LAUTERBACH
DEVELOPMENT TOOLS

XCP Debug Back-End

MANUAL

XCP Debug Back-End

TRACE32 Online Help

TRACE32 Directory

TRACE32 Index

TRACE32 Documents ..... 🗁

Debug Back-Ends ..... 🗁

XCP Debug Back-End **1**

History **4**

Introduction **5**

Related Documents and Tutorials 5

Contacting Support 5

Supported Protocol Variants 7

Target Resources 7

PowerView System Configurations **8**

Configuring XCP **11**

Reducing XCP Traffic **12**

Command Reference **13**

SYStem.CONFIG.state Open XCP configuration window 13

SYStem.CONFIG.XCP XCP specific settings 14

SYStem.CONFIG.XCP.Connect Explicitly try to connect to an XCP slave 14

SYStem.CONFIG.XCP.ConnectMode Configure automatic (dis)connect 15

SYStem.CONFIG.XCP.CONNECTTIMEOUT Configure time out connection 15

SYStem.CONFIG.XCP.DETECT.state Detect XCP slaves 16

SYStem.CONFIG.XCP.DBGWRITE Set DBGWRITE behavior manually 16

SYStem.CONFIG.XCP.DEFAULTSLAVEPORT Configure port of XCP slave 16

SYStem.CONFIG.XCP.DisConnect Explicitly disconnect from the XCP slave 17

SYStem.CONFIG.XCP.LOCK Request exclusive access 17

SYStem.CONFIG.XCP.PROTOCOL Select the protocol variant 18

SYStem.CONFIG.XCP.SLAVE Network address 18

SYStem.CONFIG.XCP.SLAVEMode Configure connect mode of XCP slave 19

SYStem.CONFIG.XCP.slaveINFO Slave information 20

SYStem.CONFIG.XCP.TEXTserviceFilter Filter text service output 20

SYStem.CONFIG.XCP.UNLOCK End exclusive access 21

SYStem.DETECT.XCPTRI Match XCP resources to access ports 21

XCP Specific Functions **22**

In This Section 22

SYStem.CONFIG.XCP.Connected() Current XCP connection state 22

SYStem.CONFIG.XCP.ConnectMode() XCP connection mode 22
SYStem.CONFIG.XCP.INFO() Numeric value of slave property 23
SYStem.CONFIG.XCP.INFO.STR() String value of slave property 23

XCP Debug Back-End

Version 06-Jun-2024

History

14-Jun-2022 Describe target resources and new command: SYStem.DETECT.XCPTRI.

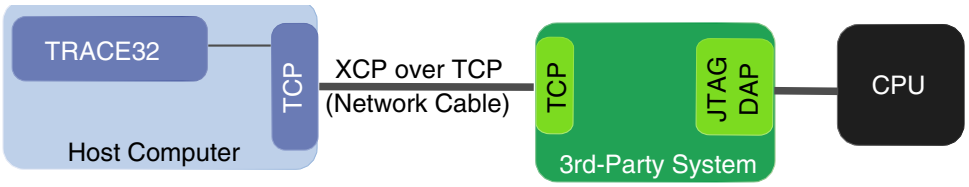
14-May-2014 Initial version.

Introduction

The XCP protocol family allows to control measurement and calibration tools of various vendors. It is standardized by the ASAM e.V. The protocol supports different transport layers, e.g., TCP/IP.

Some of these measurement and calibration tools implement extensions to the XCP protocol that allow debugging of the target CPU. TRACE32 is able to use these features.

The figure below shows a typical configuration. TRACE32 is running on a host computer. Instead of sending debugging commands directly to the target CPU, all debugging commands are encoded into XCP commands. These XCP commands are then sent over the host computer's TCP stack and a network cable to the 3rd-party system (also known as XCP slave). The 3rd-party system translates the XCP commands back into a low-level debug protocol. Typical examples are JTAG and DAP for INFINEON TriCore CPUs. The 3rd-party system may consist of multiple physical components.



This document describes how to set up and configure debugging over XCP.

Related Documents and Tutorials

- **"T32Start"** (app_t32start.pdf): The T32Start application assists you in setting up multicore / multiprocessor debug environments, and software-only debug environments. T32Start is only available for Windows.

  For more information about software-only debug environments, please refer to: **"Software-only Debugging (Host MCI)"** (app_t32start.pdf).

- For video tutorials about debugging over XCP, visit: **support.lauterbach.com/kb/articles/trace32-tutorials-debugging-over-xcp**

Contacting Support

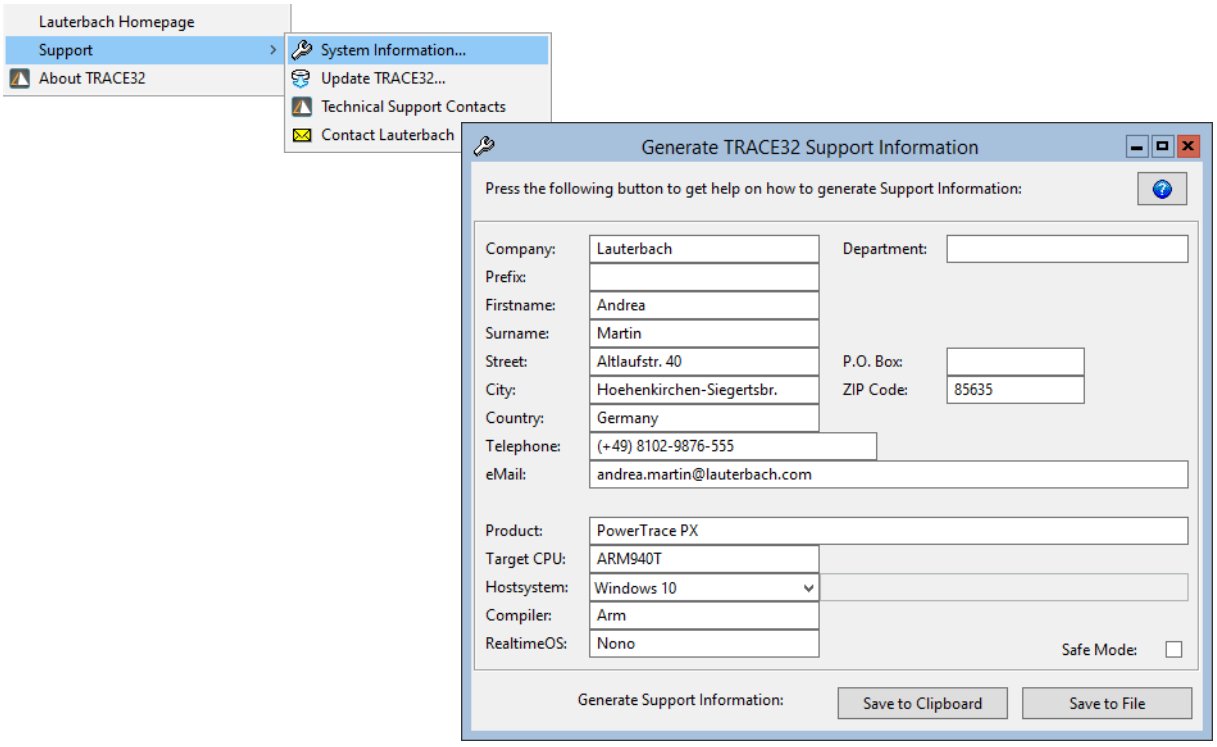
Use the Lauterbach Support Center: https://support.lauterbach.com

- To contact your local TRACE32 support team directly.
- To register and submit a support ticket to the TRACE32 global center.
- To log in and manage your support tickets.
- To benefit from the TRACE32 knowledgebase (FAQs, technical articles, tutorial videos) and our tips & tricks around debugging.

Or send an email in the traditional way to support@lauterbach.com.

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.

Supported Protocol Variants

TRACE32 supports two protocol variants for debugging over XCP:

- Software Debugging over XCP as specified in the ASAM MCD-1 (XCP) standard by ASAM e.V starting from version 1.5.
- ETAS proprietary protocol extension

Target Resources

The native function of an XCP slave often requires a high-performance access to memory mapped target resources (e.g., memory). The XCP slave can expose these target resources to the debugger as well.

In the Software Debugging over XCP standard, each target resource is identified by a target resource identifier (TRI).

TRACE32 can utilize these target resources and thus profit from the high-performance access. This requires to match these target resources to TRACE32 internal resources. While TRACE32 can match many resources automatically, others (e.g., CoreSight Access Ports) require explicit matching.

PowerView System Configurations

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

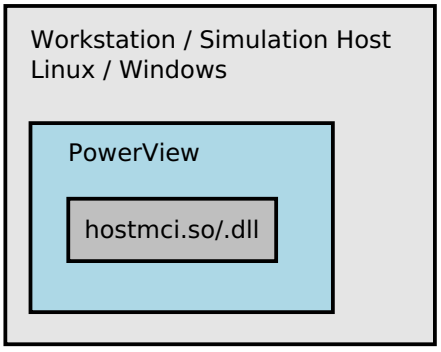
1. A single TRACE32 PowerView instance runs on the same host as the back-end, see Setup 1. This configuration can't handle AMP debug scenarios.
2. Multiple TRACE32 PowerView instances run on the same host as the back-end, see Setup 2.
3. The TRACE32 PowerView instances run on a dedicated workstation;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

The Lauterbach Debug Driver library (`hostmci.so` for Linux/Mac users and `hostmci.dll` for Windows users) can be integrated into the TRACE32 PowerView application or run as a separate process, called `t32mciserver`. Running it as a separate process provides two main benefits:

1. The MCI server can execute on one host, whilst one or more instances of TRACE32 PowerView execute on another host.
2. Multiple instances of TRACE32 PowerView can execute on a single host, sharing the MCI connection.

Setup 1

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

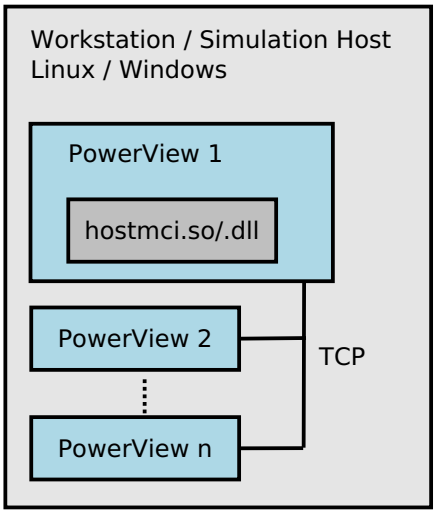


Modify the config.t32 file as follows:

```
PBI=MCILIB                          ; configure system to use hostmci.so
```

Setup 2

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

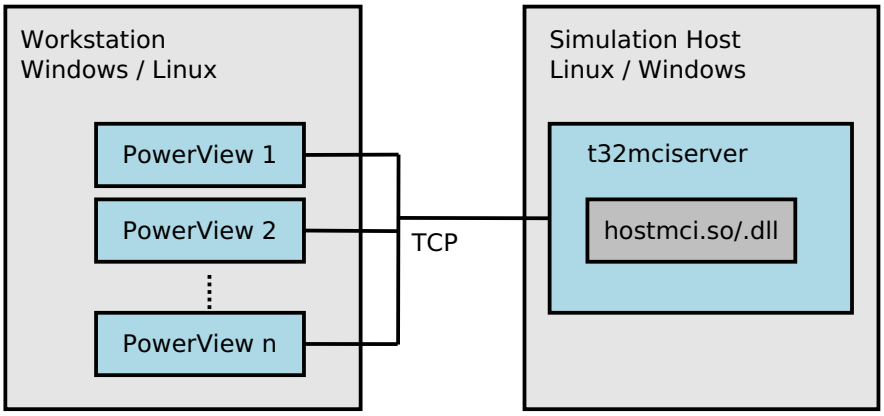


Modify the config.t32 as follows:

```
PBI=MCISERVER                ; set up the usage of hostmci.so and open
PORT=30000                   ; server at 30000 for the first instance.
INSTANCE=AUTO                ; consecutive number of instance or AUTO
```

Setup 3

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



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
```

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

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

A typical start sequence is shown below. This sequence can be written to a PRACTICE script file (*.cmm, ASCII format) and executed with the command **DO** *<file>*.

```
RESet

; select XCP as back-end
SYStem.CONFIG.DEBUGPORT XCP0

; set IP Address (192.168.0.1) and port (12345) of XCP slave
SYStem.CONFIG XCP.SLAVE 192.168.0.1:12345

; continue with CPU configuration
SYStem.CPU TC277T          ; select CPU
SYStem.Attach              ; connect to XCP slave and CPU
```

The IP address is the device that is running the XCP tools that TRACE32 PowerView will connect to. The port number is the port that is exposed by this device for incoming connections. The user should verify these and enter the correct values in the example script above.

For CPUs using CoreSight Access Ports, **SYStem.DETECT.XCPTRI** should be called between SYStem.CPU and SYStem.Attach.

Reducing XCP Traffic

Compared to a direct debug connection to a target using a PowerDebug module, the XCP debug connection will experience significantly higher latency. This can cause longer delays when polling the CPU system state, executing TRACE32 commands, or accessing the target's resources.

Several options are available that can help mitigate this extra latency by reducing the amount of XCP traffic generated.

- The most important setting is **SETUP.UpdateRATE** to configure the update rate of the TRACE32 windows. The processors state is also polled by this rate.

```
SETUP.URATE 10s                    ; screen will be updated every 10s
```

- The command **MAP.UpdateOnce** can be used to read memory regions only one time after a break is detected.

```
MAP.UpdateOnce 0x0++0x1000         ; read memory of regions
                                   ; 0x0--0x1000
                                   ; only one time after break
```

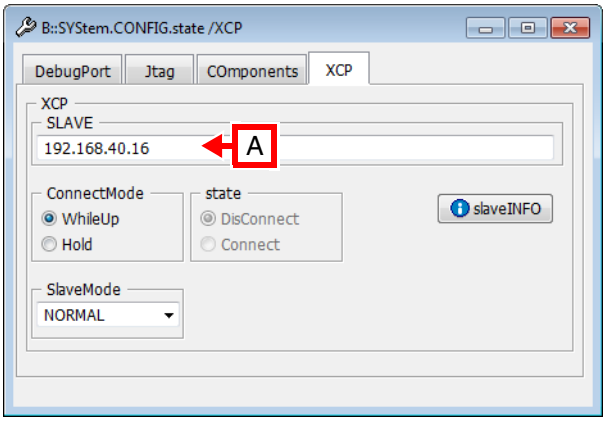
- For analysis and data display purposes it is recommended that you use the code from the TRACE32 virtual memory (VM:) instead of the code from the target memory. Therefore, the code needs to be copied to the virtual memory when an *.elf file is being loaded.

```
Data.Load.ELF *.elf /PlusVM        ; download code to target and
                                   ; to VM:

Data.List VM:                      ; open source window, but use
                                   ; VM: memory

Onchip.Access VM                   ; use VM memory for trace analysis
```

SYStem.CONFIG.state — Open XCP configuration window

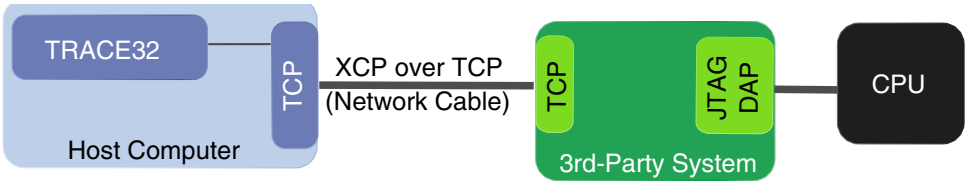
Format:	**SYStem.CONFIG.state /XCP**

Opens the **XCP** tab of the **SYStem.CONFIG.state** window, where you can configure the XCP connection to your target.



A For descriptions of the commands on the **SYStem.CONFIG.state /XCP** tab, please refer to the **SYStem.CONFIG XCP.*** commands in this chapter.
Example: For information about the **SLAVE** edit box, see **SYStem.CONFIG XCP.SLAVE**.

The **SYStem.CONFIG XCP** command group allows to set up and configure debugging over XCP.



The command group is available after **XCP0** has been selected as debug port.

```
;selecting the XCP back-end activates the SYStem.CONFIG XCP commands
SYStem.CONFIG.DEBUGPORT XCP0
```

See also

- SYStem.CONFIG.XCP.Connect
- SYStem.CONFIG.XCP.CONNECTTIMEOUT
- SYStem.CONFIG.XCP.DEFAULTSLAVEPORT
- SYStem.CONFIG.XCP.LOCK
- SYStem.CONFIG.XCP.SLAVE
- SYStem.CONFIG.XCP.SLAVEMode
- SYStem.CONFIG.XCP.UNLOCK
- SYStem.CONFIG.XCP.ConnectMode
- SYStem.CONFIG.XCP.DBGWRITE
- SYStem.CONFIG.XCP.DisConnect
- SYStem.CONFIG.XCP.PROTOCOL
- SYStem.CONFIG.XCP.slaveINFO
- SYStem.CONFIG.XCP.TEXTserviceFilter

▲ 'XCP Specific Functions' in 'XCP Debug Back-End'
▲ 'Release Information' in 'Legacy Release History'

SYStem.CONFIG.XCP.Connect Explicitly try to connect to an XCP slave

Format:	**SYStem.CONFIG XCP.Connect**

Connects to the XCP slave configured by **SYStem.CONFIG XCP.SLAVE**. Only available when **SYStem.CONFIG XCP.ConnectMode** is set to **Hold**.

See also

- SYStem.CONFIG.XCP

Format:	**SYStem.CONFIG XCP.ConnectMode** *<mode>*
<mode>:	**WhileUp** **Hold**

Configures when TRACE32 automatically connects and disconnects to or from the XCP slave.

WhileUp	• Connects when **SYStem.Mode Up**, **SYStem.Mode Attach** or a similar command is executed. • Disconnects when all cores are in **SYStem.Mode Down** or **SYStem.Mode NoDebug**.
Hold	This mode observes the pin states of the XCP slave without connecting to the target. • Connects when **SYStem.CONFIG XCP.Connect**, **SYStem.Mode Up**, **SYStem.Mode Attach** or a similar command is executed. • Disconnects when **SYStem.CONFIG XCP.DisConnect** is executed.

See also

■ SYStem.CONFIG.XCP ❏ SYStem.CONFIG.XCP.ConnectMode()

SYStem.CONFIG.XCP.CONNECTTIMEOUT Configure time out connection

[build 131621 - DVD 09/2021]

Format:	**SYStem.CONFIG.XCP.CONNECTTIMEOUT** *<time>*

Default: 1.s

Configures the time out of the initial TCP connect.

See also

■ SYStem.CONFIG.XCP

Format: **SYStem.CONFIG.XCP.DETECT.state**

Opens a window for interactive detection of XCP slaves. Requires the slave to implement the “slave detection on ethernet” procedure specified in XCP1.3.

SYStem.CONFIG.XCP.DBGWRITE Set DBGWRITE behavior manually

Format: **SYStem.CONFIG.XCP.DBGWRITE Auto** | **Normal** | **Verify**

Allows you to manually inform TRACE32 about the implementation of the DBGWRITE command in the XCP slave. In most cases, however, no manual intervention is necessary; TRACE32 configures the behavior automatically.

Auto (default)	TRACE32 derives the implementation from a list of known XCP slaves.
Verify	The XCP slave will read back the written value and verify it.
Normal	The XCP slave will not read back the written value.

See also

- SYStem.CONFIG.XCP

SYStem.CONFIG.XCP.DEFAULTSLAVEPORT Configure port of XCP slave

Format: **SYStem.CONFIG XCP.DEFAULTSLAVEPORT** *<port>*

Default: 1802.

Configures the port of the XCP slave you want to use. This configuration is not used if the port is explicitly specified by **SYStem.CONFIG XCP.SLAVE**.

<port> Port in decimal notation.

Example:

```
SYStem.CONFIG XCP.DEFAULTSLAVEPORT 12345.
```

See also

- SYStem.CONFIG.XCP

SYStem.CONFIG.XCP.DisConnect Explicitly disconnect from the XCP slave

Format: **SYStem.CONFIG XCP.DisConnect**

Disconnects from the XCP slave. Only available when **SYStem.CONFIG XCP.ConnectMode** is set to **Hold**.

See also

- SYStem.CONFIG.XCP

SYStem.CONFIG.XCP.LOCK Request exclusive access

Format: **SYStem.CONFIG XCP.LOCK FLASH | GENERIC**

Requests exclusive target access for TRACE32. For example, this can be used to protect the flash programming sequence.

FLASH Request exclusive access for flash programming.

GENERIC Request exclusive access for an action except flash programming.

See also

- SYStem.CONFIG.XCP
- SYStem.CONFIG.XCP.UNLOCK

SYStem.CONFIG.XCP.PROTOCOL Select the protocol variant

Format: **SYstem.CONFIG XCP.PROTOCOL Auto** | **ETAS** | **SWDBG** | **baseCAL**

Selects the protocol variant to be used. For information about the protocol variants, see **"Supported Protocol Variants"**, page 7.

Auto (default)	Automatic protocol selection. Selects the protocol based on information transmitted by the XCP slave. If the XCP slave indicates to have a debug resource, then the SWDBG protocol is used. Otherwise the ETAS protocol is used.
ETAS	ETAS proprietary protocol.
SWDBG	Software Debugging over XCP standard.
baseCAL	Uses standard **UPLOAD** and **DOWNLOAD** commands to access memory.

See also

- SYStem.CONFIG.XCP

SYStem.CONFIG.XCP.SLAVE Network address

Format: **SYStem.CONFIG XCP.SLAVE** *<address>*[**:***<port>*]

Configures the network address for the XCP slave you want to use.

<address>	IP address or network name of XCP slave.
<port>	Port used by XCP slave in decimal notation. If omitted the port configured by **SYStem.CONFIG XCP.DEFAULTSLAVEPORT** is used.

Example 1:

```
SYStem.CONFIG XCP.SLAVE 192.168.0.1
```

Example 2:

```
SYStem.CONFIG XCP.SLAVE 192.168.0.1:54321
```

See also

- SYStem.CONFIG.XCP

SYStem.CONFIG.XCP.SLAVEMode Configure connect mode of XCP slave

Format: **SYStem.CONFIG XCP.SLAVEMode NORMAL** | *<value>*

Configures the value of the XCP `MODE` parameter that is sent by TRACE32 to the XCP slave in the XCP `CONNECT` command. The exact meaning of this parameters depends on the XCP slave. Use **NORMAL** if not instructed otherwise by your XCP slave vendor.

NORMAL (default) Normal mode as specified in the XCP standard.

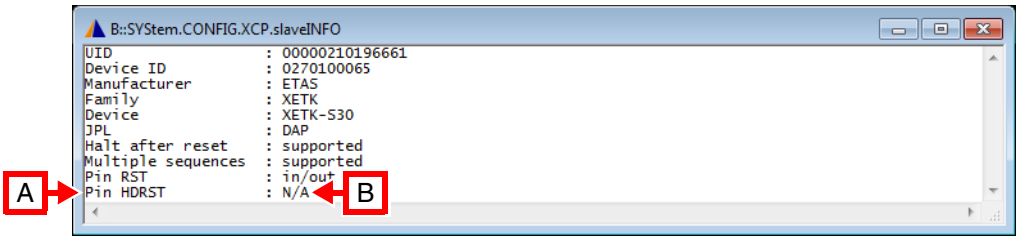
<value> Numerical value.

See also

- SYStem.CONFIG.XCP

Format: **SYStem.CONFIG XCP.slaveINFO**

Displays information about the properties of the connected XCP slave.



A Property name

B Property value

See also

- SYStem.CONFIG.XCP
- SYStem.CONFIG.XCP.INFO()
- SYStem.CONFIG.XCP.INFO.STR()

SYStem.CONFIG.XCP.TEXTserviceFilter Filter text service output

Format: **SYStem.CONFIG XCP.TEXTserviceFilter** *<filter1> … <filter8>*

XCP slaves can print text messages to the TRACE32 **AREA** window. The XCP slave sends messages using the XCP protocol to TRACE32. This command allows the user to filter these text messages:

- A message will be printed if it matches any filter.
- You can specify up to 8 filters.
- A filter can consist of text and the asterisk wildcard operator `*`.

Example 1: Print only messages starting with “ERROR:” and “WARNING”:

```
SYStem.CONFIG XCP.TEXTserviceFilter "ERROR:*" "WARNING:*"
```

Example 2: Print all messages:

```
SYStem.CONFIG XCP.TEXTserviceFilter "*"
```

See also

- SYStem.CONFIG.XCP

SYStem.CONFIG.XCP.UNLOCK End exclusive access

Format: **SYStem.CONFIG.XCP.UNLOCK**

Ends the exclusive access.

See also

- SYStem.CONFIG.XCP
- SYStem.CONFIG.XCP.LOCK

SYStem.DETECT.XCPTRI Match XCP resources to access ports

Format: **SYStem.DETECT.XCPTRI**

Automatically configures access ports to access the corresponding XCP resources.

XCP Specific Functions

In This Section

See also

- SYStem.CONFIG.XCP
- SYStem.CONFIG.XCP.ConnectMode()
- SYStem.CONFIG.XCP.INFO.STR()
- SYStem.CONFIG.XCP.Connected()
- SYStem.CONFIG.XCP.INFO()

SYStem.CONFIG.XCP.Connected() Current XCP connection state

[build 80958 - DVD 02/2017]

Syntax: **SYStem.CONFIG.XCP.Connected()**

Returns whether TRACE32 is currently connected to the XCP slave or not.

Return Value Type: Boolean.

Example:

```
PRINT SYStem.CONFIG.XCP.Connected()
```

SYStem.CONFIG.XCP.ConnectMode() XCP connection mode

[build 80988 - DVD 02/2017]

Syntax: **SYStem.CONFIG.XCP.ConnectMode()**

Returns the current mode configured by **SYStem.CONFIG XCP.ConnectMode**.

Return Value Type: String.

Example:

```
IF SYStem.CONFIG.XCP.ConnectMode()=="HOLD"
(
    SYStem.CONFIG XCP.DisConnect
)
```

SYStem.CONFIG.XCP.INFO() Numeric value of slave property

[build 70548 - DVD 09/2016]

Syntax: **SYStem.CONFIG.XCP.INFO(**<*property*>**)**

Returns the numeric value of a property shown in the **SYStem.CONFIG.XCP.slaveINFO** window.

Parameter Type: String. Any property name from the left column in the **SYStem.CONFIG.XCP.slaveINFO** window.

Return Value Type: Decimal value.

Example:

```
PRINT SYStem.CONFIG.XCP.INFO(Device ID)
```

SYStem.CONFIG.XCP.INFO.STR() String value of slave property

[build 70548 - DVD 09/2016]

Syntax: **SYStem.CONFIG.XCP.INFO.STR(**<*property*>**)**

Returns the string value of a property shown in the **SYStem.CONFIG.XCP.slaveINFO** window.

Parameter Type: String. Any property name from the left column in the **SYStem.CONFIG.XCP.slaveINFO** window.

Return Value Type: String.

Example:

```
PRINT SYStem.CONFIG.XCP.INFO.STR(Manufacturer)
```