AN14211

How to Utilize Trace Components on i.MX RT1180

Rev. 1.0 — 10 December 2024 Application note

Document information

Information	Content
Keywords	AN14211, Trace, i.MX RT1180, IAR, Lauterbach, Arm Cortex-M, Debug, Arm CoreSight
Abstract	In systems based on Microcontrollers (MCUs) or System-on-Chip (SoC), errors often appear in the development phase. The errors, bugs, and performance issues can be removed by simple debugging or by trace components, which offers many features for that purpose.

1 Introduction

In systems based on Microcontrollers (MCUs) or System-on-Chip (SoC), errors often appear in the development phase. These errors must be diagnosed and removed. The basic way to debug the system is to use a breakpoint, step through the code, and inspect the memory state using a memory and register viewer. If the error occurs only in some specific conditions, this kind of debugging is not sufficient. The tracing components are dedicated hardware and they collect data from the core/buses. Tracing enables you to find instructions that were executed before the application code crashed. It is also possible to inspect how often a part of memory is read/written or the time duration of certain functions can be inspected.

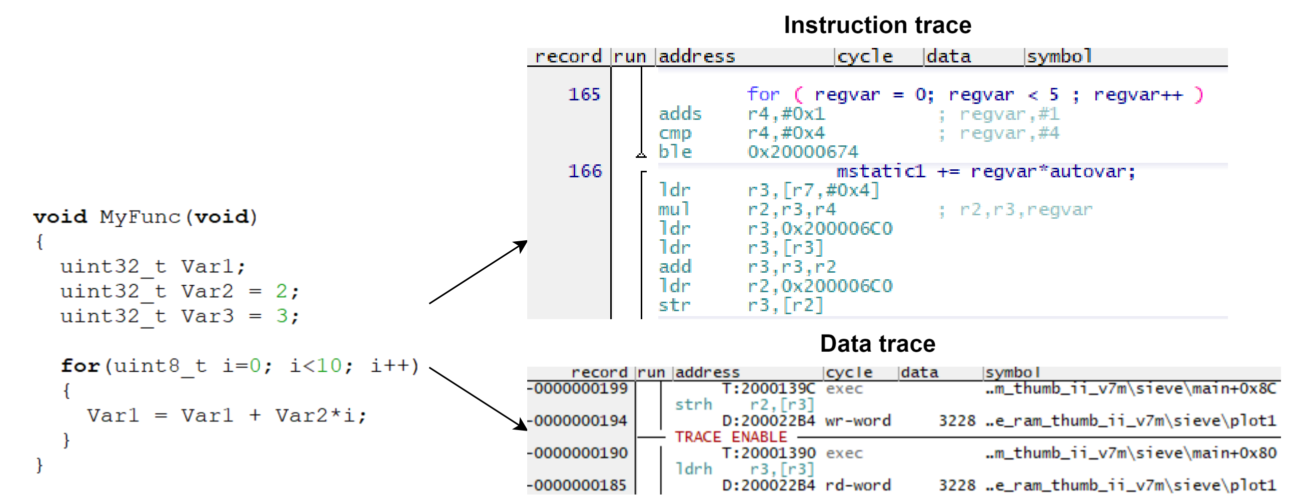

Figure 1. Types of information from trace components – instruction flow and data flow

The main advantage of tracing against usual means of debugging is that it is nonintrusive. That means that this dedicated hardware does not influence the application running on an SoC, so real conditions can be met. In usual debugging, the code is stepped and then the whole chip is stopped from running. Tracing can be understood as an advanced means of debugging.

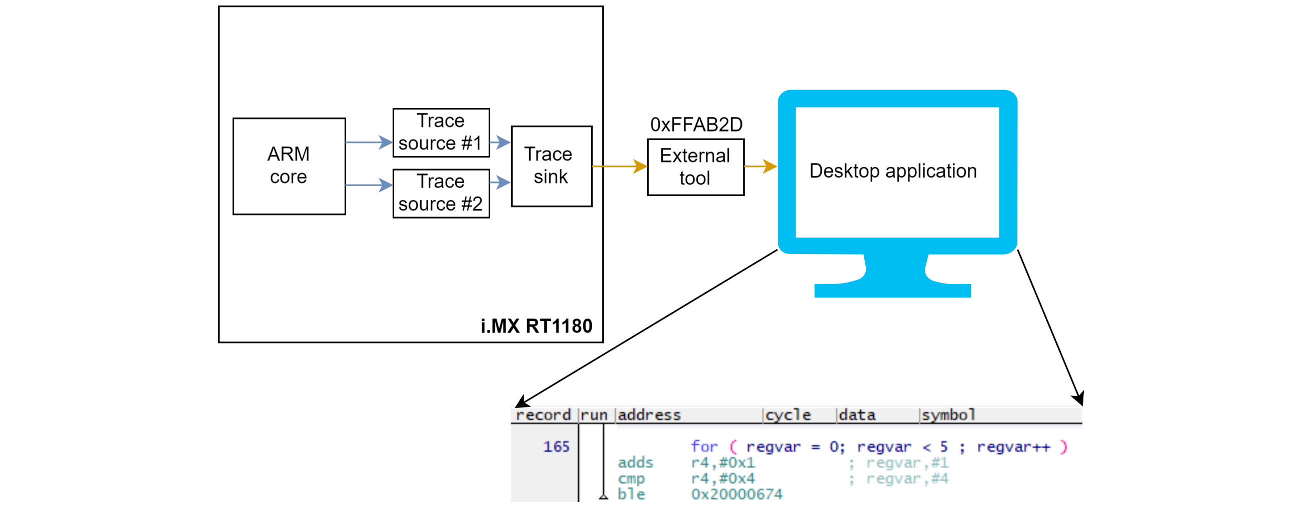

Figure 2. General trace data flow

2 Tracing capabilities on i.MX RT1180

Trace data are generated in trace source components. The trace components have registers for configuration and can be accessed via the Advanced Peripheral Bus (APB). Data are then sent in a specific format through the Advanced Trace Bus (ATB) and its components to the trace sink. The data can be read from the trace sink. On the PC site, data are decoded and visualized in a readable format in a trace desktop application. In this application note, the Lauterbach TRACE32 is used, but desktop applications from other vendors are available on the market. There is an important difference between i.MX RT1170 and i.MX RT1180. i.MX RT1180 is missing a parallel TPIU port, so the trace data cannot be streamed directly to the device. Figure 3 shows the block diagram of trace components of the i.MX RT1180 for the dual-core variant. For the single-core variant, there is no difference from a programmer's perspective.

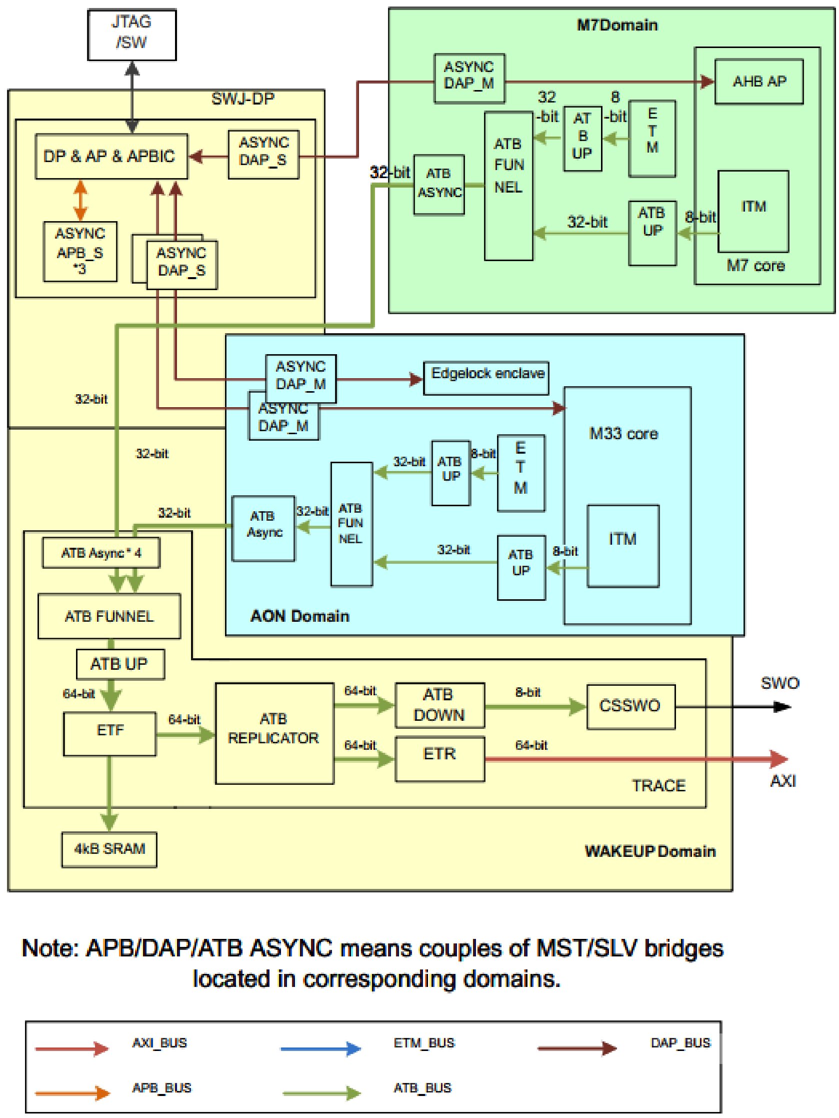

Figure 3. Block scheme of trace components on i.MX RT1180

Trace source: Trace source is the component that generates trace data.

- Embedded Trace Macrocell (ETM)
- Instrumentation Trace Macrocell (ITM)

- Data Watchpoint and Trace (DWT)

Trace sink: Trace sinks are the endpoints for trace data.

- On-chip trace sink: The trace data are stored in the dedicated on-chip memory or redirected to the system memory. Then the debugger can read the trace data via a debug interface.
  - Embedded Trace Router (ETR)
  - Embedded Trace FIFO (ETF)
- Off-chip trace sink: The trace data are streamed to the trace tool through a physical interface.
  - Serial Wire Output (SWO)
  - Trace Port Interface Unit (TPIU)

Trace link: The trace link is a component that links the trace and non-trace components together.

- Trace funnel: The trace funnel combines multiple ATBs into a single ATB.
- Replicator: The ATB replicator enables two trace sinks to be wired together and to operate from the same incoming trace stream.
- Bridge: The ATB asynchronous bridge enables the data transfer between two asynchronous clock domains.
- Cross-trigger network: It consists of a Cross-Trigger Interface (CTI) and Cross-Trigger Matrices (CTMS). The CTI can send triggers between trace components.

Timestamp generator: The timestamp generator is a simple counter that generates timestamps. This allows for a later alignment of trace information.

2.1 Embedded Trace Macrocell (ETM)

This trace component permits the instruction trace and the data trace. However, in the i.MX RT1180 implementation, it permits only the instruction trace. This means that the value of the Program Counter (PC) is periodically sampled. The ETM can also insert timestamps into trace data. This means that the function can be measured from a time perspective. To generate timestamps, the ETM uses a system counter.

Note: *The maximum frequency of this counter is 24 MHz. The DWT/ITM timestamp counter runs at the core-clock frequency.*

2.2 Instrumentation Trace Macrocell (ITM)

The following three sources can generate data for the ITM:

- Software trace: `printf`-like debugging. Writing to the stimulus register of the ITM initiates the emitting of a packet.
- Hardware trace: The DWT generates packets and the ITM emits them.
- Time stamping: Timestamps are generated relative to a packet.

2.3 Data and Watchpoint Trace (DWT)

This trace component provides watchpoints, data tracing, and system profiling for the processor. The DWT also includes comparators that can compare user predefined values with these real-time values:

- Data address
- Instruction address
- Data value
- Cycle-count value

3 Trace in IAR

The following sections explain how to set and use some useful trace components in the IAR EW with the J-Link debugger. The code examples are provided. By executing the steps in the following sections, these examples can be also used in other development tools.

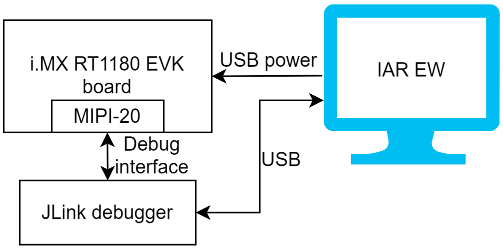

Figure 4. Configuration for the demonstration in IAR EW

3.1 Using DWT cycle counter

The DWT cycle counter is a free-running 32-bit cycle counter. This counter is incremented on each cycle of the core clock. When the counter overflows, it wraps to zero. The event triggers are derived from this counter; for example, the trigger for PC sampling. If the SysTick timer is not available or already utilized, it can also be used to measure the time of functions, data transports, and so on.

1. Enable the clock for trace components:
   - `CLOCK_EnableClock(kCLOCK_Cstrace);`
2. Enable the trace in the debug exception and in the monitor control register:
   - `CoreDebug->DEMCR |= 0x01000000;`
3. Enable the DWT in the DWT CTRL register:
   - `DWT->CTRL |= 0x00000001;`
4. Reset the DWT counter by writing zero to the CYCCNT register:
   - `DWT->CYCCNT = 0;`
5. Read the actual CYCCNT register value:
   - `cycleCnt = DWT->CYCCNT;`
6. The `cycleCnt` stores the number of core cycles between the two events by setting `CYCCNT` to zero and then reading the `CYCCNT` value.

3.2 Using Serial Wire Output (SWO)

The SWO is an extension to the SWD. The SWO is an additional pin to the SWD interface that allows the target core to send data to the master system (a computer running an IDE). The pin is unidirectional, so the data can be only sent in one direction (from the core to the host computer). The SWO supports two data formats (NRZ and Manchester encoding). The SWO can be used to record the function/interrupt entry and exit, periodic PC sampling, event notification, variable/memory changes over time, and sending messages. The last mentioned use case is an alternative to using UART when debugging the code, because you can use the ITM to send a string to the SWO. The ITM supports up to 32 stimulus ports. The SWO is more suitable for sending messages via the ITM than sending the trace data (like ETM), because it is a single-wire interface with no big data throughput, which can lead to a potential overflow (see Figure 5).

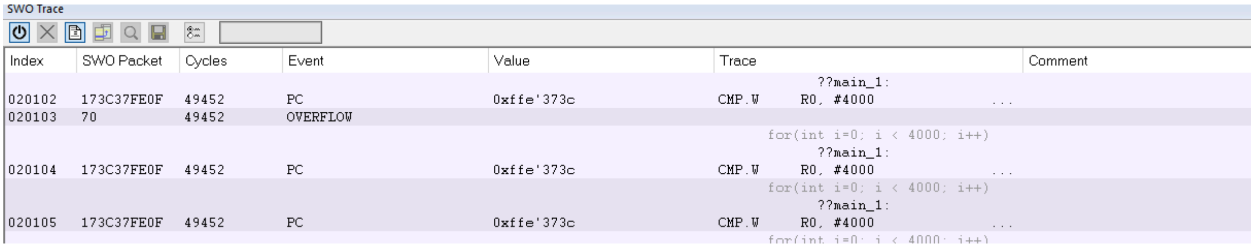

Figure 5. IAR IDE displaying trace information from DWT over SWO and OVERFLOW packet due to SWO bandwidth limitation

3.2.1 ITM sending messages

Using the ITM to send debug messages has the following advantages:

- It preserves the SoC resources, because the target application may use all UARTs.
- The ITM is less intrusive, because you only write data to the ITM stimulus register and you do not need to handle UART functions.
- The UART functions occupy space in the RAM/FLASH memory.

The following steps explain how to set the ITM to send strings via the ITM stimulus port 0. To use other ports (in step 3), the port must be enabled and the `PrintChar/PrintString` functions must be modified to have the channel number as the input parameter.

1. Enable the clock for SWO and trace components:
   - `CLOCK_EnableClock(kCLOCK_Csswo);`
   - `CLOCK_EnableClock(kCLOCK_Cstrace);`
2. Enable the trace in the debug exception and in the monitor control register:
   - `CoreDebug->DEMCR |= 0x01000000;`
3. Set the ITM stimulus port (in this example, the stimulus port 0 is set):
   - `ITM->ENA = 0x1;`
4. Select the encoding (NRZ/Manchester):
   - `SWO->SPPR = 0x0;`
5. Select the clock divider for the SWO.

```
SWO_ACPR = 23;              // Divisor for TRACECLKIN is Prescaler + 1
```

6. Set the ETF:

```
ETF->FFCR = 0x0;            // Set the ETF in the normal mode
ETF->MODE = 0x2;            // Hardware mode to drain the trace data on the
 ATB bus
ETF->BUFWM = 0x0;           // Set Buffer level Water Mark
ETF->CTL = 0x1;             // Enable the trace capture
ETF->FFCR = 0x1;            // Enable the formatter
void SWO_PrintChar(char c);
```

7. Use these functions for printing messages:
   - The following function writes one character into the stimulus register:

```
void SWO_PrintChar(char c)
{
    /* Wait until STIMx is ready, then send data */
     while ((ITM_STIM_0 & 1) == 0);
    ITM_STIM_0 = c;
```

```
}
```

- The following function prints a string using the `PrintChar` function:

```
void SWO_PrintString(const char *s);
{
    /* Print out character per character */
    while (*s)
    {
        SWO_PrintChar(*s++);
    }
}
```

To view the data from the ITM stimulus port, the channel must be enabled, as shown in Figure 6. The setting can be done in the "JLink->SWO Configuration" menu.

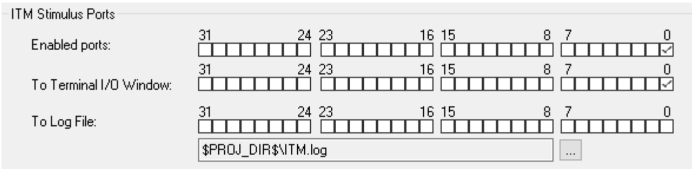

Figure 6. Enabling stimulus port 0

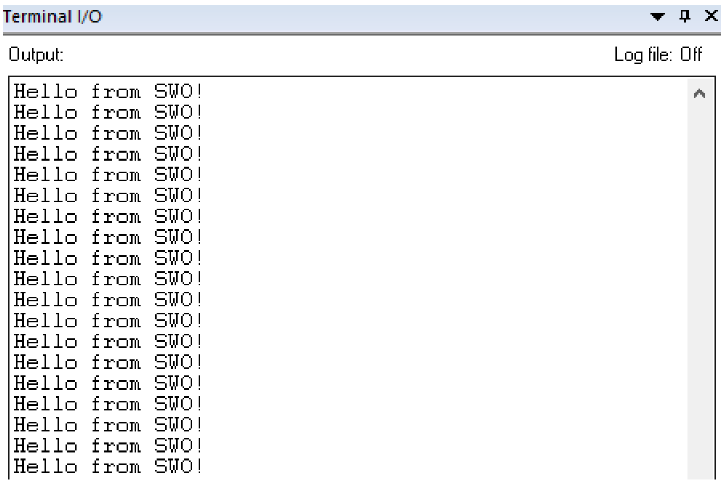

Figure 7. Printing messages via ITM from code to IAR terminal

4 Lauterbach tool

The principle of setting trace peripherals in tools from other vendors is very similar. The behavior of the Lauterbach environment is set by a script file or in the user interface of the TRACE32 program. This application note explains the basic settings of trace components in the TRACE32 user interface for a better illustration on how to use these trace components. Simple trace use cases can be set in the user interface. For a more specific use case, it is recommended to write your own script and see the *Lauterbach General Commands Reference Guide* for the particular commands.

```
WinCLEAR

; --------------------------------------------------------------------------------
; initialize and start the debugger
RESet
SYStem.CPU IMXRT1187-CM33
SYStem.CONFIG.DEBUGPORTTYPE SWD
SYStem.Option DUALPORT ON
SYStem.JtagClock 10MHz
Trace.DISable
SYStem.Up

; --------------------------------------------------------------------------------
; Init ECC SRAM
Data.Set 0x20000000++0x1FFFF %Long 0x0

; --------------------------------------------------------------------------------
; load demo program (uses internal RAM only)
Data.LOAD.Elf "~~~~/sieve_ram_thumb_ii_v7m.elf"

; --------------------------------------------------------------------------------
; initialize ONCHIP trace (ETF, ETM, ITM)
Data.Set ASD:0x44460010 %Long 0yXXXXxxxxXXXXxxxxXXXXxxxxXXXXxxx1   ; SRC_GENERAL.SCR[BT_RELEASE_M7] = 1

Trace.METHOD Onchip
Trace.TraceCONNECT ETF
Trace.AutoInit ON
ITM.DataTrace CorrelatedData
ITM.ON
ETM.Trace ON
ETM.COND ALL
ETM.ON

; --------------------------------------------------------------------------------
; start program execution
Go.direct main
WAIT !STATE.RUN()

; --------------------------------------------------------------------------------
; open some windows
WinCLEAR
Mode.Hll
WinPOS 0. 0. 116. 26.
List.auto
WinPOS 120. 0. 100. 8.
Frame.view
WinPOS 120. 14.
Var.Watch
Var.AddWatch %SpotLight ast flags
WinPOS 120. 25.
Trace.List
WinPOS 0. 32.
Var.DRAW %DEFault sinewave

ENDDO
```

Figure 8. Example of script file for Lauterbach

This section explains the basic operation of the i.MX RT1180 trace peripherals using Lauterbach with μTrace and running a demo project from Lauterbach. The instructions go step by step and the most important settings are explained. Before starting, download and install the TRACE32 tool from Lauterbach. Open the TRACE32 program and perform the following steps:

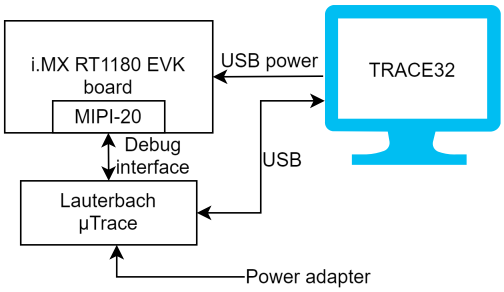

Figure 9. Configuration of Lauterbach TRACE32 tool

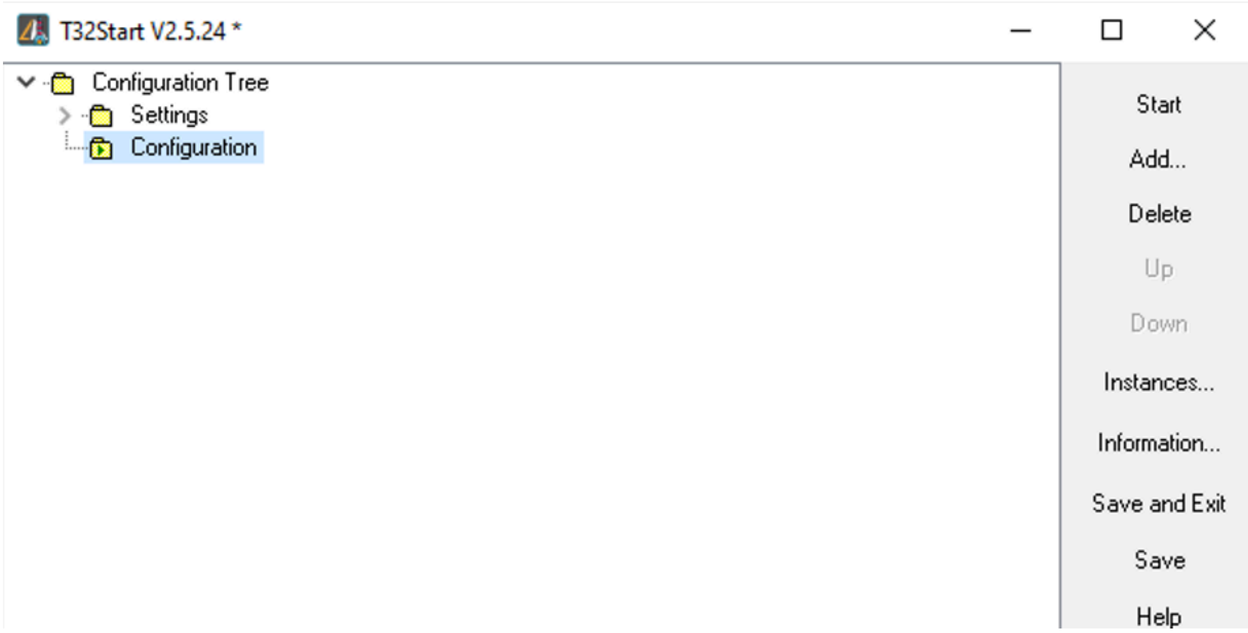

Figure 10. Step 1

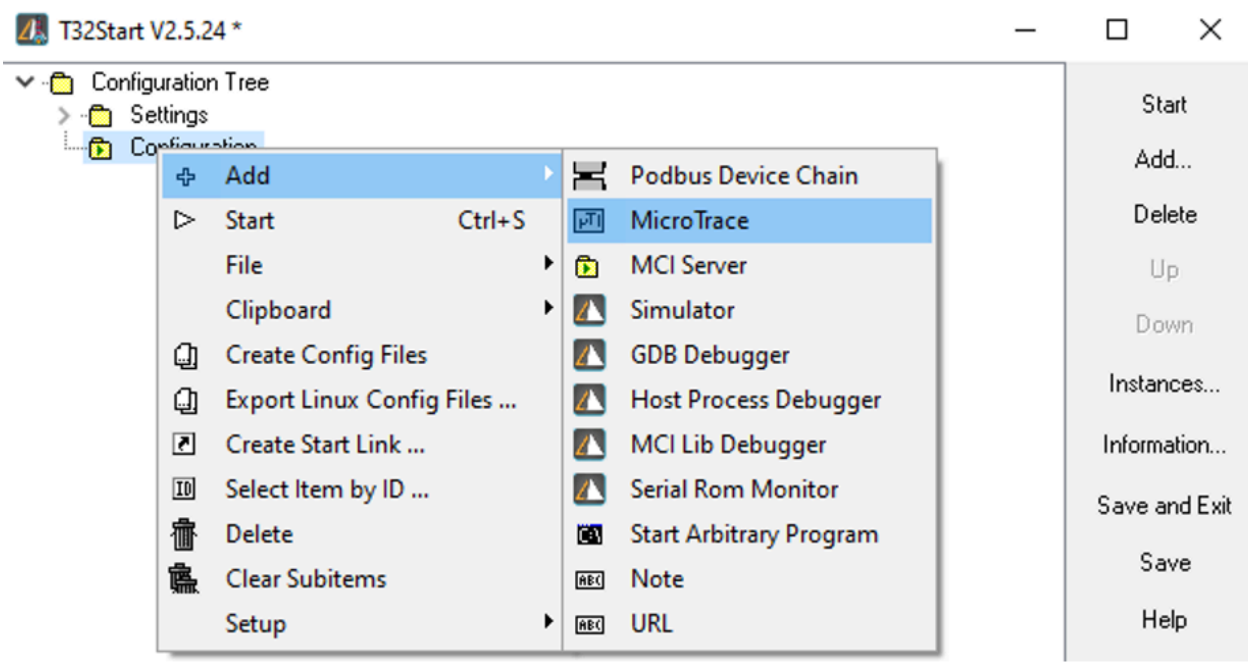

Figure 11. Step 2

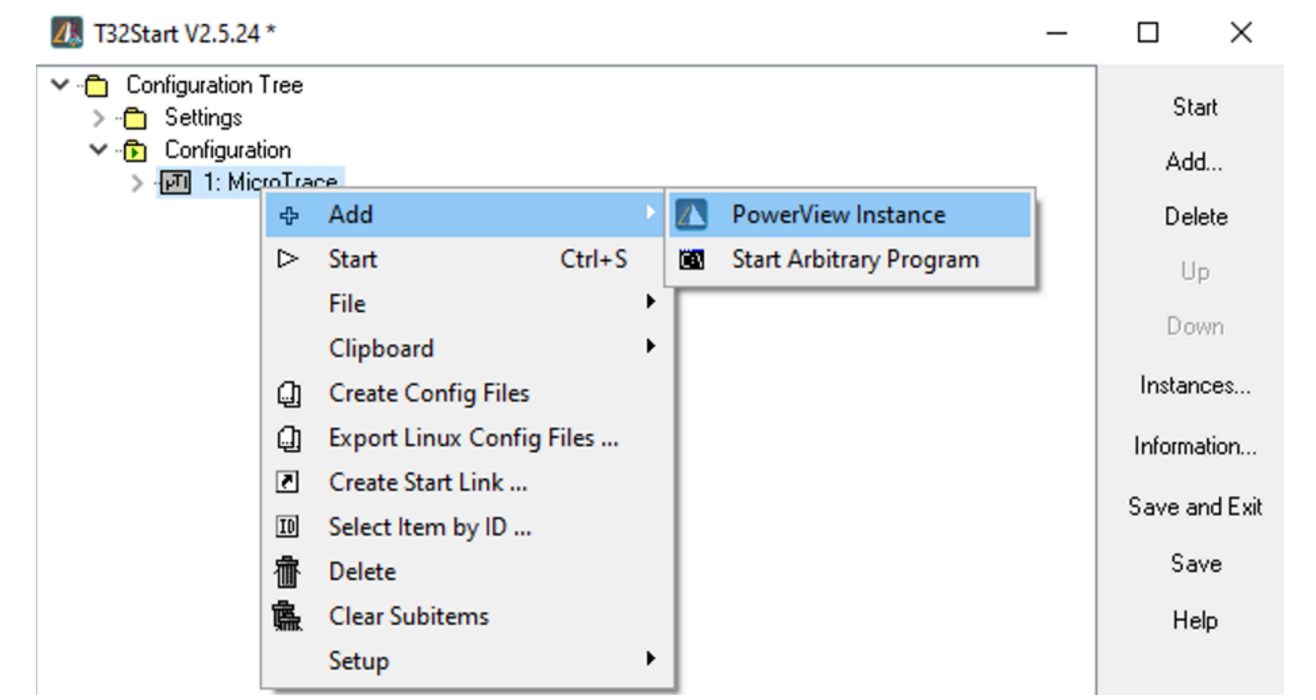

Figure 12. Step 3

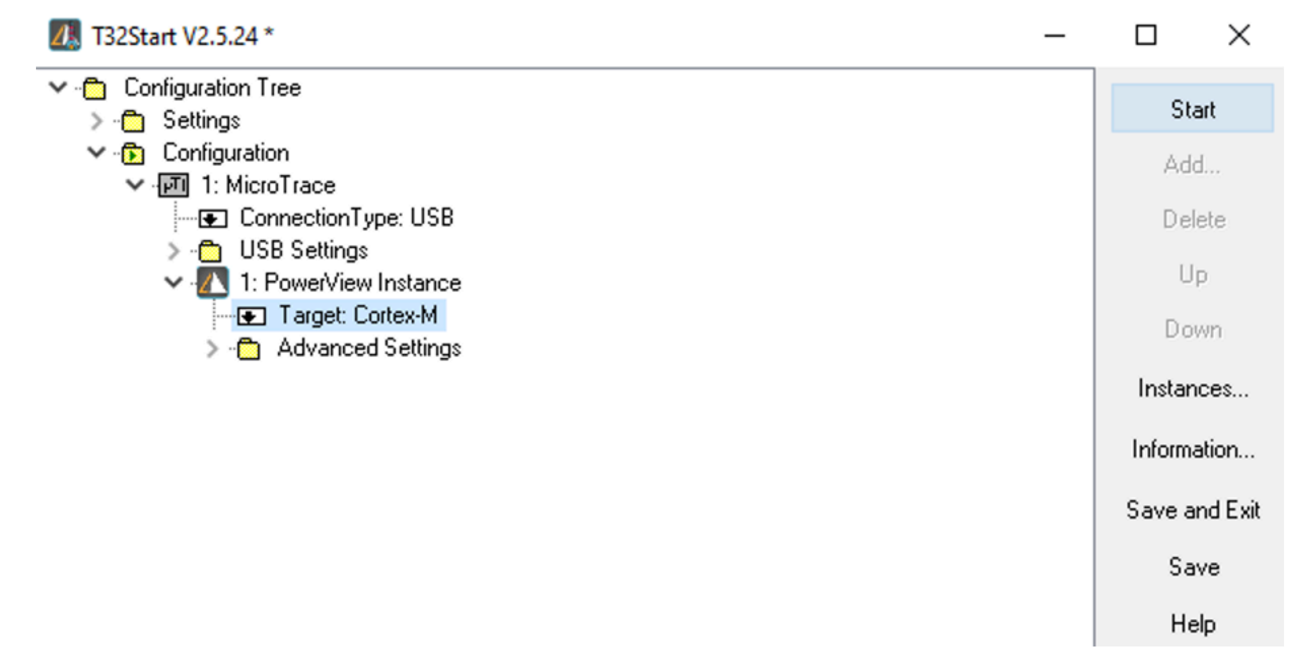

Figure 13. Step 4

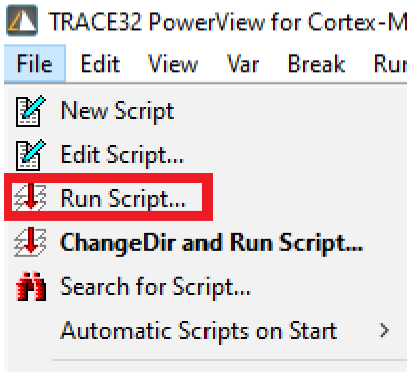

Figure 14. Step 5

Note: *By clicking the "Edit Script" option, you can inspect and modify the script code.*

Open the ...`\T32\demo\arm\hardware\imxrt` folder. This folder contains other folders for a corresponding SoC from the RT 4 digit family. Because this application note is focused on i.MX RT118x, these two folders are important: `imxrt1181` for the 144 single-core variant and `imxrt1187` for the multi-core variant. Navigate to the `imxrt1187-cm33` folder. You can derive some information from the name of the demo scripts:

- If the script name includes an ETF, the trace data are stored in the ETF (dedicated SRAM memory for trace).
- If the script name includes ETR, the trace data are stored in the on-chip RAM memory, because ETR has the AXI master interface.

Now, the demo project is stopped at the start of the `main()` function.

B.List.auto:

- This window shows the actual position of the executed program. This window is similar to the development IDE. Control the program using the "start/pause program" or "step through the code" buttons.

Trace.List:

- This window shows the history of the executed instructions.

Var.Watch:

- Inspect the values of variables in this window.

B.Var.Draw:

- This window displays the contents of an array or a structure element graphically. In this demo, it is the sinewave array. To visualize the variable over time, use `Var.PROfile`.
- You can also set the method how the Lauterbach tool reads the memory. This setting can be done in the "CPU -> System Settings" menu.
  - **MemAccess:**
    - DAP: The data is read by the `CoreSight` memory access port in runtime.
    - StopAndGo: The program execution is shortly stopped so that the memory can be read.
    - Denied: Both accesses mentioned above are denied.

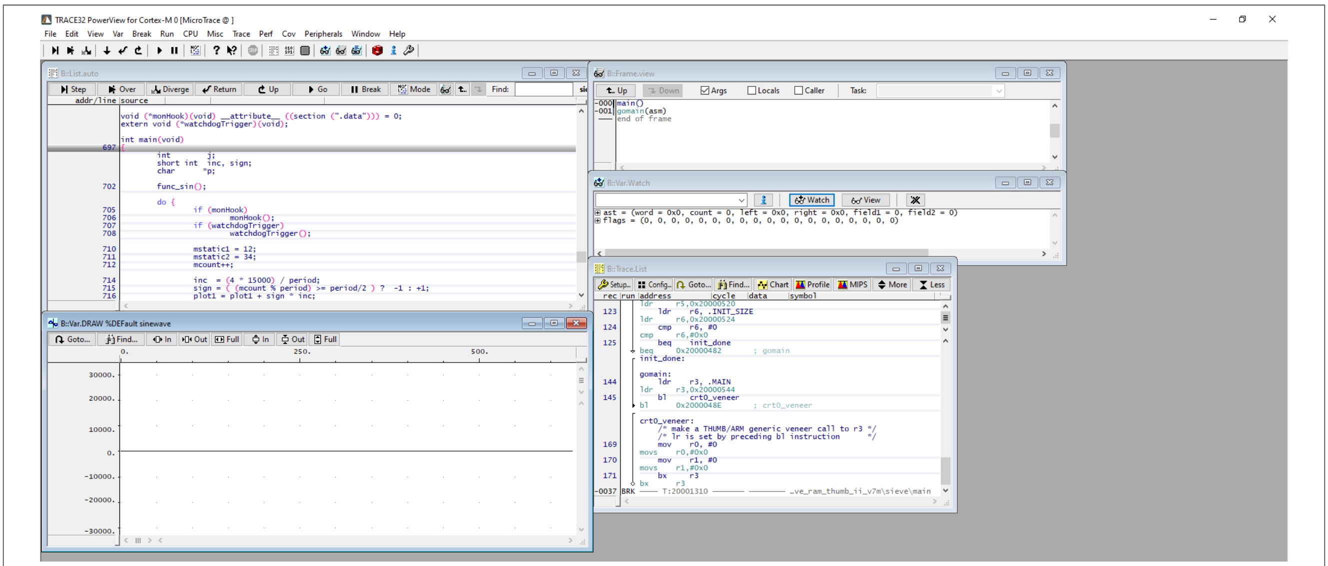

Figure 15. Default view of TRACE32 for ETF trace demo

You can run the program by clicking the "Go" button. The tracing is started simultaneously by clicking the "Go" button if AutoArm is enabled. Now we look closer on the setting trace component in the "Trace->Configuration" menu.

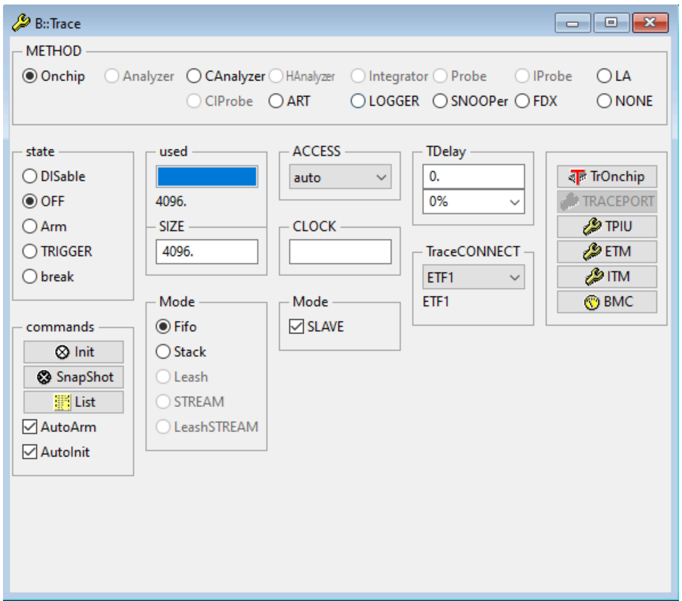

Figure 16. Setting window for trace

Method:

You can set how the trace data are obtained and where the trace data are stored. For this example and for i.MX RT1180, the on-chip method is the most suitable. Some examples of methods are as follows:

- **On-chip:** The trace data are saved in the on-chip trace buffer/memory.
- **CAnalyzer:** The trace memory is provided by the TRACE32 tool (suitable for devices with TPIU).
- **Snooper:** This method enables you to gain runtime information with just a debugger. It reads out information such as memory/variable contents, the program counter, or other system information while the program is running.

Note: *There are more methods available. For more details, see the Lauterbach documentation.*

Mode:

You can choose between the FIFO or Stack modes. This represents how the trace data are stored in the memory. If the Stack mode is set, the trace data are stored chronologically until the memory is full. If the FIFO mode is set and the memory is full, the oldest data are rewritten by the newest data, so it acts like a circular buffer.

TraceCONNECT:

It sets the destination of data. If ETF is set, the trace data are stored in a dedicated 4 kB SRAM trace memory. If ETR is set, the trace data is sent via the AXI interface to the on-chip memory, which allows to allocate more memory space.

In the right part of the "Trace" window, there are buttons to set the trace components - TPIU, ETM, ITM, and so on.

TPIU:

In this window, you can set the SWO as a trace sink and output the trace data through this interface.

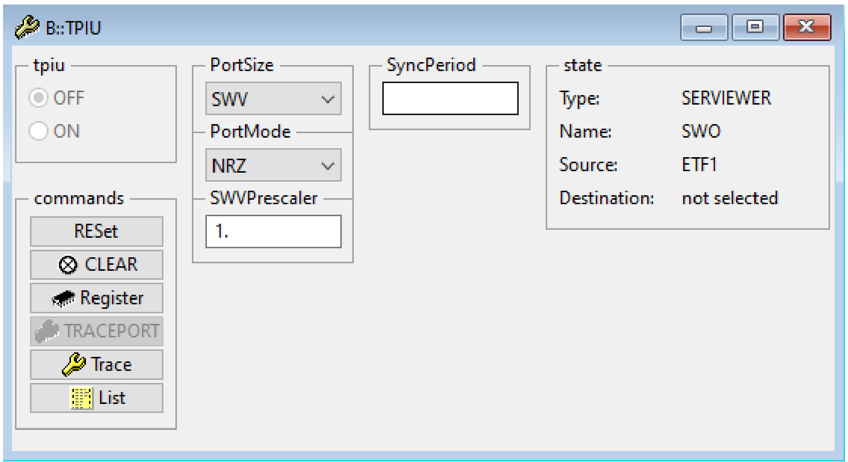

Figure 17. Setting window for TPIU

ETM:

In this window, you can set the ETM. More detailed information about the ETM, such as its version, number of comparators, and so on are in the resources. By clicking the "Advanced" button, the additional settings for the ETM appear. TraceInclude configures the ETM to generate a program trace only for the specified address range(s). The TraceExclude works in a different way.

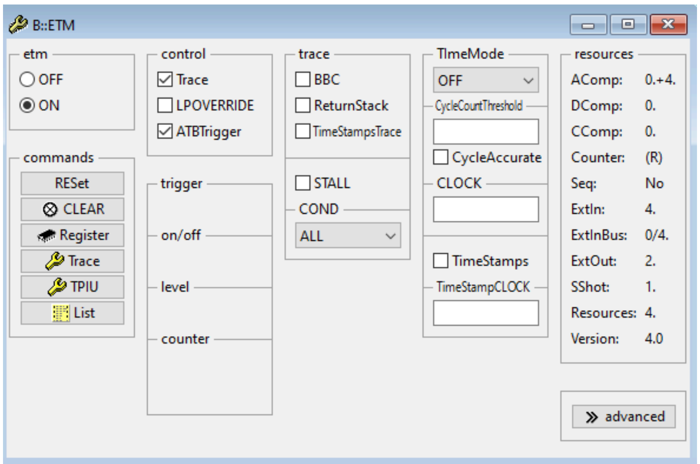

Figure 18. Setting window for ETM

In the "COND" menu, you can set if the ETM emits the information about the execution of the following conditional non-branch instructions:

- **OFF:** The conditional instruction tracing is disabled.
- **Loads:** The conditional load instructions are traced.
- **Stores:** The conditional store instructions are traced.
- **LoadsAndStores:** The conditional load and store instructions are traced.
- **ALL:** All conditional instructions are traced.

In the "TimeMode" menu, you can set the source of the timestamp. You can see the order of functions and also the duration of each function.

Figure 19. Function duration using timestamping in ETM

By clicking the "List" option, you can inspect the order of instructions. By clicking the "Config…" button, you can add or remove the columns that have additional information.

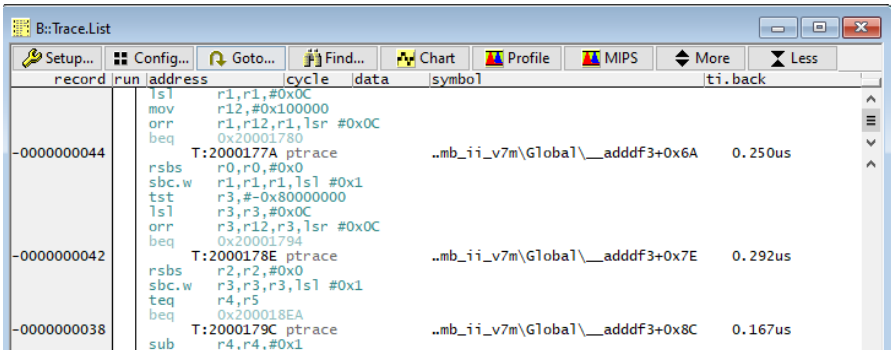

Figure 20. Trace list for ETM

ITM and DWT:

Through ITM, you can send the debug `printf`-like messages. The ITM also encapsulates the DWT data and it is controlled through the ITM window. The ITM provides data and address comparators and also a PC sampler. The DWT can emit the PC value at specific intervals, which can be set in the "PCSampler" drop-down menu.

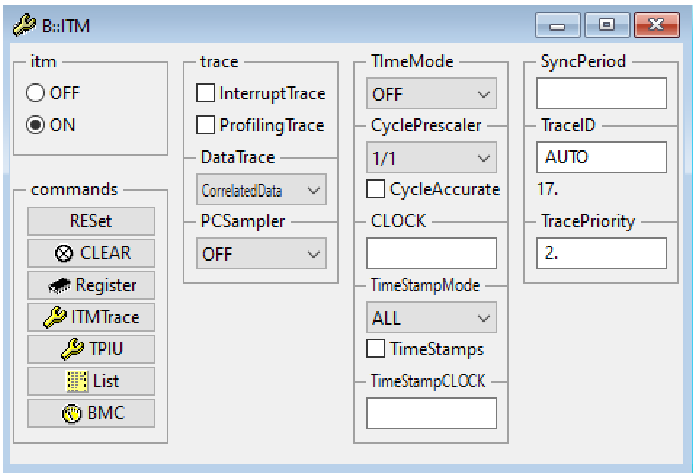

Figure 21. Setting window for ITM

In the following example, the DWT comparator is set to generate trace only if a specific address is accessed. In the "DataTrace" drop-down list, select the "Data" option. Then go to "Break->Set" and write the name of the expression or its address to the "address/expression" field. It is also possible to select the type of access, so the trace data can be emitted, for example, only if it is written to the location. The "action" field is set to the "TraceData" option.

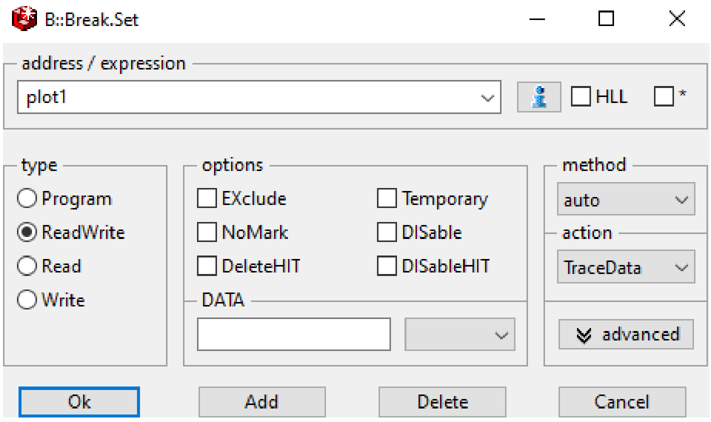

Figure 22. Settings for data access tracing

Figure 23 is the result of this tracing and it shows accesses to the memory location of the `plot1` variable. The "cycle" column shows information about the type of access. The "data" column shows the value of the data that were manipulated.

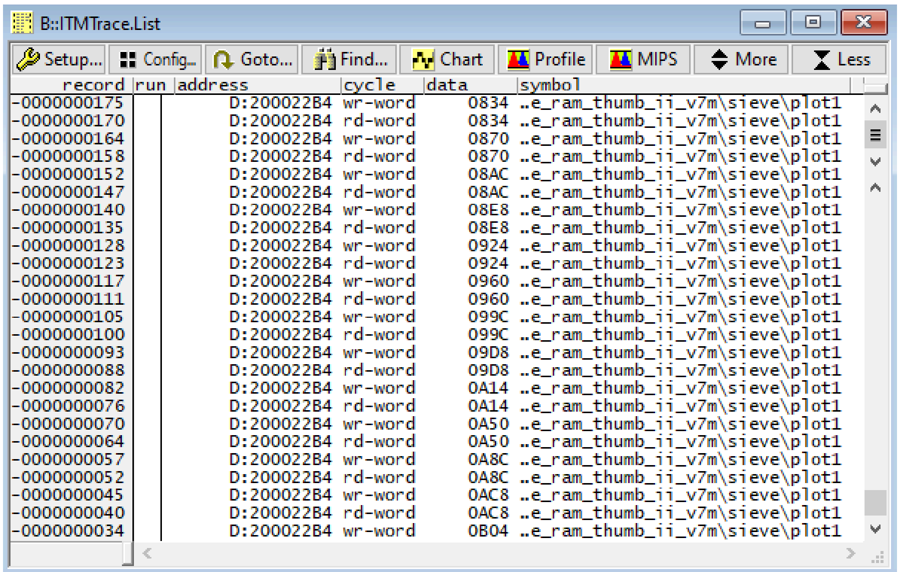

Figure 23. Trace information about accesses to specific memory location

It is also useful to emit other additional information on the comparator match. This can be done in the "ITM setting" window, in the "DataTrace" drop-down menu, by selecting the "DataPC" option. There are more options for what should happen on the address comparator match (see Table 1).

Table 1. Methods for DataTrace

Method	Description
OFF	No information on accesses to data addresses that match a DTW comparator are emitted.
ON	If data address matches a DWT comparator: address and data value information on the data access are emitted by the ITM.
Address	If data address matches a DWT comparator: address information on the data access is emitted by the ITM.
Data	If data address matches a DWT comparator: data value information on the data access is emitted by the ITM.

Table 1. Methods for DataTrace*...continued*

Method	Description
DataPC	If data address matches a DWT comparator: address and data value information on the data access are emitted by the ITM. Additionally the address of the instruction that performed the data access is emitted.
OnlyPC	If data address matches a DWT comparator: address of the instruction that performed the data access is emitted by the ITM.
CorrelatedData	Emits the same information as DataPC, but if the command Trace.List is used, the information on the data access is merged into the ETM instruction flow display.

The useful feature is "CorrelatedData..". This allows to use the DWT and ETM in parallel. The DWT then emits information about the "PC" value for data access. The information about this data access is merged with information about the program flow from ETM.

5 Note about the source code in the document

Example code shown in this document has the following copyright and BSD-3-Clause license:

Copyright 2024 NXP Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials must be provided with the distribution.
3. Neither the name of the copyright holder nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

6 Revision history

Table 2 summarizes the revisions to this document.

Table 2. Revision history

Document ID	Release date	Description
AN14211 v.1.0	10 December 2024	• Initial public release

Legal information

Definitions

Draft — A draft status on a document indicates that the content is still under internal review and subject to formal approval, which may result in modifications or additions. NXP Semiconductors does not give any representations or warranties as to the accuracy or completeness of information included in a draft version of a document and shall have no liability for the consequences of use of such information.

Disclaimers

Limited warranty and liability — Information in this document is believed to be accurate and reliable. However, NXP Semiconductors does not give any representations or warranties, expressed or implied, as to the accuracy or completeness of such information and shall have no liability for the consequences of use of such information. NXP Semiconductors takes no responsibility for the content in this document if provided by an information source outside of NXP Semiconductors.

In no event shall NXP Semiconductors be liable for any indirect, incidental, punitive, special or consequential damages (including - without limitation - lost profits, lost savings, business interruption, costs related to the removal or replacement of any products or rework charges) whether or not such damages are based on tort (including negligence), warranty, breach of contract or any other legal theory.

Notwithstanding any damages that customer might incur for any reason whatsoever, NXP Semiconductors' aggregate and cumulative liability towards customer for the products described herein shall be limited in accordance with the Terms and conditions of commercial sale of NXP Semiconductors.

Right to make changes — NXP Semiconductors reserves the right to make changes to information published in this document, including without limitation specifications and product descriptions, at any time and without notice. This document supersedes and replaces all information supplied prior to the publication hereof.

Suitability for use — NXP Semiconductors products are not designed, authorized or warranted to be suitable for use in life support, life-critical or safety-critical systems or equipment, nor in applications where failure or malfunction of an NXP Semiconductors product can reasonably be expected to result in personal injury, death or severe property or environmental damage. NXP Semiconductors and its suppliers accept no liability for inclusion and/or use of NXP Semiconductors products in such equipment or applications and therefore such inclusion and/or use is at the customer's own risk.

Applications — Applications that are described herein for any of these products are for illustrative purposes only. NXP Semiconductors makes no representation or warranty that such applications will be suitable for the specified use without further testing or modification.

Customers are responsible for the design and operation of their applications and products using NXP Semiconductors products, and NXP Semiconductors accepts no liability for any assistance with applications or customer product design. It is customer's sole responsibility to determine whether the NXP Semiconductors product is suitable and fit for the customer's applications and products planned, as well as for the planned application and use of customer's third party customer(s). Customers should provide appropriate design and operating safeguards to minimize the risks associated with their applications and products.

NXP Semiconductors does not accept any liability related to any default, damage, costs or problem which is based on any weakness or default in the customer's applications or products, or the application or use by customer's third party customer(s). Customer is responsible for doing all necessary testing for the customer's applications and products using NXP Semiconductors products in order to avoid a default of the applications and the products or of the application or use by customer's third party customer(s). NXP does not accept any liability in this respect.

Terms and conditions of commercial sale — NXP Semiconductors products are sold subject to the general terms and conditions of commercial sale, as published at https://www.nxp.com/profile/terms, unless otherwise agreed in a valid written individual agreement. In case an individual agreement is concluded only the terms and conditions of the respective agreement shall apply. NXP Semiconductors hereby expressly objects to applying the customer's general terms and conditions with regard to the purchase of NXP Semiconductors products by customer.

Export control — This document as well as the item(s) described herein may be subject to export control regulations. Export might require a prior authorization from competent authorities.

Suitability for use in non-automotive qualified products — Unless this document expressly states that this specific NXP Semiconductors product is automotive qualified, the product is not suitable for automotive use. It is neither qualified nor tested in accordance with automotive testing or application requirements. NXP Semiconductors accepts no liability for inclusion and/or use of non-automotive qualified products in automotive equipment or applications.

In the event that customer uses the product for design-in and use in automotive applications to automotive specifications and standards, customer (a) shall use the product without NXP Semiconductors' warranty of the product for such automotive applications, use and specifications, and (b) whenever customer uses the product for automotive applications beyond NXP Semiconductors' specifications such use shall be solely at customer's own risk, and (c) customer fully indemnifies NXP Semiconductors for any liability, damages or failed product claims resulting from customer design and use of the product for automotive applications beyond NXP Semiconductors' standard warranty and NXP Semiconductors' product specifications.

HTML publications — An HTML version, if available, of this document is provided as a courtesy. Definitive information is contained in the applicable document in PDF format. If there is a discrepancy between the HTML document and the PDF document, the PDF document has priority.

Translations — A non-English (translated) version of a document, including the legal information in that document, is for reference only. The English version shall prevail in case of any discrepancy between the translated and English versions.

Security — Customer understands that all NXP products may be subject to unidentified vulnerabilities or may support established security standards or specifications with known limitations. Customer is responsible for the design and operation of its applications and products throughout their lifecycles to reduce the effect of these vulnerabilities on customer's applications and products. Customer's responsibility also extends to other open and/or proprietary technologies supported by NXP products for use in customer's applications. NXP accepts no liability for any vulnerability. Customer should regularly check security updates from NXP and follow up appropriately.

Customer shall select products with security features that best meet rules, regulations, and standards of the intended application and make the ultimate design decisions regarding its products and is solely responsible for compliance with all legal, regulatory, and security related requirements concerning its products, regardless of any information or support that may be provided by NXP.

NXP has a Product Security Incident Response Team (PSIRT) (reachable at PSIRT@nxp.com) that manages the investigation, reporting, and solution release to security vulnerabilities of NXP products.

NXP B.V. — NXP B.V. is not an operating company and it does not distribute or sell products.

Trademarks

Notice: All referenced brands, product names, service names, and trademarks are the property of their respective owners.

NXP — wordmark and logo are trademarks of NXP B.V.

IAR — is a trademark of IAR Systems AB.

J-Link — is a trademark of SEGGER Microcontroller GmbH.

Microsoft, Azure, and ThreadX — are trademarks of the Microsoft group of companies.

Contents