STM32MP1 Platform trace and debug environment overview for Android

Contents

Stable: 06.04.2021 - 08:02 / Revision: 29.03.2021 - 20:38

A quality version of this page, approved on *12 December 2019*, was based off this revision.

The block diagram below shows the **STM32MP1 Platform trace and debug environment for Android** components and their possible interfaces. Click the block diagram to directly jump to one of the sub-levels listed below:

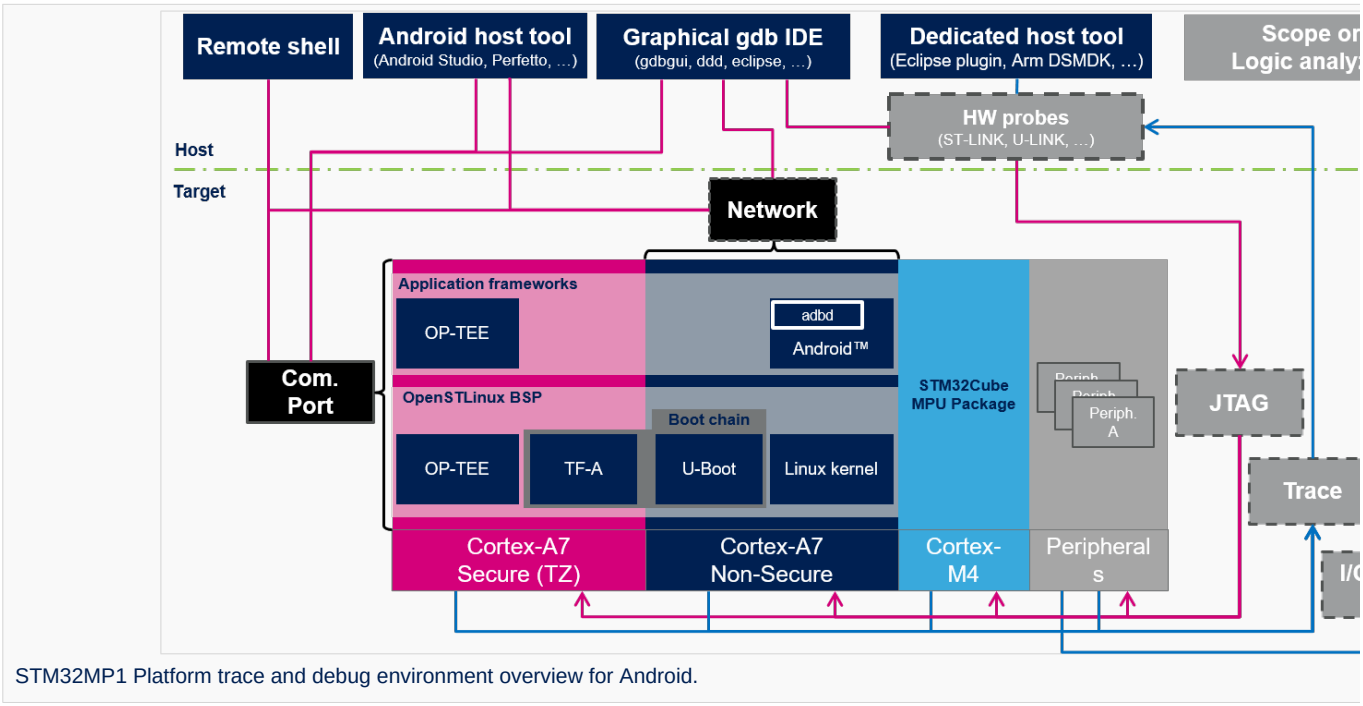
- The **STM32MPU Embedded Software** package (see STM32MPU Embedded Software for Android architecture overview) that comprises:
  - the **STM32MPU distribution for Android™** running on the Arm®Cortex®-A and including:
    - the **OpenSTLinux BSP** with:
      - the boot chain based on TF-A and U-Boot.
      - the OP-TEE secure OS running on the Arm®Cortex®-A core in Secure mode.
      - the **Linux® kernel** running on the Arm®Cortex®-A core in Non-secure mode.
    - the **application frameworks** composed of middleware components relying on the BSP and providing:
      - **OP-TEE** APIs to run **Trusted Applications (TA)** that allow manipulating secrets (not visible from the Linux® and STM32Cube MPU Package).
      - **Android** APIs to run **applications** that typically interact with the user via a display or a touchscreen.
  - the **STM32Cube MPU Package** runs on the Arm®Cortex®-M core: like other STM32 microcontrollers, it is based on HAL drivers and middleware components. Iit is completed with the coprocessor management.
- The **STM32MPU peripherals** shared between Cortex®-A and Cortex®-M cores (such as GPIO, I2C and SPI).
- The **user interfaces or tools**, which allow interacting with different trace and debug Tools, such as:
  - The **remote shell** using terminal console
  - The **Android host tools** (such as Android Studio)
  - The **debugger tools** (such as GDB)
  - The **graphical IDE** (such as GDBGUI or SystemWorkbench)
- The **trace and debug interfaces or hardware paths** that provide access to trace and debug components through:
  - the **network** interface (e.g. Ethernet)
  - the **communication port** (e.g UART)
  - the hardware connector interfaces:
    - **JTag** port
    - **Trace** port to access ETM, STM, ITM and SWD
    - **I/O probes** to access HDP
- The **hardware probes** (such as ST-Link).

This block diagram also illustrates the Arm® debugging modes:

- **Invasive debug**: debug process that allows controlling and monitoring the processor. Most debug features are considered invasive because they enable you to halt the processor and modify its state.
- **Non-invasive debug**: debug process that allows monitoring the processor but not controlling it. The embedded trace macrocell (ETM) interface and the performance monitor registers are non-invasive debug features.

Click the figure below to directly jump to the component you want to trace, monitor or debug:

- Select a **hardware component** to be redirected to the corresponding hardware board article and check if the hardware connector is supported on your board.
- Select a **target software component** to be redirected to an article that explains in details how to trace, monitor or debug the corresponding component.
- Select a **host software component** to be redirected to an article that explains how to use the corresponding remote tool.



STM32MP1 Platform trace and debug environment overview for Android.

Arm® is a registered trademark of Arm Limited (or its subsidiaries) in the US and/or elsewhere.

Cortex®

Board support package

Operating System

Linux® is a registered trademark of Linus Torvalds.

Open Portable Trusted Execution Environment

Trusted Application

Microprocessor Unit

Hardware Abstraction Layer

General-Purpose Input/Output (A realization of open ended transmission between devices on an embedded level. These pins available on a processor can be programmed to be used to either accept input or provide output to external devices depending on user desires and applications requirements.)

Inter-Integrated Circuit (Bi-directional 2-wire bus standard for efficient inter-IC control.)

Serial Peripheral Interface

GNU dedugger, a portable debugger that runs on many Unix-like systems

(Software)Integrated development/design/debugging environment

Universal Asynchronous Receiver/Transmitter

Embedded Trace Macrocell

System Trace Module

Instruction Trace Macrocell

Serial Wire Debug

Hardware Debug Port

spelling for older versions of STLink, ST in-circuit debugger and programmer for the STM8 and STM32 microcontroller families

Stable: 25.09.2020 - 08:36 / Revision: 25.09.2020 - 08:35

The block diagram below shows the **STM32MP1 Platform trace and debug environment for Android** components and their possible interfaces. Click the block diagram to directly jump to one of the sub-levels listed below:

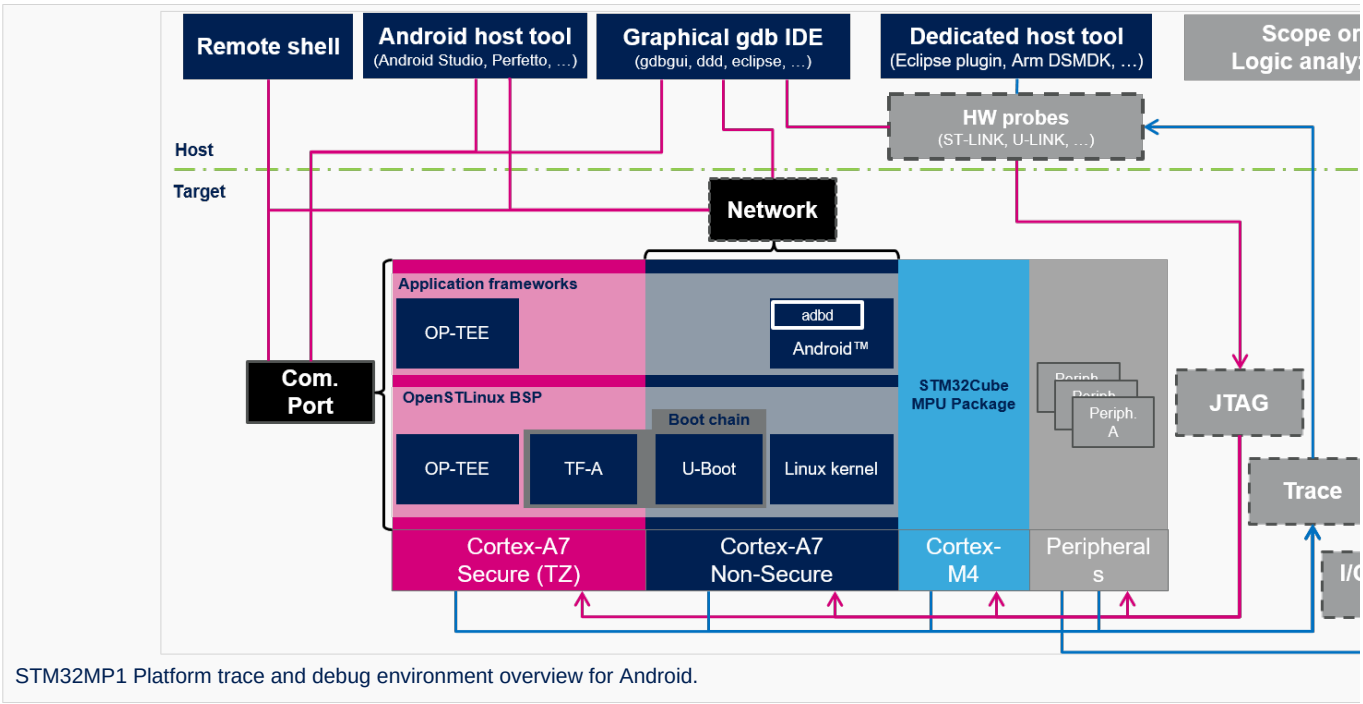
- The **STM32MPU Embedded Software** package (see STM32MPU Embedded Software for Android architecture overview) that comprises:
  - the **STM32MPU distribution for Android™** running on the Arm®Cortex®-A and including:
    - the **OpenSTLinux BSP** with:
      - the boot chain based on TF-A and U-Boot.
      - the OP-TEE secure OS running on the Arm®Cortex®-A core in Secure mode.
      - the **Linux® kernel** running on the Arm®Cortex®-A core in Non-secure mode.
    - the **application frameworks** composed of middleware components relying on the BSP and providing:
      - **OP-TEE** APIs to run **Trusted Applications (TA)** that allow manipulating secrets (not visible from the Linux® and STM32Cube MPU Package).
      - **Android** APIs to run **applications** that typically interact with the user via a display or a touchscreen.
  - the **STM32Cube MPU Package** runs on the Arm®Cortex®-M core: like other STM32 microcontrollers, it is based on HAL drivers and middleware components. Iit is completed with the coprocessor management.
- The **STM32MPU peripherals** shared between Cortex®-A and Cortex®-M cores (such as GPIO, I2C and SPI).
- The **user interfaces or tools**, which allow interacting with different trace and debug Tools, such as:
  - The **remote shell** using terminal console
  - The **Android host tools** (such as Android Studio)
  - The **debugger tools** (such as GDB)
  - The **graphical IDE** (such as GDBGUI or SystemWorkbench)
- The **trace and debug interfaces or hardware paths** that provide access to trace and debug components through:
  - the **network** interface (e.g. Ethernet)
  - the **communication port** (e.g UART)
  - the hardware connector interfaces:
    - **JTag** port
    - **Trace** port to access ETM, STM, ITM and SWD
    - **I/O probes** to access HDP
- The **hardware probes** (such as ST-Link).

This block diagram also illustrates the Arm® debugging modes:

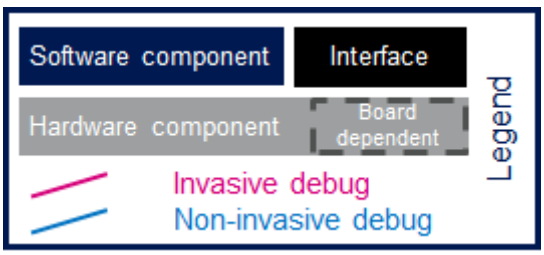
- **Invasive debug**: debug process that allows controlling and monitoring the processor. Most debug features are considered invasive because they enable you to halt the processor and modify its state.
- **Non-invasive debug**: debug process that allows monitoring the processor but not controlling it. The embedded trace macrocell (ETM) interface and the performance monitor registers are non-invasive debug features.

Click the figure below to directly jump to the component you want to trace, monitor or debug:

- Select a **hardware component** to be redirected to the corresponding hardware board article and check if the hardware connector is supported on your board.
- Select a **target software component** to be redirected to an article that explains in details how to trace, monitor or debug the corresponding component.
- Select a **host software component** to be redirected to an article that explains how to use the corresponding remote tool.



STM32MP1 Platform trace and debug environment overview for Android.



Cortex®

Board support package

Operating System

Linux® is a registered trademark of Linus Torvalds.

Open Portable Trusted Execution Environment

Trusted Application

Microprocessor Unit

Hardware Abstraction Layer

General-Purpose Input/Output (A realization of open ended transmission between devices on an embedded level. These pins available on a processor can be programmed to be used to either accept input or provide output to external devices depending on user desires and applications requirements.)

Inter-Integrated Circuit (Bi-directional 2-wire bus standard for efficient inter-IC control.)

Serial Peripheral Interface

GNU dedugger, a portable debugger that runs on many Unix-like systems

(Software)Integrated development/design/debugging environment

Universal Asynchronous Receiver/Transmitter

Embedded Trace Macrocell

System Trace Module

Instruction Trace Macrocell

Serial Wire Debug

Hardware Debug Port

spelling for older versions of STLink, ST in-circuit debugger and programmer for the STM8 and STM32 microcontroller families

Stable: 08.03.2021 - 16:07 / Revision: 16.02.2021 - 17:01

The block diagram below shows the **STM32MP1 Platform trace and debug environment for Android** components and their possible interfaces. Click the block diagram to directly jump to one of the sub-levels listed below:

- The **STM32MPU Embedded Software** package (see STM32MPU Embedded Software for Android architecture overview) that comprises:
  - the **STM32MPU distribution for Android™** running on the Arm®Cortex®-A and including:
    - the **OpenSTLinux BSP** with:
      - the boot chain based on TF-A and U-Boot.
      - the OP-TEE secure OS running on the Arm®Cortex®-A core in Secure mode.
      - the **Linux® kernel** running on the Arm®Cortex®-A core in Non-secure mode.
    - the **application frameworks** composed of middleware components relying on the BSP and providing:
      - **OP-TEE** APIs to run **Trusted Applications (TA)** that allow manipulating secrets (not visible from the Linux® and STM32Cube MPU Package).
      - **Android** APIs to run **applications** that typically interact with the user via a display or a touchscreen.
  - the **STM32Cube MPU Package** runs on the Arm®Cortex®-M core: like other STM32 microcontrollers, it is based on HAL drivers and middleware components. Iit is completed with the coprocessor management.
- The **STM32MPU peripherals** shared between Cortex®-A and Cortex®-M cores (such as GPIO, I2C and SPI).
- The **user interfaces or tools**, which allow interacting with different trace and debug Tools, such as:
  - The **remote shell** using terminal console
  - The **Android host tools** (such as Android Studio)

- The **debugger tools** (such as GDB)
- The **graphical IDE** (such as GDBGUI or SystemWorkbench)

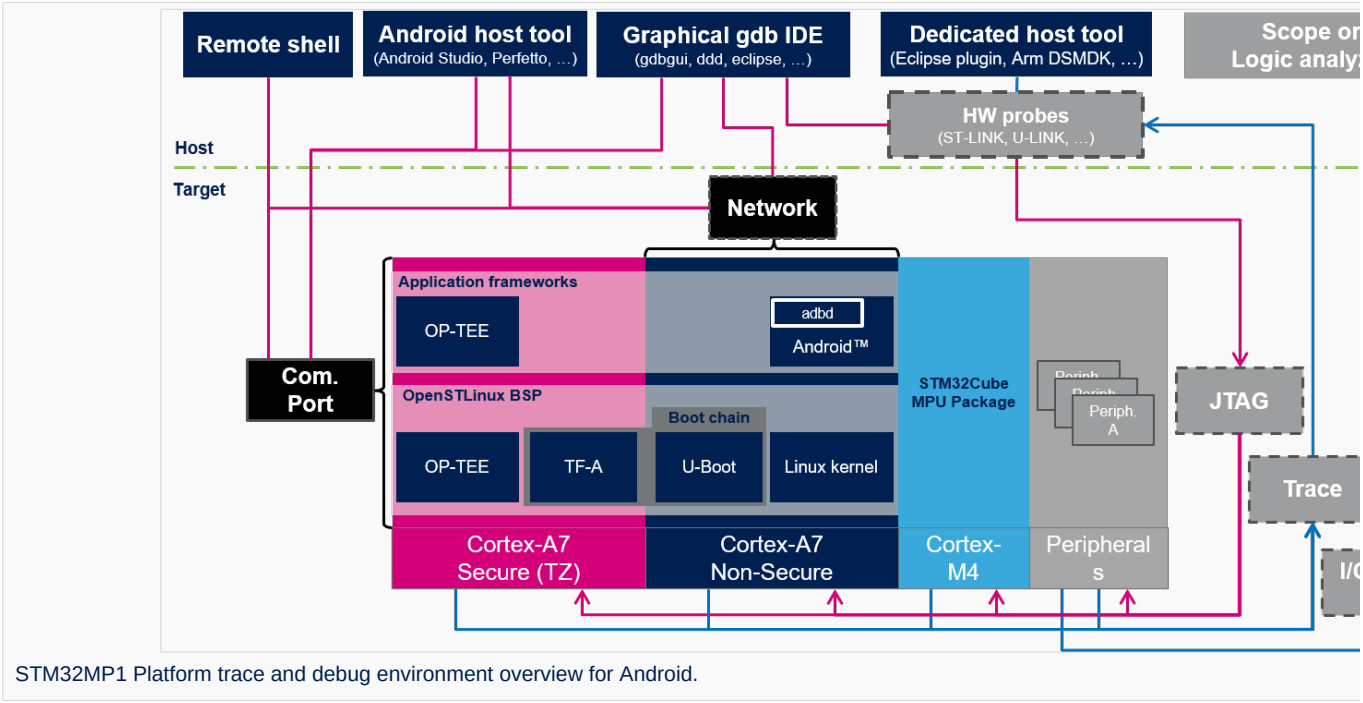
- The **trace and debug interfaces or hardware paths** that provide access to trace and debug components through:
  - the **network** interface (e.g. Ethernet)
  - the **communication port** (e.g UART)
  - the hardware connector interfaces:
    - **JTag** port
    - **Trace** port to access ETM, STM, ITM and SWD
    - **I/O probes** to access HDP
- The **hardware probes** (such as ST-Link).

This block diagram also illustrates the Arm® debugging modes:

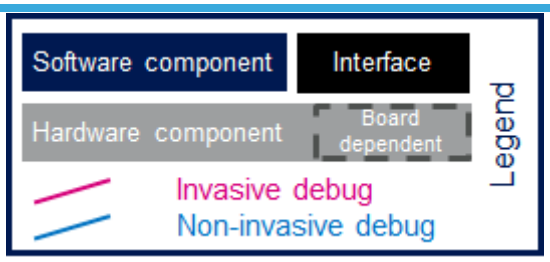
- **Invasive debug**: debug process that allows controlling and monitoring the processor. Most debug features are considered invasive because they enable you to halt the processor and modify its state.
- **Non-invasive debug**: debug process that allows monitoring the processor but not controlling it. The embedded trace macrocell (ETM) interface and the performance monitor registers are non-invasive debug features.

Click the figure below to directly jump to the component you want to trace, monitor or debug:

- Select a **hardware component** to be redirected to the corresponding hardware board article and check if the hardware connector is supported on your board.
- Select a **target software component** to be redirected to an article that explains in details how to trace, monitor or debug the corresponding component.
- Select a **host software component** to be redirected to an article that explains how to use the corresponding remote tool.



STM32MP1 Platform trace and debug environment overview for Android.



Cortex®

Board support package

Operating System

Linux® is a registered trademark of Linus Torvalds.

Open Portable Trusted Execution Environment

Trusted Application

Microprocessor Unit

Hardware Abstraction Layer

General-Purpose Input/Output (A realization of open ended transmission between devices on an embedded level. These pins available on a processor can be programmed to be used to either accept input or provide output to external devices depending on user desires and applications requirements.)

Inter-Integrated Circuit (Bi-directional 2-wire bus standard for efficient inter-IC control.)

Serial Peripheral Interface

GNU dedugger, a portable debugger that runs on many Unix-like systems

(Software)Integrated development/design/debugging environment

Universal Asynchronous Receiver/Transmitter

Embedded Trace Macrocell

System Trace Module

Instruction Trace Macrocell

Serial Wire Debug

Hardware Debug Port

spelling for older versions of STLink, ST in-circuit debugger and programmer for the STM8 and STM32 microcontroller families

Stable: 13.05.2020 - 08:56 / Revision: 13.05.2020 - 08:54

The block diagram below shows the **STM32MP1 Platform trace and debug environment for Android** components and their possible interfaces. Click the block diagram to directly jump to one of the sub-levels listed below:

- The **STM32MPU Embedded Software** package (see STM32MPU Embedded Software for Android architecture overview) that comprises:
  - the **STM32MPU distribution for Android™** running on the Arm®Cortex®-A and including:
    - the **OpenSTLinux BSP** with:
      - the boot chain based on TF-A and U-Boot.

- the OP-TEE secure OS running on the Arm®Cortex®-A core in Secure mode.
- the **Linux® kernel** running on the Arm®Cortex®-A core in Non-secure mode.

- the **application frameworks** composed of middleware components relying on the BSP and providing:
  - **OP-TEE** APIs to run **Trusted Applications (TA)** that allow manipulating secrets (not visible from the Linux® and STM32Cube MPU Package).
  - **Android** APIs to run **applications** that typically interact with the user via a display or a touchscreen.
- the **STM32Cube MPU Package** runs on the Arm®Cortex®-M core: like other STM32 microcontrollers, it is based on HAL drivers and middleware components. Iit is completed with the coprocessor management.

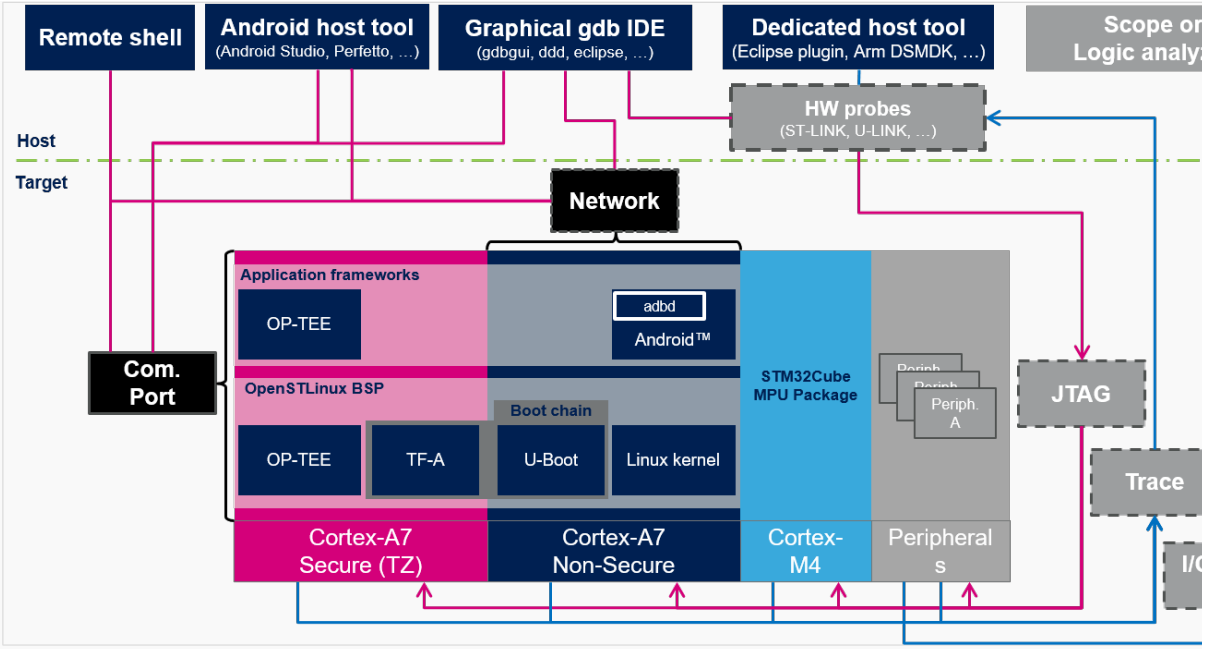
- The **STM32MPU peripherals** shared between Cortex®-A and Cortex®-M cores (such as GPIO, I2C and SPI).
- The **user interfaces or tools**, which allow interacting with different trace and debug Tools, such as:
  - The **remote shell** using terminal console
  - The **Android host tools** (such as Android Studio)
  - The **debugger tools** (such as GDB)
  - The **graphical IDE** (such as GDBGUI or SystemWorkbench)
- The **trace and debug interfaces or hardware paths** that provide access to trace and debug components through:
  - the **network** interface (e.g. Ethernet)
  - the **communication port** (e.g UART)
  - the hardware connector interfaces:
    - **JTag** port
    - **Trace** port to access ETM, STM, ITM and SWD
    - **I/O probes** to access HDP
- The **hardware probes** (such as ST-Link).

This block diagram also illustrates the Arm® debugging modes:

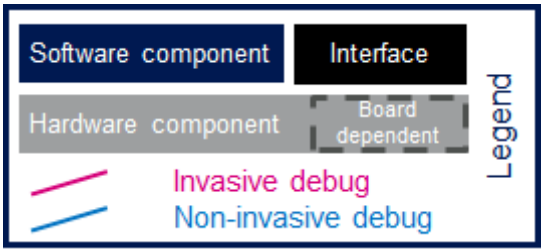
- **Invasive debug**: debug process that allows controlling and monitoring the processor. Most debug features are considered invasive because they enable you to halt the processor and modify its state.
- **Non-invasive debug**: debug process that allows monitoring the processor but not controlling it. The embedded trace macrocell (ETM) interface and the performance monitor registers are non-invasive debug features.

Click the figure below to directly jump to the component you want to trace, monitor or debug:

- Select a **hardware component** to be redirected to the corresponding hardware board article and check if the hardware connector is supported on your board.
- Select a **target software component** to be redirected to an article that explains in details how to trace, monitor or debug the corresponding component.
- Select a **host software component** to be redirected to an article that explains how to use the corresponding remote tool.



STM32MP1 Platform trace and debug environment overview for Android.



Cortex®

Board support package

Operating System

Linux® is a registered trademark of Linus Torvalds.

Open Portable Trusted Execution Environment

Trusted Application

Microprocessor Unit

Hardware Abstraction Layer

General-Purpose Input/Output (A realization of open ended transmission between devices on an embedded level. These pins available on a processor can be programmed to be used to either accept input or provide output to external devices depending on user desires and applications requirements.)

Inter-Integrated Circuit (Bi-directional 2-wire bus standard for efficient inter-IC control.)

Serial Peripheral Interface

GNU dedugger, a portable debugger that runs on many Unix-like systems

(Software)Integrated development/design/debugging environment

Universal Asynchronous Receiver/Transmitter

Embedded Trace Macrocell

System Trace Module

Instruction Trace Macrocell

Serial Wire Debug

Hardware Debug Port

spelling for older versions of STLink, ST in-circuit debugger and programmer for the STM8 and STM32 microcontroller families

Stable: 16.03.2021 - 16:12 / Revision: 11.03.2021 - 16:27

The block diagram below shows the **STM32MP1 Platform trace and debug environment for Android** components and their possible interfaces. Click the block diagram to directly jump to one of the sub-levels listed below:

- The **STM32MPU Embedded Software** package (see STM32MPU Embedded Software for Android architecture overview) that comprises:
  - the **STM32MPU distribution for Android™** running on the Arm®Cortex®-A and including:
    - the **OpenSTLinux BSP** with:
      - the boot chain based on TF-A and U-Boot.
      - the OP-TEE secure OS running on the Arm®Cortex®-A core in Secure mode.
      - the **Linux® kernel** running on the Arm®Cortex®-A core in Non-secure mode.
    - the **application frameworks** composed of middleware components relying on the BSP and providing:
      - **OP-TEE** APIs to run **Trusted Applications (TA)** that allow manipulating secrets (not visible from the Linux® and STM32Cube MPU Package).
      - **Android** APIs to run **applications** that typically interact with the user via a display or a touchscreen.
  - the **STM32Cube MPU Package** runs on the Arm®Cortex®-M core: like other STM32 microcontrollers, it is based on HAL drivers and middleware components. Iit is completed with the coprocessor management.
- The **STM32MPU peripherals** shared between Cortex®-A and Cortex®-M cores (such as GPIO, I2C and SPI).
- The **user interfaces or tools**, which allow interacting with different trace and debug Tools, such as:
  - The **remote shell** using terminal console
  - The **Android host tools** (such as Android Studio)
  - The **debugger tools** (such as GDB)
  - The **graphical IDE** (such as GDBGUI or SystemWorkbench)
- The **trace and debug interfaces or hardware paths** that provide access to trace and debug components through:
  - the **network** interface (e.g. Ethernet)
  - the **communication port** (e.g UART)
  - the hardware connector interfaces:
    - **JTag** port
    - **Trace** port to access ETM, STM, ITM and SWD
    - **I/O probes** to access HDP
- The **hardware probes** (such as ST-Link).

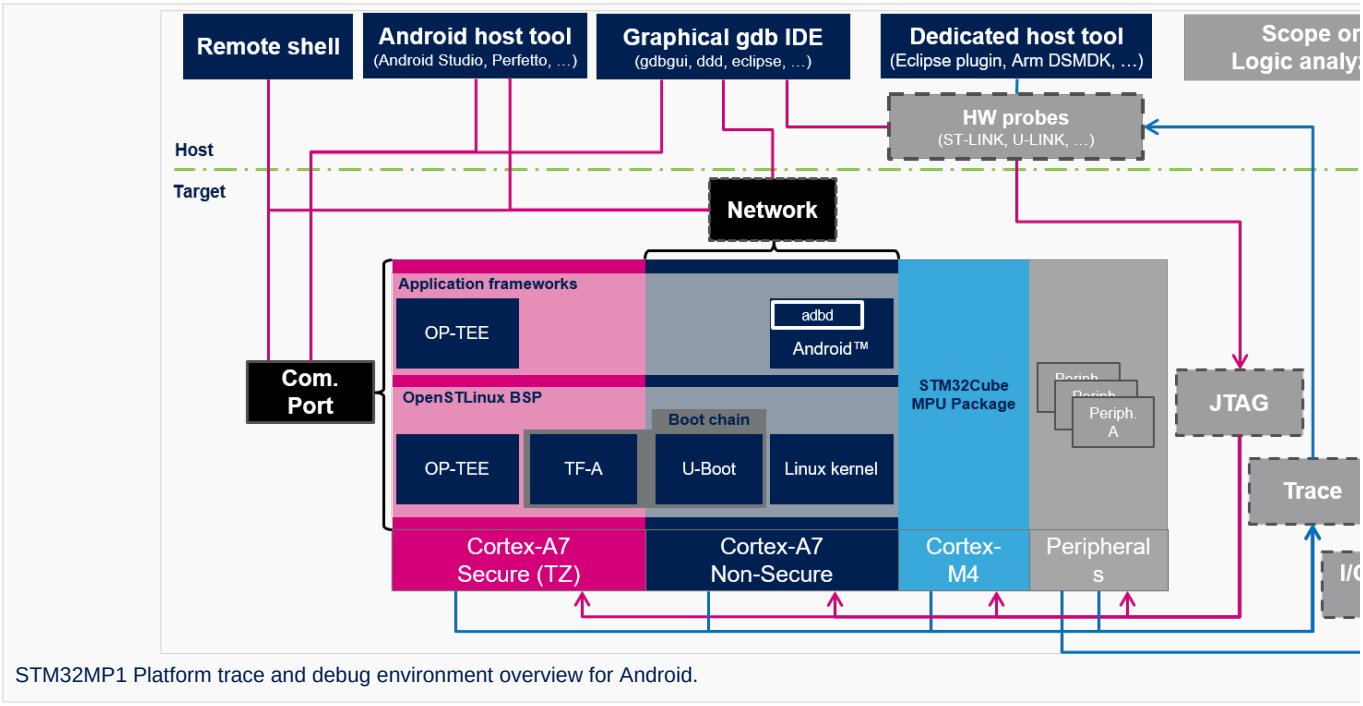
This block diagram also illustrates the Arm® debugging modes:

- **Invasive debug**: debug process that allows controlling and monitoring the processor. Most debug features are considered invasive because they enable you to halt the processor and modify its state.

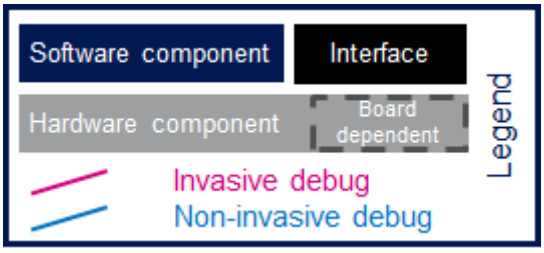
- **Non-invasive debug**: debug process that allows monitoring the processor but not controlling it. The embedded trace macrocell (ETM) interface and the performance monitor registers are non-invasive debug features.

Click the figure below to directly jump to the component you want to trace, monitor or debug:

- Select a **hardware component** to be redirected to the corresponding hardware board article and check if the hardware connector is supported on your board.
- Select a **target software component** to be redirected to an article that explains in details how to trace, monitor or debug the corresponding component.
- Select a **host software component** to be redirected to an article that explains how to use the corresponding remote tool.



STM32MP1 Platform trace and debug environment overview for Android.



Cortex®

Board support package

Operating System

Linux® is a registered trademark of Linus Torvalds.

Open Portable Trusted Execution Environment

Trusted Application

Microprocessor Unit

Hardware Abstraction Layer

General-Purpose Input/Output (A realization of open ended transmission between devices on an embedded level. These pins available on a processor can be programmed to be used to either accept input or provide output to external devices depending on user desires and applications requirements.)

Inter-Integrated Circuit (Bi-directional 2-wire bus standard for efficient inter-IC control.)

Serial Peripheral Interface

GNU dedugger, a portable debugger that runs on many Unix-like systems

(Software)Integrated development/design/debugging environment

Universal Asynchronous Receiver/Transmitter

Embedded Trace Macrocell

System Trace Module

Instruction Trace Macrocell

Serial Wire Debug

Hardware Debug Port

spelling for older versions of STLink, ST in-circuit debugger and programmer for the STM8 and STM32 microcontroller families

Stable: 22.04.2021 - 11:23 / Revision: 09.04.2021 - 13:17

The block diagram below shows the **STM32MP1 Platform trace and debug environment for Android** components and their possible interfaces. Click the block diagram to directly jump to one of the sub-levels listed below:

- The **STM32MPU Embedded Software** package (see STM32MPU Embedded Software for Android architecture overview) that comprises:
  - the **STM32MPU distribution for Android™** running on the Arm®Cortex®-A and including:
    - the **OpenSTLinux BSP** with:
      - the boot chain based on TF-A and U-Boot.
      - the OP-TEE secure OS running on the Arm®Cortex®-A core in Secure mode.
      - the **Linux® kernel** running on the Arm®Cortex®-A core in Non-secure mode.
    - the **application frameworks** composed of middleware components relying on the BSP and providing:
      - **OP-TEE** APIs to run **Trusted Applications (TA)** that allow manipulating secrets (not visible from the Linux® and STM32Cube MPU Package).
      - **Android** APIs to run **applications** that typically interact with the user via a display or a touchscreen.
  - the **STM32Cube MPU Package** runs on the Arm®Cortex®-M core: like other STM32 microcontrollers, it is based on HAL drivers and middleware components. Iit is completed with the coprocessor management.
- The **STM32MPU peripherals** shared between Cortex®-A and Cortex®-M cores (such as GPIO, I2C and SPI).
- The **user interfaces or tools**, which allow interacting with different trace and debug Tools, such as:
  - The **remote shell** using terminal console
  - The **Android host tools** (such as Android Studio)
  - The **debugger tools** (such as GDB)
  - The **graphical IDE** (such as GDBGUI or SystemWorkbench)
- The **trace and debug interfaces or hardware paths** that provide access to trace and debug components through:
  - the **network** interface (e.g. Ethernet)

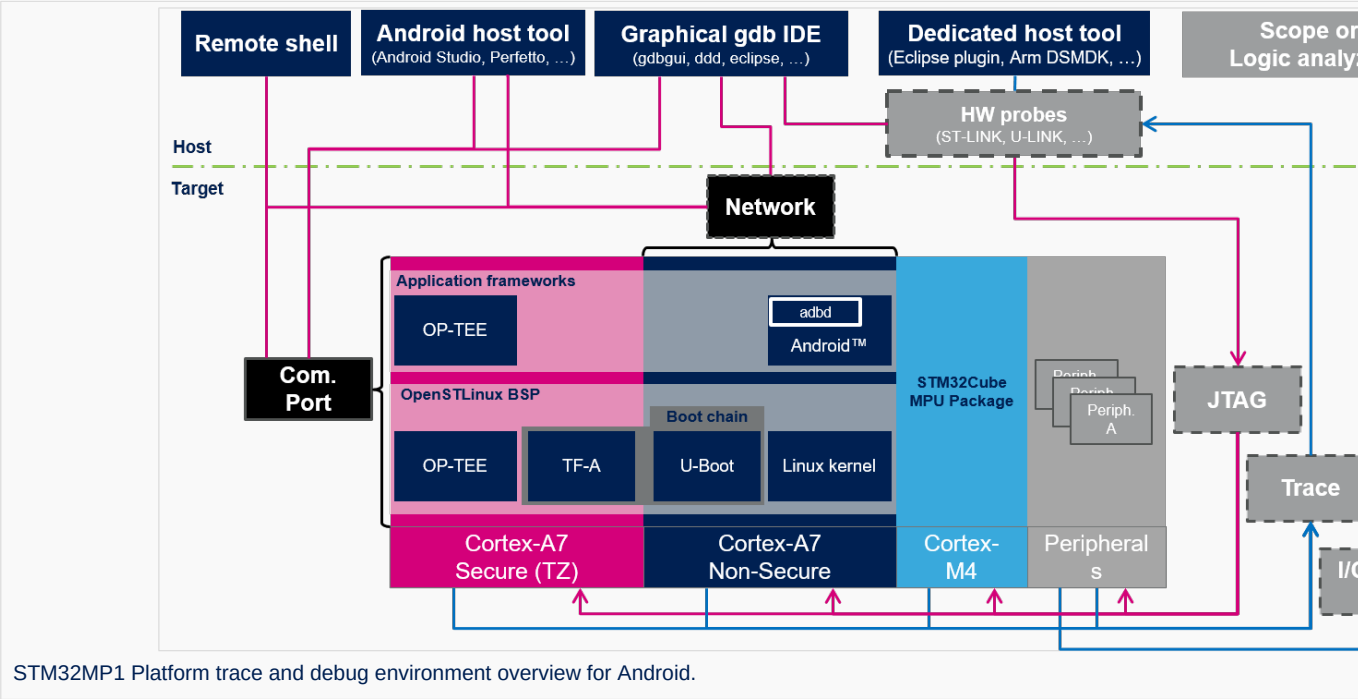
- the **communication port** (e.g UART)
- the hardware connector interfaces:
  - **JTag** port
  - **Trace** port to access ETM, STM, ITM and SWD
  - **I/O probes** to access HDP
- The **hardware probes** (such as ST-Link).

This block diagram also illustrates the Arm® debugging modes:

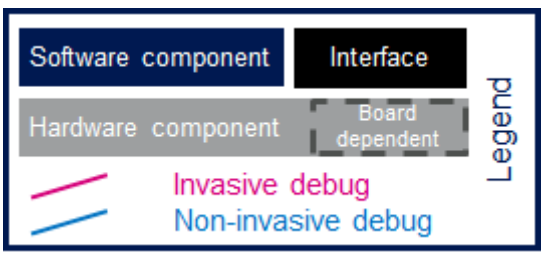
- **Invasive debug**: debug process that allows controlling and monitoring the processor. Most debug features are considered invasive because they enable you to halt the processor and modify its state.
- **Non-invasive debug**: debug process that allows monitoring the processor but not controlling it. The embedded trace macrocell (ETM) interface and the performance monitor registers are non-invasive debug features.

Click the figure below to directly jump to the component you want to trace, monitor or debug:

- Select a **hardware component** to be redirected to the corresponding hardware board article and check if the hardware connector is supported on your board.
- Select a **target software component** to be redirected to an article that explains in details how to trace, monitor or debug the corresponding component.
- Select a **host software component** to be redirected to an article that explains how to use the corresponding remote tool.



STM32MP1 Platform trace and debug environment overview for Android.



Arm® is a registered trademark of Arm Limited (or its subsidiaries) in the US and/or elsewhere.

Cortex®

Board support package

Operating System

Linux® is a registered trademark of Linus Torvalds.

Open Portable Trusted Execution Environment

Trusted Application

Microprocessor Unit

Hardware Abstraction Layer

General-Purpose Input/Output (A realization of open ended transmission between devices on an embedded level. These pins available on a processor can be programmed to be used to either accept input or provide output to external devices depending on user desires and applications requirements.)

Inter-Integrated Circuit (Bi-directional 2-wire bus standard for efficient inter-IC control.)

Serial Peripheral Interface

GNU dedugger, a portable debugger that runs on many Unix-like systems

(Software)Integrated development/design/debugging environment

Universal Asynchronous Receiver/Transmitter

Embedded Trace Macrocell

System Trace Module

Instruction Trace Macrocell

Serial Wire Debug

Hardware Debug Port

spelling for older versions of STLink, ST in-circuit debugger and programmer for the STM8 and STM32 microcontroller families

Stable: 19.10.2021 - 13:54 / Revision: 19.10.2021 - 13:54

The block diagram below shows the **STM32MP1 Platform trace and debug environment for Android** components and their possible interfaces. Click the block diagram to directly jump to one of the sub-levels listed below:

- The **STM32MPU Embedded Software** package (see STM32MPU Embedded Software for Android architecture overview) that comprises:
  - the **STM32MPU distribution for Android™** running on the Arm®Cortex®-A and including:
    - the **OpenSTLinux BSP** with:
      - the boot chain based on TF-A and U-Boot.
      - the OP-TEE secure OS running on the Arm®Cortex®-A core in Secure mode.
      - the **Linux® kernel** running on the Arm®Cortex®-A core in Non-secure mode.
    - the **application frameworks** composed of middleware components relying on the BSP and providing:
      - **OP-TEE** APIs to run **Trusted Applications (TA)** that allow manipulating secrets (not visible from the Linux® and STM32Cube MPU Package).
      - **Android** APIs to run **applications** that typically interact with the user via a display or a touchscreen.

- the **STM32Cube MPU Package** runs on the Arm®Cortex®-M core: like other STM32 microcontrollers, it is based on HAL drivers and middleware components. Iit is completed with the coprocessor management.

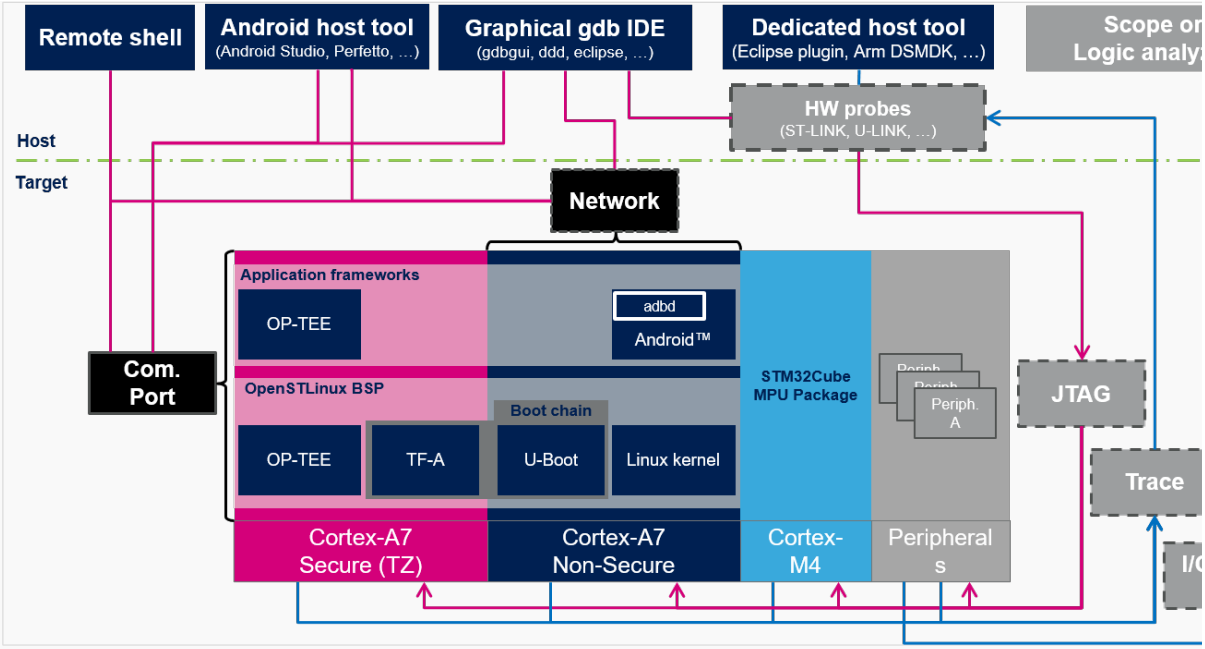
- The **STM32MPU peripherals** shared between Cortex®-A and Cortex®-M cores (such as GPIO, I2C and SPI).
- The **user interfaces or tools**, which allow interacting with different trace and debug Tools, such as:
  - The **remote shell** using terminal console
  - The **Android host tools** (such as Android Studio)
  - The **debugger tools** (such as GDB)
  - The **graphical IDE** (such as GDBGUI or SystemWorkbench)
- The **trace and debug interfaces or hardware paths** that provide access to trace and debug components through:
  - the **network** interface (e.g. Ethernet)
  - the **communication port** (e.g UART)
  - the hardware connector interfaces:
    - **JTag** port
    - **Trace** port to access ETM, STM, ITM and SWD
    - **I/O probes** to access HDP
- The **hardware probes** (such as ST-Link).

This block diagram also illustrates the Arm® debugging modes:

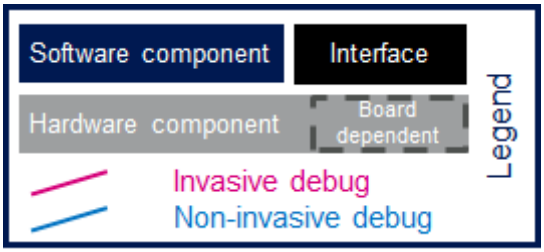
- **Invasive debug**: debug process that allows controlling and monitoring the processor. Most debug features are considered invasive because they enable you to halt the processor and modify its state.
- **Non-invasive debug**: debug process that allows monitoring the processor but not controlling it. The embedded trace macrocell (ETM) interface and the performance monitor registers are non-invasive debug features.

Click the figure below to directly jump to the component you want to trace, monitor or debug:

- Select a **hardware component** to be redirected to the corresponding hardware board article and check if the hardware connector is supported on your board.
- Select a **target software component** to be redirected to an article that explains in details how to trace, monitor or debug the corresponding component.
- Select a **host software component** to be redirected to an article that explains how to use the corresponding remote tool.



STM32MP1 Platform trace and debug environment overview for Android.



Arm® is a registered trademark of Arm Limited (or its subsidiaries) in the US and/or elsewhere.

Cortex®

Board support package

Operating System

Linux® is a registered trademark of Linus Torvalds.

Open Portable Trusted Execution Environment

Trusted Application

Microprocessor Unit

Hardware Abstraction Layer

General-Purpose Input/Output (A realization of open ended transmission between devices on an embedded level. These pins available on a processor can be programmed to be used to either accept input or provide output to external devices depending on user desires and applications requirements.)

Inter-Integrated Circuit (Bi-directional 2-wire bus standard for efficient inter-IC control.)

Serial Peripheral Interface

GNU dedugger, a portable debugger that runs on many Unix-like systems

(Software)Integrated development/design/debugging environment

Universal Asynchronous Receiver/Transmitter

Embedded Trace Macrocell

System Trace Module

Instruction Trace Macrocell

Serial Wire Debug

Hardware Debug Port

spelling for older versions of STLink, ST in-circuit debugger and programmer for the STM8 and STM32 microcontroller families