	PIPESTATA1
PIPESTATB0	37	38	PIPESTATA0

ETMv1/2 with 8/16 bit Demultiplexed Mode

Connector 1:

Signal	Pin	Pin	Signal
N/C	1	2	N/C
N/C	3	4	N/C
N/C	5	6	TRACECLK
DBGREQ	7	8	DBGACK
SRST-	9	10	EXTRIG
TDO	11	12	VTREF
RTCK	13	14	VCC
TCK	15	16	TRACEPKTA7
TMS	17	18	TRACEPKTA6
TDI	19	20	TRACEPKTA5
TRST-	21	22	TRACEPKTA4
TRACEPKTA15	23	24	TRACEPKTA3
TRACEPKTA14	25	26	TRACEPKTA2
TRACEPKTA13	27	28	TRACEPKTA1
TRACEPKTA12	29	30	TRACEPKTA0
TRACEPKTA11	31	32	TRACESYNCA
TRACEPKTA10	33	34	PIPESTATA2
TRACEPKTA9	35	36	PIPESTATA1
TRACEPKTA8	37	38	PIPESTATA0

Connector 2:

Signal	Pin	Pin	Signal
N/C	1	2	N/C
N/C	3	4	N/C
N/C	5	6	N/C
N/C	7	8	N/C
N/C	9	10	N/C
N/C	11	12	N/C
N/C	13	14	N/C
N/C	15	16	TRACEPKTB7
N/C	17	18	TRACEPKTB6
N/C	19	20	TRACEPKTB5
N/C	21	22	TRACEPKTB4
TRACEPKTB15	23	24	TRACEPKTB3
TRACEPKTB14	25	26	TRACEPKTB2
TRACEPKTB13	27	28	TRACEPKTB1
TRACEPKTB12	29	30	TRACEPKTB0
TRACEPKTB11	31	32	TRACESYNCB
TRACEPKTB10	33	34	PIPESTATB2
TRACEPKTB9	35	36	PIPESTATB1
TRACEPKTB8	37	38	PIPESTATB0

Signal	Pin	Pin	Signal
N/C	1	2	N/C
N/C	3	4	N/C
TRACECLKB	5	6	TRACECLKA
DBGREQ	7	8	DBGACK
SRST-	9	10	EXTRIG
TDO	11	12	VTREF
RTCK	13	14	VCC
TCK	15	16	ATRACEPKT7
TMS	17	18	ATRACEPKT6
TDI	19	20	ATRACEPKT5
TRST-	21	22	ATRACEPKT4
BTRACEPKT3	23	24	ATRACEPKT3
BTRACEPKT2	25	26	ATRACEPKT2
BTRACEPKT1	27	28	ATRACEPKT1
BTRACEPKT0	29	30	ATRACEPKT0
BTRACESYNC	31	32	ATRACESYNC
BPIPESTAT2	33	34	APIPESTAT2
BPIPESTAT1	35	36	APIPESTAT1
BPIPESTAT0	37	38	APIPESTAT0

Connector 1:

Signal	Pin	Pin	Signal
N/C	1	2	N/C
N/C	3	4	N/C
GND	5	6	TRACECLK
DBGRQ	7	8	DBGACK
RESET-	9	10	EXTRIG
TDOI-ISWO	11	12	VREF-TRACE
RTCK	13	14	VREF-DEBUG
TCKITCKCISWCLK	15	16	TRACEDATA[7]
TMSITMSCISWDIO	17	18	TRACEDATA[6]
TDI	19	20	TRACEDATA[5]
TRST-	21	22	TRACEDATA[4]
TRACEDATA[15]	23	24	TRACEDATA[3]
TRACEDATA[14]	25	26	TRACEDATA[2]
TRACEDATA[13]	27	28	TRACEDATA[1]
TRACEDATA[12]	29	30	GND
TRACEDATA[11]	31	32	GND
TRACEDATA[10]	33	34	VCC
TRACEDATA[9]	35	36	TRACECTL
TRACEDATA[8]	37	38	TRACEDATA[0]

Connector 2:

Signal	Pin	Pin	Signal
N/C	1	2	N/C
N/C	3	4	N/C
GND	5	6	N/C
N/C	7	8	N/C
N/C	9	10	N/C
N/C	11	12	N/C
N/C	13	14	N/C
N/C	15	16	TRACEDATA[23]
N/C	17	18	TRACEDATA[22]
N/C	19	20	TRACEDATA[21]
N/C	21	22	TRACEDATA[20]
TRACEDATA[31]	23	24	TRACEDATA[19]
TRACEDATA[30]	25	26	TRACEDATA[18]
TRACEDATA[29]	27	28	TRACEDATA[17]
TRACEDATA[28]	29	30	GND
TRACEDATA[27]	31	32	GND
TRACEDATA[26]	33	34	VCC
TRACEDATA[25]	35	36	GND
TRACEDATA[24]	37	38	TRACEDATA[16]

20 pin JTAG Connector

For a detailed description of the 20-pin JTAG connector, please refer to “[Target Adaption](#)” (debugger_arm.pdf).