Performing ETM and ITM trace on PSOC™ Control C3 and PSOC™ Edge E8 MCUs

About this document

Scope and purpose

This application note introduces the trace features of the PSOC™ Control C3 and PSOC™ Edge E8 MCUs, helping you get started with performing instruction (ETM) and instrumentation (ITM) tracing on these devices. This application note also guides you to explore more features of trace and other resources available online to accelerate your learning. If you are new to the ModusToolbox™ software environment and these devices, see:

- AN238329 – Getting started with PSOC™ Control C3 MCU on ModusToolbox™ software
- AN235935 – Getting started with PSOC™ Edge E8 MCU on ModusToolbox™ software

Intended audience

This application note is intended for advanced engineers who want to use the trace capabilities of the PSOC™ Control C3 and PSOC™ Edge E8 MCUs.

Acronyms

Acromyn	Definition
DWT	Data Watchpoint and Trace
ETB	Embedded Trace Buffer
ETF	Embedded Trace FIFO
ETM	Embedded Trace Macrocell
ITM	Instrumentation Trace Macrocell
RTT	Real-Time Transfer
SoC	System on Chip
SWJ	Serial Wire / JTAG
SWO	Single Wire Output
SWV	Serial Wire Viewer
TMC	Trace Memory Controller
TPIU	Trace Port Interface Unit

Table of contents

Table of contents

About this document **1**
Table of contents **2**
1 Introduction **3**
1.1 Debug/trace features 3
1.2 What are ETM and ITM tracing? 4
1.3 Benefits of ETM/ITM Tracing compared to breakpoints/logging 4
1.4 Key differences 5
1.5 Typical use cases 5
1.6 Additional information 5
1.6.1 No SWO pin 5
1.6.2 Probe capability 5
1.6.3 Trace sinks 6
1.6.3.1 Off-chip capture: TPIU (Parallel Trace Output) 6
1.6.3.2 On-chip capture: Trace Memory Controller (TMC) 7
1.6.3.3 Comparison: TPIU (Parallel Trace Port) vs TMC/ETB 7
2 Getting started **8**
2.1 Software requirements 8
2.2 Hardware requirements 8
2.3 General limitations and constraints 9
2.4 Keil µVision limitations and constraints 9
2.4.1 ULINKpro 9
2.5 IAR Embedded Workbench limitations and constraints 10
2.5.1 I-jet Trace 10
3 Performing trace on PSOC™ Control C3 and PSOC™ Edge E8 MCU **11**
3.1 Keil µVision 11
3.1.1 Configure the debugger script and debugger 11
3.1.2 Configure the debugger script with ULINKpro Cortex® 11
3.1.2.1 Configure trace via TPIU 14
3.1.2.2 Configure trace via TMC/ETB 15
3.1.3 Perform ETM trace 16
3.1.4 Perform ITM trace (printf-style debugging) 17
3.2 IAR Embedded Workbench for ARM 19
3.2.1 Configure the debugger script and debugger 19
3.2.2 Perform ETM trace 21
3.2.3 Perform ITM trace (printf-style debugging) 23
4 Troubleshooting **24**
4.1 Keil µVision 24
4.1.1 Recommended fixes 24
5 References **26**
Revision history **27**
Disclaimer **28**

1 Introduction

This application note discusses the PSOC™ Control C3 and PSOC™ Edge E8 MCU trace architecture and the associated development tools with new features and limitations. It also covers various general aspects of Arm® trace architecture to help you understand and perform trace on PSOC™ Control C3 and PSOC™ Edge E8 MCUs to its full potential.

Although this application note focuses on demonstrating the ETM and ITM trace, third-party tools generally offer a wide range of additional trace features, including function profiling, data watchpoints, interrupt logging, and code coverage. Refer to the respective third-party tool documentation to understand all the features they offer and learn how to use them effectively for complex embedded application debugging and development.

1.1 Debug/trace features

Arm® CoreSight is a suite of tools for debugging and tracing software on Arm®-based devices. Debug capabilities let you observe or modify the state of system components, while trace capabilities continuously collect system information for later offline analysis.

Debug features include:

- Run Control of the processor allows you to start and stop programs.
- Single Step one source or assembler line.
- Set breakpoints while the processor is running.
- Read/write memory contents and peripheral registers on-the-fly.

Trace features include:

- Embedded Trace Macrocell (ETM)
- Instruction Trace Macrocell (ITM)
- Data Watchpoint and Trace (DWT)

ARM® CoreSight tracing is the architecture-level mechanism that streams execution and event information from the CPU in real time for debugging and analysis. Because it’s built into the core (ETM/DWT/ITM with ETB/TPIU via SWO/Parallel Trace Output), it works without halting the target and without altering program behavior.

The CoreSight features are available via JTAG and Serial Wire Debug interfaces.

Introduction

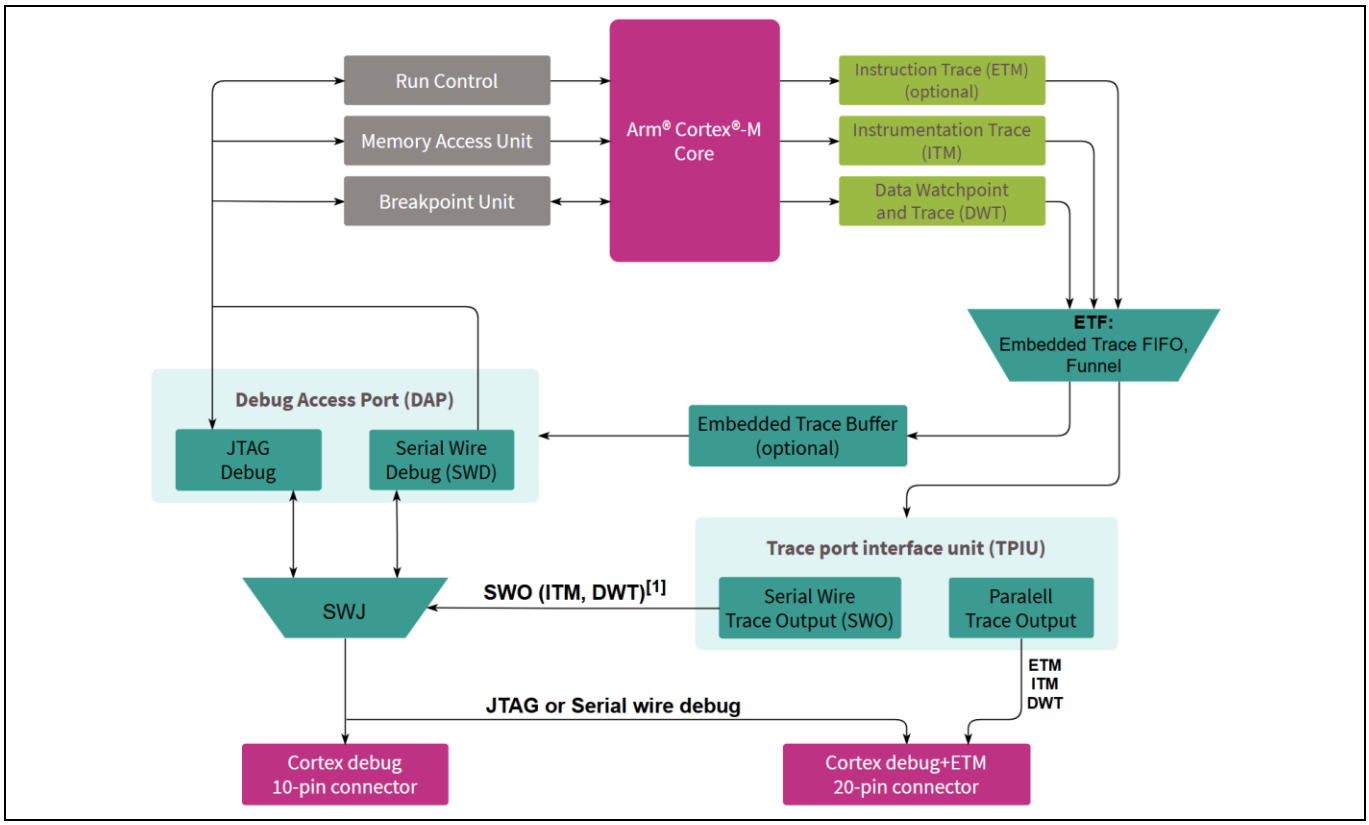


Figure 1 ARM® CoreSight Technology (Debug and Trace Support)

1.2 What are ETM and ITM tracing?

- **ETM (Embedded Trace Macrocell)** is one of the tracing types, that provides hardware program-flow trace. It records the actual instruction stream executed by the CPU (branches, exceptions, returns), allowing the debugger to reconstruct “where the core went” precisely, even across interrupts and context switches. ETM is ideal for runtime analysis, timing, software profiling, and code coverage.
- **ITM (Instrumentation Trace Macrocell)** is another of the tracing type, that provides software-generated trace. Your firmware writes small messages/events into ITM stimulus ports (0–31). These are timestamped and transported in the CoreSight trace stream, enabling non-intrusive printf, event logging, RTOS awareness, counters, and unit-test annotations. ITM also multiplexes DWT events (PC sampling, cycle counters, exceptions).

1.3 Benefits of ETM/ITM Tracing compared to breakpoints/logging

- **Non-intrusive:** trace runs continuously while the system executes at full speed — no instrumented UART, no blocking I/O, no periodic polling.
- **Deterministic:** ETM reconstructs the exact path, essential for intermittent faults, race conditions, and exception analysis.
- **Rich timing:** DWT (Data Watchpoint & Trace Unit) counters and timestamps correlate events; ITM can annotate domain-specific data in-line.
- **Profiling and coverage:** ETM enables function/line coverage and hot-path detection without recompiling with heavy instrumentation.

[1] The devices with Arm® CoreSight SoC-600 don’t support SWO pin.

1.4 Key differences

- **ETM** is hardware program-flow trace — high bandwidth, requires Parallel Trace Output via 20-pin Cortex Debug+ETM connector or ETB. Offers code coverage and detailed execution history.
- **ITM** is software trace over stimulus ports — lighter weight, designed for application messages. On PSOC™ Control C3 and PSOC™ Edge E8 MCU devices, ITM must go over the Parallel Trace Port (or ETB).
- **DWT** is hardware events and sampling — can be included in the ITM/TPIU stream for timing and counters.

1.5 Typical use cases

Usually, ETM is used when you need exact instruction-level history and timing. ITM is used when you need lightweight, application-driven messages, timestamped annotations from firmware-messages, state changes, and counters—merged with DWT events in the same trace stream.

The items below illustrate common scenarios.

- **ETM**: crash reconstruction, timing across interrupts/tasks, function execution profiling, code coverage certification.
- **ITM:** printf-style logging without UART, RTOS task switching visibility, counters, unit test progress markers, app-specific telemetry. Use ITM to emit diagnostic trace data and debug messages.

1.6 Additional information

For deeper architectural flow and background, review Infineon AN235279 (Performing ETM and ITM trace on PSOC™ 6 MCU), where the same CoreSight principles apply to PSOC™ Control C3 and PSOC™ Edge E8. However, PSOC™ Control C3 and PSOC™ Edge E8 MCU have some differences from the PSOC™ 6 MCU that you need to keep in mind.

To learn more about the specific device you are working with, we recommend studying the corresponding Technical Reference Manual (TRM) to understand which CoreSight components are available:

- For PSOC™ Control C3 MCUs – 002-38337: PSoC™ Control C3 MCU Architecture Technical Reference Manual
- For PSOC™ Edge E8 MCUs – 002-38931: PSoC™ Edge E84 MCU Family Architecture Technical Reference Manual

For example, PSOC™ Control C3 supports trace output only via the TPIU (parallel trace port), while PSOC™ Edge E8 allows trace capture through both the TPIU and the on-chip ETB (via TMC) for its Cortex®-M33 and Cortex®-M55 cores.

1.6.1 No SWO pin

The devices with Arm® CoreSight SoC-600 like PSOC™ Control C3 and PSOC™ Edge E8 MCUs do not have the **SWO pin**. Therefore, ITM must be routed via the **Parallel Trace Output** or **ETB**. Any “SWV console” expecting SWO will not work.

1.6.2 Probe capability

ETM trace data can be output via the **Parallel Trace Output** (TPIU) or captured on-chip using the **ETB (via TMC)**. Both paths require a compatible probe:

- For the **Parallel Trace Output**, a trace-capable probe with a Cortex Debug + ETM connector is required, such as ULINKpro or I-jet Trace.

Introduction

- For **ETB**, a probe that supports on-chip trace buffer readout is required, such as J-Link Plus (V12 or later), ULINKpro, or I-jet Trace.

This rule also applies to **ITM**, because basic probes (ULINK2 and I-jet) only have the capability to output ITM + DWT via SWO pin, **which has been removed**, as mentioned earlier.

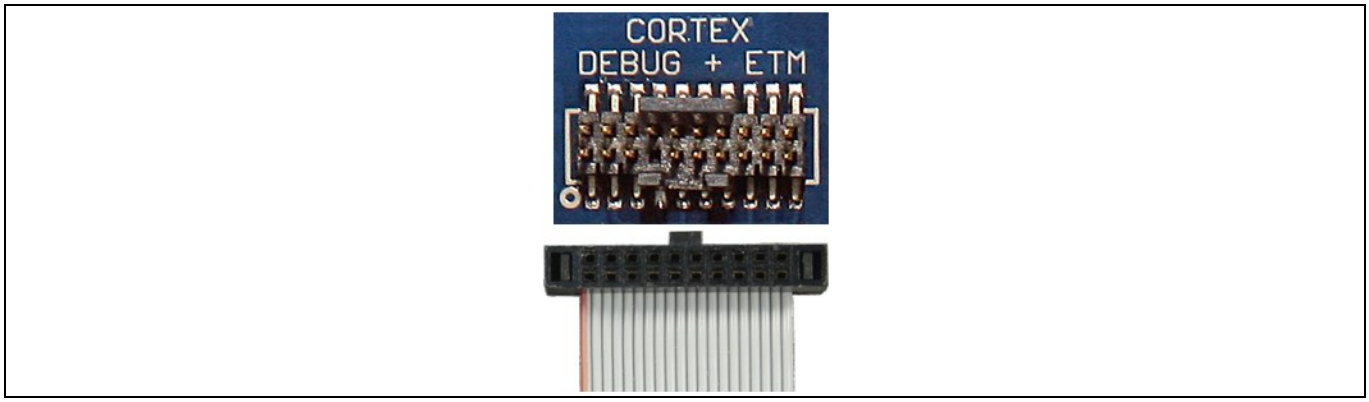


Figure 2 ARM® Cortex® Debug+ETM 20-pin connector

1.6.3 Trace sinks

After trace data is generated by ETM, ITM, and DWT, it must be routed to a **trace sink** — a destination that either outputs or stores the trace stream. According to Arm® CoreSight™ architecture, trace sinks are divided into two categories:

- **Off-chip capture** – routes trace data outside the MCU to an external debug probe in real time via physical pins.
- **On-chip capture** – stores trace data inside the MCU in a dedicated buffer for later retrieval via the debug interface.

1.6.3.1 Off-chip capture: TPIU (Parallel Trace Output)

The **Trace Port Interface Unit (TPIU)** formats and serializes the trace stream from the on-chip trace bus and outputs it via the **Parallel Trace Output** (TRACECLK + TRACEDATA[0..3]) through the 20-pin Cortex Debug+ETM connector.

To use the TPIU, the following must be configured:

- **Trace pins** (TRACECLK, TRACEDATA[0..3]) must be assigned their CoreSight function via the device pinmux/HSIOM.
- **Trace clock (TRACECLK)** must be set to a frequency supported by both the MCU and the probe.
- **TPIU formatter** must be enabled and configured for parallel trace mode (not SWO/Manchester).
- A **trace-capable probe** with a 20-pin connector (ULINKpro, I-jet Trace) and a compatible ribbon cable are required.

1.6.3.2 On-chip capture: Trace Memory Controller (TMC)

The **Trace Memory Controller (TMC)** is a CoreSight component that captures and buffers trace data on chip. According to Arm® documentation, the TMC can be configured in one of four modes depending on the target device design:

- **ETB (Embedded Trace Buffer)** – on-chip SRAM circular buffer. Stores trace data until full, then overwrites the oldest data. Simple and widely supported. Typical size: 4 KB to 64 KB.
- **ETF (Embedded Trace FIFO)** – on-chip SRAM that can operate as a circular buffer, hardware FIFO, or software FIFO. More flexible than ETB; often used to smooth trace bursts before forwarding to TPIU.
- **ETR (Embedded Trace Router)** – routes trace over AXI to system memory (DDR/SRAM), allowing much larger capture than ETB/ETF developer.
- **ETS (Embedded Trace Streamer)** – streams trace over AXI4-Stream to a high-speed link (e.g., HSSTP). No local storage; designed for continuous high-bandwidth output.

Note: *For more information about the Trace Memory Controller and its configuration modes, refer to the Arm® CoreSight™ TMC Technical Reference Manual: https://developer.arm.com/documentation/ddi0461/b?lang=en*

1.6.3.3 Comparison: TPIU (Parallel Trace Port) vs TMC/ETB

Table 1 TPIU (4-Pin) vs TMC/ETB

Parameter	TPIU (off-chip)	TMC/ETB (on-chip)
Probe required [2]	ULINKpro, I-jet Trace	J-Link Plus/Ultra, ULINKpro, I-jet/I-jet Trace
Hardware required	20-pin Cortex Debug+ETM connector + ribbon cable	Standard 10-pin SWJ connector, also accessible via 20-pin connector
ETM support	Yes	Yes
ITM/DWT support	Yes	Yes
Live trace in IDE	Yes	No (post-capture readout)
Capture type	Real-time streaming	Snapshot (circular buffer)
Output path	TRACECLK + TRACEDATA[0..3] via 20-pin connector	On-chip SRAM; read back via SWD or JTAG
Trace depth	Unlimited (continuous at TRACECLK rate)	Limited by SRAM size (typically 4–64 KB); wraps oldest data
Pin configuration	Required (TRACECLK, TRACEDATA[0..3])	Not required
Signal integrity risk	Sensitive to cable length, skew, and setup/hold timing	None (on-chip path)
Best for	Code coverage, profiling, long-running sessions	Quick capture without trace connector

[2] For detailed specifications and supported configurations, refer to the official documentation provided by the IDE vendor.

2 Getting started

Create a ModusToolbox™ application for a PSOC™ Control C3 or PSOC™ Edge E8 device that will be used with either IAR Embedded Workbench or Keil µVision. This application note will use a simple Empty_App application for demonstration purposes. For more information on creating applications targeting third-party IDEs, see:

- Keil µVision (MDK-ARM) – Keil µVision for ModusToolbox™ user guide
- IAR Embedded Workbench for ARM (EWARM) – IAR Embedded Workbench for ModusToolbox™ user guide

2.1 Software requirements

This application note uses the following software tools:

- ModusToolbox™ tools package: ModusToolbox™ tools package v3.6 or later is required to create or import a project for PSOC™ Control C3 and PSOC™ Edge E8 MCUs for use with IAR Embedded Workbench or Keil µVision.
- IAR Embedded Workbench for Arm®: IAR Embedded Workbench (tested with v9.70.1) and I-jet Trace can be used to perform tracing on PSOC™ Control C3 and PSOC™ Edge E8 MCUs.
- Keil µ Vision: Keil µVision (tested with v5.43) and ULINKpro can be used to perform tracing on PSOC™ Control C3 and PSOC™ Edge E8 MCUs.

2.2 Hardware requirements

Trace probes (external debugger)

This application note uses the following trace probes:

- **I-jet Trace for Arm® Cortex®-M:** This probe is used with IAR Embedded Workbench software tools. For more information, see I-jet Trace for Arm® Cortex® -M.
- **ULINKpro debug and trace unit:** This probe is used with Keil µVision software tools. For more information, see ULINKpro.

Development board

This application note uses the following kits:

- For PSOC™ Control C3 MCUs – PSOC™ Control C3M5 Evaluation Kit;
- For PSOC™ Edge E8 MCUs – PSOC™ Edge E84 Evaluation Kit

Typically, on evaluation kits, the trace pins are multiplexed with other peripherals due to limited I/Os.

For instance, for PSC3M5-EVK, open the KIT-PSC3M5-EVK schematics file and search for the term "trace". You will find a box with trace pins, as shown in Figure 3.

Getting started

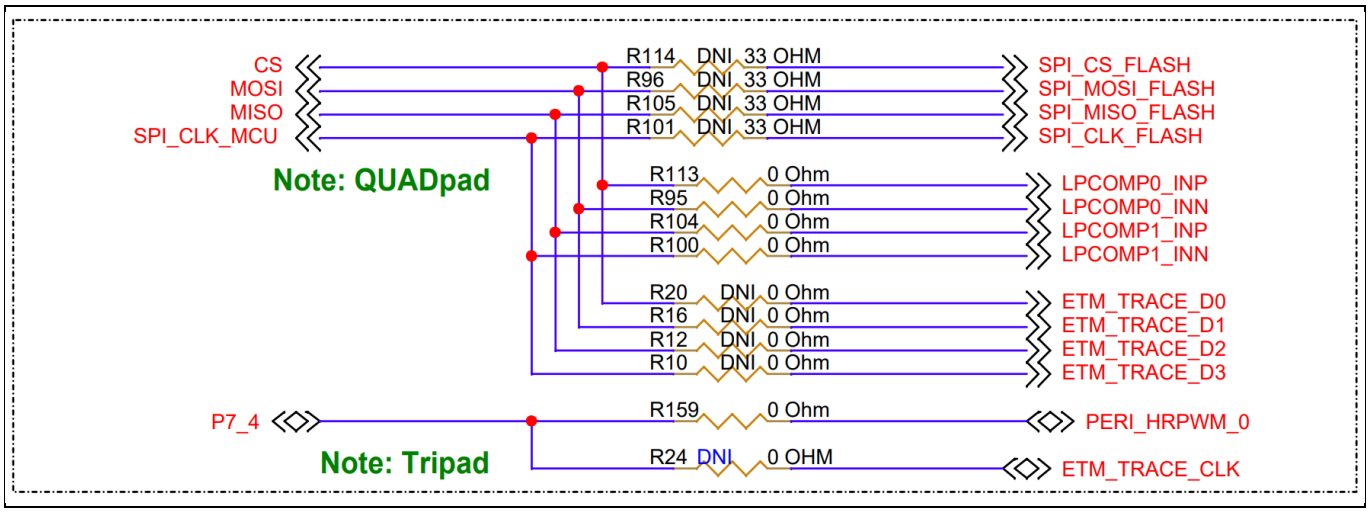


Figure 3 KIT-PSC3M5-EVK kit trace pin schematics

To expose the trace pins to the J14 ETM header, do the following:

1. Unload the resistances R113, R95, R104, R100, and R159.
2. Load 0 Ω resistances to R20, R16, R12, R12, and R24.
3. Solder the ETM header (20-pin FRC male connector) in place.

2.3 General limitations and constraints

MCU security and access:

TrustZone/security policies can block trace/debug until unlocked (device option bytes, debug authentication). If trace stays empty, verify the device is in an unlocked debug state.

Note: For details about PSOC™ Control C3 and PSOC™ Edge E8 MCU security, refer to Application Notes:

- *AN240106 – Getting started with PSOC™ Control C3 security*
- *AN237849 – Getting started with PSOC™ Edge security*

2.4 Keil µVision limitations and constraints

2.4.1 ULINKpro

Debug interface:

- **Issue:** JTAG is **not supported** by ULINKpro.
- **Impact:**
  - JTAG connection and JTAG-chain workflows (boundary-scan, multi-device daisy chain) are not available.
  - Debug and trace must use SWD with the parallel trace port; any setup expecting JTAG will fail.
- **Resolution/Status:** N/A.
- **Recommended action:** Use SWD as the only supported interface.

Getting started

Tracing of Cortex-M55 (for PSOC™ Edge E8 MCU):

- **Issue:** ETM, DWT, and ITM are not supported for Cortex-M55 on **Keil µVision v5.43.0.**
- **Impact:**
  - Instruction trace (ETM), hardware events/timing (DWT), and ITM messages from Cortex-M55 cannot be captured/decoded.
  - Features relying on these (code coverage, profiling, timelines, ITM consoles) are unavailable on Cortex-M55 for PSOC Edge E8 with v5.43.0.
- **Resolution/Status:** N/A. Monitor Keil µVision release notes for future support changes.
- **Recommended action:** N/A.

2.5 IAR Embedded Workbench limitations and constraints

2.5.1 I-jet Trace

To use the I-jet Trace probe from IAR, it is recommended to use IAR EWARM version 9.60.3 or later, as this version provides full trace support for the PSOC™ Control C3 and PSOC™ Edge E8 MCU.

ITM Tracing:

- **Issue:** You cannot enable ITM via ETB or the Parallel Trace output in **IAR Embedded Workbench through version 9.70.1.**
- **Impact:**
  - ITM output cannot be routed via ETB or the Parallel Trace Port; SWO is not available on these MCUs, so ITM logging is effectively not possible in the affected IAR versions.
  - Any workflows depending on ITM over ETB or the Parallel Trace Port (runtime logs, profiling, visualization) will not function.
- **Resolution/Status:** N/A. Monitor IAR release notes for future support changes.
- **Recommended action:** Use alternative channels (UART, semihosting, RTT).

Performing trace on PSOC™ Control C3 and PSOC™ Edge E8 MCU

3 Performing trace on PSOC™ Control C3 and PSOC™ Edge E8 MCU

ETM and ITM are supported on the CM33 core for PSOC™ Control C3 MCUs and the CM33/CM55 for PSOC™ Edge E8 MCUs.

3.1 Keil µVision

3.1.1 Configure the debugger script and debugger

The debugger configures the TPIU port and necessary clocks before starting the trace. The CMSIS CAT1B and PSE8XXX packs have the necessary files to perform this configuration. The DBGCONF files in the DebugConfig folder, located under the project directory, contain the TPIU port selection information and clock trace clock prescaler option.

3.1.2 Configure the debugger script with ULINKpro Cortex®

To override the default values in the DBGCONF file, do the following:

1. In Keil µVision, select **File > Open**.
2. Navigate to the DebugConfig folder in the Empty App project directory, select the DBGCONF file, and click **Open**.

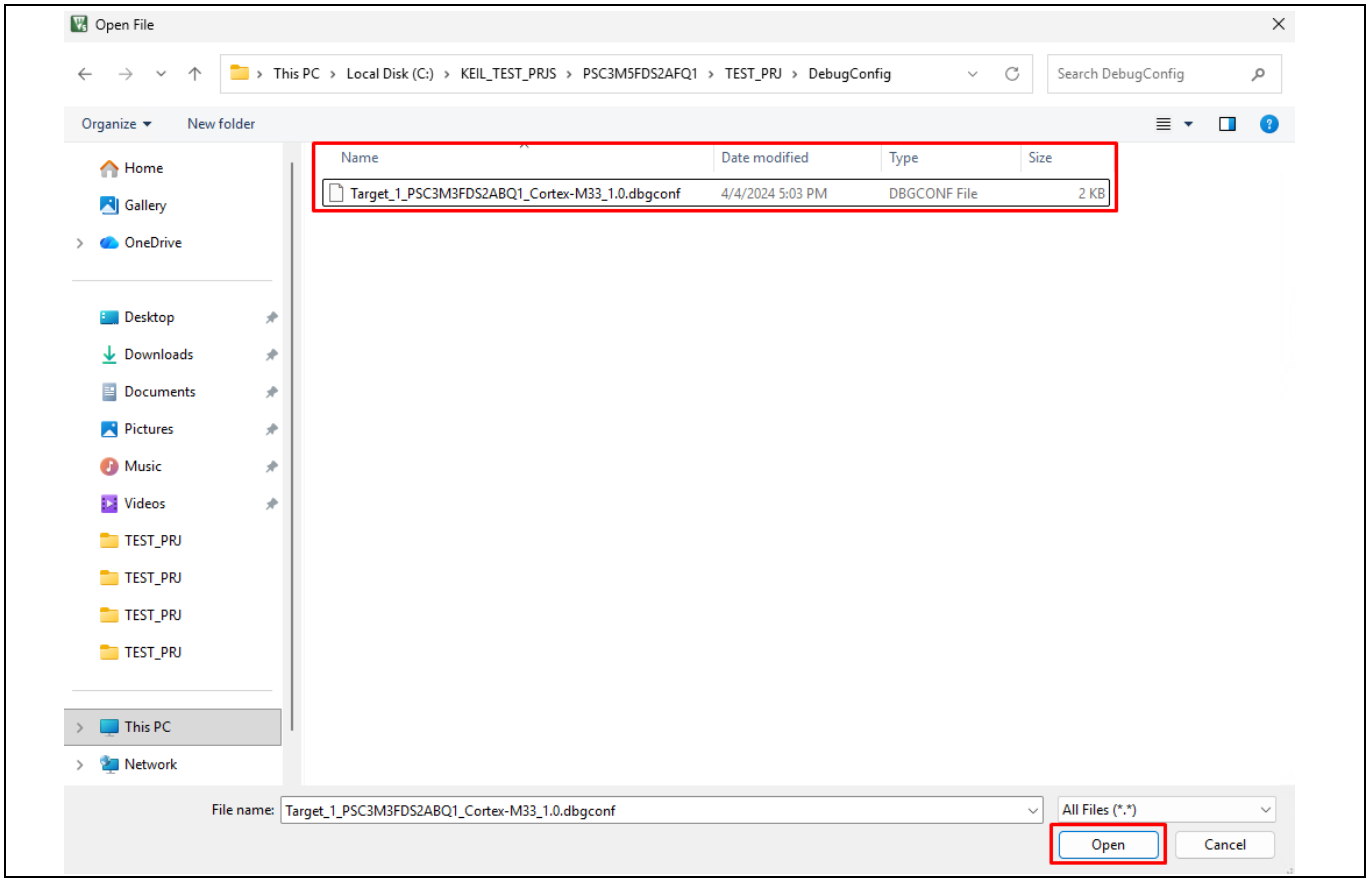


Figure 4 DBGCONF file for PSOC™ Control C3 project.

Performing trace on PSOC™ Control C3 and PSOC™ Edge E8 MCU

3. In the editor that opens with the file content, select **Configuration Wizard** at the bottom of the editor.

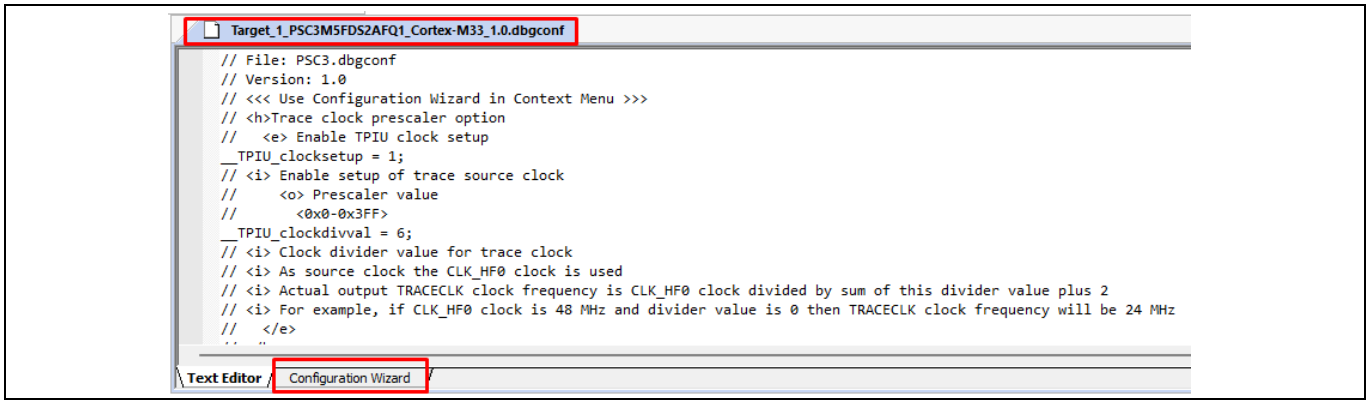


Figure 5 Opening the DBGCONF file in Configuration Wizard

4. Select **Trace Pin Setup > TPIU Pin Location > Pin Location 1** from the drop-down list. Details about the drop-down options are provided at the end of the editor.

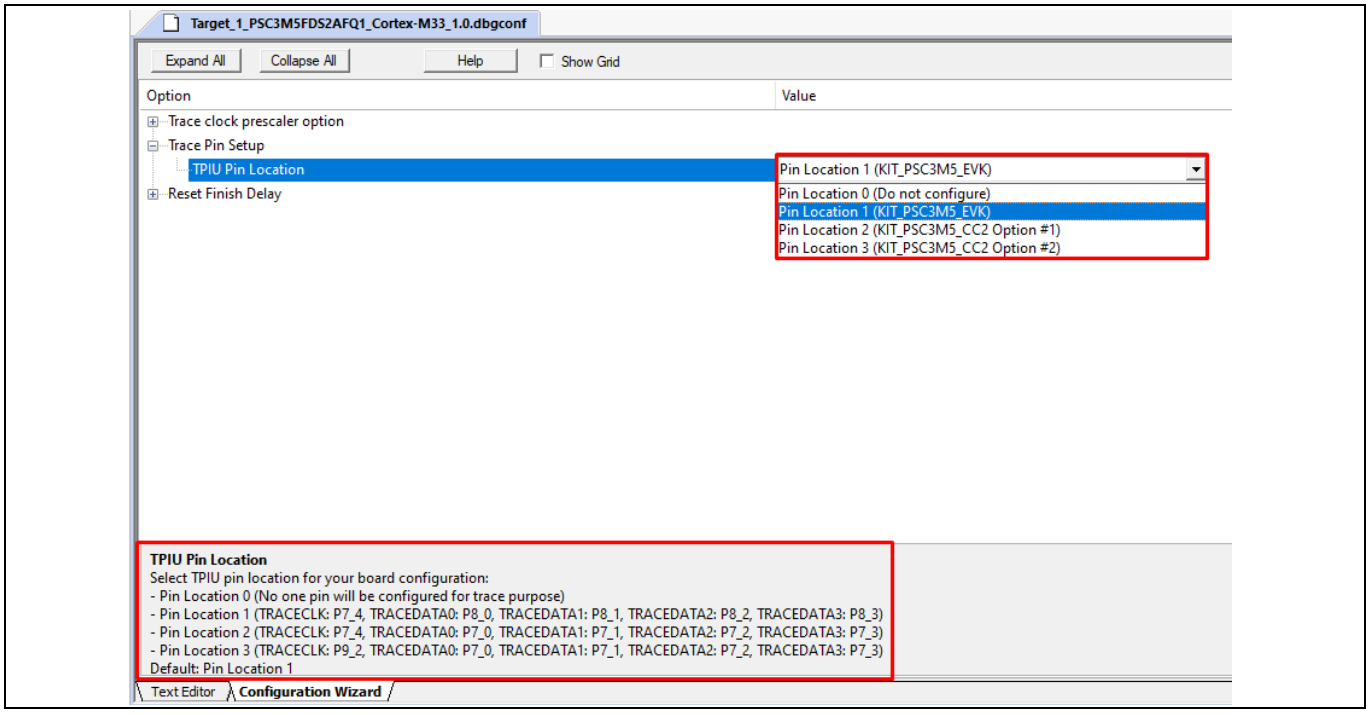


Figure 6 Select the TPIU Pin Location

Performing trace on PSOC™ Control C3 and PSOC™ Edge E8 MCU

5. Enable the **TPIU clock setup** using the check box and set a **prescaler value** if you do not want to use the default. Once the setup is complete, save and close the file.

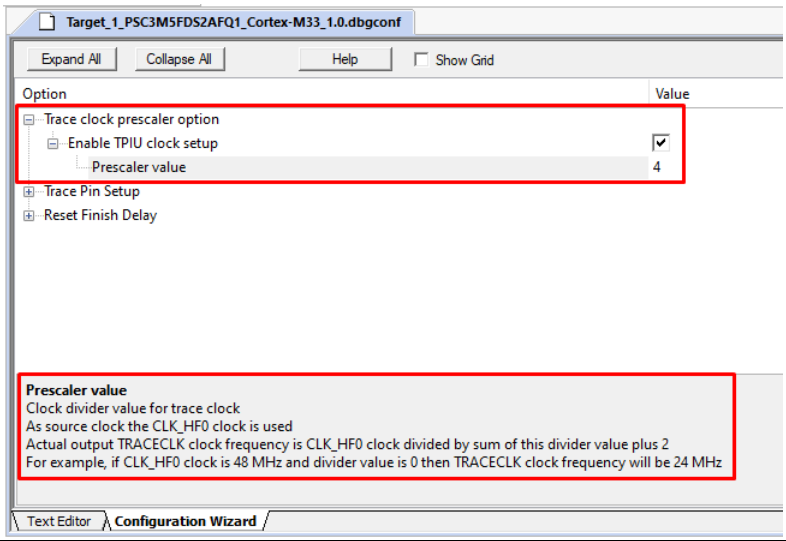


Figure 7 Enable the TPIU clock with prescaler value

Note: The CoreSight trace components (ETM, ITM, DWT, TPIU, TMC) are all driven by a shared internal trace clock derived from the system clock. This internal clock is configured via the trace clock prescaler.

When using off-chip capture (TPIU), this internal clock is also output on the TRACECLK pin to synchronize the external probe. However, for on-chip capture (ETB), the external TRACECLK pin is not required, as the components rely solely on the internal clock.

To select and configure the debugger in µVision, do the following:

1. Open the project options.
2. Select the **Debug** tab and choose **ULINK Pro ARMv8-M Debugger** from the drop-down menu.
3. Click the **Settings** button next to it.

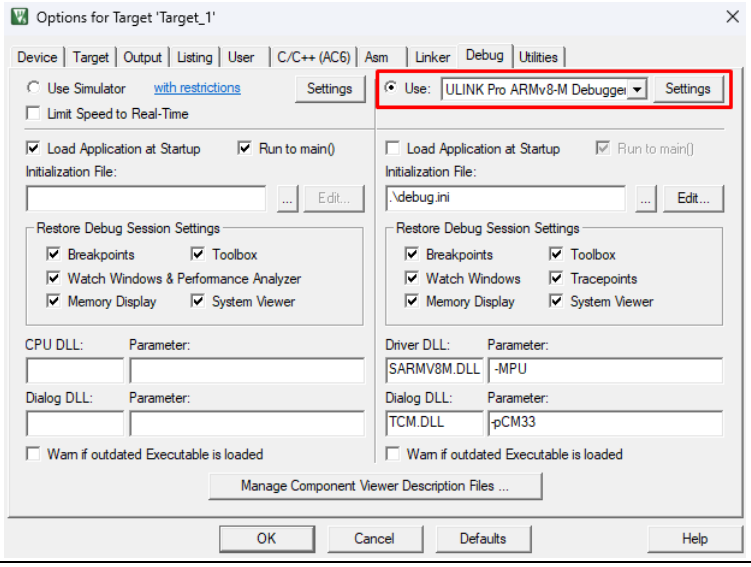


Figure 8 Configurations in Debug tab

4. In the new window, configure the settings for the **Debug** tab to match Figure 9.
   - **Serial No.**: Any
   - **Port**: SW
   - **Max Clock**: 1MHz
   - **Connect**: Normal
   - **Reset**: SYSRESETREQ
   - **Reset after Connect**: selected

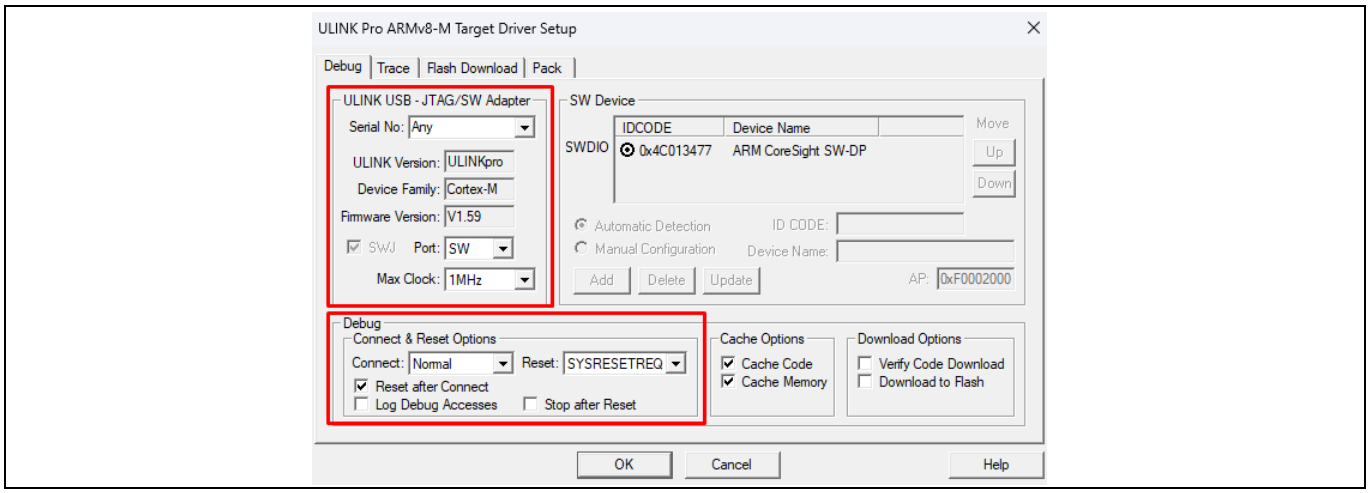


Figure 9 Configurations in Debug > Debug tab

5. Select the **Flash Download** tab and select the check boxes for **Program, Verify, Reset,** and **Run**.
6. Select the **Pack** tab and **Enable** check box, and then click **OK**.

3.1.2.1 Configure trace via TPIU

Select the **Trace** tab and configure the settings to match Figure 10.

- **Trace Enable**: selected
- **ETM Trace Enable**: selected
- **Trace Port**: Sync Trace Port with 1/2/4-bit Data
- **ITM Stimulus Ports**: 0 selected

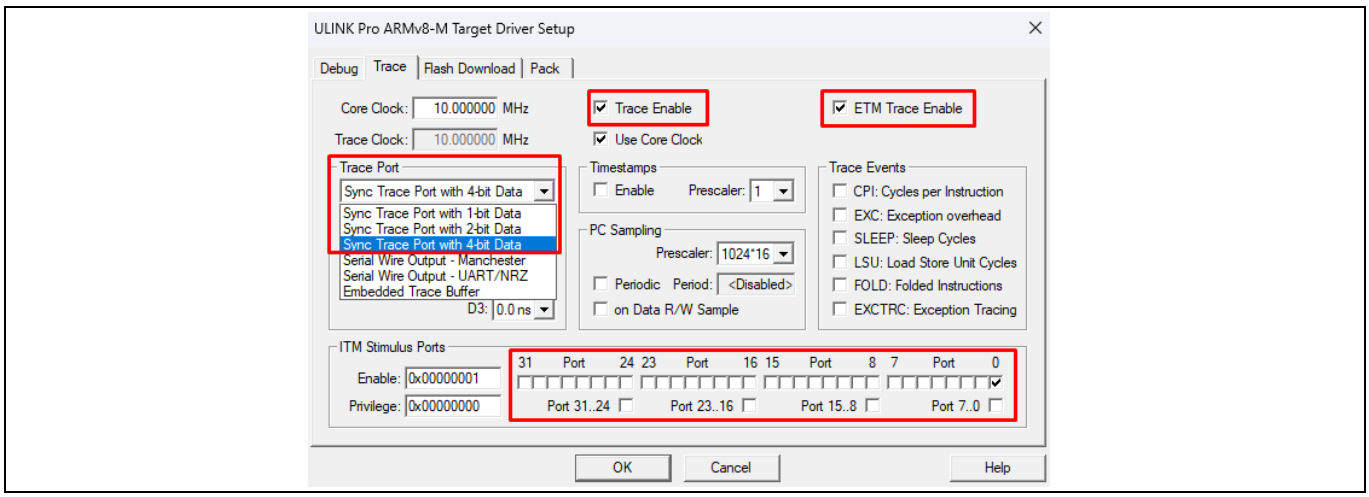


Figure 10 Configurations in Debug > Trace tab

Performing trace on PSOC™ Control C3 and PSOC™ Edge E8 MCU

3.1.2.2 Configure trace via TMC/ETB

Select the **Trace** tab and configure the settings to match Figure 11.

- **Trace Enable**: selected
- **ETM Trace Enable**: selected
- **Trace Port**: Embedded Trace Buffer
- **ITM Stimulus Ports**: 0 selected

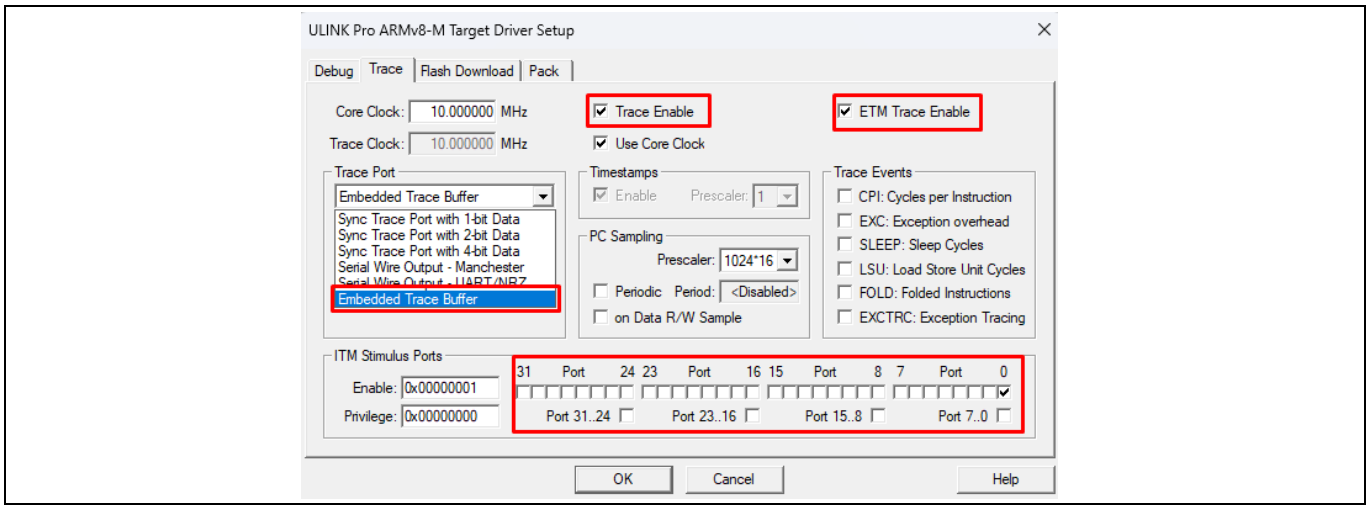


Figure 11 Configurations in Debug > Trace tab

Performing trace on PSOC™ Control C3 and PSOC™ Edge E8 MCU

3.1.3 Perform ETM trace

In the *main.c* file, modify the main function as follows. This modification will toggle the user LED on the kit every second.

```
int main(void)
{
    cy_rslt_t result;

    /* Initialize the device and board peripherals */
    result = cybsp_init();

    /* Board init failed. Stop program execution */
    if (result != CY_RSLT_SUCCESS)
    {
        CY_ASSERT(0);
    }

    /* Enable global interrupts */
    __enable_irq();

    /* Initialize the User LED */
    mtb_hal_gpio_t led;
    mtb_hal_gpio_setup(&led, CYBSP_USER_LED_PORT_NUM, CYBSP_USER_LED_PIN);
    mtb_hal_gpio_configure(&led, MTB_HAL_GPIO_DIR_OUTPUT, MTB_HAL_GPIO_DRIVE_STRONG);
    mtb_hal_gpio_write(&led, CYBSP_LED_STATE_OFF);


    for (;;)
    {
        mtb_hal_system_delay_ms(1000);

        /* Invert the USER LED state */
        mtb_hal_gpio_toggle(&led);
    }
}
```

To perform an ETM trace, do the following:

1. Click the **Build** icon on the µVision toolbar to rebuild the code.
2. Click the **Start or Stop Debug Session** icon on the µVision toolbar to program the image and start the debug perspective.
3. To open the ETM trace window, select **View > Trace > Trace Data**.
4. Click the **Run** icon on the toolbar to start the execution and collection of ETM trace data.

Performing trace on PSOC™ Control C3 and PSOC™ Edge E8 MCU

You should see the user LED toggling every second. When you pause the program execution, the collected trace data will populate in the ETM trace data window, as shown in Figure 12.

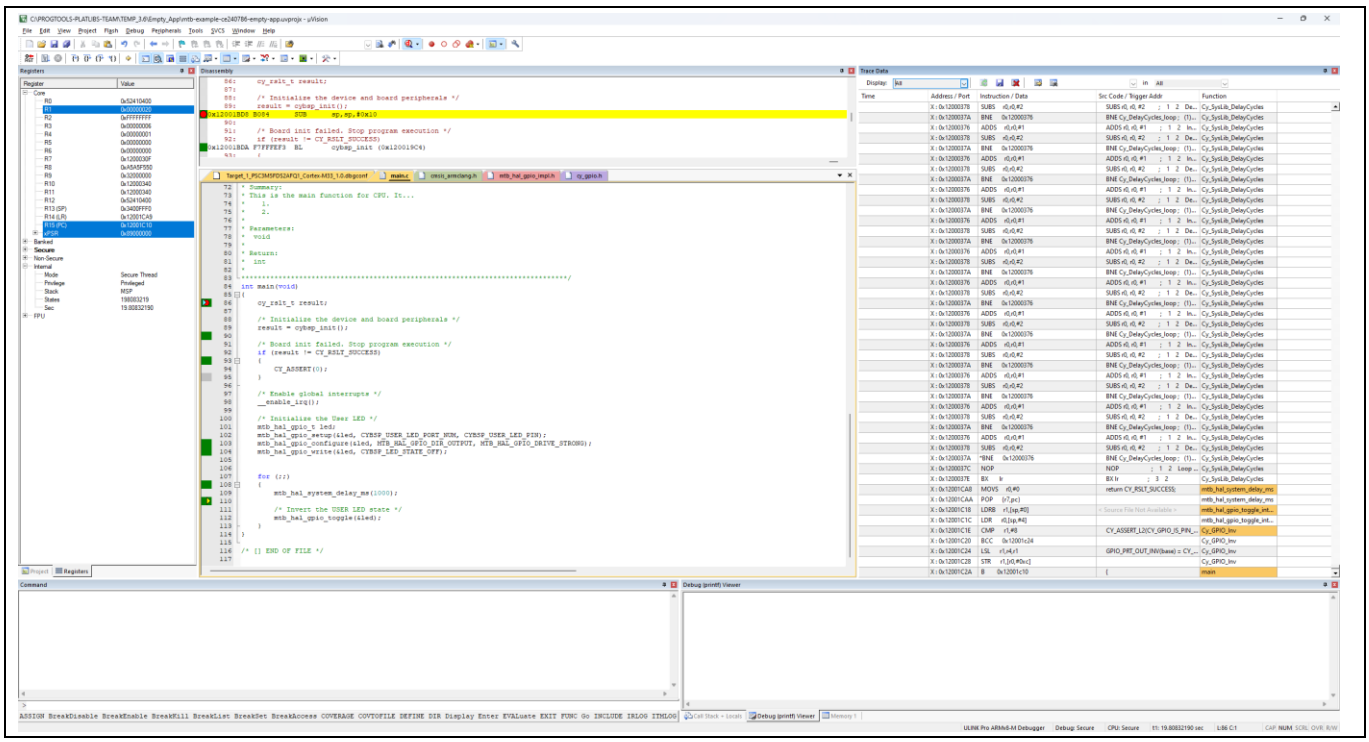


Figure 12 µVision trace data windows showing the ETM data

The ULINKpro debugger automatically calibrates to the new trace clock if the trace clock changes during code execution.

3.1.4 Perform ITM trace (printf-style debugging)

To configure for printf-style debugging, do the following:

1. In µVision, go to **Project > Manage > Run-Time Environment**.
2. Once the Manage Run-Time Environment window opens, select **Compiler > IO**, and then select the check boxes for **STDIN** and **STDOUT**. In the drop-down list for STDIN and STDOUT, select **ITM**.
3. In the *main.c* file, modify the main function and header list as follows:

Performing trace on PSOC™ Control C3 and PSOC™ Edge E8 MCU

```c
#include "mtb_hal.h"
#include "cybsp.h"
#include "stdio.h"

int main(void)
{
    cy_rslt_t result;

    /* Initialize the device and board peripherals */
    result = cybsp_init();

    /* Board init failed. Stop program execution */
    if (result != CY_RSLT_SUCCESS)
    {
        CY_ASSERT(0);
    }

    /* Enable global interrupts */
    __enable_irq();

    /* Initialize the User LED */
    mtb_hal_gpio_t led;
    mtb_hal_gpio_setup(&led, CYBSP_USER_LED_PORT_NUM, CYBSP_USER_LED_PIN);
    mtb_hal_gpio_configure(&led, MTB_HAL_GPIO_DIR_OUTPUT, MTB_HAL_GPIO_DRIVE_STRONG);
    mtb_hal_gpio_write(&led, CYBSP_LED_STATE_OFF);

    int a = 0;
    int b = 1;
    printf("Fibonacci Sequence:\n");

    for (;;)
    {
        mtb_hal_system_delay_ms(1000);

        /* Invert the USER LED state */
        mtb_hal_gpio_toggle(&led);

          /* printf-style debugging */
        printf("%d, %d, ", a, b);
        a += b;
        b += a;
    }
}
```

4. If not already done, follow the steps described in the Configure the debugger script with ULINKpro Cortex® section.
5. Click the **Build** icon on the µVision toolbar to rebuild the code.
6. Click the **Start or Stop Debug Session** icon on the µVision toolbar to program the image and start the debug perspective.
7. Select **View > Serial Windows > Debug (printf) Viewer**.
8. Click the **Run** icon on the toolbar to start the execution and collection of ITM data.

Performing trace on PSOC™ Control C3 and PSOC™ Edge E8 MCU

You should see the user LED toggling every second. The Debug Viewer displays logs as shown in Figure 13.

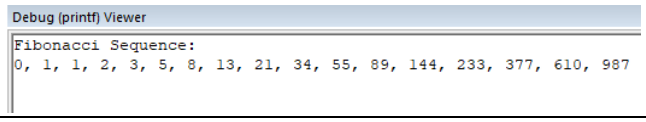


Figure 13 µVision Debug Viewer showing printf outputs

3.2 IAR Embedded Workbench for ARM

3.2.1 Configure the debugger script and debugger

The debugger configures the TPIU port and necessary clocks before starting the trace. This configuration is done through functions listed in a debugger script. The debugger scripts for PSOC™ Control C3 and PSOC™ Edge E8 MCUs are located in the IAR EW installation directory in accordance:

- <IAR Installation path> \Embedded Workbench\arm\config\debugger\Infineon\CAT1B.
- <IAR Installation path> \Embedded Workbench\arm\config\debugger\Infineon\CAT1D.

The debugger scripts with .dmac extension have default values for the TPIU port and clock divider selection. However, the default value for the TPIU port is usually not applicable for all development kits. Therefore, the script file must be edited to choose the correct TPIU port.

The PSOC™ Edge E84 Evaluation Kit uses CAT1D_trace.dmac files for configuration of TPIU port. To configure the TPIU port, open this debugger script from the IAR EWARM installation directory. In the file, search for the following parameters:

- __param _TRACE_int_div_value – parameter, that defines the clock divider value for trace clock
- __param _TRACE_d0_path – parameter, that defines the TRACEDATA[0] pin.
- __param _TRACE_d1_path – parameter, that defines the TRACEDATA[1] pin.
- __param _TRACE_d2_path – parameter, that defines the TRACEDATA[2] pin.
- __param _TRACE_d3_path – parameter, that defines the TRACEDATA[3] pin.
- __param _TRACE_clk_path – parameter, that defines the TRACECLK pin.

Note: The CoreSight trace components (ETM, ITM, DWT, TPIU, TMC) are all driven by a shared internal trace clock derived from the system clock. This internal clock is configured via the trace clock prescaler.

When using off-chip capture (TPIU), this internal clock is also output on the TRACECLK pin to synchronize the external probe. However, for on-chip capture (ETB), the external TRACECLK pin is not required, as the components rely solely on the internal clock.

Performing trace on PSOC™ Control C3 and PSOC™ Edge E8 MCU

The parameters and their comments are shown in Figure 14.

```
25  //
26  //  Trace pins route path - determine the route path for each trace pin.
27  //  _TRACE_xx_path = ( PORT_NUM << 8 ) + PIN_NUM
28  //  Default values are suitable for PSoC EDGE E8 EVALUATION KIT
29  //
30  __param _TRACE_d0_path = 0x1404;   // default - P20_4 (0x1404), alternate - P4_4 (0x404)
31  __param _TRACE_d1_path = 0x1403;   // default - P20_3 (0x1403), alternate - P4_5 (0x405)
32  __param _TRACE_d2_path = 0x1402;   // default - P20_2 (0x1402), alternate - P4_6 (0x406)
33  __param _TRACE_d3_path = 0x1401;   // default - P20_1 (0x1401), alternate - P4_7 (0x407)
34  __param _TRACE_clk_path = 0x1400;  // default - P20_0 (0x1400), alternate - P4_3 (0x403)
35
```

Figure 14 IAR debugger script snapshot for PSOC™ Edge E8 MCU

Trace data pins are routed to Port 20 of the kit, which is the default setting. However, if you want to override the parameter values in the debugger script without directly editing the file, please follow these steps:

1. In the IAR Embedded Workbench, go to **Project > Options > Debugger > Extra Options** and select **Use command line options**.
2. In the text box, enter the following command:

```
--macro_param _TRACE_d0_path=0x404
--macro_param _TRACE_d1_path=0x405
--macro_param _TRACE_d2_path=0x406
--macro_param _TRACE_d3_path=0x407
--macro_param _TRACE_clk_path=0x403
--macro_param _TRACE_int_div_value=6
```

Note: This command is space-sensitive; extra spaces should not be added around the equals sign.

You can modify any other parameter in the script in a similar way if required. For the flow followed in this application note, modifying the trace path is sufficient.

To select the debugger, in the IAR EWARM, go to **Project > Options > Debugger > Setup** and select **Driver(I-jet)** from the **Driver** drop-down menu.

To enable tracing, go to **Project > Options > Debugger > I-jet > Trace** and select the appropriate trace port depending on your setup as shown in Figure 15:

- **Parallel (ETM)** – for trace output via the TPIU and 20-pin connector
- **On-Chip (MTB/ETB)** – for trace capture via the on-chip trace buffer

Performing trace on PSOC™ Control C3 and PSOC™ Edge E8 MCU

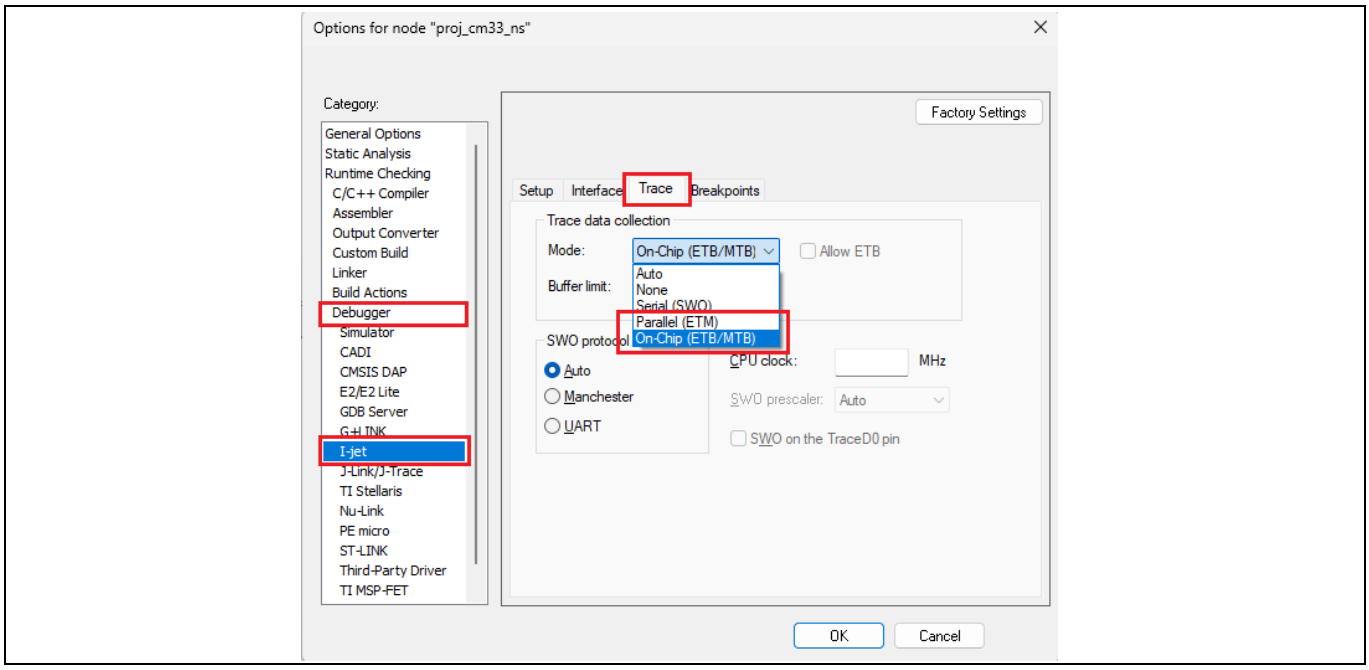


Figure 15 Configurations in Debugger > I-jet > Trace

3.2.2 Perform ETM trace

In the main.c file, modify the code as described in the Perform ETM trace section. This modification will toggle the user LED on the kit every second.

To start the ETM trace, follow these steps:

1. Click one of the buttons:
   - **Download and Debug** icon on the IAR EW toolbar to program the image and start the debug perspective.
   - **Debug without Downloading** icon on the IAR EW toolbar to start the debug perspective without programming of MCU.
2. To open ETM Trace Window for I-jet, select **I-jet > ETM trace**. The ETM trace window appears.
3. Right-click the ETM trace window and select **Enable**.
4. Click the **Go** icon on the toolbar to start the execution and collection of ETM trace data. You should see the user LED toggling every second. The debug log window is shown in Figure 16 (TPIU) and Figure 17 (TMC/ETB).

Performing trace on PSOC™ Control C3 and PSOC™ Edge E8 MCU

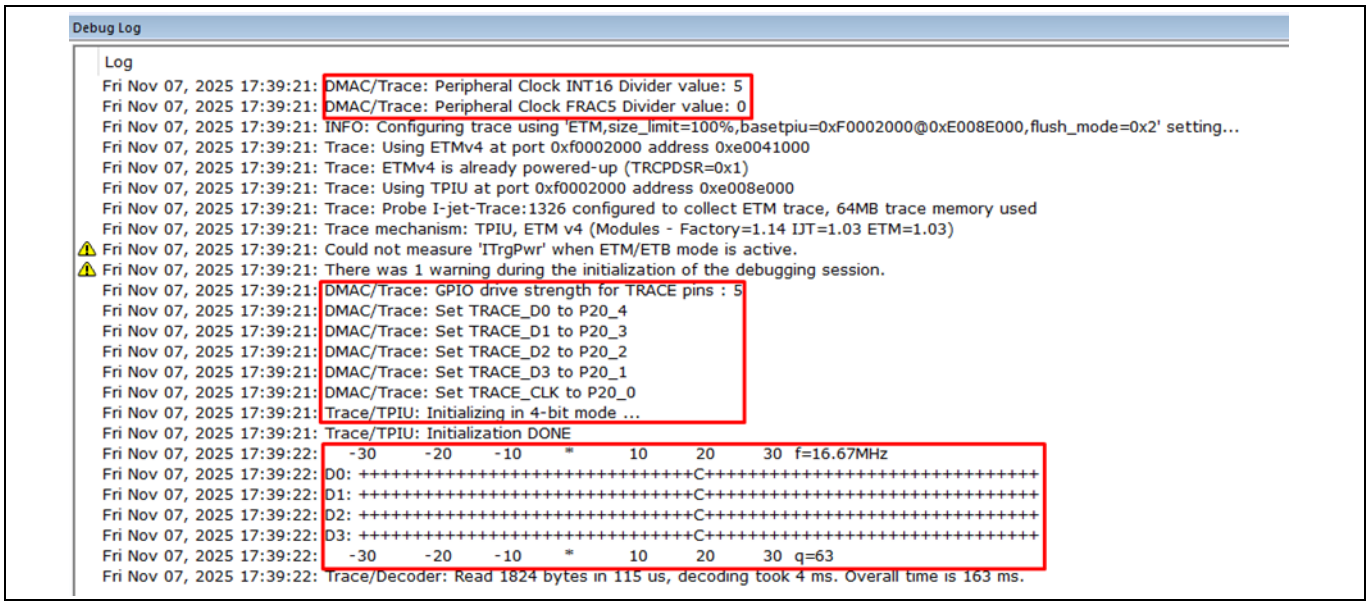


Figure 16 IAR debug log for ETM trace for first initialization using TPIU

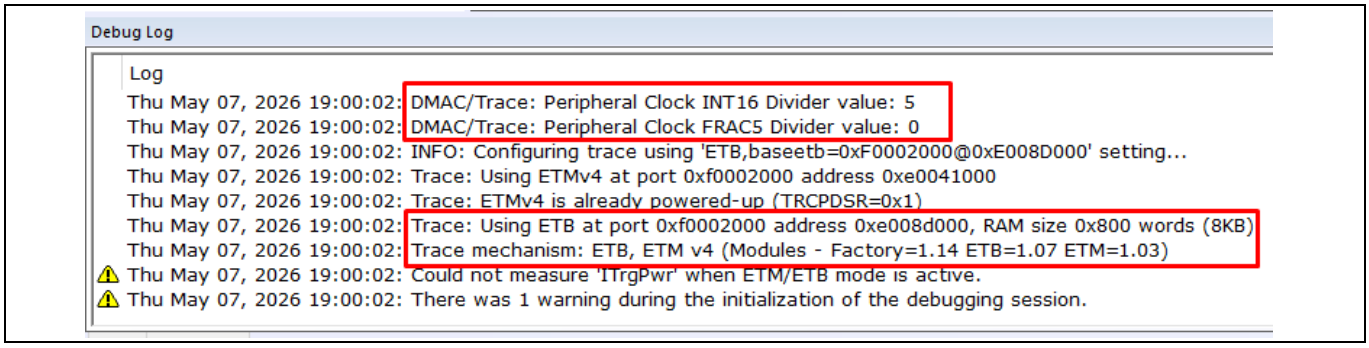


Figure 17 IAR debug log for ETM trace for first initialization using TMC/ETB

Sometimes the trace clock can have another clock after the first initialization, so we recommend recalibrating the trace clock by changing the number of pins.

The ideal way to trace is to set a breakpoint right after the cybsp_init() function. Once the breakpoint is hit, tracing stops. If you then restart the trace, the debugger recalibrates to the new frequency and starts capturing data correctly. See Figure 18.

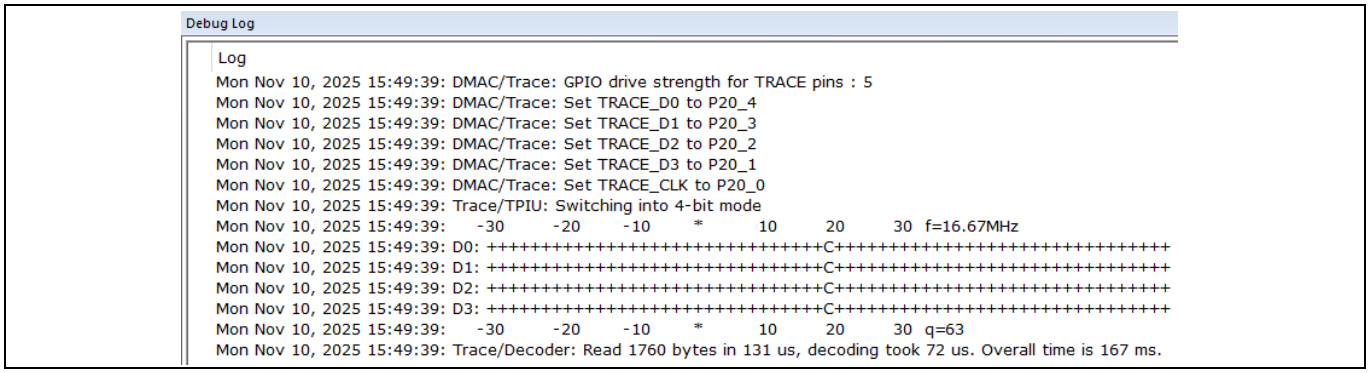


Figure 18 IAR debug log for ETM trace recalibration for TPIU

Performing trace on PSOC™ Control C3 and PSOC™ Edge E8 MCU

If you pause the program execution, the trace data collected will be populated in the ETM trace window as shown in Figure 19.

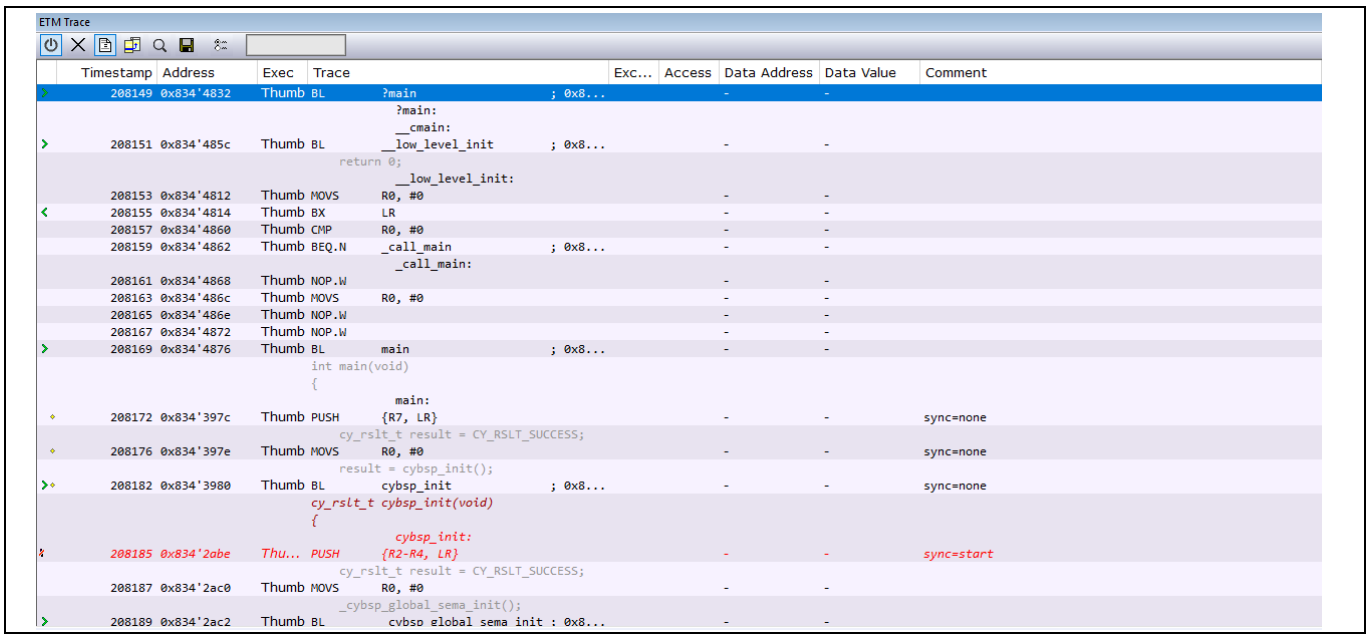


Figure 19 IAR ETM trace window showing ETM data

3.2.3 Perform ITM trace (printf-style debugging)

As previously stated, at this moment IAR Embedded Workbench **does not provide** support for ITM Trace (printf-style debugging) via Parallel Trace Output or over ETB/MTB.

4 Troubleshooting

In this section, we will examine how we can resolve common issues encountered while performing ETM and ITM tracing on the PSOC™ Control C3 and PSOC™ Edge E8 MCU.

4.1 Keil µVision

During debugging, you may encounter some common µVision Trace Status errors when using Parallel Trace on PSOC™ Control C3 and PSOC™ Edge E8 MCUs:

- Communication Error
- No Synchronization
- Data Stream Error
- Data Overflow
- Processing aborted

Note: *Detailed definitions are available at Arm's page:* https://developer.arm.com/documentation/101416/0100/-Vision-Windows/Trace-Status?lang=en

Before applying fixes, we recommend reviewing these items:

1. Ensure cybsp_init() did not reconfigure or clear the trace pins. Confirm TRACECLK and TRACEDATA[0..3] remain routed, enabled, and in trace mode after board initialization.
2. In Options for **Target > Debug > Settings > Trace**, verify:
   - Trace Port is Sync Trace Port with 1/2/4-bit Data (Parallel). Do not select SWO.
   - Enter the correct prescaler value for TRACECLK.
   - ETM is enabled; ITM/DWT are enabled if needed.

4.1.1 Recommended fixes

Reduce the number of active trace data pins

If the MCU supports configurable TPIU port size, set the trace port width from 4 to 2 or 1 bit. This lowers bandwidth and eases signal integrity and timing.

Increase compensating signal delays for Sync Trace Port

The default is 0.0 ns and often works, but if you see No Synchronization or Data Stream Error, increase these delays to move the sampling point toward the center of the valid window and compensate cable/board skew. Tune in small steps (for example, +0.5 to +2.0 ns). Start with the same value for D0..D3; if one data lane is still unstable, fine-tune it individually.

In ULINKpro Trace settings, adjust the input sampling delays as defined by Keil:

- CLK: signal delay (ns) applied to TRACECLK.
- D0..D3: signal delay (ns) applied to TRACEDATA[0]..[3].

Note: *For more information, see Arm® guidance on compensating signal delays:* *https://developer.arm.com/documentation/101416/0100/Troubleshooting/Compensate-Signal-Delays?lang=en*

Troubleshooting

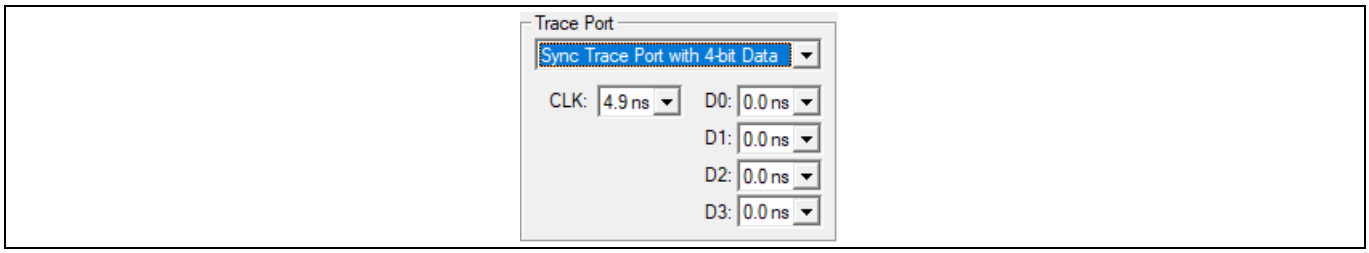


Figure 20 Sync Trace Port setting for ULINKpro probe

Adjust the trace clock divider

Lower the TPIU TRACECLK using the device's trace clock divider or reduce system clock feeding TPIU. If synchronization fails, try stepping down in increments.

If the probe reports underflow or stalls, you can also try modestly increasing TRACECLK to improve synchronization.

Change the 20-pin ribbon cable

Use a shorter shielded trace cable; verify correct orientation and a solid ground reference. Avoid long fly-wires or adapters without ground stitching.

Attach after the system is fully running

Use the **Start Debug Session**, but configure µVision to avoid a reset on connect. Go to **Project > Options > Debug > Settings** and deselect **Reset after connect**. This attaches without resetting the device, so late trace-clock initialization is preserved and helps prevent issues.

5 References

[1] AN235279: Performing ETM and ITM trace on PSOC™ 6 MCU

[2] AN238329: Getting started with PSOC™ Control C3 MCU on ModusToolbox™ software

[3] 630-60703-01: PSOC CONTROL C3M5 EVALUATION KIT Schematic

[4] AN235935 : Getting started with PSOC™ Edge E8 MCU on ModusToolbox™ software

[5] ModusToolbox™ tools package user guide (v3.6)

[6] Learn the architecture: Understanding trace by Arm® Developer

[7] IAR Embedded Workbench for ModusToolbox™ user guide

[8] Keil µVision for ModusToolbox™ user guide

[9] AN240106: Getting started with PSOC™ Control C3 security

[10] AN237849: Getting started with PSOC™ Edge security

[11] PSOC™ Control C3 architecture PSOC™ Control C3 architecture reference manual

[12] PSOC™ Edge E8x2, E8x3, E8x5, E8x6 architecture reference manual

[13] CoreSight Trace Memory Controller Technical Reference Manual

Revision history

Document revision	Date	Description of changes
**	2025-11-20	New document.
*A	2026-05-14	Added/updated the following sections: • Acronyms Table • Trace sink: TPIU (Section 2.6.3.1) • Trace sink: TMC (Section 2.6.3.2) • TPIU vs TMC/ETB comparison (Section 2.6.3.3) • ETB trace configuration for Keil µVision (Section 4.1.2.2) • ETB trace configuration for IAR Embedded Workbench • References section

Trademarks
All referenced product or service names and trademarks are the property of their respective owners.

Edition 2026-05-14

Published by

Infineon Technologies AG

81726 Munich, Germany

© 2026 Infineon Technologies AG. All Rights Reserved.

Do you have a question about this document?

Email: erratum@infineon.com

Document reference

002-42338 Rev. *A

Important Notice

Products which may also include samples and may be comprised of hardware or software or both ("Product(s)") are sold or provided and delivered by Infineon Technologies AG and its affiliates ("Infineon") subject to the terms and conditions of the frame supply contract or other written agreement(s) executed by a customer and Infineon or, in the absence of the foregoing, the applicable Sales Conditions of Infineon. General terms and conditions of a customer or deviations from applicable Sales Conditions of Infineon shall only be binding for Infineon if and to the extent Infineon has given its express written consent.

For the avoidance of doubt, Infineon disclaims all warranties of non-infringement of third-party rights and implied warranties such as warranties of fitness for a specific use/purpose or merchantability.

Infineon shall not be responsible for any information with respect to samples, the application or customer's specific use of any Product or for any examples or typical values given in this document.

The data contained in this document is exclusively intended for technically qualified and skilled customer representatives. It is the responsibility of the customer to evaluate the suitability of the Product for the intended application and the customer's specific use and to verify all relevant technical data contained in this document in the intended application and the customer's specific use. The customer is responsible for properly designing, programming, and testing the functionality and safety of the intended application, as well as complying with any legal requirements related to its use.

Unless otherwise explicitly approved by Infineon, Products may not be used in any application where a failure of the Products or any consequences of the use thereof can reasonably be expected to result in personal injury. However, the foregoing shall not prevent the customer from using any Product in such fields of use that Infineon has explicitly designed and sold it for, provided that the overall responsibility for the application lies with the customer.

Infineon expressly reserves the right to use its content for commercial text and data mining (TDM) according to applicable laws, e.g. Section 44b of the German Copyright Act (UrhG).

If the Product includes security features:

Because no computing device can be absolutely secure, and despite security measures implemented in the Product, Infineon does not guarantee that the Product will be free from intrusion, data theft or loss, or other breaches ("Security Breaches"), and Infineon shall have no liability arising out of any Security Breaches.

If this document includes or references software:

The software is owned by Infineon under the intellectual property laws and treaties of the United States, Germany, and other countries worldwide. All rights reserved. Therefore, you may use the software only as provided in the software license agreement accompanying the software.

If no software license agreement applies, Infineon hereby grants you a personal, non-exclusive, non-transferable license (without the right to sublicense) under its intellectual property rights in the software (a) for software provided in source code form, to modify and reproduce the software solely for use with Infineon hardware products, only internally within your organization, and (b) to distribute the software in binary code form externally to end users, solely for use on Infineon hardware products. Any other use, reproduction, modification, translation, or compilation of the software is prohibited. For further information on the Product, technology, delivery terms and conditions, and prices, please contact your nearest Infineon office or visit https://www.infineon.com