AN12877

How to Enable ETM Trace for i.MXRT10xx Series

Rev. 0 — June 2020

Application Note

Contents

1 Introduction

ETM trace is a high-speed trace. It offers a powerful debug mode to solve the most difficult problems. This document introduces how to enable ETM trace for i.MXRT10XX silicon and the basic steps of using utrace debugger.

2 Installing software

The TRACE32 installation package can be found from https://www.lauterbach.com/frames.html?download_overview.html. Download the *TRACE32_201909.7z* to the computer and install it.

1. Because the installation package is relatively large, you can install software components according to the target processor to save hard disk space.
2. You can find installed driver at *C:\T32\bin\windows64\drivers*.

3 Connecting hardware

The TRACE32 debugger hardware always consists of:

- Universal debugger hardware
- Debug cable specific to the processor architecture

Figure 1 is a schematic diagram of hardware connection.

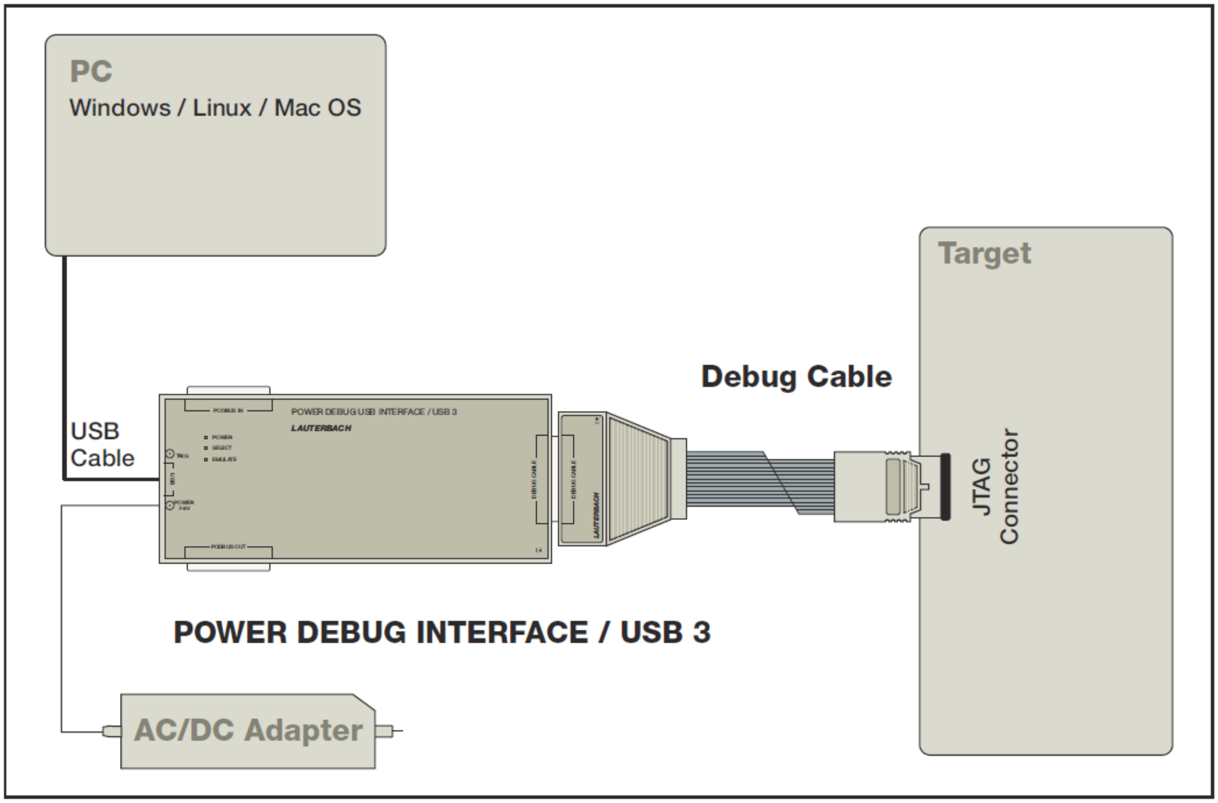


Figure 1. Schematic diagram of hardware connection

1. Taking i.MX RT1010 Validation Board (RAM) as an example, Figure 2 shows the i.MX RT1010 validation board hardware connection.

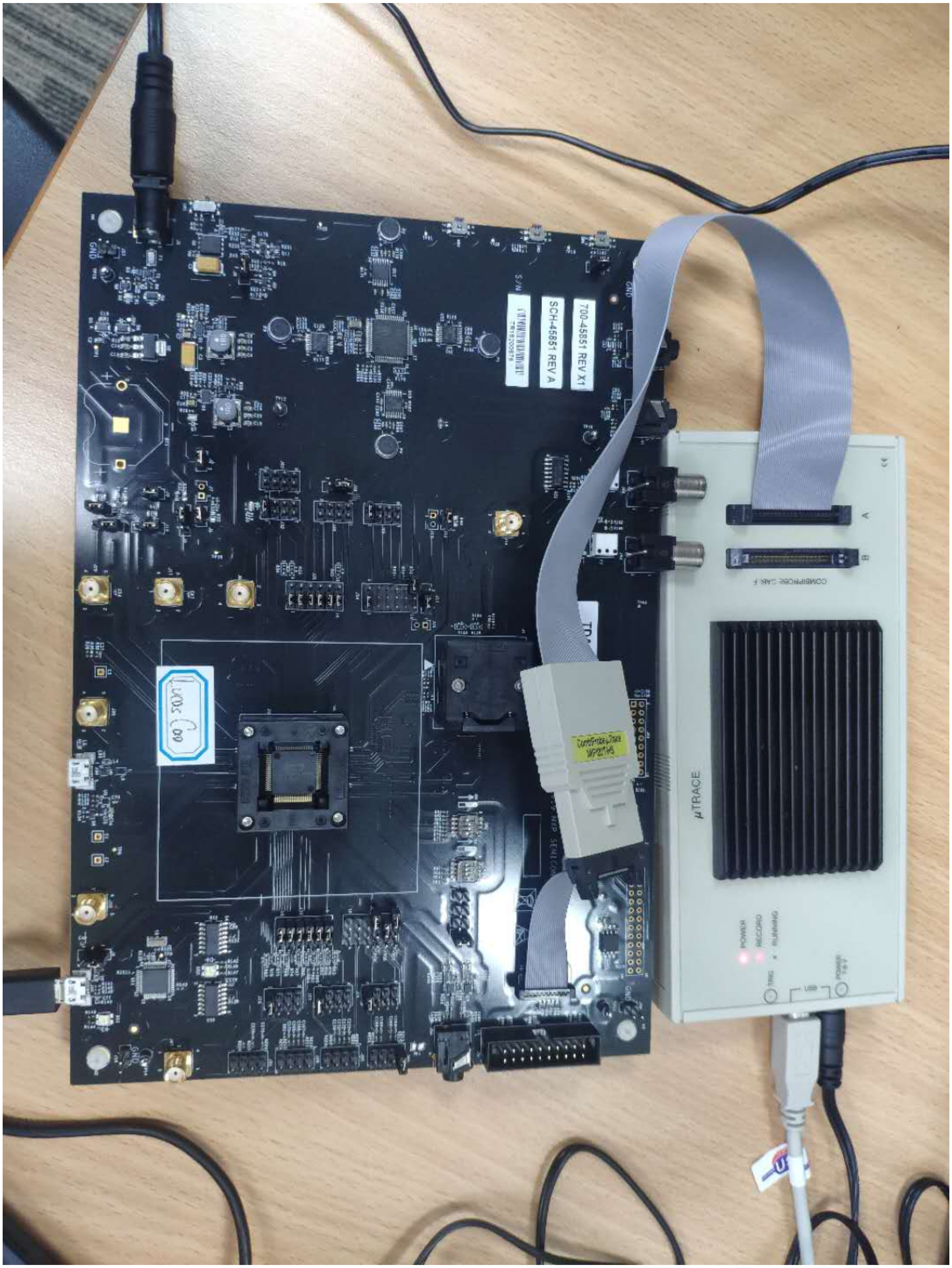


Figure 2. i.MX RT1010 validation board hardware connection

2. Write `efuse` to i.MX RT1010, change the debugging mode to JTAG, and solder the related resistors: R62, R63, R64, R65, R67, as shown in Figure 3 (enabling trace function is not related with debug port, so choosing either JTAG or SWD is fine).

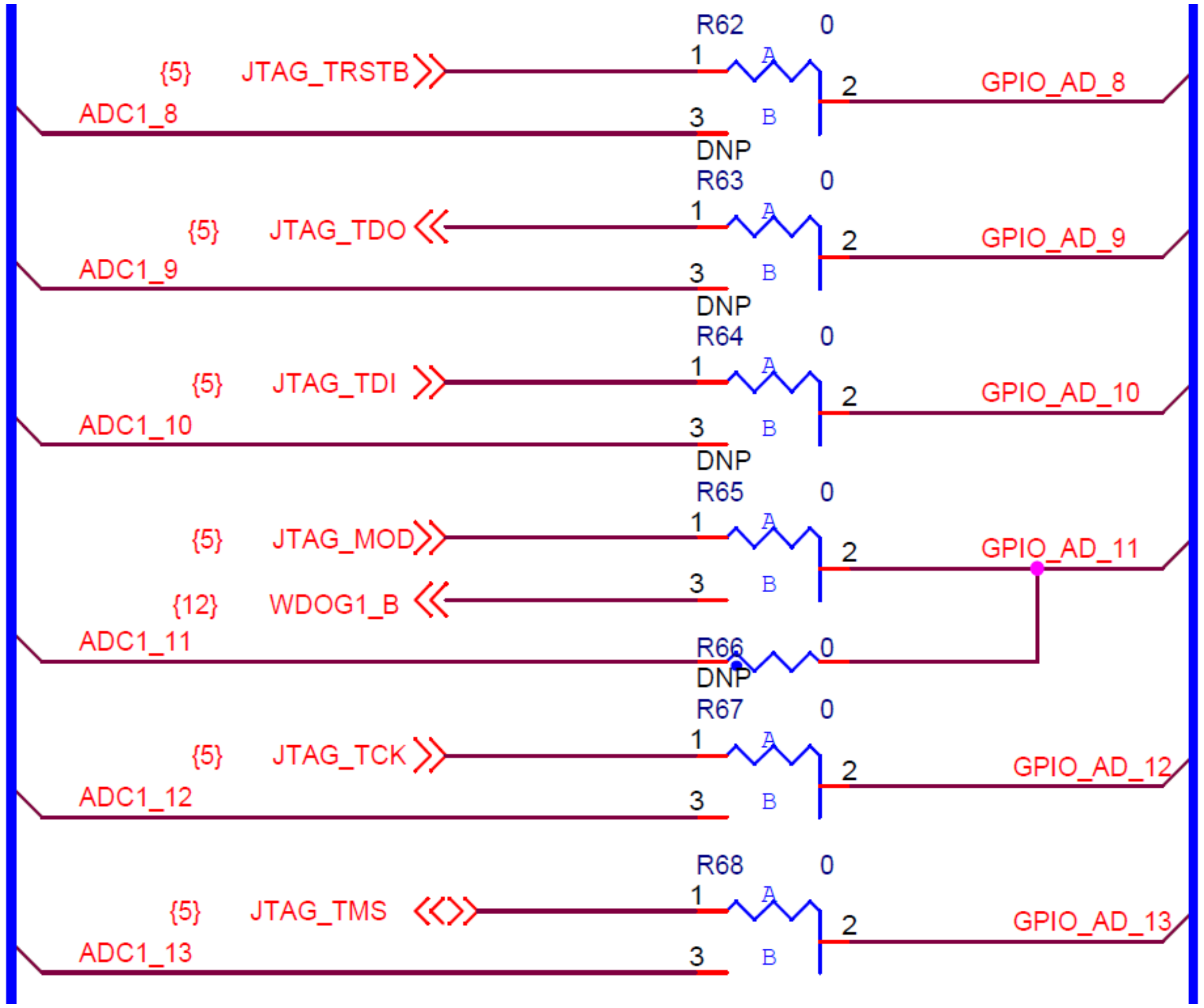


Figure 3. Related resistors to be connected

3. Weld trace-related resistors, `TRACE_CLK` (R59), `TRACE0` (R57), `TRACE1` (R73), `TRACE2` (R72), and `TRACE3` (R69). Disconnect other signal jumpers on the signal line, as shown in Figure 4.

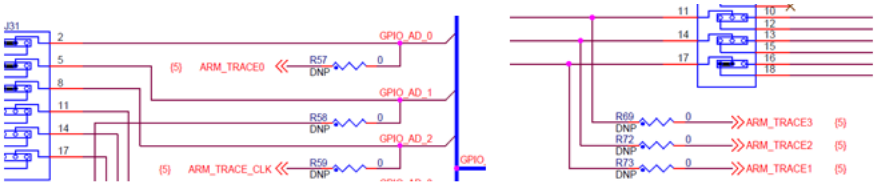
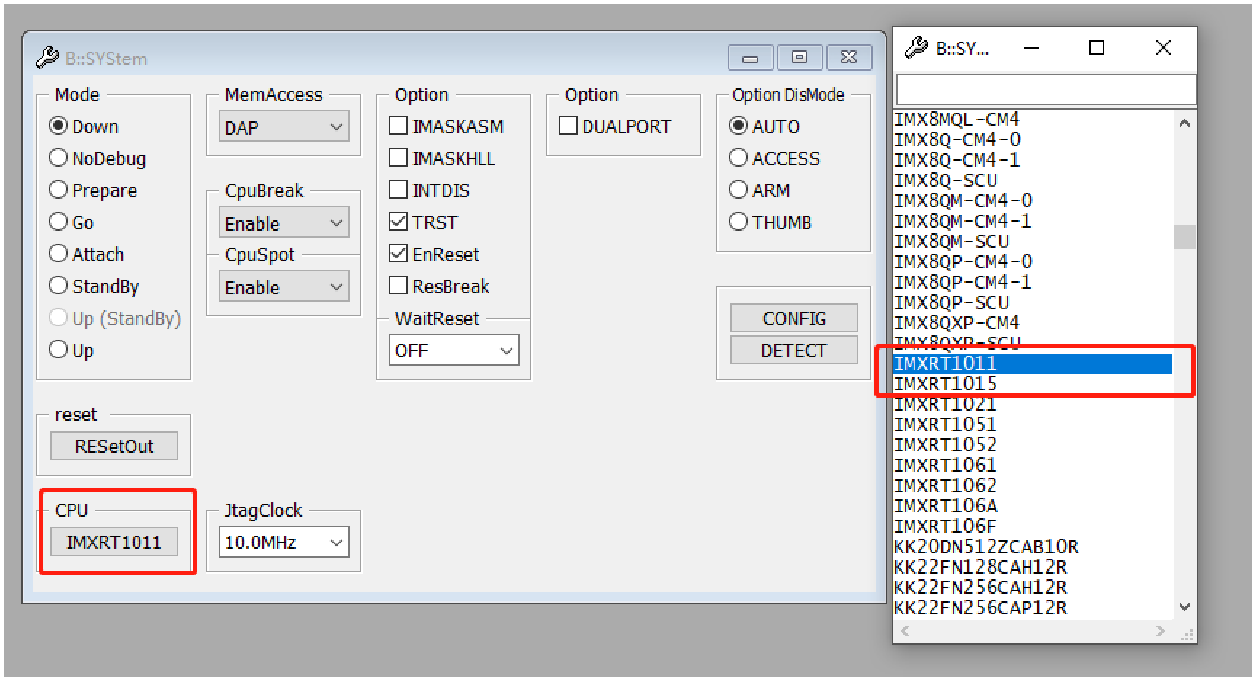


Figure 4. Trace-related resistors

4 Operating software

1. Select **CPU->System settings** and then click **CPU** to select the CPU type, as shown in Figure 5.



B::SYStem

Mode: ● Down ○ NoDebug ○ Prepare ○ Go ○ Attach ○ StandBy ○ Up (StandBy) ○ Up

MemAccess: DAP

CpuBreak: Enable

CpuSpot: Enable

Option: □ IMASKASM □ IMASKHLL □ INTDIS ☑ TRST ☑ EnReset □ ResBreak

WaitReset: OFF

Option: □ DUALPORT

Option DisMode: ● AUTO ○ ACCESS ○ ARM ○ THUMB

reset: RESetOut

CPU: IMXRT1011

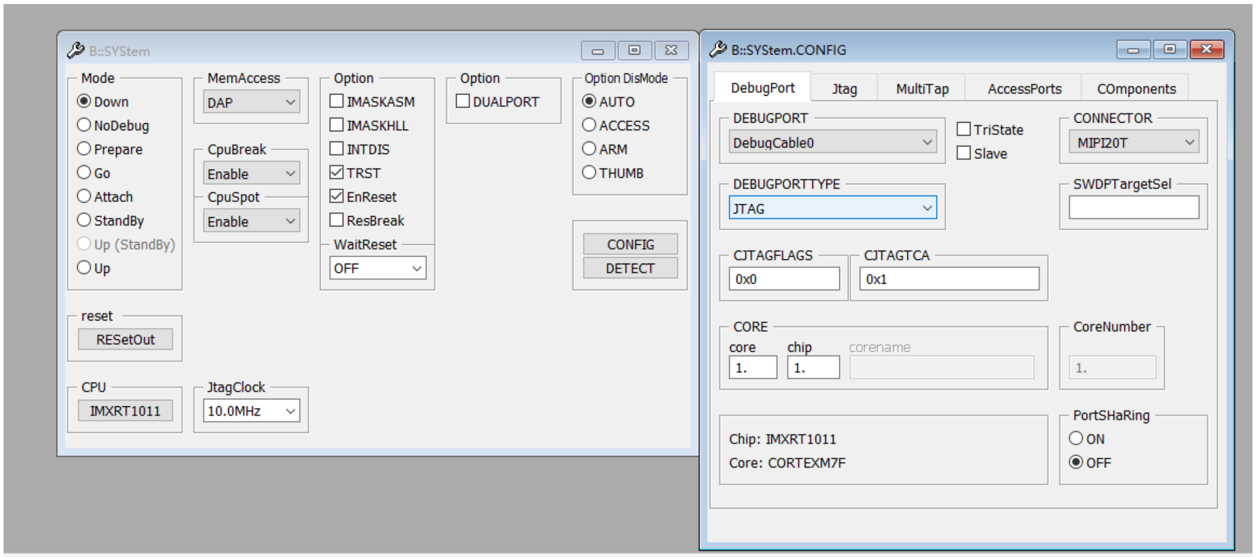
JtagClock: 10.0MHz

B::SY...

```
IMX8MQL-CM4
IMX8Q-CM4-0
IMX8Q-CM4-1
IMX8Q-SCU
IMX8QM-CM4-0
IMX8QM-CM4-1
IMX8QM-SCU
IMX8QP-CM4-0
IMX8QP-CM4-1
IMX8QP-SCU
IMX8QXP-CM4
IMX8QXP-SCU
IMXRT1011
IMXRT1015
IMXRT1021
IMXRT1051
IMXRT1052
IMXRT1061
IMXRT1062
IMXRT106A
IMXRT106F
KK20DN512ZCAB10R
KK22FN128CAH12R
KK22FN256CAH12R
KK22FN256CAP12R
```

Figure 5. Selecting CPU type

2. Set debug port type to **JTAG**, as shown in Figure 6 (Here uses JTAG as an example and by default, SWD is used).



B::SYStem.CONFIG

DEBUGPORT: DebugCable0

□ TriState □ Slave

CONNECTOR: MIPI20T

DEBUGPORTTYPE: JTAG

SWDPTargetSel:

CJTAGFLAGS: 0x0

CJTAGTCA: 0x1

CORE: core 1. chip 1. corename

CoreNumber: 1.

Chip: IMXRT1011

Core: CORTEXM7F

PortSHaRing: ○ ON ● OFF

Figure 6. Setting debug port type to JTAG

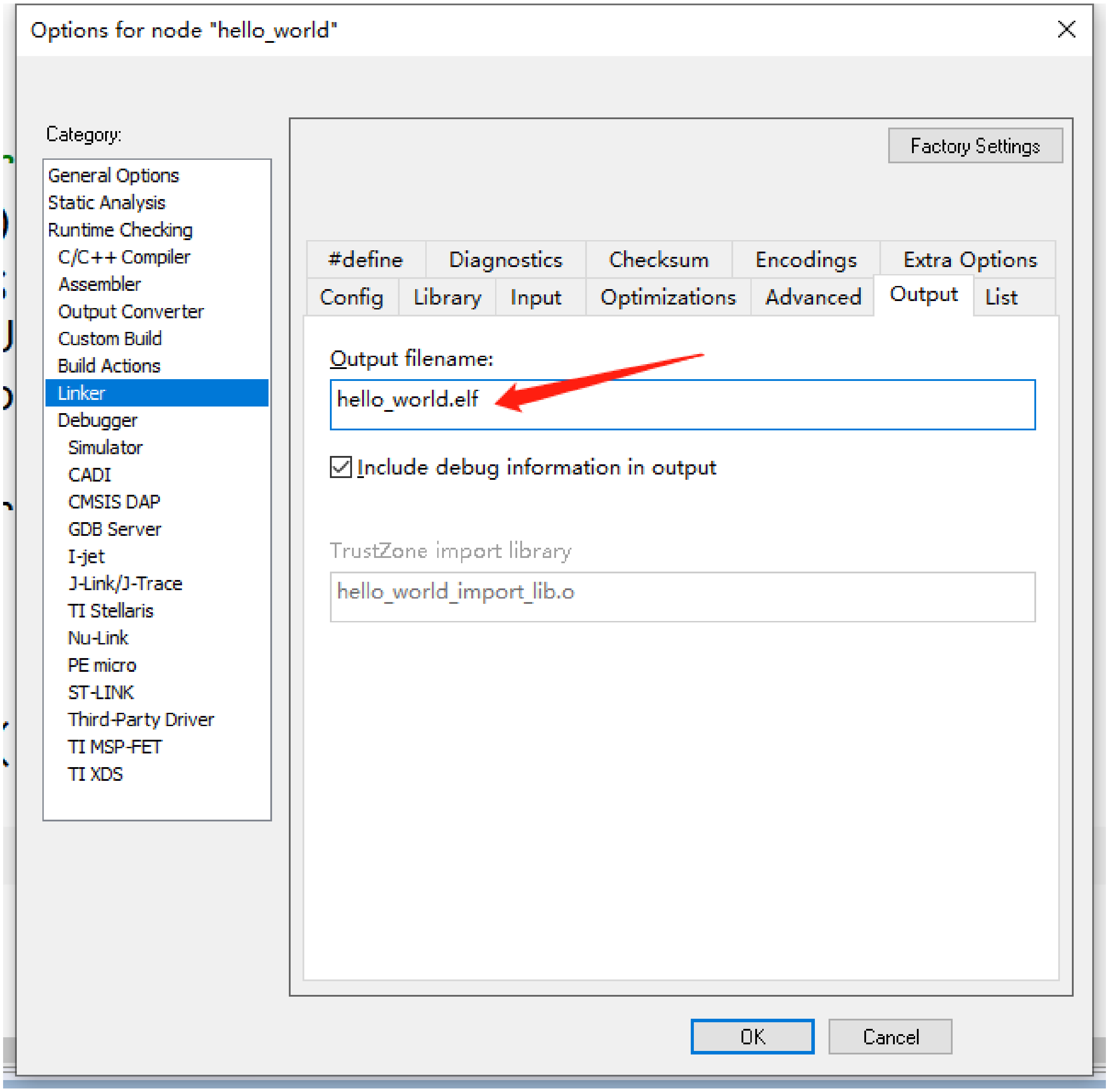
3. To debug the simulation, select **Up** on the **Mode** tab. To establish the communication between the debugger and the CPU, restart the CPU.

5 Generating APP

Taking `SDK_2.6.1_EVK-MIMX RT1010` as an example, to generate an application, perform the following operations.

1. Open IAR `hello_world` projects.
2. Set it to the debug mode.

3. Set the suffix of the generated file to `.elf`, as shown in Figure 7.



Options for node "hello_world"

Output filename:

hello_world.elf

☑ Include debug information in output

TrustZone import library

hello_world_import_lib.o

Figure 7. Generating .elf file in IAR

Now, you see the generated *hello_world.elf* file is in the debug folder.

6 Creating script

We should save the following script as a *.cmm* format file and name it to *hello_world.cmm*.

```
WinCLEAR

; --------------------------------------------------------------------------------
; initialize and start the debugger
RESet
SYStem.RESet
SYStem.CPU IMXRT1010
SYStem.CONFIG.DEBUGPORTTYPE JTAG
SYStem.Option DUALPORT ON
SYStem.MemAccess DAP
SYStem.JtagClock CTCK 10MHz
Trace.DISable
SYStem.Up

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

; --------------------------------------------------------------------------------
; initialize OFFCHIP trace (ETM, ITM)
IF COMBIPROBE()||UTRACE()||Analyzer()
(
  ; set PinMux and enable Clocks
  ; TRACECLK - IOMUX_GPIO_AD_02 – ALT7
  ; TRACEDATA0 - IOMUX_GPIO_AD_00 – ALT7
  ; TRACEDATA1 - IOMUX_GPIO_AD_13 – ALT7
  ; TRACEDATA2 - IOMUX_GPIO_AD_12 – ALT7
  ; TRACEDATA3 - IOMUX_GPIO_AD_11 – ALT7
  Data.Set AD:0x401F8040 %Long 0x7
  Data.Set AD:0x401F8048 %Long 0x7
  Data.Set AD:0x401F8088 %Long 0x7
  Data.Set AD:0x401F808C %Long 0x7
  Data.Set AD:0x401F8090 %Long 0x7

  TPIU.PortSize 1
  TPIU.PortMode Continuous
  ITM.DataTrace CorrelatedData
  ITM.ON
  ETM.Trace ON
  ETM.COND ALL
  ETM.ON
)
IF COMBIPROBE()||UTRACE()
(
  Trace.METHOD CAnalyzer
  Trace.AutoInit ON
  IF VERSION.BUILD.BASE()>=74752.
  (
    CAnalyzer.AutoFocus
  )
  ELSE
  (
    ; for uTrace & Combiprobe use manual calibration
    ; CAnalyzer.ClockDELAY Large
  )
)
IF Analyzer()
(
  Trace.METHOD Analyzer
  Trace.AutoInit ON
  Trace.AutoFocus
)

; --------------------------------------------------------------------------------
; start program execution
Go.direct main
```

```
WAIT !STATE.RUN()

; --------------------------------------------------------------------------------
; setup ITM based datatrace of variable ch
;Var.Break.Set ch /Write /TraceData

; --------------------------------------------------------------------------------
; 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.
;Trace.DRAW.Var %DEFault ch

ENDDO
```

7 Loading APP

Open the TRACE32 software, select **File->Load File**. Find the generated *hello_world.elf* file and run it.

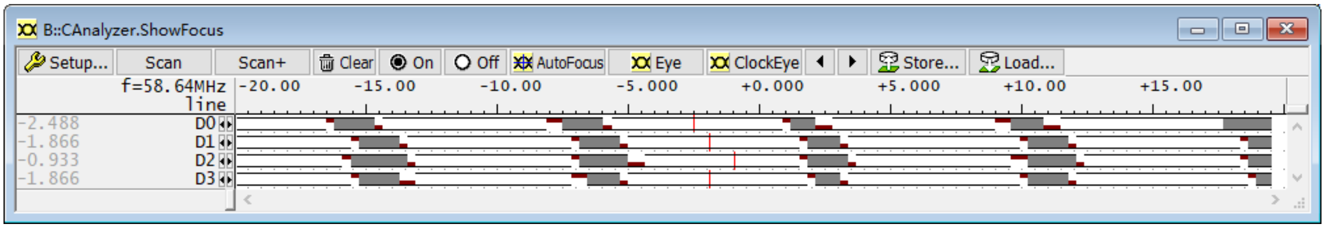
1. To open the code debugging window, select **View-> List Source**.
2. To step through the program, click **Step** or press **F2**.
3. To run the program directly, click **Go** or press **F7**.
4. To pause the program, click **Break** or press **F8**.

8 Tracing debug

TRACE32 offers a powerful feature called *Trace.ShowFocus* to analyze the signal integrity of the trace port. The functionality is similar to a sampling scope.

The horizontal axis reflects time line in nanoseconds. On the left side, the current delay is shown for each trace signal. The red line shows the sampling point. It can be different for each signal. If values smaller than zero are set or not all sampling points are equal, data lines are delayed. If values are larger than zero, the clock line is delayed.

Pressing **SCAN** to execute *Analyzer.TestFocus* to update the window. Figure 8 shows the example of the best case.



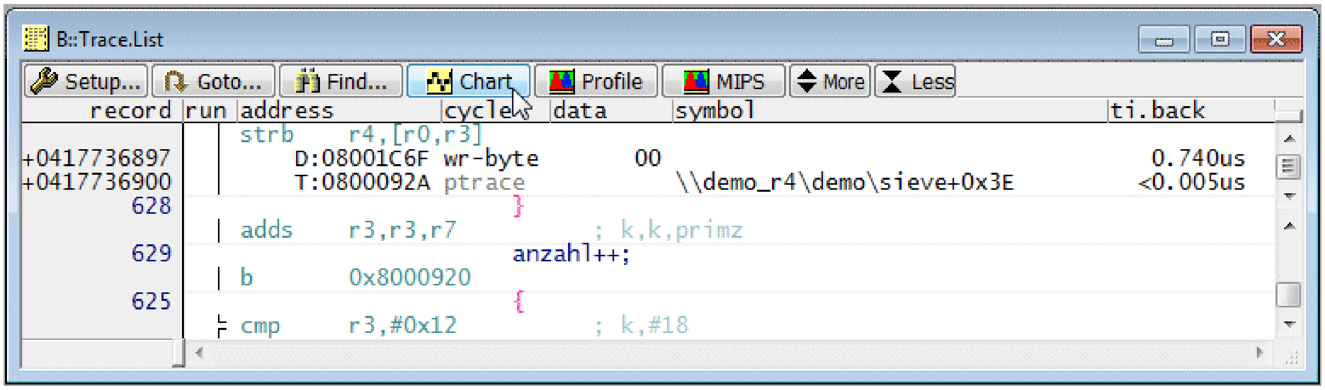
B::CAnalyzer.ShowFocus

f=58.64MHz
line
-2.488 D0
-1.866 D1
-0.933 D2
-1.866 D3

Figure 8. CAnalyzer.TestFocus

TRACE32 PowerView provides a timing diagram which shows when the program counters were in which function/symbol range.

To go to Figure 10, press **Chart** in Figure 9.



```
      record run address        cycle   data      symbol                          ti.back
              strb    r4,[r0,r3]
+0417736897        D:08001C6F  wr-byte       00                                    0.740us
+0417736900        T:0800092A  ptrace              \\demo_r4\demo\sieve+0x3E      <0.005us
         628                          }
             | adds    r3,r3,r7              ; k,k,primz
         629                          anzahl++;
             | b       0x8000920
         625                          {
             ├ cmp     r3,#0x12              ; k,#18
```

Figure 9. Trace.List window

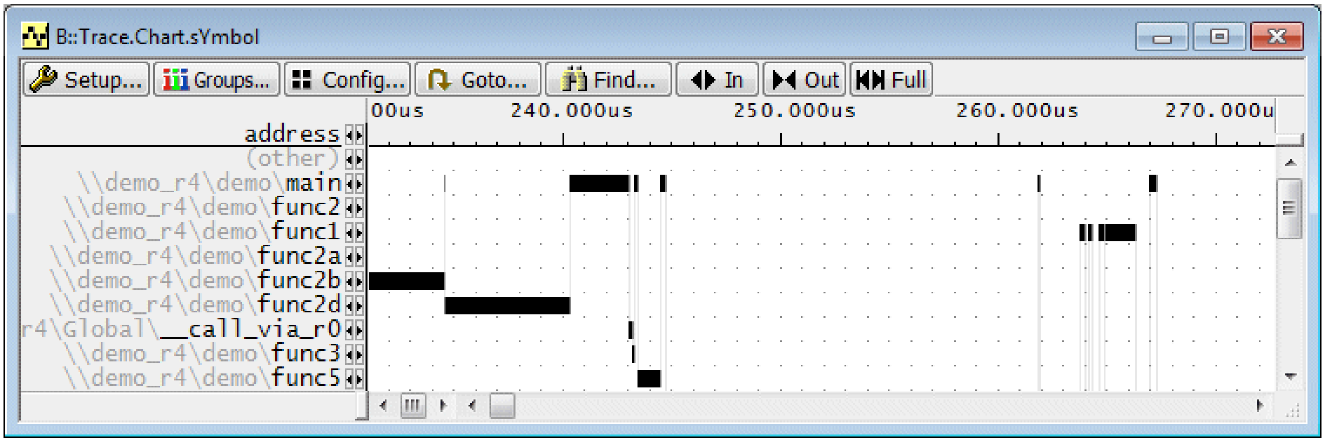


Figure 10. Chart.sYmbol window

9 Others

For other i.MX RT platforms, make the following changes.

- i.MXRT1050 validation board:
  1. Program efuse to change the debugging mode to JTAG.
  2. Welding Trace-related resistance, TRACE_CLK (R140 R592), TRACE0 (R583), TRACE1 (R270), TRACE2 (R294 R547), TRACE3 (R268 R688).
  3. Replace the following code in the script:
     - `Data.Set AD:0x401F816C %Long 0x2`
     - `Data.Set AD:0x401F814C %Long 0x3`
     - `Data.Set AD:0x401F8150 %Long 0x3`
     - `Data.Set AD:0x401F8154 %Long 0x3`
     - `Data.Set AD:0x401F8158 %Long 0x3`
- i.MXRT1020 validation board:
  1. Program efuse to change the debugging mode to JTAG.
  2. Welding Trace related resistance, TRACE_CLK (R140), TRACE0 (R815).

3. Replace the following code in the script:
   - `; set PinMux and enable Clocks`
   - `Data.Set AD:0x401F80E4 %Long 0x6`
   - `Data.Set AD:0x401F80EC %Long 0x6`

NOTE

RT1020 can only enables 1-bit ETM trace due to the SOC limitation.

How To Reach Us

Home Page:

nxp.com

Web Support:

nxp.com/support

Limited warranty and liability — Information in this document is provided solely to enable system and software implementers to use NXP products. There are no express or implied copyright licenses granted hereunder to design or fabricate any integrated circuits based on the information in this document. NXP reserves the right to make changes without further notice to any products herein.

NXP makes no warranty, representation, or guarantee regarding the suitability of its products for any particular purpose, nor does NXP assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. “Typical” parameters that may be provided in NXP data sheets and/or specifications can and do vary in different applications, and actual performance may vary over time. All operating parameters, including “typicals,” must be validated for each customer application by customer's technical experts. NXP does not convey any license under its patent rights nor the rights of others. NXP sells products pursuant to standard terms and conditions of sale, which can be found at the following address: nxp.com/SalesTermsandConditions.

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.

Security — Customer understands that all NXP products may be subject to unidentified or documented vulnerabilities. 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.

For more information, please visit: http://www.nxp.com
For sales office addresses, please send an email to: salesaddresses@nxp.com

Date of release: June 2020
Document identifier: AN12877