Application Note

Enabling Trace on AM26x Devices with Lauterbach®

Shaunak Deshpande, Nilabh Anand, Aakash Kedia

ABSTRACT

Tracing is a technique that monitors software in real-time to help developers debug and diagnose the problems, exceptions and run-time behavior of applications. Tracing can also be used for performance bench-marking or logging. Real-time tracing is an excellent choice for resolving complex issues.

Lauterbach® is a globally recognized provider of development tools for embedded systems, specializing in high-performance debugging and tracing designs. The TRACE32® tool suite offered by Lauterbach integrates hardware and software to deliver comprehensive support for high-speed trace and debugging, code analysis, and real-time tracing. The tool suite is widely used across industries for optimizing and validating embedded software. For AM26x devices, Lauterbach tools facilitate seamless debugging and enable detailed visibility into system behavior, empowering developers to efficiently address challenges in embedded software development. Tracing with Lauterbach's TRACE32 tools enables developers to gain a detailed understanding of software execution on AM26x devices. By recording the sequence of executed instructions, memory accesses, and peripheral interactions, the trace functionality helps identify performance issues, debug complex scenarios, and verify software correctness. This capability is critical for real-time embedded systems where timing and execution flow are pivotal. Lauterbach's non-intrusive tracing approach, combined with features like event time stamping, supports in-depth analysis while preserving system integrity, ultimately enhancing development productivity and system reliability.

Lauterbach Tracing is supported on the ARM R5F Cores as well as the ARM M4 cores. This document provides a step-by-step guide to enable Lauterbach ETM Trace for Texas Instruments high performance AM26x micro controllers.

Note

This document version is valid only for the AM263x and AM263Px devices.

Table of Contents

Trademarks

Code Composer Studio™ is a trademark of Texas Instruments.
FreeRTOS™ is a trademark of Amazon Web Services, Inc.
Lauterbach® and TRACE32® are registered trademarks of Lauterbach GmbH.
Windows® is a registered trademark of Microsoft Corporation.
Linux® is a registered trademark of Linus Torvalds.
All trademarks are the property of their respective owners.

1 List of Acronyms

1. ETM - Embedded Trace Macrocell
2. MCU - MicroController Unit
3. PRU - Programmable Real-time Unit
4. SDK - Software Development Kit
5. SBL - Secondary BootLoader
6. OSPI - Octal Serial Peripheral Interface
7. QSPI - Quad Serial Peripheral Interface
8. GPIO - General Purpose Input Output
9. IOMUX - Pin Multiplexing
10. I2C - Inter Integrated Circuit
11. ROM - Read only memory
12. CPU - Central Processing Unit

2 Software Setup

Install the following software and tools required to run MCU_PLUS_SDK applications on AM26x devices.

1. MCU_PLUS_SDK (Version 10.01 and above)
   a. AM263x: Download link for the AM263x MCU_PLUS_SDK
   b. AM263Px: Download link for the AM263Px MCU_PLUS_SDK
2. Code Composer Studio™: Download link for Code Composer Studio (CCS)
3. Syscfg: Download link for Sysconfig tool
4. TI-ARM-CLANG compiler: Download link for the TI-ARM-CLANG Compiler
5. TI UniFlash tool (optional): Download link for the TI UniFlash tool
6. Python3: Download link for Python3
7. OpenSSL: Download link for OpenSSL

If users need further help, then refer to the Download, Install and Setup SDK and Tools page from the official documentation.

Lauterbach software - The Trace32 software package can be downloaded from: Lauterbach Support and Training. Install this at C:\T32 for Windows® and install at the default location for Linux®.

3 Hardware Setup

The hardware needed to enable Lauterbach® trace are listed below:

1. AM26x microcontroller:
   a. AM263x controlCARD: TMDSCNCD263.
   b. AM263Px controlCARD: TMDSCNCD263P.
2. HSEC dock and breakout board: TMDSHSECDOCK-AM263.
3. Standard power supply for the HSEC Dock breakout board.
4. USB Type A to USB micro-B cable for JTAG, XDS110 connection to the AM26x microcontroller.
5. Lauterbach connector adapter.
6. Trace probe.
7. PowerView Trace.
8. JTAG cable.
9. Power supply to Lauterbach.
10. Trace ribbon cable.

3.1 AM263x Connections

1. Dock the AM263x device on the HSEC Dock breakout board.
2. Connect the power supply to the HSEC Dock breakout board.
3. Connect the USB Type-A to micro-B from the Host PC to the AM26x microcontroller.
4. On the AM263Px, put the SW-5 switch to OFF to disconnect the on-board debugger.

Switch	State
SW-5	Low

5. Power on the HSEC Dock breakout board. Users now see the LD1, LD6, LD14, LD15 glowing on the AM263x.

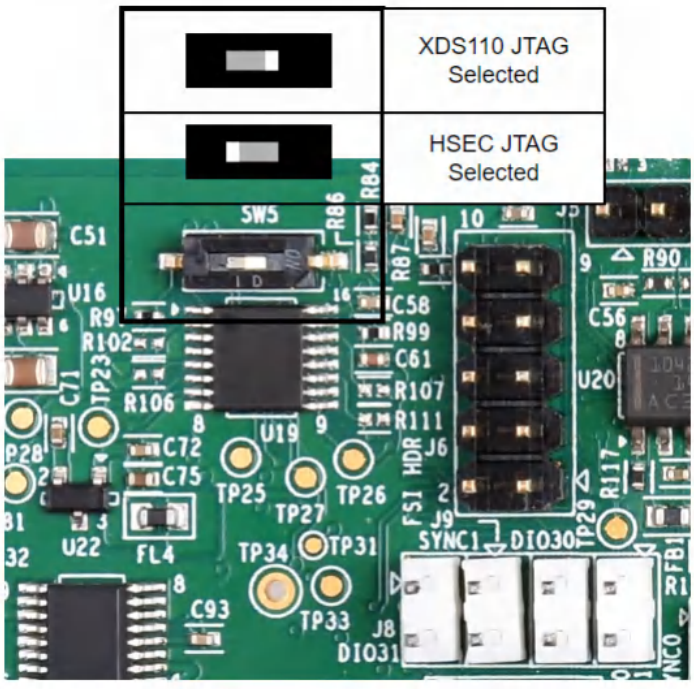

Figure 3-1. AM263x PCB# PROC E2 SW-5 Switch

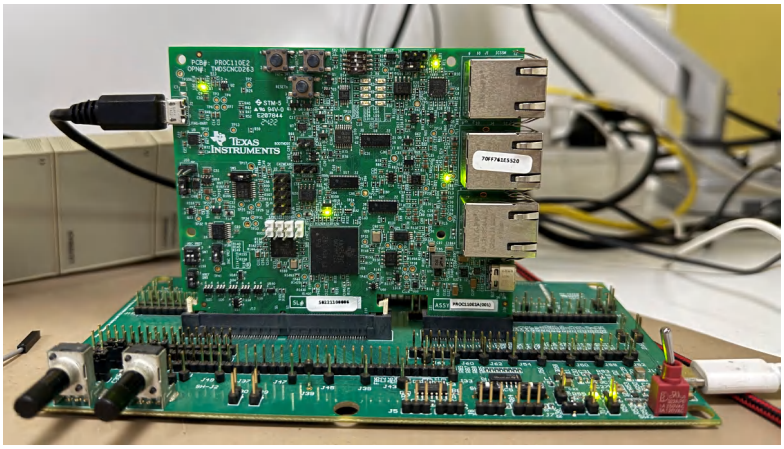

Figure 3-2. AM263x Control Card Mounted on HSEC Dock Board

3.2 AM263Px Connections

The AM263Px schematics differ from the AM263x. As a result, some of the steps can be different in-case of AM263Px.

1. Dock the AM263Px device on the HSEC Dock breakout board.
2. Connect the power supply to the HSEC Dock breakout board.
3. Connect the USB Type-A to micro-B from the Host PC to the AM26x microcontroller.
4. On the AM263Px, put the SW-1 switch to OFF to disconnect the onboard debugger. Next, put SW-14 switch to OFF, SW-15 switch to OFF and SW-16 switch to ON state. This is necessary to route the signals to the HSEC board to which the Lauterbach trace pins are connected.

Switch	State
SW-1	Low
SW-14	Low
SW-15	Low
SW-16	High

5. Power on the HSEC Dock breakout board. LD2, LD4, LD5, LD9 glows on the AM263Px.

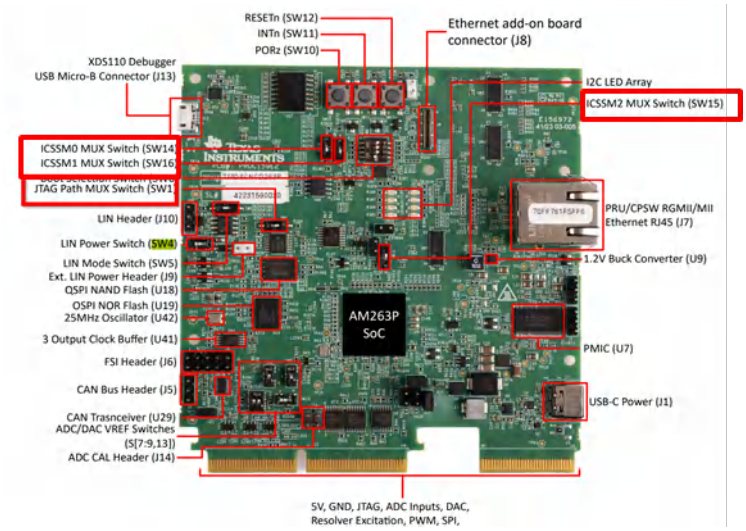

Figure 3-3. AM263Px PCB# PROC E2 Switches

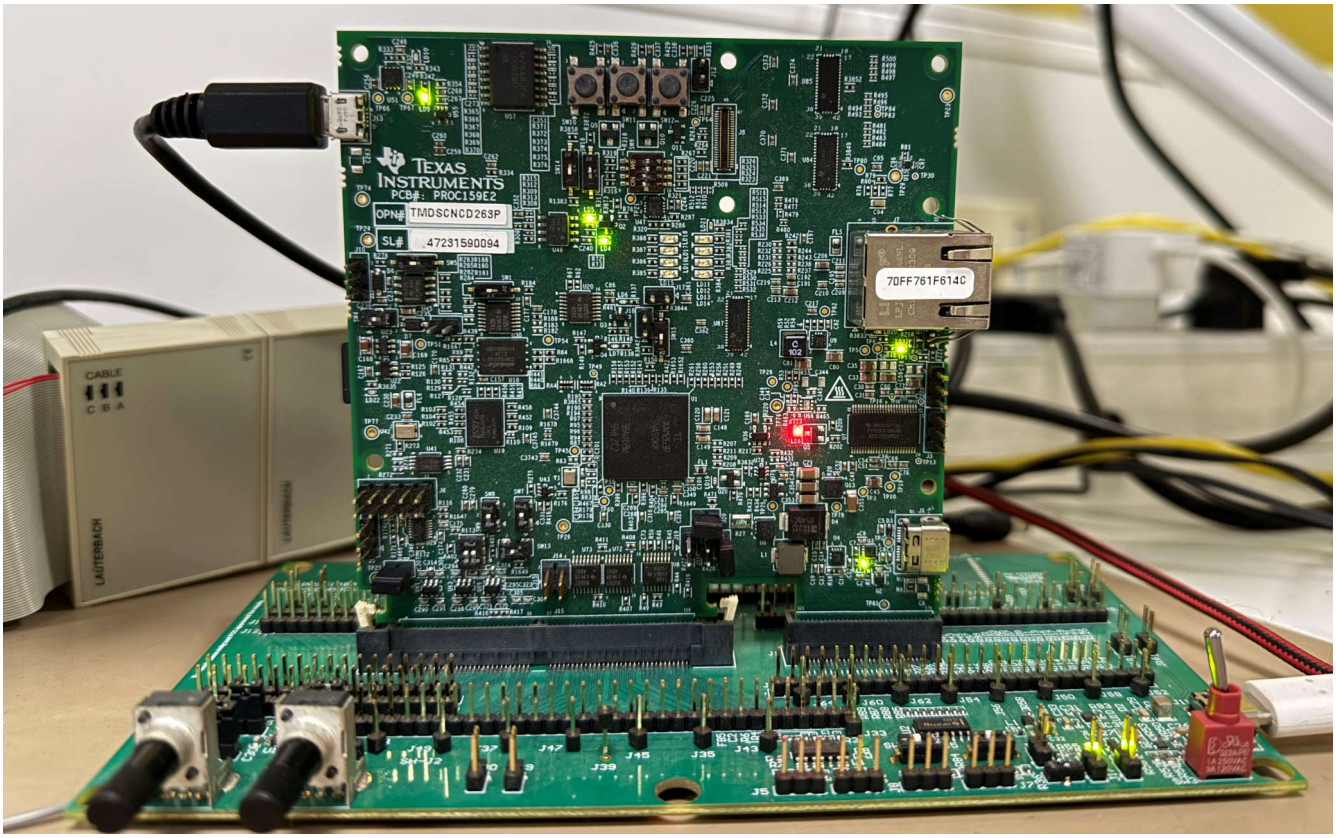

Figure 3-4. AM263Px Control Card Mounted on HSEC Dock Board

3.3 Lauterbach® Connections

1. Connect the Lauterbach adapter to the Dock board.
2. Connect the Lauterbach trace probe to the same port (A or B) as on the adapter connected to the Dock board.
3. Make sure the Trace probe and PowerView trace are connected properly to the adapter.
4. Connect the JTAG cable between PowerView Trace and adapter.
5. Connect the Lauterbach power supply to the PowerView trace and power on the hardware.
6. Connect the Trace ribbon cable between the Trace probe and adapter.

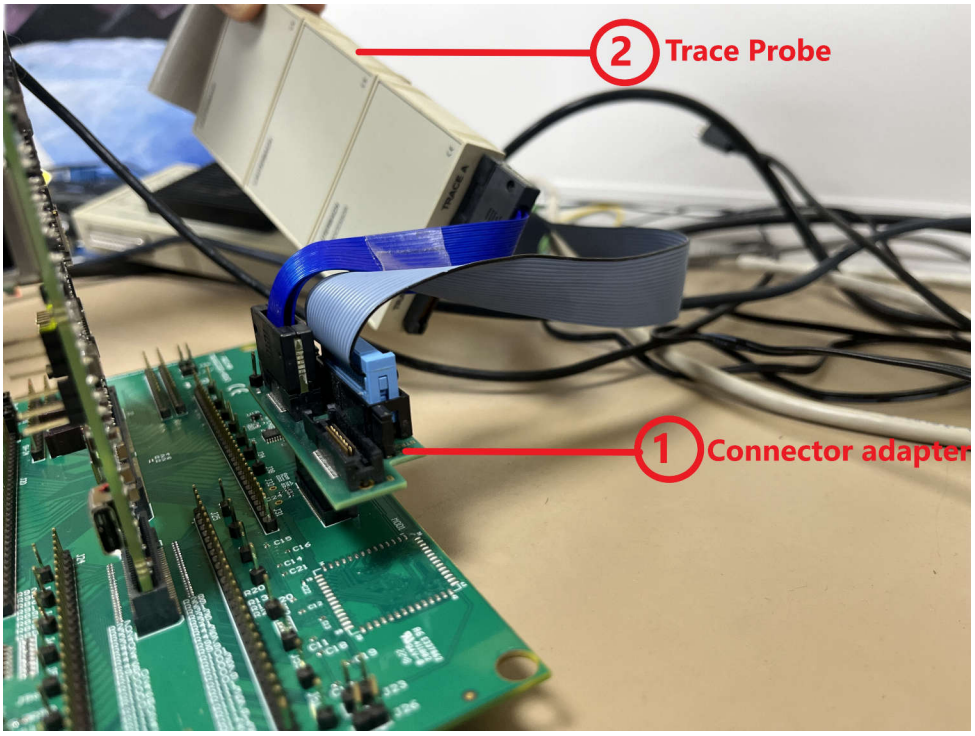

Figure 3-5. Lauterbach Setup Connections

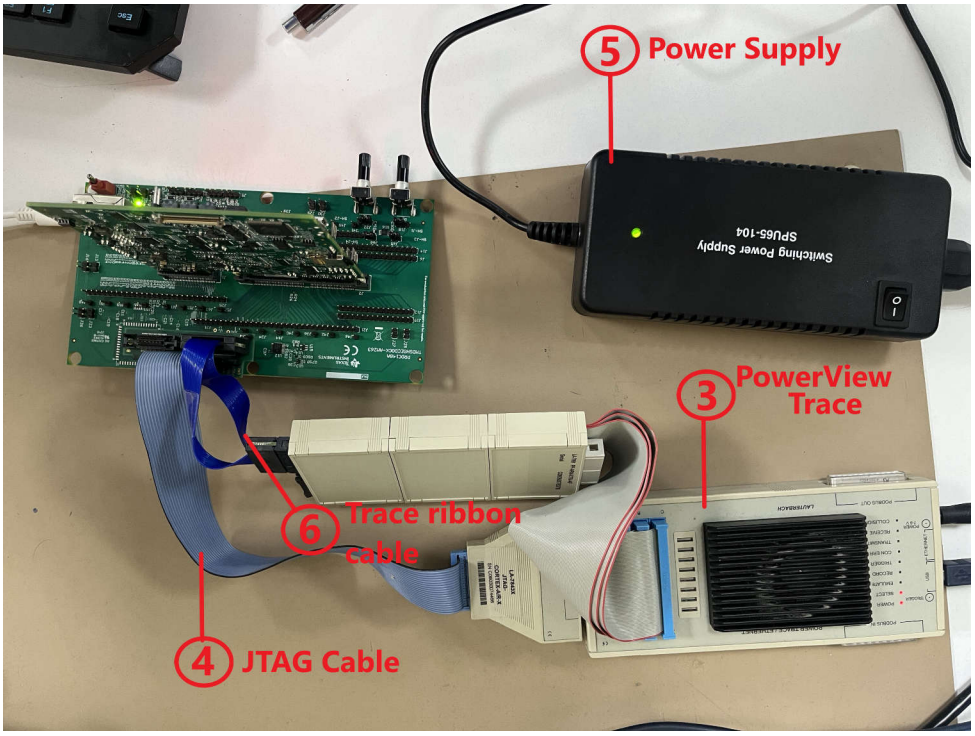

Figure 3-6. Lauterbach Setup Connections

Now, users can power on the Lauterbach power supply.

4 Building MCU_PLUS_SDK Examples

CAUTION

The debug firewalls are opened by the hsmRtImg. This image is present at <mcu_plus_sdk>/source/security/security_common/drivers/hsmclient/soc/<device_name>/hsmRtImg.h

If users are using an older MCU_PLUS_SDK version (before v10.01), then replace the hsmRtImg.h with the updated files:

AM263x - Github

AM263Px- Github

As a part of this application note, users use MCU_PLUS_SDK examples, run them on the AM26x MCU and get the trace. In case users want to use a different application, skip this section. Make sure there is a *.debug* configuration build for consistent trace results.

4.1 CCS Import And Build

1. Import the application in CCS using the steps mentioned here: Using SDK with CCS projects
2. Right-click on the project in the project-view window and build the application in debug configuration. This generates a .out binary file, which is loaded to the AM26x device for debugging.

4.2 Command Line Build

1. From the top level MCU_PLUS_SDK folder, open a terminal window and using the GNU Make commands, build the application. For example,

```
#TO CLEAN
gmake -sj -C examples/drivers/gpio/gpio_led_blink/am263px-cc/r5fss0-0_nortos/ti-arm-clang/
PROFILE=debug clean

#TO SCRUB
gmake -sj -C examples/drivers/gpio/gpio_led_blink/am263px-cc/r5fss0-0_nortos/ti-arm-clang/
PROFILE=debug scrub

#TO BUILD
gmake -sj -C examples/drivers/gpio/gpio_led_blink/am263px-cc/r5fss0-0_nortos/ti-arm-clang/
PROFILE=debug all
```

For more details, refer to Building SDK with makefiles.

Note

To run the above commands for AM263x, replace the device name to use am263x instead of am263px.

5 CMM Scripts

CMM is a batch type scripting language used by debuggers. The CMM scripts below handle the reset and connection of cores, configuring I2C clocks, trace pins, IO Expander configurations, generating off-chip trace results and displaying them in a window. By default, this works for the R5F Core-0, and can be modified to run for other ARM R5F and ARM M4 cores.

CAUTION

The scripts are validated for NoRTOS based applications. Some additional steps are required to get FreeRTOS™ task table and FreeRTOS component details. Refer to the OS awareness manual FreeRTOS.

5.1 AM263x CMM Script

AM263x CMM Script

```
; --------------------------------------------------------------------------
; Copyright (C) 2023-2024 Texas Instruments Incorporated
;
; Redistribution and use in source and binary forms, with or without
; modification, are permitted provided that the following conditions
; are met:
;
; Redistributions of source code must retain the above copyright
; notice, this list of conditions and the following disclaimer.
;
; 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 provided with the
; distribution.
;
; Neither the name of Texas Instruments Incorporated nor the names of
; contributors can 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
; OWNER 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.
; --------------------------------------------------------------------------
;
; --------------------------------------------------------------------------
; @Title: Simple demo script for Cortex-R5F on TMDSCNCD263
; @Description:
;   Prompts the user to load application into RAM and sets up a demo debug
;   scenario for the first Cortex-R5F of the SS0 cluster.
;   In addition, configures the off-chip trace.
;   Prerequisites:
;    * Plug TMDSCNCD263 onto TI debug&trace adapter TMDSHSECDOCK-AM263
;    * Connect Debug Cable to J13 via adapter LA-3818
;    * SW5=OFF to deactivate onboard debugger and enable external JTAG debugger
; @Keywords: ARM, Cortex-R5F
; @Board: TMDSCNCD263
; @Chip: AM2634
; @Copyright: 2023-2024 Texas Instruments Incorporated
; --------------------------------------------------------------------------

&sbl=TRUE() ; When running in QSPI Bootmode

RESet
SYStem.CPU AM2634-SS0
CORE.ASSIGN 1.

IF &sbl==FALSE() ; In case of QSPI bootmode, SBL takes care of the below config
(
  ; --------------------------------------------------------------------------
  ; Attach to system bus and make some preparations
  SYStem.Mode PREPARE

  ; Unlock MSS_CTRL register
  Data.Set EAHB:0x50D01008 %Long 0x01234567 ; MSS_CTRL_LOCK0_KICK0
  Data.Set EAHB:0x50D0100c %Long 0x0fedcba8 ; MSS_CTRL_LOCK0_KICK1

  ; Eclipse ROM, use RAM in ATCM
  Data.Set EAHB:0x50D00080 %Long 0x00000007 ; MSS_CTRL_R5SS0_ROM_ECLIPSE

  ; Let core run
  Data.Set EAHB:0x50D00024 %Long 0x00000000 ; MSS_CTRL_R5SS0_CORE0_HALT
)

; --------------------------------------------------------------------------
; Attach to the cores
```

```
SYStem.Mode Attach
Break

IF &sbl==TRUE()
(
  Data.Set EAHB:0x50D01008 %Long 0x01234567 ; MSS_CTRL_LOCK0_KICK0
  Data.Set EAHB:0x50D0100c %Long 0x0fedcba8 ; MSS_CTRL_LOCK0_KICK1
)

; --------------------------------------------------------------------------
; Disable the MPU and the caches that have been enabled by the firmware (SCTLR)
Data.Set C15:0x1 %Long (Data.Long(C15:0x1)&(~0x1005))

; --------------------------------------------------------------------------
; Load demo program, opens up a file explorer. Navigate and select the application binary you wish
data.LOAD.Elf *

; --------------------------------------------------------------------------
; Configure off-chip trace
IF Analyzer()||CAnalyzer()
(
  ; Unlock TOP_RCM register
  Data.Set EAHB:0x53201008 %Long 0x01234567 ; TOP_RCM_LOCK0_KICK0
  Data.Set EAHB:0x5320100C %Long 0x0fedcba8 ; TOP_RCM_LOCK0_KICK1
  WAIT 1.MS

  IF &sbl==FALSE() ; Incase of OSPI Bootmode, SBL does the PLL clock configuration
  (
    ; Config core PLL
    Data.Set EAHB:0x53200410 %Long 0x00010009 ; TOP_RCM_PLL_CORE_M2NDIV
    Data.Set EAHB:0x53200414 %Long 0x00000320 ; TOP_RCM_PLL_CORE_MN2DIV
    Data.Set EAHB:0x53200408 %Long 0x00000001 ; TOP_RCM_PLL_CORE_TENABLE
    Data.Set EAHB:0x53200404 %Long 0x00095001 ; TOP_RCM_PLL_CORE_CLKCTRL
    Data.Set EAHB:0x53200430 %Long 0x00000103 ; TOP_RCM_PLL_CORE_HSDIVIDER_CLKOUT1
  )

  ; Select trace clock source and divider
  Data.Set EAHB:0x53200C20 %Long 0x00000222 ; TOP_RCM_TRCCLKOUT_CLK_SRC_SEL
  Data.Set EAHB:0x53200C24 %Long 0x00000222 ; TOP_RCM_TRCCLKOUT_DIV_VAL

  ; Unlock IOMUX register and configure IOs
  Data.Set EAHB:0x53100298 %Long 0x83e70b13 ; IOMUX_IO_CFG_KICK0
  Data.Set EAHB:0x5310029C %Long 0x95a4f1e0 ; IOMUX_IO_CFG_KICK1
  WAIT 1.MS

  Data.Set EAHB:0x53100064 %Long 0x00000501 ; IOMUX_UART0_RTSN_CFG_REG
  Data.Set EAHB:0x53100068 %Long 0x00000501 ; IOMUX_UART0_CTSN_CFG_REG

  Data.Set EAHB:0x531000B0 %Long 0x000007F7 ; IOMUX_EPWM0_B_CFG_REG (GPIO44 -> input + pull-high)
  Data.Set EAHB:0x531000BC %Long 0x000007F7 ; IOMUX_EPWM2_A_CFG_REG (GPIO47 -> input + pull-high)

  Data.Set EAHB:0x53100298 %Long 0x83e70b13 ; IOMUX_IO_CFG_KICK0
  Data.Set EAHB:0x5310029C %Long 0x95a4f1e0 ; IOMUX_IO_CFG_KICK1
  WAIT 1.MS

  Data.Set EAHB:0x531001DC %Long 0x004 ; IOMUX_PR0_PRU1_GPO19_CFG_REG (TRC_CLK)
  Data.Set EAHB:0x531001E0 %Long 0x204 ; IOMUX_PR0_PRU1_GPO18_CFG_REG (TRC_CTL)
  Data.Set EAHB:0x5310019C %Long 0x204 ; IOMUX_PR0_PRU1_GPO5_CFG_REG (TRC_DATAD0)
  Data.Set EAHB:0x531001A0 %Long 0x204 ; IOMUX_PR0_PRU1_GPO9_CFG_REG (TRC_DATAD1)
  Data.Set EAHB:0x531001A4 %Long 0x204 ; IOMUX_PR0_PRU1_GPO10_CFG_REG (TRC_DATAD2)
  Data.Set EAHB:0x531001A8 %Long 0x204 ; IOMUX_PR0_PRU1_GPO8_CFG_REG (TRC_DATAD3)

  IF Analyzer()
  (
    Data.Set EAHB:0x531001AC %Long 0x204 ; IOMUX_PR0_PRU1_GPO6_CFG_REG (TRC_DATAD4)
    Data.Set EAHB:0x531001B0 %Long 0x204 ; IOMUX_PR0_PRU1_GPO4_CFG_REG (TRC_DATAD5)
    Data.Set EAHB:0x531001B4 %Long 0x204 ; IOMUX_PR0_PRU1_GPO0_CFG_REG (TRC_DATAD6)
    Data.Set EAHB:0x531001B8 %Long 0x204 ; IOMUX_PR0_PRU1_GPO1_CFG_REG (TRC_DATAD7)
    Data.Set EAHB:0x531001BC %Long 0x204 ; IOMUX_PR0_PRU1_GPO2_CFG_REG (TRC_DATAD8)
    Data.Set EAHB:0x531001C0 %Long 0x204 ; IOMUX_PR0_PRU1_GPO3_CFG_REG (TRC_DATAD9)
    Data.Set EAHB:0x531001C4 %Long 0x204 ; IOMUX_PR0_PRU1_GPO16_CFG_REG (TRC_DATAD10)
    Data.Set EAHB:0x531001C8 %Long 0x204 ; IOMUX_PR0_PRU1_GPO15_CFG_REG (TRC_DATAD11)
    Data.Set EAHB:0x531001CC %Long 0x204 ; IOMUX_PR0_PRU1_GPO11_CFG_REG (TRC_DATAD12)
    Data.Set EAHB:0x531001D0 %Long 0x204 ; IOMUX_PR0_PRU1_GPO12_CFG_REG (TRC_DATAD13)
    Data.Set EAHB:0x531001D4 %Long 0x204 ; IOMUX_PR0_PRU1_GPO13_CFG_REG (TRC_DATAD14)
    Data.Set EAHB:0x531001D8 %Long 0x204 ; IOMUX_PR0_PRU1_GPO14_CFG_REG (TRC_DATAD15)
  )
```

```
  ; Use I2C to control the GPIO expander (TCA6416) on the control card to route signals to the
docking station
  Data.Set EAHB:0x52502024 %Long 0x00004620 ; I2C2_ICMDR
  Data.Set EAHB:0x5250200C %Long 0x00000009 ; I2C2_ICCLKL
  Data.Set EAHB:0x52502010 %Long 0x00000009 ; I2C2_ICCLKH
  Data.Set EAHB:0x5250201C %Long 0x00000020 ; I2C2_ICSAR
  Data.Set EAHB:0x52502020 %Long 0x00000006 ; I2C2_ICDXR;
  Data.Set EAHB:0x52502024 %Long 0x00006e20 ; I2C2_ICMDR;
  Data.Set EAHB:0x52502020 %Long 0x00000003 ; I2C2_ICDXR;
  Data.Set EAHB:0x52502024 %Long 0x00006c20 ; I2C2_ICMDR

  IF Analyzer()
  (
    Trace.METHOD Analyzer
    TPIU.PortSize 16
  )
  ELSE
  (
    Trace.METHOD CAnalyzer
    TPIU.PortSize 4
  )
  TPIU.PortMode Continuous
  Trace.TERMination ON
  Trace.AutoFocus
)

; --------------------------------------------------------------------------
; 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
```

5.2 AM263Px CMM Script

AM263Px CMM Script

```
; --------------------------------------------------------------------------
; Copyright (C) 2023-2024 Texas Instruments Incorporated
;
; Redistribution and use in source and binary forms, with or without
; modification, are permitted provided that the following conditions
; are met:
;
; Redistributions of source code must retain the above copyright
; notice, this list of conditions and the following disclaimer.
;
; 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 provided with the
; distribution.
;
; Neither the name of Texas Instruments Incorporated nor the names of
; the contributors can 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
; OWNER 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.
; -------------------------------------------------------------------------
;
; -------------------------------------------------------------------------
; @Title: Simple demo script for Cortex-R5F on TMDSCNCD263P
; @Description:
;   Prompts the user to load application into RAM and sets up a demo debug
;   scenario for the first Cortex-R5F of the SS0 cluster.
;   In addition, configures the off-chip trace.
;   Prerequisites:
;    * Plug TMDSCNCD263P onto TI debug&trace adapter TMDSHSECDOCK-AM263
;    * Connect Debug Cable to J13 via adapter LA-3818
;    * SW1=OFF to deactivate onboard debugger and enable external JTAG debugger
;    * SW14=OFF, SW15=OFF, SW16=ON to have signals routed from PRU1 to HSEC
; @Keywords: ARM, Cortex-R5F
; @Board: TMDSCNCD263P
; @Chip: AM263P4
; @Copyright: 2023-2024 Texas Instruments Incorporated
; -------------------------------------------------------------------------

&sbl=TRUE() ; When running in OSPI Bootmode

RESet
SYStem.CPU AM263P4-SS0
CORE.ASSIGN 1.

IF &sbl==FALSE() ; In case of OSPI bootmode, SBL takes care of the below config
(
  ; -----------------------------------------------------------------------
  ; Attach to system bus and make some preparations
  SYStem.Mode PREPARE

  ; Unlock MSS_CTRL register
  Data.Set EAHB:0x50D01008 %Long 0x01234567 ; MSS_CTRL_LOCK0_KICK0
  Data.Set EAHB:0x50D0100c %Long 0x0fedcba8 ; MSS_CTRL_LOCK0_KICK1

  ; Eclipse ROM, use RAM in ATCM
  Data.Set EAHB:0x50D00080 %Long 0x00000007 ; MSS_CTRL_R5SS0_ROM_ECLIPSE

  ; Let core run
  Data.Set EAHB:0x50D00024 %Long 0x00000000 ; MSS_CTRL_R5SS0_CORE0_HALT
)

; -------------------------------------------------------------------------
; Attach to the cores
SYStem.Mode Attach
Break

IF &sbl==TRUE()
(
  Data.Set EAHB:0x50D01008 %Long 0x01234567 ; MSS_CTRL_LOCK0_KICK0
  Data.Set EAHB:0x50D0100c %Long 0x0fedcba8 ; MSS_CTRL_LOCK0_KICK1
)

; -------------------------------------------------------------------------
; Disable the MPU and the caches that have been enabled by the firmware (SCTLR)
Data.Set C15:0x1 %Long (Data.Long(C15:0x1)&(~0x1005))

; -------------------------------------------------------------------------
; Load demo program, opens up a file explorer. Navigate and select the application binary you wish
data.LOAD.Elf *

; -------------------------------------------------------------------------
; Configure off-chip trace
IF Analyzer()||CAnalyzer()
(
  ; Unlock TOP_RCM register
  Data.Set EAHB:0x53201008 %Long 0x01234567 ; TOP_RCM_LOCK0_KICK0
  Data.Set EAHB:0x5320100C %Long 0x0fedcba8 ; TOP_RCM_LOCK0_KICK1
  WAIT 1.MS

  IF &sbl==FALSE() ; Incase of OSPI Bootmode, SBL does the PLL clock configuration
  (
    ; Config core PLL
    Data.Set EAHB:0x53200410 %Long 0x00010009 ; TOP_RCM_PLL_CORE_M2NDIV
    Data.Set EAHB:0x53200414 %Long 0x00000320 ; TOP_RCM_PLL_CORE_MN2DIV
    Data.Set EAHB:0x53200408 %Long 0x00000001 ; TOP_RCM_PLL_CORE_TENABLE
```

```
    Data.Set EAHB:0x53200404 %Long 0x00095001 ; TOP_RCM_PLL_CORE_CLKCTRL
    Data.Set EAHB:0x53200430 %Long 0x00000103 ; TOP_RCM_PLL_CORE_HSDIVIDER_CLKOUT1
  )

  ; Select trace clock source and divider
  Data.Set EAHB:0x53200C20 %Long 0x00000222 ; TOP_RCM_TRCCLKOUT_CLK_SRC_SEL
  Data.Set EAHB:0x53200C24 %Long 0x00000222 ; TOP_RCM_TRCCLKOUT_DIV_VAL

  ; Unlock IOMUX register and configure IOs
  Data.Set EAHB:0x53100298 %Long 0x83e70b13 ; IOMUX_IO_CFG_KICK0
  Data.Set EAHB:0x5310029C %Long 0x95a4f1e0 ; IOMUX_IO_CFG_KICK1
  WAIT 1.MS

  Data.Set EAHB:0x53100064 %Long 0x00000501 ; IOMUX_UART0_RTSN_CFG_REG
  Data.Set EAHB:0x53100068 %Long 0x00000501 ; IOMUX_UART0_CTSN_CFG_REG

  Data.Set EAHB:0x531000B0 %Long 0x000007F7 ; IOMUX_EPWM0_B_CFG_REG (GPIO44 -> input + pull-high)
  Data.Set EAHB:0x531000BC %Long 0x000007F7 ; IOMUX_EPWM2_A_CFG_REG (GPIO47 -> input + pull-high)

  Data.Set EAHB:0x53100298 %Long 0x83e70b13 ; IOMUX_IO_CFG_KICK0
  Data.Set EAHB:0x5310029C %Long 0x95a4f1e0 ; IOMUX_IO_CFG_KICK1
  WAIT 1.MS

  Data.Set EAHB:0x531001DC %Long 0x004 ; IOMUX_PR0_PRU1_GPO19_CFG_REG (TRC_CLK)
  Data.Set EAHB:0x531001E0 %Long 0x204 ; IOMUX_PR0_PRU1_GPO18_CFG_REG (TRC_CTL)
  Data.Set EAHB:0x5310019C %Long 0x204 ; IOMUX_PR0_PRU1_GPO5_CFG_REG (TRC_DATAD0)
  Data.Set EAHB:0x531001A0 %Long 0x204 ; IOMUX_PR0_PRU1_GPO9_CFG_REG (TRC_DATAD1)
  Data.Set EAHB:0x531001A4 %Long 0x204 ; IOMUX_PR0_PRU1_GPO10_CFG_REG (TRC_DATAD2)
  Data.Set EAHB:0x531001A8 %Long 0x204 ; IOMUX_PR0_PRU1_GPO8_CFG_REG (TRC_DATAD3)

  IF Analyzer()
  (
    Data.Set EAHB:0x531001AC %Long 0x204 ; IOMUX_PR0_PRU1_GPO6_CFG_REG (TRC_DATAD4)
    Data.Set EAHB:0x531001B0 %Long 0x204 ; IOMUX_PR0_PRU1_GPO4_CFG_REG (TRC_DATAD5)
    Data.Set EAHB:0x531001B4 %Long 0x204 ; IOMUX_PR0_PRU1_GPO0_CFG_REG (TRC_DATAD6)
    Data.Set EAHB:0x531001B8 %Long 0x204 ; IOMUX_PR0_PRU1_GPO1_CFG_REG (TRC_DATAD7)
    Data.Set EAHB:0x531001BC %Long 0x204 ; IOMUX_PR0_PRU1_GPO2_CFG_REG (TRC_DATAD8)
    Data.Set EAHB:0x531001C0 %Long 0x204 ; IOMUX_PR0_PRU1_GPO3_CFG_REG (TRC_DATAD9)
    Data.Set EAHB:0x531001C4 %Long 0x204 ; IOMUX_PR0_PRU1_GPO16_CFG_REG (TRC_DATAD10)
    Data.Set EAHB:0x531001C8 %Long 0x204 ; IOMUX_PR0_PRU1_GPO15_CFG_REG (TRC_DATAD11)
    Data.Set EAHB:0x531001CC %Long 0x204 ; IOMUX_PR0_PRU1_GPO11_CFG_REG (TRC_DATAD12)
    Data.Set EAHB:0x531001D0 %Long 0x204 ; IOMUX_PR0_PRU1_GPO12_CFG_REG (TRC_DATAD13)
    Data.Set EAHB:0x531001D4 %Long 0x204 ; IOMUX_PR0_PRU1_GPO13_CFG_REG (TRC_DATAD14)
    Data.Set EAHB:0x531001D8 %Long 0x204 ; IOMUX_PR0_PRU1_GPO14_CFG_REG (TRC_DATAD15)
  )

  ; Use I2C to control the GPIO expander (TCA6424) on the control card to route signals to the
docking station
  Data.Set EAHB:0x52502024 %Long 0x00004620 ; I2C2_ICMDR
  Data.Set EAHB:0x5250200C %Long 0x00000009 ; I2C2_ICCLKL
  Data.Set EAHB:0x52502010 %Long 0x00000009 ; I2C2_ICCLKH
  Data.Set EAHB:0x5250201C %Long 0x00000022 ; I2C2_ICSAR
  Data.Set EAHB:0x52502020 %Long 0x0000000C ; I2C2_ICDXR
  Data.Set EAHB:0x52502024 %Long 0x00006e20 ; I2C2_ICMDR
  Data.Set EAHB:0x52502020 %Long 0x00000006 ; I2C2_ICDXR
  Data.Set EAHB:0x52502024 %Long 0x00006c20 ; I2C2_ICMDR

  IF Analyzer()
  (
    Trace.METHOD Analyzer
    TPIU.PortSize 16
  )
  ELSE
  (
    Trace.METHOD CAnalyzer
    TPIU.PortSize 4
  )
  TPIU.PortMode Continuous
  Trace.TERMination ON
  Trace.AutoFocus
)

; --------------------------------------------------------------------------
; 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
```

6 Flashing SBL Null

TI recommends that users have SBL Null flashed to the AM26x device.

6.1 Using UniFlash tool

To use the UniFlash tool to flash the SBL Null image to the device, follow the steps below.

1. Follow the steps in this video: TI Video.
2. Make sure to select the files as uart_uniflash and sbl_null for flashing.

6.2 Using Command Line Python Scripts

To use the SDK python scripts for flashing, follow the steps below.

1. Power off the device and switch to UART boot mode.
2. From the following folder: *mcu_plus_sdk/tools/boot* run the below commands in the command prompt. (Check the UART COM Port from the device manager).
3. *python uart_uniflash.py -p COM<your_port_number> --cfg=sbl_prebuilt/<device>/default_sbl_null.cfg*.
4. Confirm the successful flashing and execution of both the steps in the above script.
5. Power off the device and switch to OSPI/QSPI Bootmode based on the AM26x device.

For more detailed steps, refer to Flashing applications on AM26x devices.

7 Debugging with Trace32 Software

The Lauterbach hardware, AM26x device, and the application are prepared to be traced. To extract trace for an application, follow the steps below:

1. Launch the TRACE32 software.
2. From the T32Start screen, select the PowerView Instance and click on *Start*.

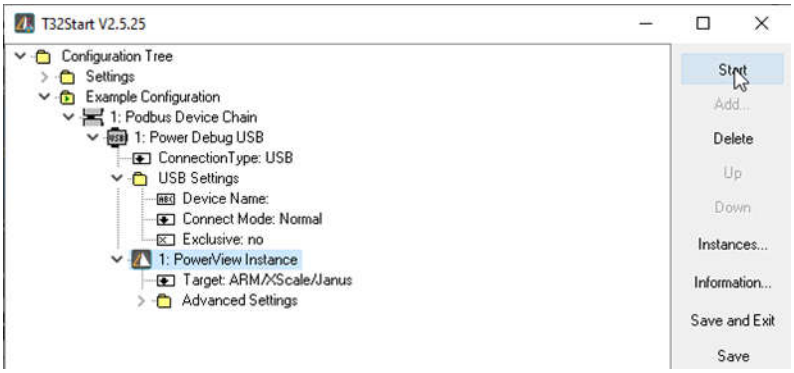

Figure 7-1. T32 Start Window

3. Set the device in OSPI, QSPI boot mode and power cycle your device (HSEC Dock board).
4. Click on File → Run Script → Select the .cmm script *am263px-trace.cmm*.
5. First, this file resets the CPU, attaches to the system bus and unlocks MSS_CTRL registers, then eclipse ROM and runs the R5F core. The file then prompts users to choose an application binary to debug. Browse the file explorer and select the file.
6. The script then configures trace clock source, IOMUX, and I2C to route signals to the docking station.
7. The script opens up windows to show the debug trace.

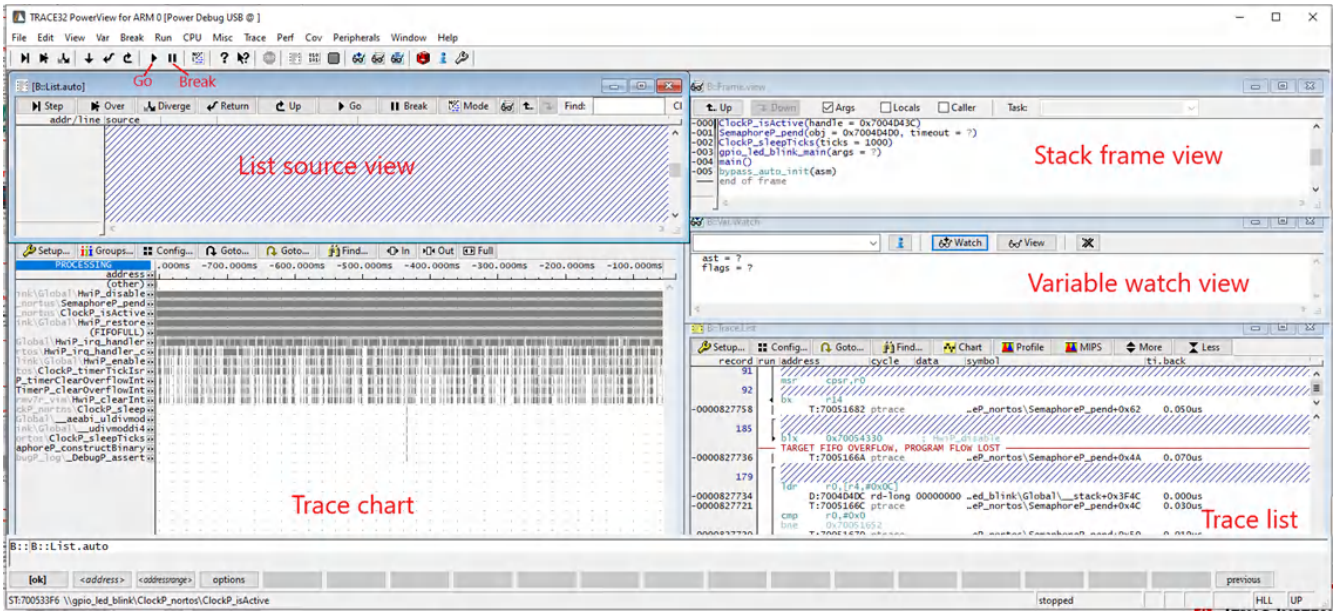

Figure 7-2. Trace Result Windows

See Figure 8-1 for the debug trace of GPIO Led blink example on the AM263Px controlCard.

8 Summary

This document provides a step-by-step guide to enable Lauterbach ETM Trace for Texas Instruments high-performance AM26x micro controllers. By following the hardware and software setup steps above and using the .cmm scripts with the Trace32 software, Lauterbach trace is enabled on the AM26x microcontroller. Figure 8-1 shows the trace for the GPIO LED blink example from the MCU_PLUS_SDK, running on the R5F Core-0, on the AM263P4 in OSPI boot mode.

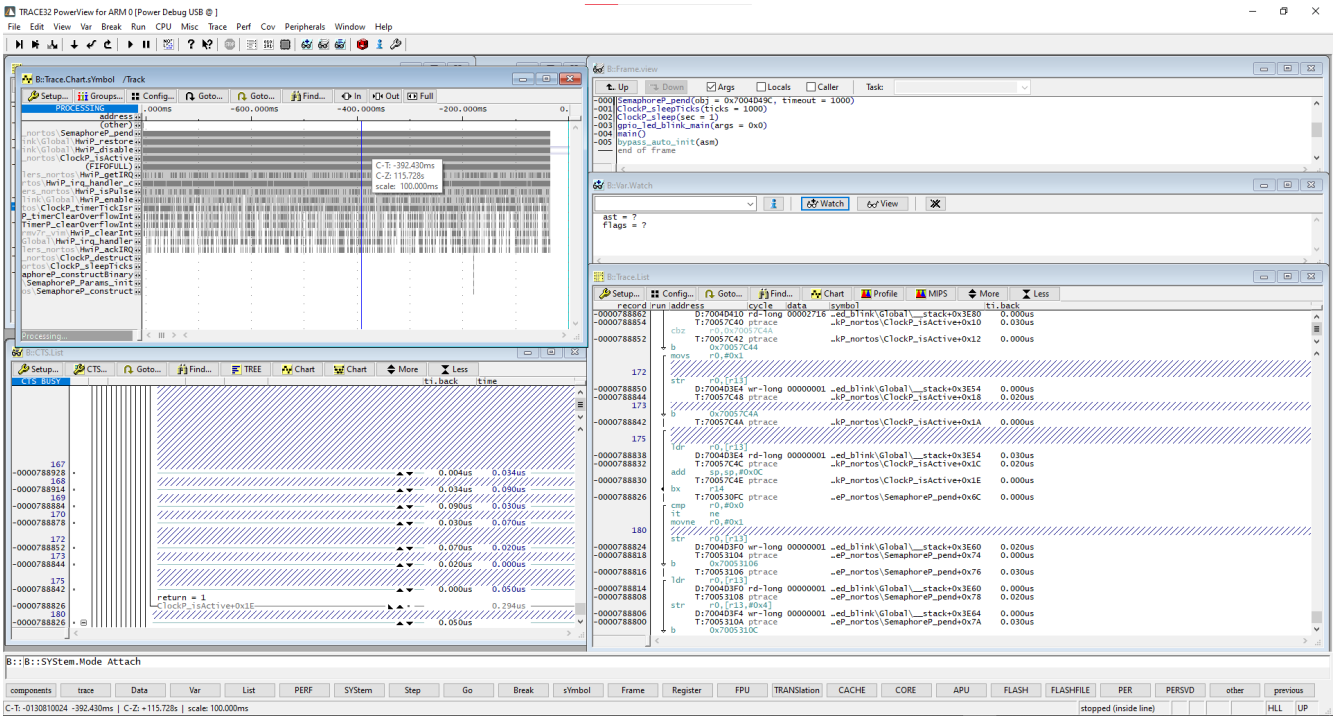

Figure 8-1. Sample Trace Output

9 References

- Lauterbach AM2634 support, Official Lauterbach support for AM2634 devices, webpage
- Lauterbach, Official Lauterbach error messages guide, error guide
- Lauterbach, OS awareness manual FreeRTOS, FreeRTOS manual
- Texas Instruments, Official MCU_PLUS_SDK documentation for AM263x devices, AM263x MCU_PLUS_SDK
- Texas Instruments, Official MCU_PLUS_SDK documentation for AM263Px devices, AM263Px_MCU_PLUS_SDK
- Texas Instruments, AM26x Academy to learn more and get started with development, TI AM26x Academy
- Texas Instruments, Collection of AM26x video trainings, AM26x video trainings

IMPORTANT NOTICE AND DISCLAIMER

TI PROVIDES TECHNICAL AND RELIABILITY DATA (INCLUDING DATA SHEETS), DESIGN RESOURCES (INCLUDING REFERENCE DESIGNS), APPLICATION OR OTHER DESIGN ADVICE, WEB TOOLS, SAFETY INFORMATION, AND OTHER RESOURCES “AS IS” AND WITH ALL FAULTS, AND DISCLAIMS ALL WARRANTIES, EXPRESS AND IMPLIED, INCLUDING WITHOUT LIMITATION ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT OF THIRD PARTY INTELLECTUAL PROPERTY RIGHTS.

These resources are intended for skilled developers designing with TI products. You are solely responsible for (1) selecting the appropriate TI products for your application, (2) designing, validating and testing your application, and (3) ensuring your application meets applicable standards, and any other safety, security, regulatory or other requirements.

These resources are subject to change without notice. TI grants you permission to use these resources only for development of an application that uses the TI products described in the resource. Other reproduction and display of these resources is prohibited. No license is granted to any other TI intellectual property right or to any third party intellectual property right. TI disclaims responsibility for, and you will fully indemnify TI and its representatives against, any claims, damages, costs, losses, and liabilities arising out of your use of these resources.

TI’s products are provided subject to TI’s Terms of Sale or other applicable terms available either on ti.com or provided in conjunction with such TI products. TI’s provision of these resources does not expand or otherwise alter TI’s applicable warranties or warranty disclaimers for TI products.

TI objects to and rejects any additional or different terms you may have proposed.

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265