Debugging via USB User´s Guide

Debugging via USB User´s Guide

Debugging via USB User´s Guide

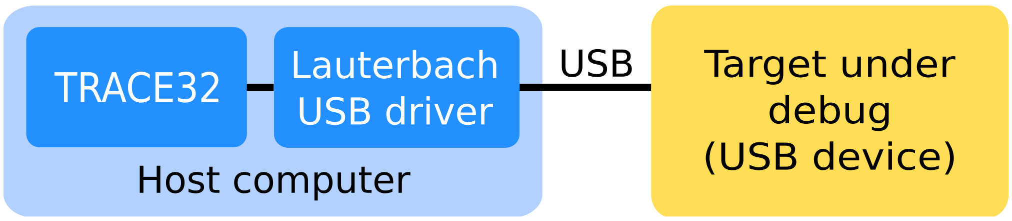
Version 06-Jun-2024

History

06-Feb-2017	Last major revision of the manual.

Introduction

Some debug target systems support debugging via the USB protocol. This can for example be debugging via TRACE32 with only a USB cable as the connection between PC and target (depending on the target architecture, additional hardware or particular cables might be necessary).



This document gives an introduction on how to configure and get started with a TRACE32 system for debugging via the USB protocol.

Therefore the following chapters provide a walkthrough on **the installation of the necessary USB drivers** and on **how to configure the TRACE32 system**. In addition, they cover the **USB commands in TRACE32**.

System Requirements

Hardware requirements:

- **USB Cable**: The necessary cable type depends on the target system:
  - **USB device targets**: If the target acts as a USB *device*, then a standard USB 3.0 cable can be used. Example for targets that are mostly USB devices: smarthphones.
  - **USB host targets**: If the target acts as a USB *host*, then a so-called “A-to-A” or “Male-to-Male” USB 3.0 cable must be used. Such cables typically omit the “Vbus”, “D+” and “D-” wires and are made of particularly high quality. Examples for targets that are mostly USB hosts: laptops and workstations.

Software requirements:

- **USB driver installation files**: On Windows host PCs, a dedicated USB target driver is necessary for USB debugging. For more information about the driver, see chapter “**Installation of the USB Driver**”.
- **TRACE32 version**: Whether the build number of your TRACE32 installation already supports debugging via the USB protocol does depend on the target system. For more details, please contact Lauterbach support.

Installation of the USB Driver

In order to use TRACE32 for USB debugging, it is necessary that the required USB driver is installed on the host PC. Depending on the operating system of the host PC, continue with **"Install the USB Driver on Windows"** or with **"Install the USB Driver on Linux"**.

NOTE:	The TRACE32 Target USB Driver is used to address a debug target. Please note that this driver is distinct from the Lauterbach USB driver provided for Lauterbach **hardware modules**.

Install the USB Driver on Windows

To install the TRACE32 Target USB Driver:

1. Make sure that your target USB device is powered and connected to the host PC.

2. Navigate to the TRACE32 system directory (by default c:\t32) and double-click ~~/bin/windows/drivers/setup_t32_target_usb_driver.exe

3. In the **TRACE32 Target USB Driver Installation** window, follow the instructions of the installation wizard.

Once the TRACE32 Target USB Driver has been installed on the host PC, continue with **"Start a TRACE32 Session for USB Debugging"**, page 7.

Install the USB Driver on Linux

On a Linux host operating system, no USB driver installation is necessary. Continue with **"Start a TRACE32 Session for USB Debugging"**, page 7.

Start a TRACE32 Session for USB Debugging

In this section:

- Overview of Configuration Scenarios: Introduces the three different USB debugging scenarios.
- Start the TRACE32 Session via T32Start: Describes how to configure software-only debug environments in T32Start for use in TRACE32 PowerView. The T32Start application assists you in configuring the desired setup. This way you do not need to manually edit any config.t32 file.
- Start the TRACE32 Session without T32Start: Describes how to configure software-only debug environments by manually creating a configuration file for use in TRACE32 PowerView. Choose a manual configuration if you do not want to use T32Start, or if you work with Linux.

NOTE:	The T32Start application is currently only available for Windows. For more information about T32Start, refer to **"T32Start"** (app_t32start.pdf).

Overview of Configuration Scenarios

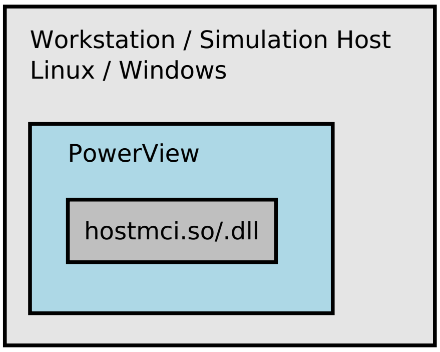
The TRACE32 PowerView instances can be set up in different ways.

1. A single TRACE32 PowerView instance runs on the same host as the back-end, see Setup 1. This configuration cannot handle AMP debug scenarios.
2. Multiple TRACE32 PowerView instances run on the same host as the back-end, see Setup 2.
3. The TRACE32 PowerView instances run on a dedicated workstation; the back-end runs on another host, see Setup 3.

Simply choose the setup you need, and then follow one of the two cross-references at the bottom of the chosen setup diagram.

Setup 1

Setup with a single TRACE32 PowerView instance running on the same host as the back-end:

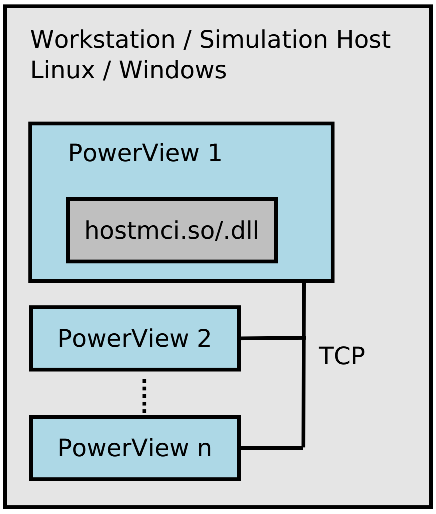


There are two configuration options:

- For step-by-step instructions on how to configure the above setup in T32Start, see **Start TRACE32 Session via T32Start**.
- For instructions on how to manually create a configuration file for the above setup, see **Start TRACE32 Session without T32Start**.

Setup 2

Setup with multiple TRACE32 PowerView instances (AMP) running on the same host as the back-end:

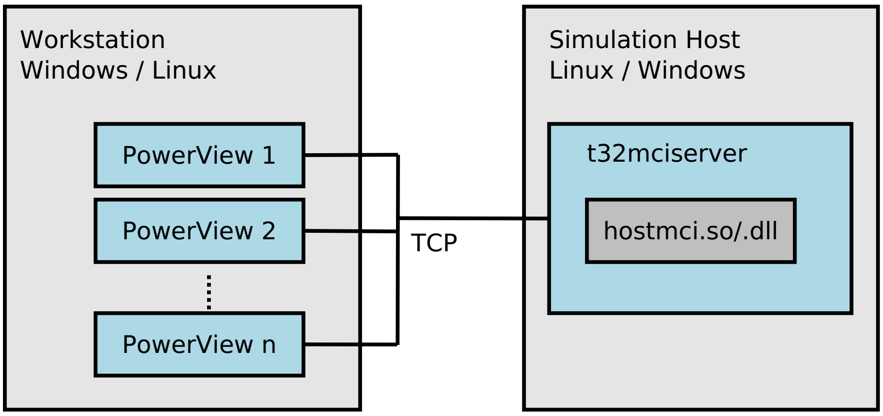


There are two configuration options:

- For step-by-step instructions on how to configure the above setup in T32Start, see **Start TRACE32 Session via T32Start**.
- For instructions on how to manually create a configuration file for the above setup, see **Start TRACE32 Session without T32Start**.

Setup 3

Setup with multiple TRACE32 PowerView instances (AMP) running on another host:



There are two configuration options:

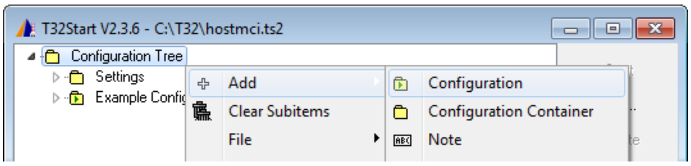
- For step-by-step instructions on how to configure the above setup in T32Start, see **Start TRACE32 Session via T32Start**.
- For instructions on how to manually create a configuration file for the above setup, see **Start TRACE32 Session without T32Start**.

Start the TRACE32 Session via T32Start

- Navigate to the TRACE32 system directory (by default c:\t32) and double-click ~~/bin/windows/t32start.exe.

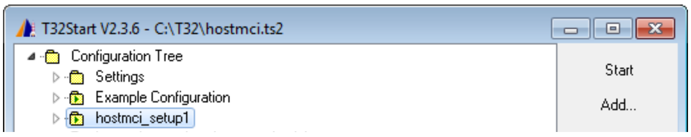
Debug Environment for Setup 1 (Single Instance)

1. In the **T32Start** window, right-click **Configuration Tree**, point to **Add**, and then select **Configuration**.



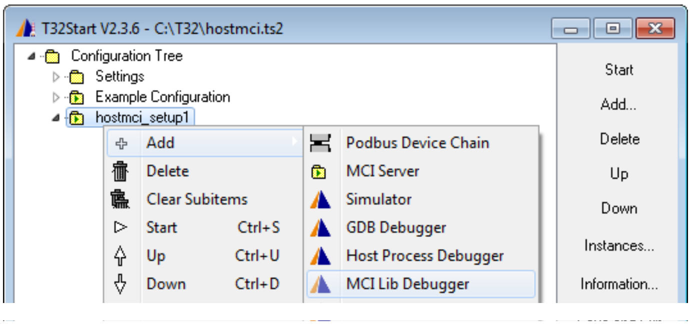
A tree item named **Configuration<*number*>** is displayed.

2. Press **F2** to rename the new **Configuration<*number*>** tree item to a meaningful name, for example, **hostmci_setup1**.

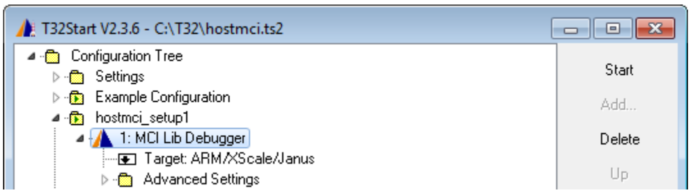


MCI Lib Debugger and Target

3. Right-click the renamed tree item, point to **Add**, and then select **MCI Lib Debugger**.



4. Click the little triangle next to **MCI Lib Debugger** to expand the tree node and navigate to **Target**.



5. Right-click **Target**, and then select the target name you want from the **Target** drop-down list.

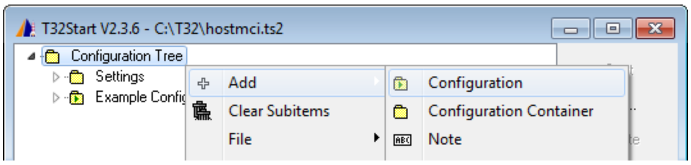
6. Configure the **Advanced Settings** as required by your environment. See **"Default Advanced Settings"** (app_t32start.pdf) for more details.

7. Click **Save** when you are done.

8. Click **Start** in order to start the debug session. A TRACE32 PowerView instance opens.

Debug Environment for Setup 2 (Integrated Server)

1. In the **T32Start** window, right-click **Configuration Tree**, point to **Add**, and then select **Configuration**.

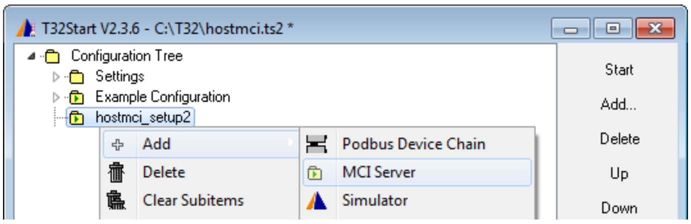


A new tree item named **Configuration<*number*>** is displayed.

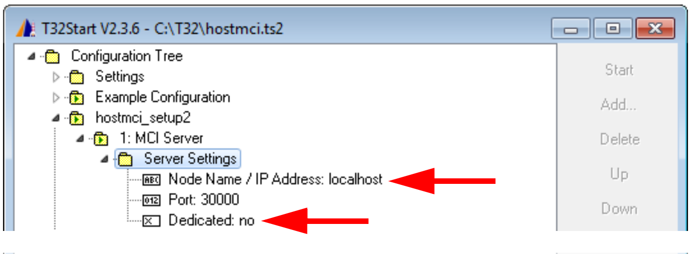
2. Press **F2** to rename the tree item to a meaningful name, for example, **hostmci_setup2**.

MCI Server and Server Settings (Setup 2)

3. Right-click the renamed tree item, point to **Add**, and then select **MCI Server**.



4. Navigate from **MCI Server** to **Server Settings** by clicking the little triangles next to the tree items.

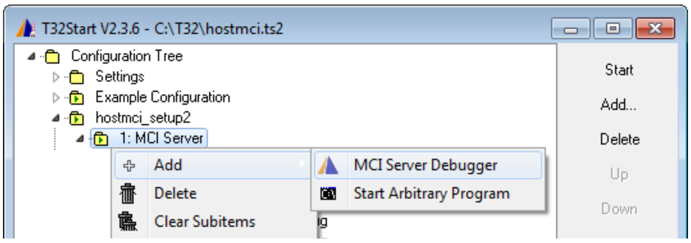


Node Name / IP Address: localhost
Port: 30000
Dedicated: no

5. Under **Server Settings**, make these settings:

- **Node Name / IP Address**: Leave empty or type `localhost`
- **Port**: Type any free port number
- **Dedicated**: `no`

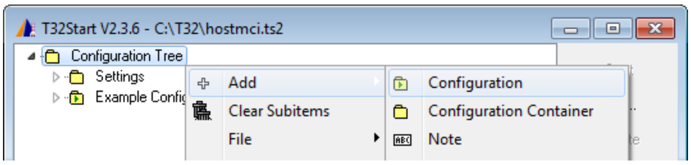
6. Click the **MCI Server** tree item, point to **Add**, and then select **MCI Server Debugger**.



7. Click the little triangle next to **MCI Server Debugger** to navigate to **Target**.
8. Right-click **Target**, and then select the target name you want from the **Target** drop-down list.
9. Repeat the steps in this section for each **MCI Server Debugger** required for your project.
10. Configure the **Advanced Settings** as required by your environment. See **"Default Advanced Settings"** (app_t32start.pdf) for more details.
11. Click **Save** when you are done.
12. Click **Start** in order to start the debug session. A TRACE32 PowerView instance opens.

Debug Environment for Setup 3 (Dedicated Server)

1. In the **T32Start** window, right-click **Configuration Tree**, point to **Add**, and then select **Configuration**.

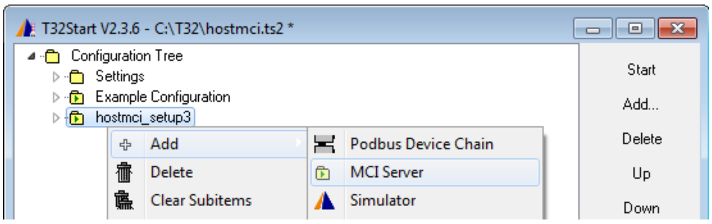


A tree item named **Configuration<*number*>** is displayed.

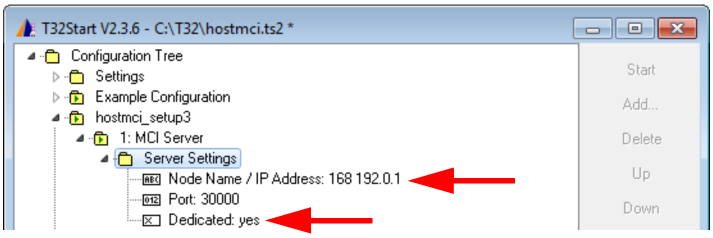
2. Press **F2** to rename the tree item to a meaningful name, for example, **hostmci_setup3**.

MCI Server and Server Settings (Setup 3)

3. Right-click the renamed tree item, point to **Add**, and then select **MCI Server**.



4. Navigate from **MCI Server** to **Server Settings** by clicking the little triangles next to the tree items.

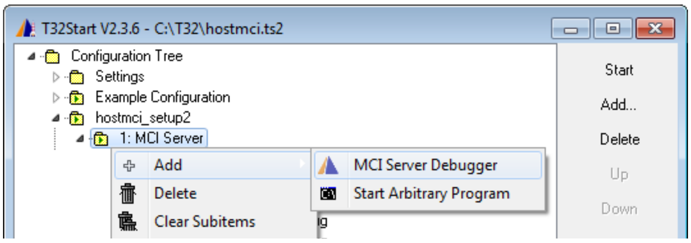


Node Name / IP Address: 168 192.0.1
Port: 30000
Dedicated: yes

5. Under **Server Settings**, make these settings:

- **Node Name / IP Address**: Type IP address of the workstation where t32mciserver runs
- **Port**: Type any free port number
- **Dedicated**: `yes`

6. Click the **MCI Server** tree item, point to **Add**, and then select **MCI Server Debugger**.



7. Click the little triangle next to **MCI Server Debugger** to navigate to **Target**.

8. Right-click **Target**, and then select the target name you want from the **Target** drop-down list.

9. Repeat the steps in this section for each **MCI Server Debugger** required for your project.

10. Configure the **Advanced Settings** as required by your environment. See **"Default Advanced Settings"** (app_t32start.pdf) for more details.

11. Click **Save** when you are done.

12. Click **Start** in order to start the debug session. A TRACE32 PowerView instance opens.

Start the TRACE32 Session without T32Start

If you work on Linux or you do not want to use T32Start.exe, then you can manually create or edit your TRACE32 configuration file.

For more information about the TRACE32 configuration file, refer to **“Configuration File”** (training_debugger.pdf).

To make the settings in the configuration file:

1. Open your TRACE32 configuration file (by default it is *<trace32_directory>/config.t32*),

2. Proceed with one of the following three chapters, depending on which scenario you have chosen in chapter “**Overview of Configuration Scenarios**”.

Debug Environment for Setup 1 (Single Instance)

Modify the configuration file (e.g. config.t32) as follows:

```
PBI=MCILIB                          ; configure software-only debugging
```

Debug Environment for Setup 2 (Integrated Server)

Modify the configuration file (e.g. config.t32) as follows:

```
PBI=MCISERVER                  ; configure software-only debugging and open
PORT=30000                     ; server at 30000 for the first instance.
INSTANCE=AUTO                  ; consecutive number of instance or AUTO
```

Debug Environment for Setup 3 (Dedicated Server)

Start t32mciserver on the simulation host:

```
./t32mciserver port=30000           ; start t32mciserver at port 30000
```

Modify the config.t32 file as follows:

```
PBI=MCISERVER                       ; set up connection to t32mciserver
NODE=192.168.0.1                    ; connect to IP 192.168.0.1
PORT=30000                          ; at port 30000
INSTANCE=AUTO                       ; consecutive number of instances
DEDICATED                           ; avoid to fall into Setup2 case
```

To start TRACE32 PowerView with a specific config file, use e.g.:

```
bin/pc_linux/t32marm -c config.t32
```

Troubleshooting

The following describes some possible error scenarios, along with suggestions how to resolve them:

Communication problem

- While attempting to connect to the target via TRACE32 PowerView by going into **SYStem.Up**, the following error occurs:

```
debug port fail
Could not get the device handle (error -5)
Could not connect to this device: VID=xxxx, PID=xxxx (error -5)
```

- **(Possible) solution:** This error indicates a communication problem with the device. Especially for USB3.x this output can indicate problems with the used USB cable or chipset.
  - Please use high-quality USB cables, the length should be as short as possible.
  - Double check if you have chosen the correct USB cable for the desired USB device. For more details, see chapter “**System Requirements**”.
  - Check if there is a firmware upgrade for your host USB chipset available.
  - Connect the device to another USB port. There is a chance that other ports are connected to another type of USB chipset that might fully support all USB features.

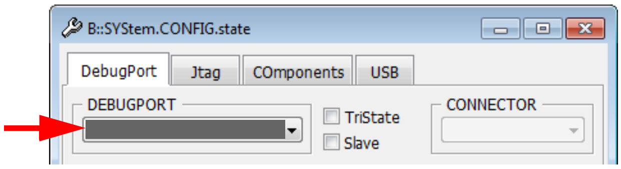
Select a USB Device via the GUI

The following step-by-step procedure describes how to select a USB device for debugging by using the TRACE32 PowerView GUI. All steps described below can alternatively also be executed via the TRACE32 command line.

Remember that USB debugging is not available for all architectures.

Prerequisites:

- You have started TRACE32, as described in **"Start a TRACE32 Session for USB Debugging"**, page 7.
- You have selected the CPU type of the target (**SYStem.CPU**).
- You have chosen the debug port (**SYStem.CONFIG.DEBUGPORT**). For more information about this architecture-specific command, please refer to your **Processor Architecture Manual**.

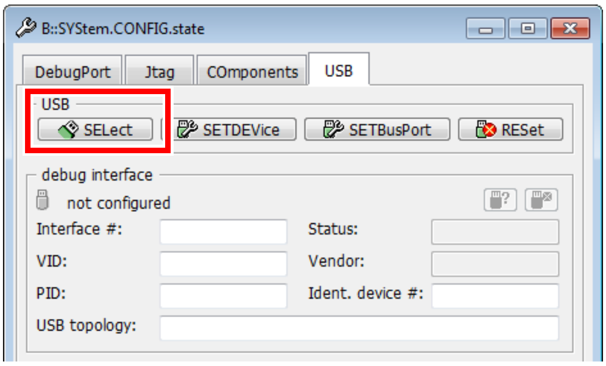


To select a USB device via the TRACE32 PowerView GUI:

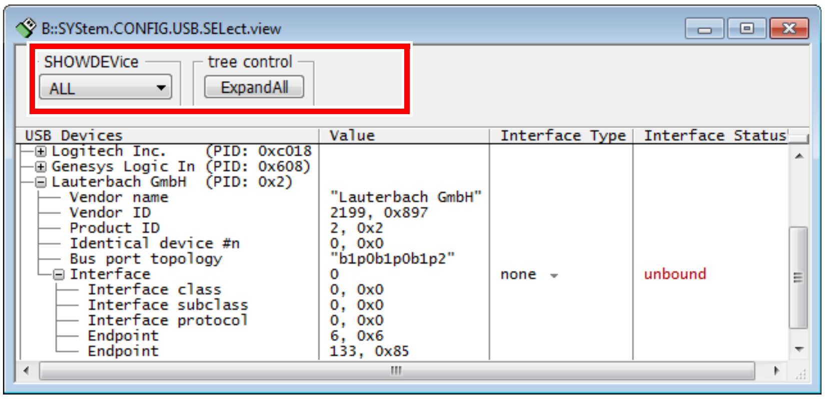
1. Open the **SYStem.CONFIG.state** window by typing at the TRACE32 command line:

```
SYStem.CONFIG.state
```

2. In the **SYStem.CONFIG.state** window, click the **USB** tab.
3. Under **USB**, click the **SELect** button to open the **SYStem.CONFIG.USB.SELect.view** window.



4. From the **SHOWDEVice** drop-down list, select **ALL** to list all USB devices.

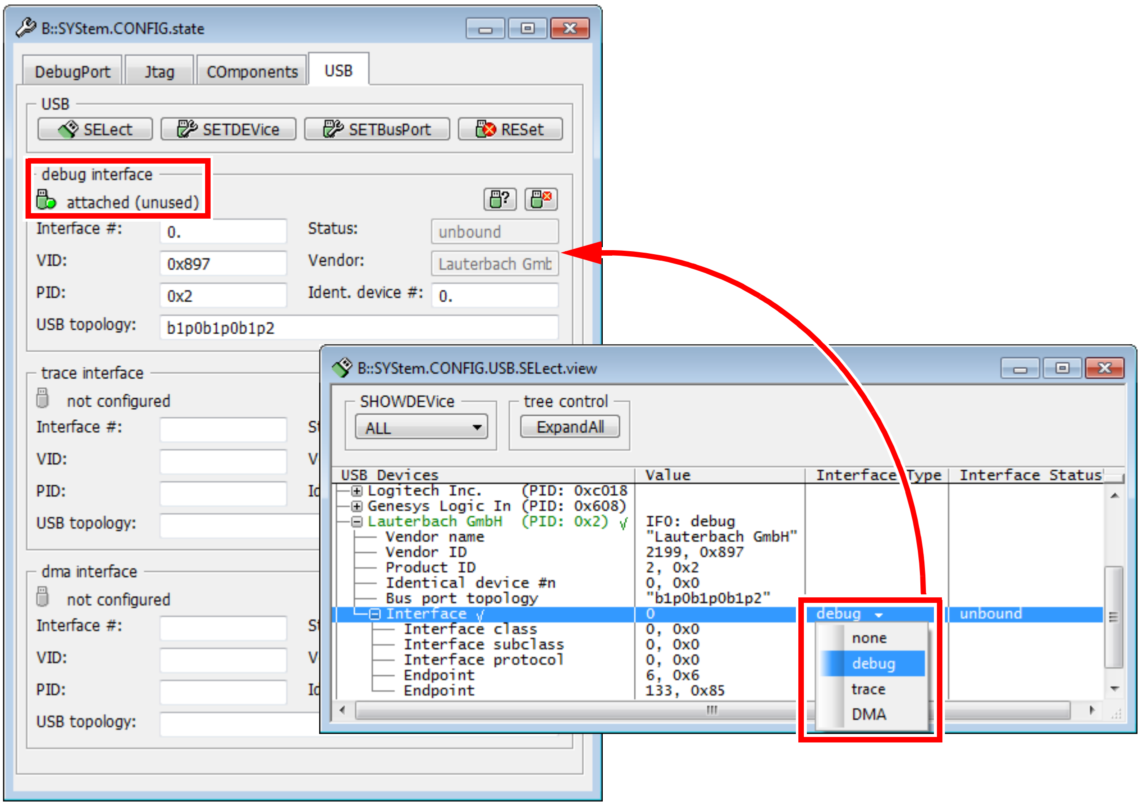


```
USB Devices                             Value                Interface Type  Interface Status
⊞ Logitech Inc.      (PID: 0xc018)
⊞ Genesys Logic In   (PID: 0x608)
⊟ Lauterbach GmbH    (PID: 0x2)
   Vendor name                          "Lauterbach GmbH"
   Vendor ID                            2199, 0x897
   Product ID                           2, 0x2
   Identical device #n                  0, 0x0
   Bus port topology                    "b1p0b1p0b1p2"
  ⊟ Interface                           0                    none            unbound
     Interface class                    0, 0x0
     Interface subclass                 0, 0x0
     Interface protocol                 0, 0x0
     Endpoint                           6, 0x6
     Endpoint                           133, 0x85
```

5. Search for the desired USB device in the device tree.

   In the **Interface Type** column of the chosen device, right-click and then select the TRACE32 interface type from the popup menu, e.g. **debug**.

   The values are transferred to the **USB** tab of the **SYStem.CONFIG.state** window and displayed under **debug interface**.



```
USB
debug interface
  attached (unused)
Interface #:   0.            Status:           unbound
VID:           0x897         Vendor:           Lauterbach Gmb
PID:           0x2           Ident. device #:  0.
USB topology:  b1p0b1p0b1p2

trace interface
  not configured
Interface #:
VID:
PID:
USB topology:

dma interface
  not configured
Interface #:
VID:
PID:
USB topology:
```

```
USB Devices                               Value                Interface Type  Interface Status
⊞ Logitech Inc.      (PID: 0xc018)
⊞ Genesys Logic In   (PID: 0x608)
⊟ Lauterbach GmbH    (PID: 0x2) √         IF0: debug
   Vendor name                            "Lauterbach GmbH"
   Vendor ID                              2199, 0x897
   Product ID                             2, 0x2
   Identical device #n                    0, 0x0
   Bus port topology                      "b1p0b1p0b1p2"
  ⊟ Interface √                           0                    debug           unbound
     Interface class                      0, 0x0
     Interface subclass                   0, 0x0
     Interface protocol                   0, 0x0
     Endpoint                             6, 0x6
     Endpoint                             133, 0x85
```

Result: You have successfully selected an interface of a USB device. It can now be used for the chosen purpose (e.g. debug, trace, or DMA).

Select a USB Device via the TRACE32 Commands

There are three commands that allow to identify the USB target device based on certain parameters. In addition, the commands are used to configure the assignment of TRACE32 interface type to USB device interface. This assignment defines that a particular interface of the USB device is either reserved for debugging or tracing or DMA in TRACE32.

The three commands are:

- **SYStem.CONFIG.USB.SET**
- **SYStem.CONFIG.USB.SETDEVice**
- **SYStem.CONFIG.USB.SETBusPort**

While all three commands have the same purpose, **SYStem.CONFIG.USB.SET** can be seen as the "generic" command that allows to set all available parameters for the description of a device. The commands **SYStem.CONFIG.USB.SETDEVice** and **SYStem.CONFIG.USB.SETBusPort** can be seen as subsets of **SYStem.CONFIG.USB.SET**, which only need a small number of predefined parameters.

The decision which command is the most suitable one mainly depends on the amount and the properties of USB devices that are currently connected to the host PC. In more complex scenarios with multiple similar or even identical USB devices, the **SYStem.CONFIG.USB.SET** command is recommended.

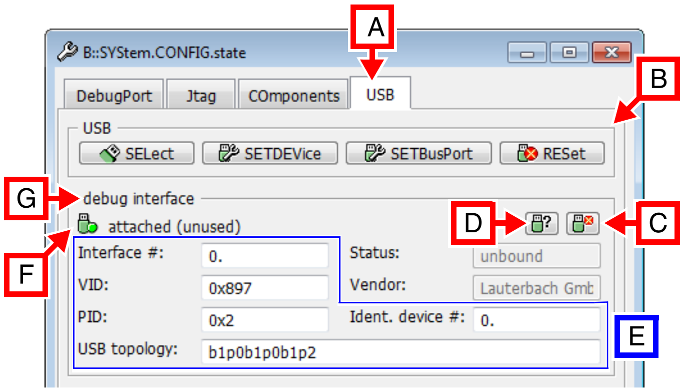
The configuration for a USB device can also be made at a point in time when the device is not attached to the host PC (yet). In this case, the configuration will be displayed in the **SYStem.CONFIG.state /USB** window as "configured but not attached". As soon as a device gets connected that uniquely matches the previously made configuration, the status of the TRACE32 interface type will be updated and it will be displayed as "attached". See the chapter **SYStem.CONFIG.state /USB** for more details.

SYStem.CONFIG.state Open configuration window for USB debugging

Format: **SYStem.CONFIG.state /USB**

The **SYStem.CONFIG.state /USB** tab provides an overview of the USB configuration and its status [**F**]. It displays:

- Supported TRACE32 interface types
- USB device configuration for a certain interface type
- Status of each interface type



Interface #:	0.	Status:	unbound
VID:	0x897	Vendor:	Lauterbach Gmb
PID:	0x2	Ident. device #:	0.
USB topology:	b1p0b1p0b1p2		

A USB tab in the **SYStem.CONFIG.state** window.

B Buttons for the most important **SYStem.CONFIG.USB** commands.

C Resets the configuration of the respective interface type (executes **SYStem.CONFIG.USB.RESet**).

D Copies the current configuration settings to the TRACE32 command line (will be printed as **SYStem.CONFIG.USB.SET** command).

E Edit boxes that allow to set identification parameters for the desired target device. Therefore, each edit box represents one (optional) parameter of the **SYStem.CONFIG.USB.SET** command.

F Indicator for the status of the respective interface type. Possible states are:

- **not configured**: No configuration for this interface type has been made yet.
- **configured, not attached**: A configuration for this interface type has been made, but the parameters do not match any USB device that is currently attached to the host PC.
- **configured (no unique device)**: The configuration parameters for this interface type can be matched to more than one of the USB devices that are connected to the host PC. To solve this, either remove unnecessary USB devices, or add more configuration parameters in order to make the description unique.
- **configured (no unique interface)**: The configuration parameters for the interface type match exactly one attached USB device. However, the selection of the USB interface is not unique. Possible reasons are:

    - The selected USB device has more than one interface, and the interface parameter is not set.

    - At least two configurations (for different interface types) were made for the same USB interface of the same device. Note that each USB interface can only be assigned to a single type.
- **attached (unused)**: The configuration parameters for this USB mode match exactly one attached USB device, and also match a unique device interface. This configuration is sufficient, however the interface type is currently *not* in use.
- **attached (in use)**: The configuration parameters for this interface type match exactly one attached USB device, and also match a unique device interface. The interface type *is* currently in use.

G The entire **debug interface** box displays the configuration state of the TRACE32 interface type **Debug**.

Using the command group **SYStem.CONFIG.USB**, you can configure TRACE32 in order to ensure proper identification of the desired target device. In addition, the command group allows to specify which of the target’s various USB device interfaces shall be used for which purpose (**Debug**, **Trace**, or **DMA**).

Parameters and Expressions

The following table describes parameters and expressions that are used in the **SYStem.CONFIG.USB** commands and on the graphical user interface TRACE32 PowerView:

<type>	Specify the TRACE32 interface *<type>* you want to assign to the selected USB device interface. Available TRACE32 interface *<types>* are **Debug**, **Trace**, or **DMA**. Depending on the target architecture, only a subset of the TRACE32 interface *<types>* can be selected. **NOTE**: To select a USB device interface, specify its *<interface_number>*, as explained below.
InterFace *<interface_number>*	The *<interface_number>* allows you to assign a particular USB interface device to a TRACE32 interface *<type>*. For an example, see **SYStem.CONFIG.USB.SET**. In this example, the USB device interface **1.** is assigned to the TRACE32 interface *<type>* **Debug**, i.e. the USB device interface **1.** is now reserved for debugging (and not for tracing or DMA).

Bus port topology **BusPort** *<busport>*	USB port topology representation. Physical position of the host PC USB port to which the target is connected in the bus. *<busport>* syntax: The string must match the pattern “(bXpY)+”, with X and Y being integers from 0-9999. Valid examples: “b2p3”, “b12p18”, "b18p1b33p39b2p73". As an alternative to the manual data input, you can proceed as described in **“Select a USB Device via the GUI”**, page 18.
DeviceNum *<device_number>*	See **Ident. device #**

Ident. device #	Identical device number. If multiple identical USB devices are connected to the computer, it is not trivial any more to distinguish them by their 'usual' identification parameters such as vendor ID or product ID, as these parameters are identical as well. In order to ease the differentiation in such scenarios, each identical device is assigned its own 'identical-device number' by TRACE32. This number is an ID that is unique for each of the attached devices. The IDs start at 0.
Interface class	Numeric description of the USB interface class.
Interface protocol	Numeric description of the USB interface protocol.
Interface subclass	Numeric description of the USB interface subclass.
Product ID **PID** *<pid>*	ID of the device, which is contained in the device descriptor of the USB device.
Vendor ID **VID** *<vid>*	ID of the device vendor, which is contained in the device descriptor of the USB device.
Vendor name	Name of the device vendor.

See also

- SYStem.CONFIG.USB.RESet
- SYStem.CONFIG.USB.SET
- SYStem.CONFIG.USB.SETDEFaults
- SYStem.CONFIG.USB.SELect
- SYStem.CONFIG.USB.SETBusPort
- SYStem.CONFIG.USB.SETDEVice

Format:	**SYStem.CONFIG.USB.RESet** [<*type*>]
<*type*>:	**ALL** **Debug** **Trace** **DMA**

Resets the configuration of a selected or all TRACE32 interface types.

ALL	Resets the configuration for all TRACE32 interface types (such as **Debug**, **Trace**, etc.). Any USB device that has already been selected will be unselected.
Debug	Resets the configuration for the debug interface.
Trace	Resets the configuration for the trace interface.
DMA	Resets the configuration for the DMA interface (Direct Memory Access).

See also

■ SYStem.CONFIG.USB

SYStem.CONFIG.USB.SELect — Connected USB devices

Using the **SYStem.CONFIG.USB.SELect** command group, you can display a list of connected USB devices in the form of a tree structure, filter the list, as well as expand and collapse the nodes of the USB device tree.

See also

- SYStem.CONFIG.USB.SELect.CollapseAll
- SYStem.CONFIG.USB.SELect.ExpandAll
- SYStem.CONFIG.USB.SELect.SHOWDEVice
- SYStem.CONFIG.USB.SELect.view
- SYStem.CONFIG.USB

SYStem.CONFIG.USB.SELect.view — List connected USB devices

[Window Description]

Format:	**SYStem.CONFIG.USB.SELect.view** [**/**<*option*>]
<*option*>:	**SpotLight**

Opens the USB device selection window, displaying a list of all USB devices that are currently connected to the host computer. To help you identify and distinguish the USB devices, the tree of each device and its various interfaces displays additional information (partially obtained from the USB descriptors):

- Each device is described by its vendor name, vendor ID, product ID, identical device number, and bus port topology.
- Each interface of a device is described by its class, subclass, protocol, and endpoints.

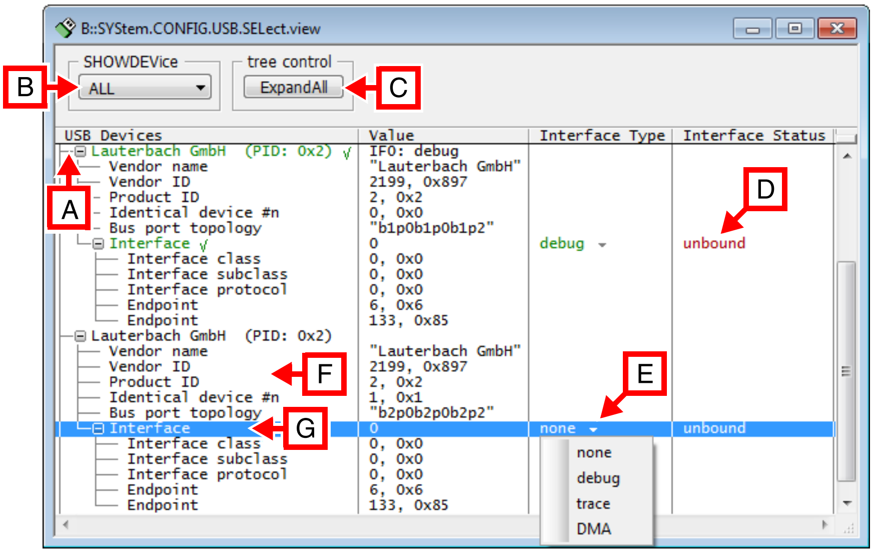
  For information about these parameters, see **SYStem.CONFIG.USB**.

SpotLight	Highlights recently attached USB devices. The USB device that was most recently attached to the host PC is highlighted in dark red. The USB device that was attached second to last is highlighted a little bit lighter. This works up to a level of 4 USB devices. The window can only highlight devices that get attached while the window is open.

Description of the SYStem.CONFIG.USB.SELect Window:

In the **SYStem.CONFIG.USB.SELect** window, you can perform the following actions:

- Filter the USB devices [**B**].
- Expand and collapse the USB device tree [**C**].
- Assign an interface of a USB device to an interface type (right-click and select an entry in the popup menu [**E**]).



```
USB Devices                          Value                 Interface Type   Interface Status
⊟ Lauterbach GmbH  (PID: 0x2) √      IF0: debug
   Vendor name                       "Lauterbach GmbH"
   Vendor ID                         2199, 0x897
   Product ID                        2, 0x2
   Identical device #n               0, 0x0
   Bus port topology                 "b1p0b1p0b1p2"
  ⊟ Interface √                      0                     debug            unbound
     Interface class                 0, 0x0
     Interface subclass              0, 0x0
     Interface protocol              0, 0x0
     Endpoint                        6, 0x6
     Endpoint                        133, 0x85
⊟ Lauterbach GmbH  (PID: 0x2)
   Vendor name                       "Lauterbach GmbH"
   Vendor ID                         2199, 0x897
   Product ID                        2, 0x2
   Identical device #n               1, 0x1
   Bus port topology                 "b2p0b2p0b2p2"
  ⊟ Interface                        0                     none             unbound
     Interface class                 0, 0x0
     Interface subclass              0, 0x0
     Interface protocol              0, 0x0
     Endpoint                        6, 0x6
     Endpoint                        133, 0x85
```

A USB device in tree structure

B Filter option for USB devices:
- **SUPPORTED** - show only supported USB devices (i.e. devices that match the supported interface class)
- **CPUVENDOR** - show only USB devices of CPU vendor
- **ALL** - show all USB devices

C Expands / collapses all tree elements.

D Status of USB interface:
- “bound” if the interface is currently in use
- “unbound” if the interface is not used.

E Interface type assignment for a specific interface. The available types are **debug**, **trace**, and **DMA**.

F Parameters of the USB device.

G Interface parameters of the USB device.

See also

■ SYStem.CONFIG.USB.SELect

SYStem.CONFIG.USB.SELect.SHOWDEVice — Filter the USB device tree

Format:	**SYStem.CONFIG.USB.SELect.SHOWDEVice** *<filter>*
<filter>:	**SUPPORTED** | **CPUVENDOR** | **ALL**

Filters the tree structure of the USB devices displayed in the **SYStem.CONFIG.USB.SELect.view** window. USB devices matching the *<filter>* criterion are visible. USB devices not matching the *<filter>* criterion are temporarily hidden.

SUPPORTED	Lists only USB devices that match the supported USB interface class.
CPUVENDOR	Lists only USB devices that match the vendor of the selected target CPU, i.e., the CPU selected with the **SYStem.CPU** command.
ALL	Lists all attached USB devices.

See also

- SYStem.CONFIG.USB.SELect

SYStem.CONFIG.USB.SELect.ExpandAll — Expand tree

Format:	**SYStem.CONFIG.USB.SELect.ExpandAll**

Expands all nodes of the USB device tree in the **SYStem.CONFIG.USB.SELect.view** window.

See also

- SYStem.CONFIG.USB.SELect

SYStem.CONFIG.USB.SELect.CollapseAll Collapse tree

Format: **SYStem.CONFIG.USB.SELect.CollapseAll**

Collapses all nodes of the USB device tree in the **SYStem.CONFIG.USB.SELect.view** window.

See also

- SYStem.CONFIG.USB.SELect

SYStem.CONFIG.USB.SET Configure all parameters of USB device

Format:	**SYStem.CONFIG.USB.SET** *<type>* [**/***<option>*]
<type>:	**Debug** | **Trace** | **DMA**
<option>:	**InterFace** *<interface_number>* **VID** *<vid>* **PID** *<pid>* **BusPort** *<busport>* **DeviceNum** *<device_number>*

Configures a USB device based on all parameters.

<type>	For a description of the parameters *<type>*, etc., see **SYStem.CONFIG.USB**.

Example: This example contains all of the optional parameters. Depending on how many devices are connected to the PC and what their properties are, some of the optional parameters can be left out.

```
SYStem.CONFIG.USB.SET Debug /InterFace 1. /VID 0x897 /PID 5. \
                                      /BusPort "b3p0b2p0b3p10" /DeviceNum 0.
```

See also

- SYStem.CONFIG.USB.SETDEFaults
- SYStem.CONFIG.USB

SYStem.CONFIG.USB.SETBusPort Configure device by bus port

Format:	**SYStem.CONFIG.USB.SETBusPort** [*<type>* *<interface_number>* **"***<busport>***"**]
<type>:	**Debug** | **Trace** | **DMA**

With arguments: Configures a USB device by bus port.

Without arguments: Opens the **SYStem.CONFIG.USB.SETBusPort** window for configuration.

<type>	For a description of the parameters *<type>*, etc., see **SYStem.CONFIG.USB**.

Example:

```
SYStem.CONFIG.USB.SETBusPort Debug 1. "b3p0b2p0b3p10"
```

See also

■ SYStem.CONFIG.USB

SYStem.CONFIG.USB.SETDEFaults — Apply default USB settings

Format:	**SYStem.CONFIG.USB.SETDEFaults**

Configures the default USB settings according to the selected **SYStem.CPU** entry (if default settings are available for the selected entry). This mainly affects the settings of the **SYStem.CONFIG.USB.SET** command.

See also

■ SYStem.CONFIG.USB.SET ■ SYStem.CONFIG.USB

SYStem.CONFIG.USB.SETDEVice — Configure device by VID/PID

Format:	**SYStem.CONFIG.USB.SETDEVice** [*<type> <interface_number> <vid> <pid>*]
<type>:	**Debug** | **Trace** | **DMA**

With arguments: Configures a USB device by vendor ID (VID) or product ID (PID).

Without arguments: Opens the **SYStem.CONFIG.USB.SETDEVice** window for configuration.

<type>	For a description of the parameters *<type>*, etc., see **SYStem.CONFIG.USB**.

See also

■ SYStem.CONFIG.USB

Example:

```
SYStem.CONFIG.USB.SETDEVice Debug 1. 0x897  5.
```