Hello, and welcome to this presentation of the STM32 debug and trace interface. It covers the debug and trace capabilities offered by STM32MP1 devices.

Overview

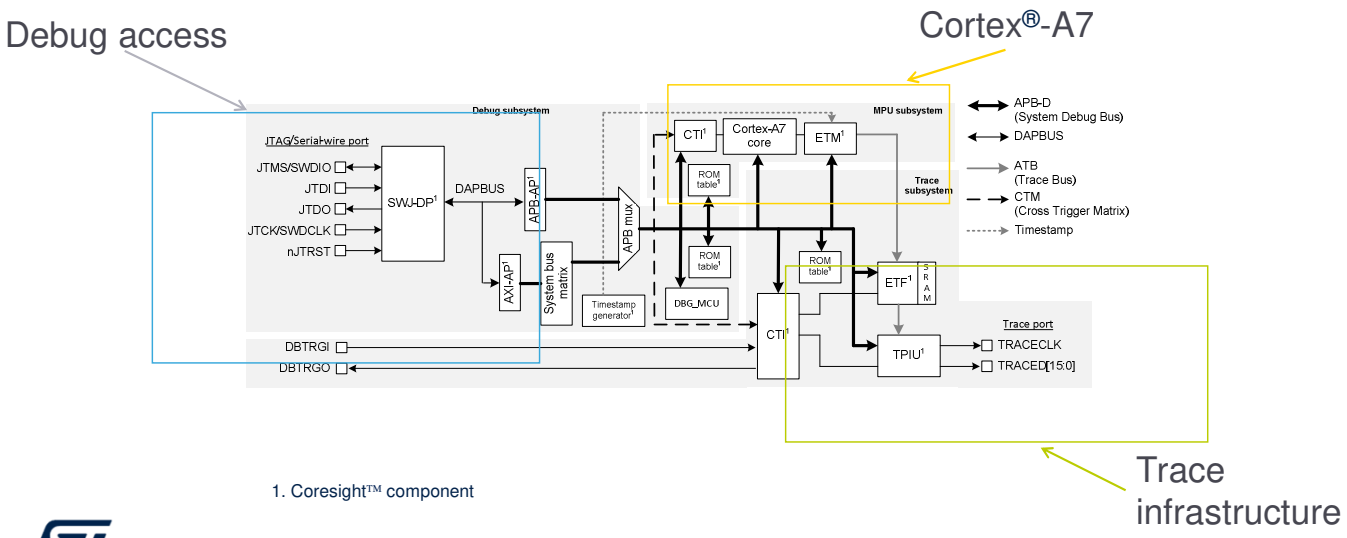
- STM32MP1xx provides rich support for debug and trace
  - Download programs into RAM or Flash memory
  - Examine memory and register contents
  - Insert breakpoints and halt the processor
  - Run or Single-step through programs
  - Trace program execution
- Based on ARM® CoreSight™ architecture
  - Wide range of compatible tools
  - Standard interface (JTAG/serial wire)

life.augmented

2

The STM32MP1 microprocessor incorporates all the familiar debug capabilities provided by the STM32 family of MCUs – flash download, breakpoint debugging, register and memory view, serial wire trace – and adds high bandwidth instruction trace as well as cross-triggering capabilities in multi-core versions of the STM32MP1 family. The debug and trace infrastructure uses the ARM® CoreSight™ standard, well supported by most tool providers.

Debug architecture – STM32MP13x



1. Coresight™ component

The debug and trace infrastructure for STM32MP13 devices is composed of three distinct functional domains:

• Debug access infrastructure – includes the debug port (SWJ-DP) and access ports (AP) which allow access by an external debugger to the target's trace and debug features.

• Trace infrastructure – includes the parallel (TPIU) trace port and the trace FIFO (ETF) used for smoothing the trace flow. There is also a System Trace Module (STM) which allows software generated debug information, as well as hardware events, to be traced.

• Cortex-A7 core – includes the processor (single- or dual-core) and embedded trace module (ETM).

In addition, there are system debug features including:

• Cross trigger interfaces and matrix (CTI, CTM) – these allow simultaneous halting of both cores, triggering of trace, etc.

• Global timestamp generator – provides a common time reference for the different trace sources

• DBG_MCU – provides proprietary features such as freezing of

timers during debug

• External trigger input/output – allows an external signal to trigger debug or trace, or generates a trigger pulse for synchronizing external equipment or components

Debug port

- The debugger accesses the STM32MP1xx via the JTAG/SWD debug port
  - Standard 5-pin JTAG port also used for boundary scan and DFT
  - Serial wire debug (SWD) port uses 2 of the JTAG port pins

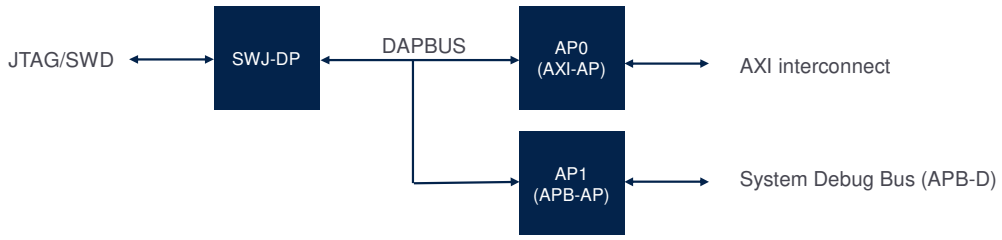
Pin	JTAG debug port	SW debug port
JTMS-SWDIO	Test Mode Select	Serial Wire Data (In/Out)
JTCK-SWCLK	Test Clock	Serial Wire Clock
JTDO	Test Data Out	-
JTDI	Test Data In	-
nJTRST	Test Reset	-

life.augmented

The debug port is available on dedicated pins on all STM32MP1 packages.
Serial wire debug uses a special serial code driven by the debugger on the SWDIO (JTMS) input. This is recognized by the SWJ-DP which switches to SWD mode (after reset, JTAG mode is configured by default).
ST-Link, and most 3rd party debug adaptors (for example, Ulink), support serial wire debug.

Access ports

- Three access ports (AP) act as bus masters allowing the debugger to perform read/write transactions to memory and registers



AP0: Allows access to the AXI interconnect. This gives the debugger direct access to all memory and peripheral registers.
AP1: Allows access to the debug and trace features on the system APB debug bus, namely the Cortex-A7 debug features and the trace subsystem.
Applications running on the processor can access the debug features located on the System Debug Bus since they are mapped in the unified address space. This includes the trace subsystem (TPIU, TSGEN), as well as the Cortex-A7 features (ETM, CTI, DBG).

Authentication and security

- 6 authentication signals from the Boot and Security (BSEC) unit control the access to debug features:
  - DEVICEEN: Controls access to debug components via the external debug port.
    - 0: External debugger has no access to debug components. Debugger access to system interconnect is not affected.
    - 1: External debugger has access to debug components
  - DBGEN: Global enable for all debug and trace features
    - 0: All debug features are disabled. External and software access to debug components is still possible. External access to system interconnect is disabled.
    - 1: Debug features in non-secure mode are enabled. External access to non-secure system interconnect is enabled. Debug features in secure mode and external access to secure system interconnect are dependent on the state of the SPIDEN signal.
  - SPIDEN: Enables debug in secure privileged mode when DBGEN = 1
    - 0: Debug features are disabled in secure privileged mode. External access to secure system interconnect is disabled.
    - 1: Debug features are enabled in secure privileged mode. External access to secure system interconnect is enabled.
  - NIDEN: Enables trace and performance monitoring (non-invasive debug)
    - 0: Trace generation is disabled.
    - 1: Trace generation in non-secure mode is enabled. Trace generation in secure mode is dependent on the state of the SPNIDEN signal.
  - SPNIDEN: Enables trace and performance monitoring in secure privileged mode when NIDEN = 1.
    - 0: Trace generation is disabled in secure privileged mode.
    - 1: Trace generation is enabled in secure privileged mode.
  - DBGSWEN: Enables self-hosted debug
    - 0: Software access to all debug components is disabled
    - 1: Software access to all debug components is enabled

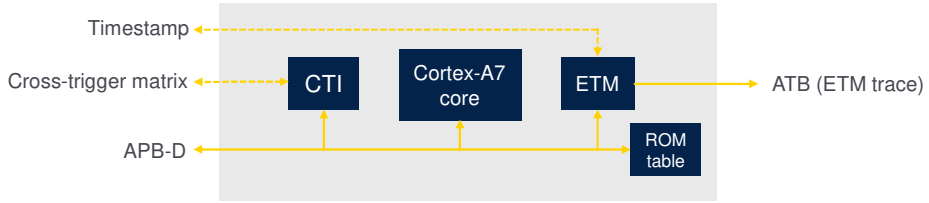
life.augmented

6

The authentication signal states are set in the Boot and Security (BSEC) unit. The default state of these signals is determined by the factory state of the device (open or closed). The state can be modified by a secure software.
The debugger must use secure privileged transactions to access secure addresses. These transactions only succeed if the SPIDEN signal is asserted.

Cortex®-A7 T&D features

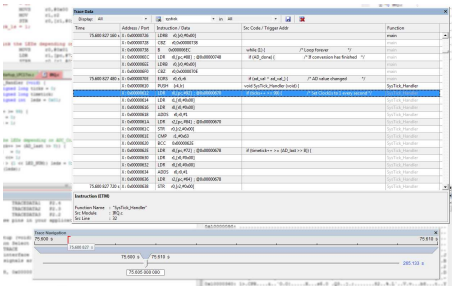
- The Cortex-A7 contains the following debug components:
  - One debug unit (DBG)
  - One Embedded Trace Macrocell (ETM)
  - One Cross-Trigger Interface (CTI)
  - ROM table



All debug related registers in the Cortex-A7 core are accessed via the system debug bus, APB-D, through access port AP1.
The ROM table contains pointers to the base addresses of each debug component in the core. ROM tables are used by some debug tools to automatically detect the topology of the CoreSight™ infrastructure in the target.
The debug unit (DBG) contains the registers for controlling the processor core while in debug mode.

Instruction Trace

- The Embedded Trace Macrocell (ETM) generates trace packets which allow the execution of the software to be observed. The trace information includes:
  - The number of instructions executed in the same cycle
  - Changes in program flow
  - The current processor instruction state
  - The addresses of memory locations accessed by load and store instructions
  - The type, direction and size of a transfer
  - Condition code information
  - Exception information
  - Wait for interrupt state information
- Trace packets are output on the ATB trace bus

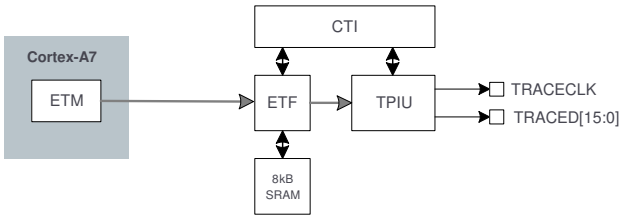


life.augmented

In the STM32MP1 series, the Embedded Trace Macrocell (ETM) is configured for instruction trace only; i.e. data accesses are not included in the trace information.

Trace Infrastructure

- Trace information from the ETM is routed via the Amba Trace Bus (ATB) to the trace port (TPIU), or can be stored in the trace FIFO (ETF)



The ETM in the Cortex-A7 generates a trace stream which is output on the trace port (TPIU).

Trace Sinks

- The trace packets are channeled to one of two destinations or “sinks”:
  - Embedded Trace FIFO (ETF)
    - This is a 4-Kbyte memory that can store trace packets in a circular buffer. The trace can be read out by software or by the debugger
  - Trace Port Interface (TPIU)
    - Trace packets are streamed out of the device via a 16-pin parallel port, accompanied by a synchronous clock signal. This requires connection of a trace port analyzer probe such as ULINKpro or DStream.

life.augmented

The ETF can be used as a trace buffer for storing traces on-chip. The trace can be read by software, or by the debugger, or flushed via the trace port. If configured as a circular buffer, the trace will be stored continuously, so the most recent trace will overwrite the oldest. Alternatively, the FIFO full flag can be used to stop a trace when the buffer is full, and hence capture a trace at a particular point in time.

The ETF also acts (in hardware mode) to smooth the flow of trace to the TPIU. Since the trace stream tends to be bursty in nature, and the instantaneous bandwidth is much higher than that of the trace port, the buffer absorbs the peaks and regulates the flow to the trace port’s maximum continuous bandwidth.

Trace Port

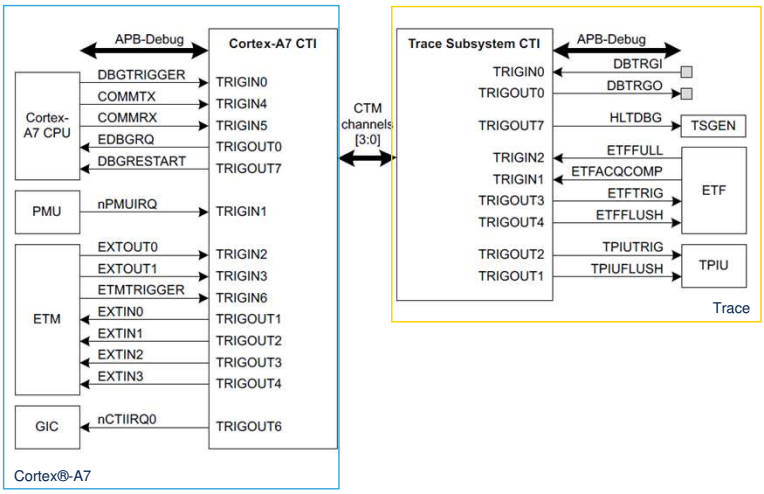
- TPIU parallel port
  - 1 to 16 data pins and a clock can be assigned to trace (GPIOs by default)
  - TRACECLK can operate at up to 133 MHz in DDR mode
  - 1Gbps maximum bandwidth (800 Mbps with ULINKpro)

life.augmented

The trace port width can be programmed from 1 to 16 pins. The bandwidth scales proportionally to the number of pins and the TRACECLK frequency (selectable via a divider in the RCC). Full dual-core instruction trace at maximum clock frequency is likely to require the maximum bandwidth. By applying filters and triggers to the trace source (ETM), the average amount of trace data can be reduced, allowing a lower clock rate or reduced number of pins.

Cross Trigger Matrix

- The CTM propagates trigger events to other debug and trace components.
  - Trigger event sources can include:
    - Data watchpoints
    - Hardware breakpoints
    - Profiling counter events
    - Trace buffer full/empty
    - External trigger signal
    - Processor halt/restart
  - Trigger events at the destination can cause:
    - Trace start/stop
    - Trace buffer flushing
    - Processor halt/restart
    - External trigger signal output
    - Processor interrupt



life.augmented

12

Cross-triggering can be used to stop and start trace based on triggers from the various debug components, such as the ETF FIFO becoming full, or from external pins.

The cross-trigger feature can also be used to halt the processor with an external trigger signal (this might be an edge on one of the IO pins), or generate an external trigger when the processor hits a breakpoint.

There is a Cross-Trigger Interface (CTI) for the processor core and debug components, as well as a system CTI connected to the trace components (ETF, TPIU) and external trigger signals.

To use any of the cross-trigger features, the CTIs must be programmed accordingly by the debugger. The required trigger input signals (TRIGINn) and trigger output signals (TRIGOUTn) need to be connected to the cross-trigger matrix (CTM). The CTM comprises up to four channels allowing four different events to be propagated in parallel. Trigger inputs can be combined in the CTI so that any one of the combined inputs will cause an event on the connected channel. Similarly, a channel can be connected to

several trigger outputs, so that one event can trigger multiple actions.

DBGMCU

- The “MCU debug” block enables device-specific debug features
  - Device identity
    - Standard location for the reading the device identity code register
  - Emulation of low power modes
    - Maintains the power and clock when the device enters a low power mode (SLEEP, STOP, STANDBY) so that debug access is still possible
  - Peripheral clock “freeze” in debug mode
    - Freezes the RTC, TIM, LPTIM and watchdog (IWDG) timer counters, as well as the SMBUS and FDCAN timeout counters, while the processor is halted
  - Domain debug clock enable
    - Disables the clocks to debug devices in each power domain when not required, to save power
  - External trigger direction
    - Controls the direction (input/output) of the bi-directional TRGIO external trigger pin

life.augmented

The DBGMCU is located on the debug APB bus and can be accessed by the debugger via the APB access port AP2. It is also accessible by the processors in the debug APB address space.
The DBGMCU_IDC register provides the device ID and revision codes in STM32 standard format. The information is also available in the debug port (DP_TARGETID register – accessible only to an external debugger) and in the system debug ROM table registers (SYSROM_PIDR[2:0] – accessible also by software).
Low power mode emulation means that the debugger connection is not lost when entering low power mode. It eliminates the need to replace the low power entry command (for example, WFI/WFE) by a while() loop. On exit, the device is in the same state as if the emulation was not active (apart from any changes made by the debugger during the low power mode emulation).
Peripheral clock freeze is particularly useful to prevent a watchdog timeout from resetting the device while debugging, without having to re-arm the watchdog with the debugger. It also allows timer values to be inspected and corresponding interrupts to be

suspended until “normal” operation is resumed.
The debug clock enable bits to ensure that the debug blocks are only clocked when needed. This avoids unnecessary power consumption, since apart from the DAP, all blocks are clocked with the ungated domain clock.
On certain packages, the TRGIN and TRGOUT pins are not available, only the bi-directional pin is used, and the direction must be chosen using the TRGOEN bit.

Thank you

life.augmented