LAUTERBACH
DEVELOPMENT TOOLS

CSWP Debug Back-End

MANUAL

CSWP Debug Back-End

TRACE32 Online Help

TRACE32 Directory

TRACE32 Index

TRACE32 Documents ..... 🗁

Debug Back-Ends ..... 🗁

CSWP Debug Back-End **1**

History **3**

Introduction **4**

Set Up the Debug and Trace Session **5**

Start TRACE32 PowerView 5

Install the USB Drivers 5

Configure the Debug Session for Arm CoreSight 5

Configure the Trace Recording 8

Command Reference **10**

SYStem.CSWP Configure CSWP communication 10

SYStem.CSWP.Connect Connect to server 10

SYStem.CSWP.DisConnect Terminate CSWP session 10

SYStem.CSWP.Init Initialize CSWP session 10

SYStem.CSWP.SETDEVICE Set device list 11

CSWP Debug Back-End

Version 06-Jun-2024

History

12-Feb-24 Initial version of the manual.

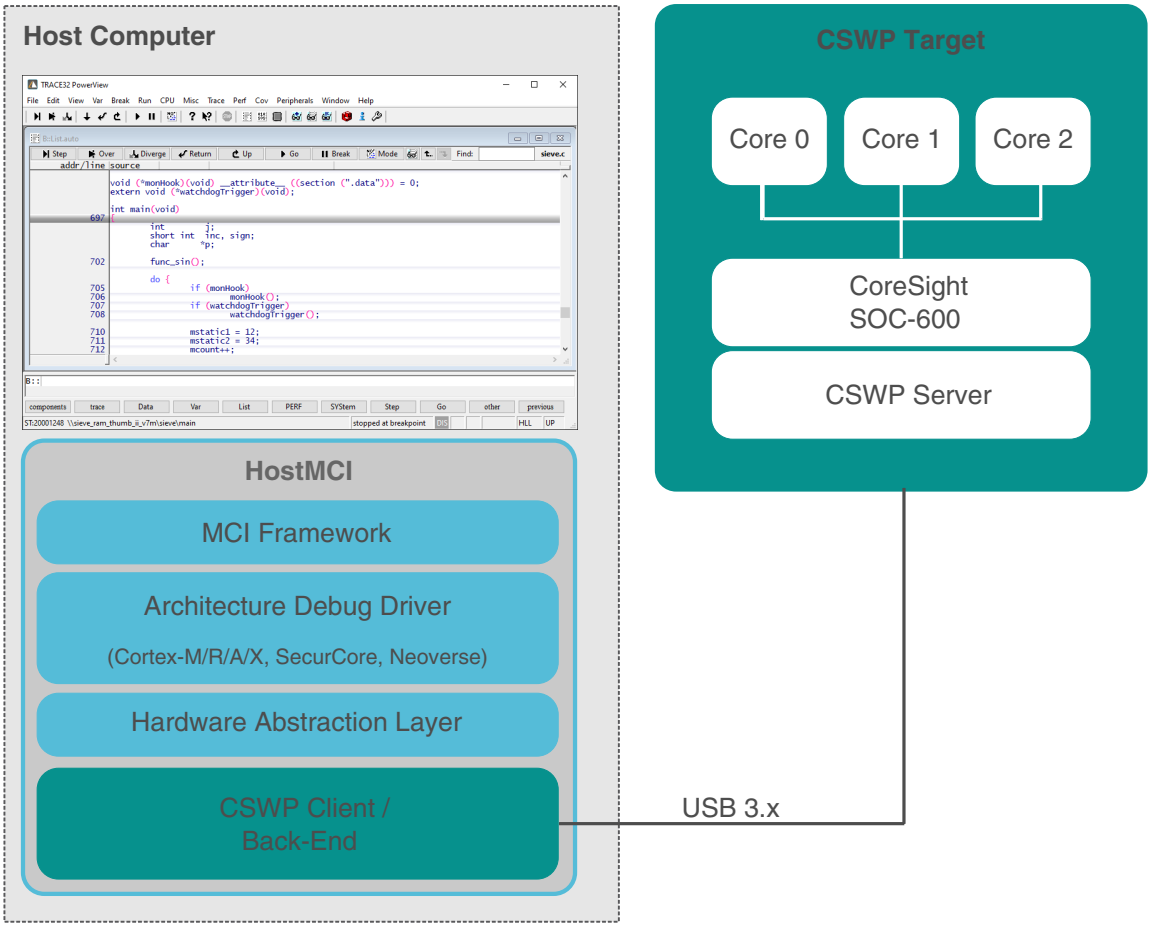
Introduction

The CoreSight Wire Protocol (CSWP) serves as a communication protocol specifically designed for debugging and tracing Arm-based System-on-Chips (SoCs) through functional interfaces like USB. The communication link between the SoC and the debug and trace software running on the host computer is managed by a CSWP server, an on-target debug agent. Lauterbach enables debugging in stop mode by using this technology and fully supporting all CoreSight components. Additionally, support for the SoC's CoreSight trace infrastructure enables the streaming of trace data, offering transparency into the internal processes of the SoC.

NOTE:	For now, TRACE32 only supports CSWP with USB connection.

The system comprises two processes:

1. The process that enables TRACE32 to debug the target and control trace generation.
2. CSWP Client/Server manage the process responsible for communication through the functional interface.



Multiple PowerView instances can be connected to one CSWP client/back-end in oder to perform AMP debugging.

Set Up the Debug and Trace Session

Start TRACE32 PowerView

Start one or more TRACE32 PowerView instances as described in **"PowerView System Configurations"** in TRACE32 Debug Back-Ends, page 6 (backend_overview.pdf).

Install the USB Drivers

In order to use TRACE32 for USB debugging, it is necessary that the required USB driver is installed on the host PC. Refer to **"Installation of the USB Driver"** in Debugging via USB User´s Guide, page 6 (usbdebug_user.pdf) for details.

Configure the Debug Session for Arm CoreSight

The following steps are necessary for each TRACE32 PowerView instance started:

1. **Specify Arm CoreSight Wire Protocol (CSWP) as debug protocol.**

   ```
   SYStem.CONFIG.DEBUGPORT CSWP0
   ```

2. **Prepare multicore debugging setup.**

   For details refer to **"Quick Start for Multicore Debugging"** in Armv8 and Armv9 Debugger, page 33 (debugger_armv8v9.pdf).

3. **Configure debugger for CoreSight components.**

   For core debug registers (mandatory):

   ```
   SYStem.CONFIG.COREDEBUG.Base <access_class>:<base_address>
   [<access_class>:<base_address> ...]
   ```

   For CTI registers (mandatory):

   ```
   SYStem.CONFIG.CTI1.Base <access_class>:<base_address>
   [<access_class>:<base_address> ...]
   ```

   Details about the configuration of the other CoreSight components can be found in **"Setup of the Debugger for a CoreSight System"** (app_arm_coresight.pdf).

4. **Open USB debug connection.**

```
SYStem.CONFIG.USB.SETDEVice Debug <interface_number> <vid>|<pid>
```

Refer to **"Debugging via USB User´s Guide"** (usbdebug_user.pdf) for additional USB debug connection details.

5. **Initiate CSWP connection.**

```
SYStem.CSWP.Connect
SYStem.CSWP.Init
```

6. **Specify CoreSight device for debugging.**

```
SYStem.CSWP.SETDEVICE <device_number> "<device_name>" "<device_type>"
```

7. **Specify APB bus for CSWP device access.**

```
SYStem.CONFIG.APBAP1.CSWPDEV <device_number>
```

8. **Establish debug communication.**

```
SYStem.Attach
Break
```

9. **Additional preparations for debugging.**

Load the application and perform any further necessary steps.

This is how a startup script might be written.

```
; enable Arm CoreSight Wire Protocol (CSWP)
SYStem.CONFIG.DEBUGPORT CSWP0

; prepare multicore debugging setup
SYStem.CPU CortexA78AE
SYStem.CONFIG.CoreNumber 16.
SYStem.CONFIG CORE 1. 1.
CORE.ASSIGN 1.

; configure debugger for CoreSight components
SYStem.CONFIG.COREDEBUG.Base APB:0x51010000 APB:0x51110000 \
APB:0x51210000 APB:0x51310000 APB:0x52010000 APB:0x52110000 \
APB:0x52210000 APB:0x52310000 APB:0x53010000 APB:0x53110000 \
APB:0x53210000 APB:0x53310000 APB:0x54010000 APB:0x54110000 \
APB:0x54210000 APB:0x54310000
SYStem.CONFIG.CTI1.Base APB:0x51020000 APB:0x51120000 APB:0x51220000 \
APB:0x51320000 APB:0x52020000 APB:0x52120000 APB:0x52220000 \
APB:0x52320000 APB:0x53020000 APB:0x53120000 APB:0x53220000 \
APB:0x53320000 APB:0x54020000 APB:0x54120000 APB:0x54220000 \
APB:0x54320000

; open USB debug connection
SYStem.CONFIG USB SETDEVice Debug 1. 0x5c0 0x2

; initiate CSWP connection
SYStem.CSWP.Connect
SYStem.CSWP.Init

; specify CoreSight device for debugging
SYStem.CSWP.SETDEVICE 0. "CSMEMAP" "mem-ap.v2"

; specify APB bus for CSWP device access
SYStem.CONFIG.APBAP1.CSWPDEV 0.

; establish debug communication
SYStem.Attach
Break

; load application
DO ~~/demo/arm/compiler/gnu-pic/demo_sieve.cmm 0x0

ENDDO
```

Configure the Trace Recording

To stream trace data off-chip, you must adjust the debug configuration accordingly.

1. **Configure the trace CoreSight components**

   For ETM registers (mandatory):

   ```
   SYStem.CONFIG.ETM.Base <access_class>:<base_address>
   [<access_class>:<base_address> …]
   ```

2. **Open USB trace connection.**

   ```
   SYStem.CONFIG.USB.SETDEVice Debug <interface_number> <vid>|<pid>
   ```

   Refer to **"Debugging via USB User´s Guide"** (usbdebug_user.pdf) for additional USB connection details.

3. **Set the trace port type to CSWP**

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

```
SYStem.CONFIG.DEBUGPORT CSWP0

SYStem.CPU CortexA78AE
SYStem.CONFIG CoreNumber 16.
SYStem.CONFIG CORE 1. 1.
CORE.ASSIGN 1.
SYStem.CONFIG.COREDEBUG.Base APB:0x51010000 APB:0x51110000 \
APB:0x51210000 APB:0x51310000 APB:0x52010000 APB:0x52110000 \
APB:0x52210000 APB:0x52310000 APB:0x53010000 APB:0x53110000 \
APB:0x53210000 APB:0x53310000 APB:0x54010000 APB:0x54110000 \
APB:0x54210000 APB:0x54310000
SYStem.CONFIG.CTI1.Base APB:0x51020000 APB:0x51120000 APB:0x51220000 \
APB:0x51320000 APB:0x52020000 APB:0x52120000 APB:0x52220000 \
APB:0x52320000 APB:0x53020000 APB:0x53120000 APB:0x53220000 \
APB:0x53320000 APB:0x54020000 APB:0x54120000 APB:0x54220000 \
APB:0x54320000

SYStem.CONFIG.ETM.Base APB:0x51040000 APB:0x51140000 APB:0x51240000 \
APB:0x51340000 APB:0x52040000 APB:0x52140000 APB:0x52240000 \
APB:0x52340000 APB:0x53040000 APB:0x53140000 APB:0x53240000 \
APB:0x53340000 APB:0x54040000 APB:0x54140000 APB:0x54240000 \
APB:0x54340000

SYStem.CONFIG.TRACEJUNCTION1.Name "NPFUNNELS"
SYStem.CONFIG.TRACEJUNCTION1.TraceSource ETM

SYStem.CONFIG.TRACEPORT.Type CSWP
SYStem.CONFIG.TRACEPORT.TraceSource TRACEJUNCTION

HAnalyzer.SIZE 1000000000.
Trace.METHOD HAnalyzer
HAnalyzer.TraceCONNECT TP-CSWP
HAnalyzer.AutoArm ON

SYStem.CONFIG USB SETDEVice Debug 1. 0x5c0 0x2
SYStem.CONFIG.USB.SETDEVice Trace 0. 0x5c0 0x2

SYStem.CSWP.Connect
SYStem.CSWP.Init

SYStem.CSWP.SETDEVICE 0. "CSMEMAP" "mem-ap.v2"

SYStem.CONFIG.APBAP1.CSWPDEV 0.

SYStem.CSWP.REQ_STREAM_SET_SINK 0. 0x83

SYStem.Mode Up
DO ~~/demo/arm/compiler/gnu-pic/demo_sieve.cmm 0x0
ENDDO
```

SYStem.CSWP — Configure CSWP communication

The **SYStem.CSWP** command group consists of a variety of request messages designed to facilitate the configuration of CSWP communication, for accessing CoreSight components via USB, for debugging and tracing purposes.

See also

- SYStem.state

SYStem.CSWP.Connect — Connect to server

Format: **SYStem.CSWP.Connect**

Uses the settings previously configured with the **SYStem.CONFIG** commands to establish a connection to the CSWP server.

SYStem.CSWP.DisConnect — Terminate CSWP session

Format: **SYStem.CSWP.DisConnect**

Disconnects from existing connection to the CSWP server to release any resources.

SYStem.CSWP.Init — Initialize CSWP session

Format: **SYStem.CSWP.Init**

Starts the CSWP session and allows the target to initialize the necessary resources.

Format:	**SYStem.CSWP.SETDEVICE** *<number> “<name>” “<type>”*
<type>:	**mem-ap.v2**

Sets device in device the list by specifying the number, name, and type, only “mem-ap.v2” is supported for the device type.

Example:

```
SYStem.CSWP.Connect
SYStem.CSWP.Init
SYStem.CSWP.SETDEVICE 0. "CSMEMAP" "mem-ap.v2"
SYStem.CONFIG.APBAP1.CSWPDEV 0.
```