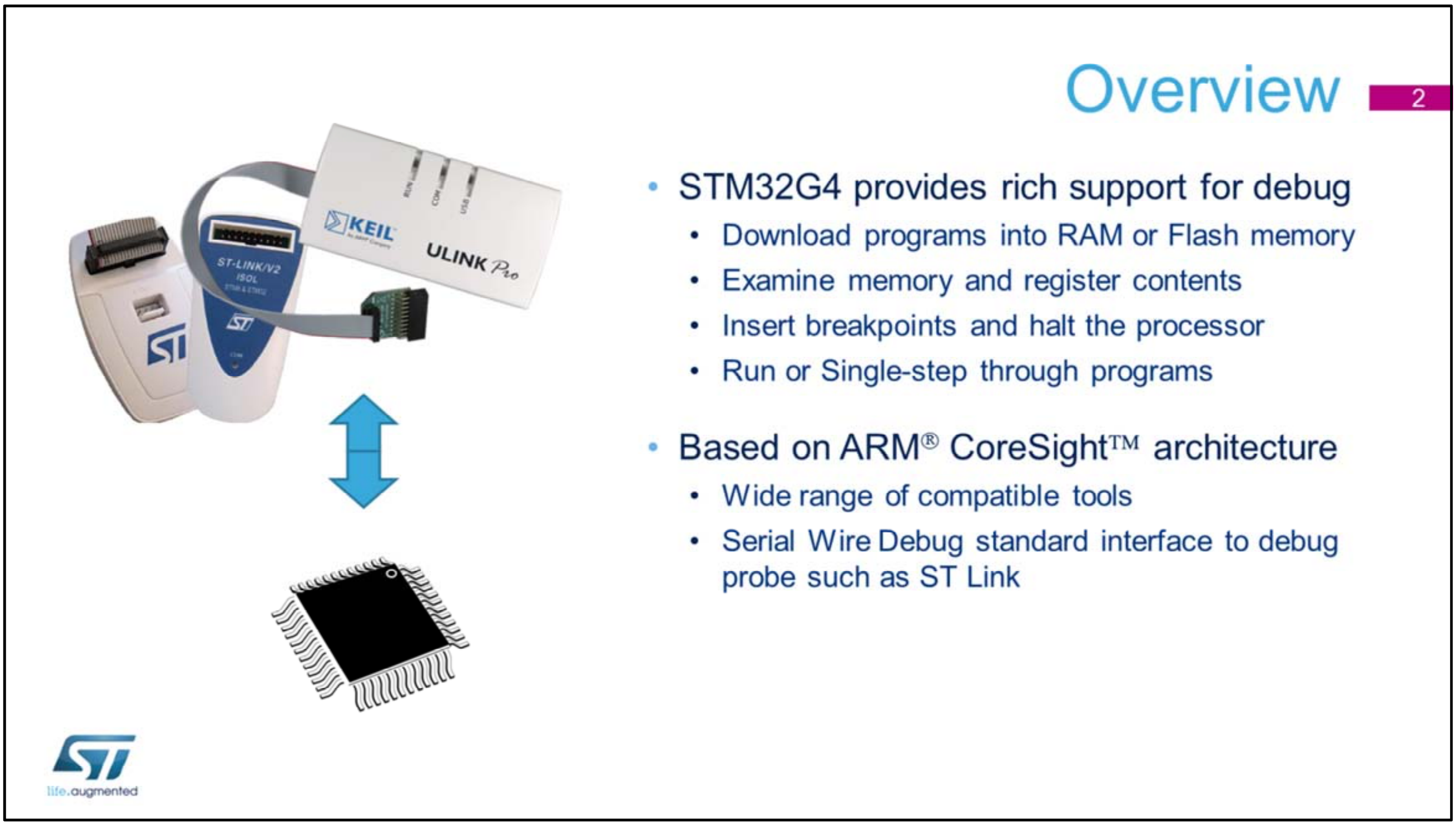
STM32G4 - DBG

Debug and trace

Revision 1.0

Hello, and welcome to this presentation of the STM32 debug interface. It covers the debug capabilities offered by STM32G4 devices.



Overview

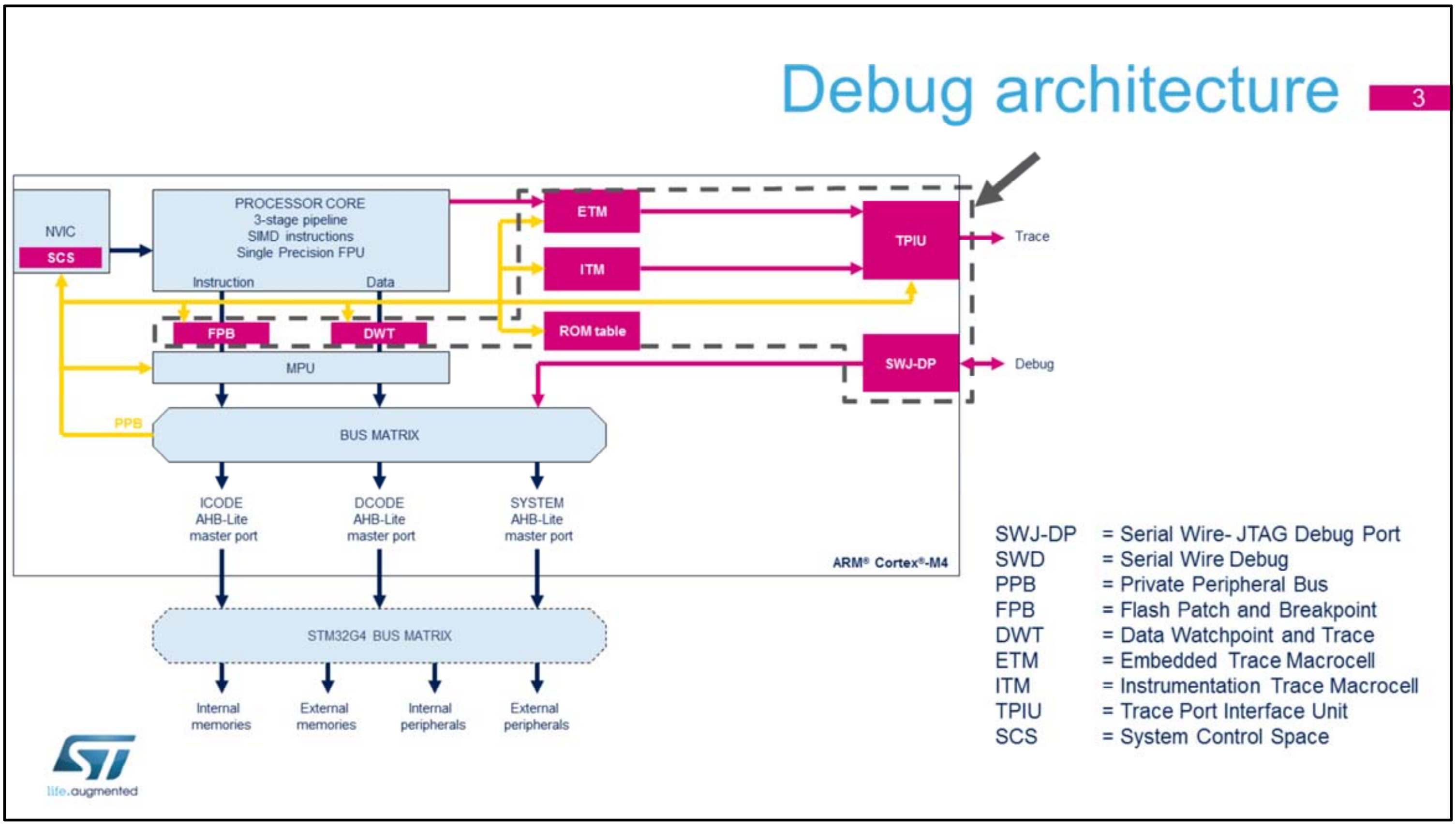
- STM32G4 provides rich support for debug
  - Download programs into RAM or Flash memory
  - Examine memory and register contents
  - Insert breakpoints and halt the processor
  - Run or Single-step through programs
- Based on ARM® CoreSight™ architecture
  - Wide range of compatible tools
  - Serial Wire Debug standard interface to debug probe such as ST Link

The STM32G4 incorporates all the familiar debug capabilities provided by the STM32 family of MCUs – flash download, breakpoint debugging, register and memory view. The debug infrastructure uses the ARM® CoreSight™ standard, well supported by most tool providers.

Slide 2

T1 Can you provide a picture of ST-Linkv3 ?

Trainer; 03/02/2019

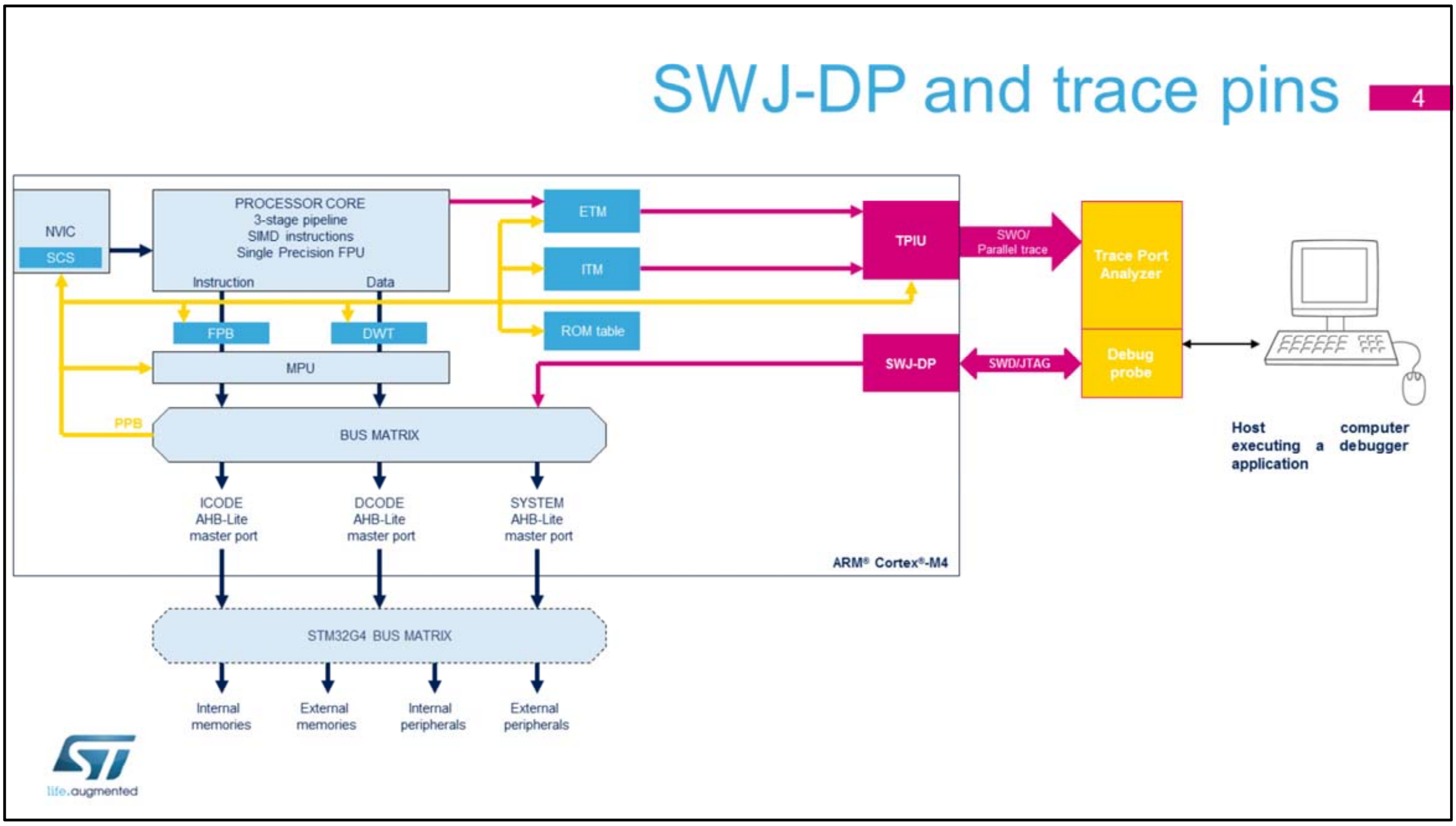


Debug architecture

SWJ-DP	= Serial Wire- JTAG Debug Port
SWD	= Serial Wire Debug
PPB	= Private Peripheral Bus
FPB	= Flash Patch and Breakpoint
DWT	= Data Watchpoint and Trace
ETM	= Embedded Trace Macrocell
ITM	= Instrumentation Trace Macrocell
TPIU	= Trace Port Interface Unit
SCS	= System Control Space

All units involved in the debug process have memory-mapped registers accessible through the Private Peripheral Bus (PPB) by both the core and the SWJ-DP.

The debugger can access memory-mapped resources while the processor is running. For example, a breakpoint can be set by the debugger by accessing the FPB unit connected to the Private Peripheral Bus while the processor is executing instructions.



SWJ-DP and trace pins

The SWJ-DP enables an external debug probe to access any memory-mapped resources also accessible from the Cortex®-M4 core.

Note that the Memory Protection Unit (MPU) does not intercept the requests initiated by the SWJ-DP.

The SWJ-DP supports two protocols to exchange data with the debug probe:

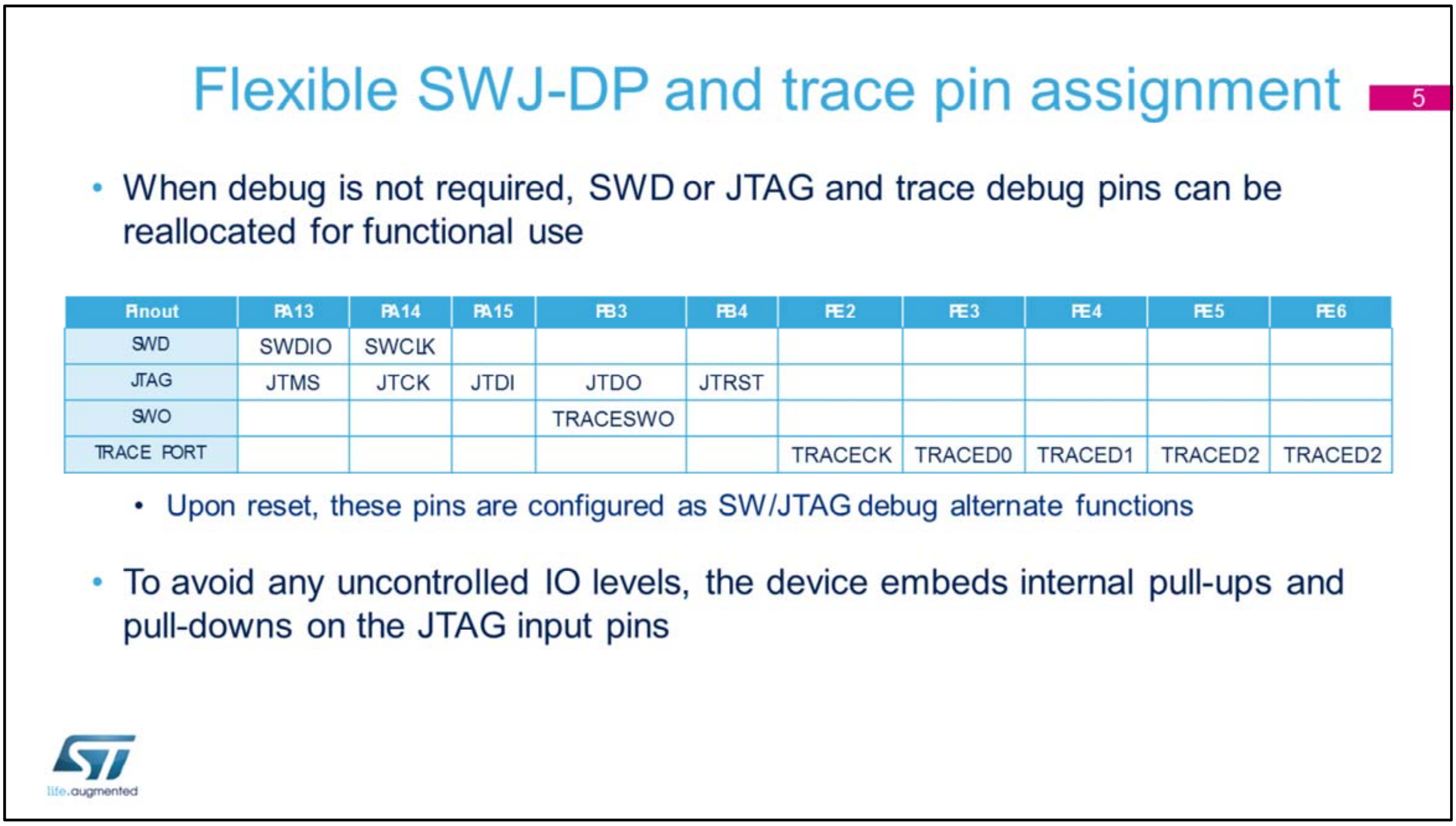
- Either the 2-wire Serial Wire Debug (SWD) protocol
- Or the 5-wire JTAG protocol.

The SWJ-DP automatically detects which protocol is used.

Regarding the trace output, the TPIU offers two possibilities:

- Either the asynchronous 1-wire trace port, called Serial Wire Output (SWO)
- Or the synchronous 5-wire trace port, including a clock signal and 1, 2 or 4 bits of data.

The SWO is multiplexed with the JTAG JTDO signal. Consequently it cannot be used at the same time as JTAG protocol. SWD must be chosen.



Flexible SWJ-DP and trace pin assignment

- When debug is not required, SWD or JTAG and trace debug pins can be reallocated for functional use

Pinout	PA13	PA14	PA15	PB3	PB4	PE2	PE3	PE4	PE5	PE6
SWD	SWDIO	SWCLK								
JTAG	JTMS	JTCK	JTDI	JTDO	JTRST					
SWO				TRACESWO						
TRACE PORT						TRACECK	TRACED0	TRACED1	TRACED2	TRACED2

  - Upon reset, these pins are configured as SW/JTAG debug alternate functions

- To avoid any uncontrolled IO levels, the device embeds internal pull-ups and pull-downs on the JTAG input pins

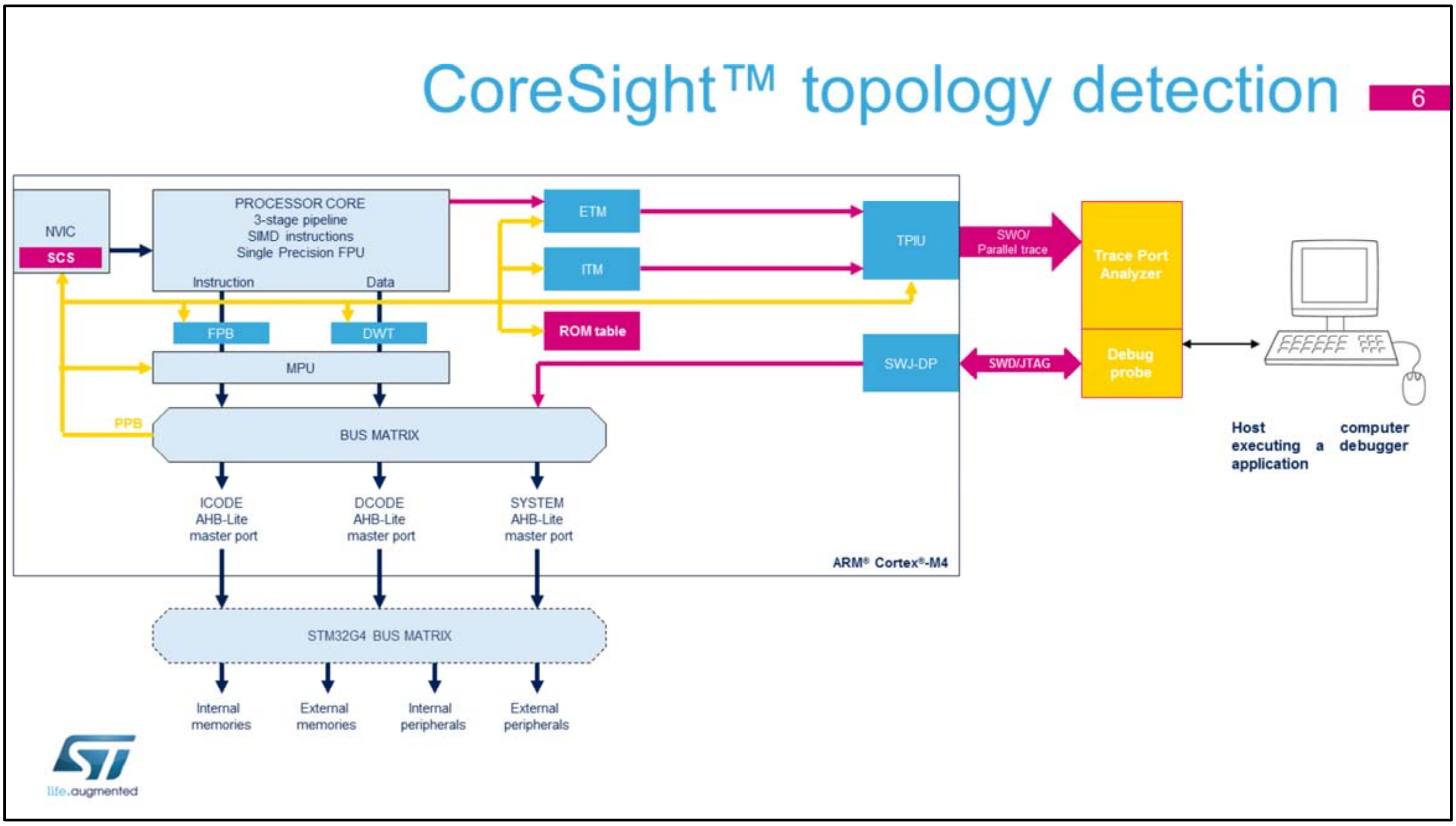
Five pins are used as outputs from the STM32G4 for the SWJ-DP as alternate functions of general-purpose IOs. These pins are available on all packages.

After RESET, all five pins used for the SWJ-DP are assigned as dedicated pins immediately usable by the debugger host. However, the STM32G4 MCUs offer the possibility of disabling some or all of the SWJ-DP ports, and therefore the possibility of releasing the associated pins for general-purpose IO (GPIO) usage, except for NJTRST that can be left disconnected but cannot be used as general purpose GPIO without loosing debugger connection.

Note that the parallel trace port is not assigned except if explicitly programmed by the debugger host.

JTAG pins have internal pull-ups and pull-downs which are by default active:

- Pull-up on NJTRST, JTDI, JTMS/SWDIO
- Pull-down on TCK/SWCLK.

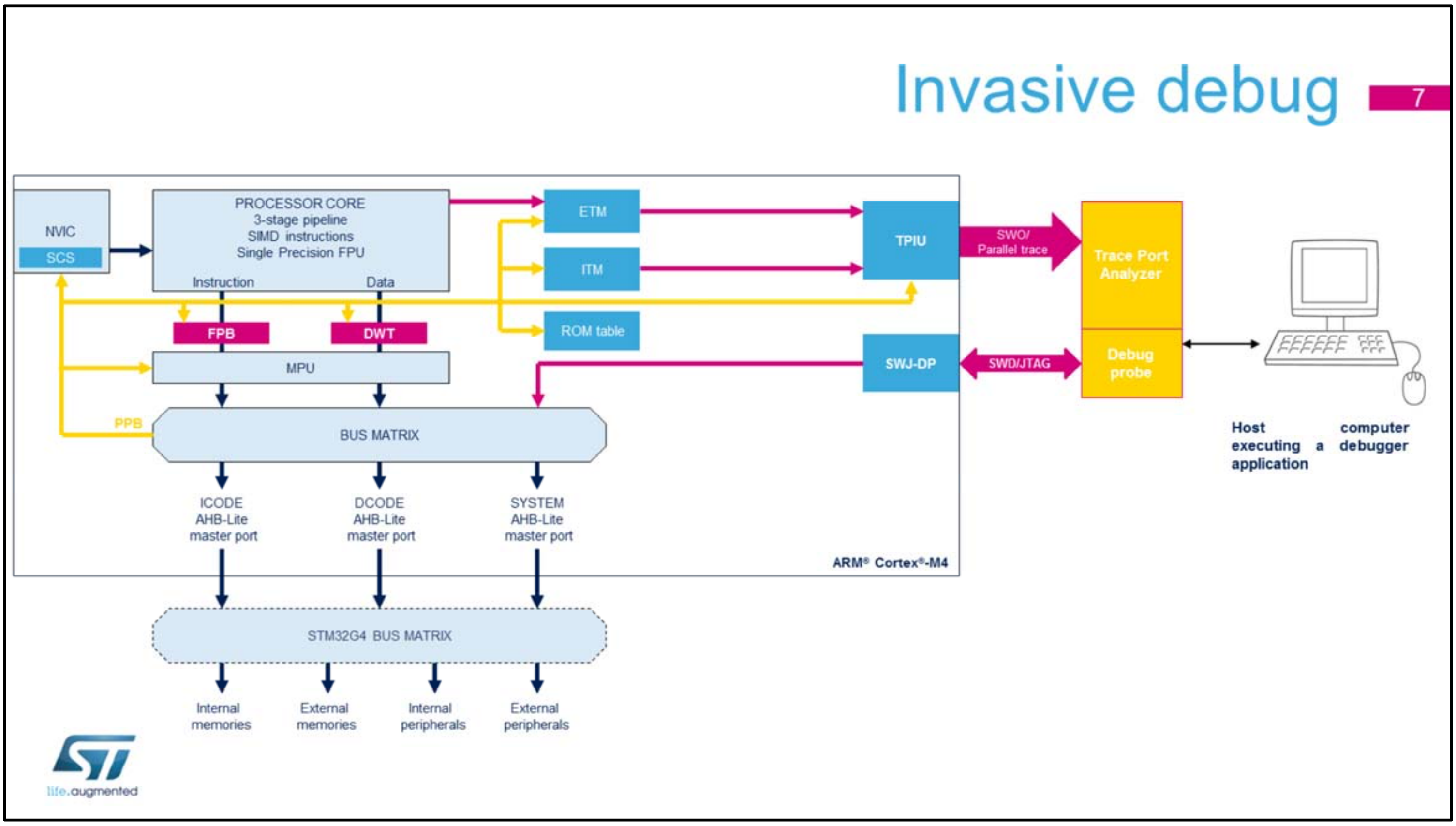


CoreSight™ topology detection

The ROM table contains pointers to the base addresses of each debug component visible from the Core. They are used by some debug tools to automatically detect the topology of the CoreSight™ infrastructure in the target.

The SWJ-DP contains a read-only register pointing to the ROM table.

The SCS (system control space) is a block of registers, including the CPUID register that indicates the reference and revision of the CPU, used by the debugger to control entry to and exit from Halt mode.



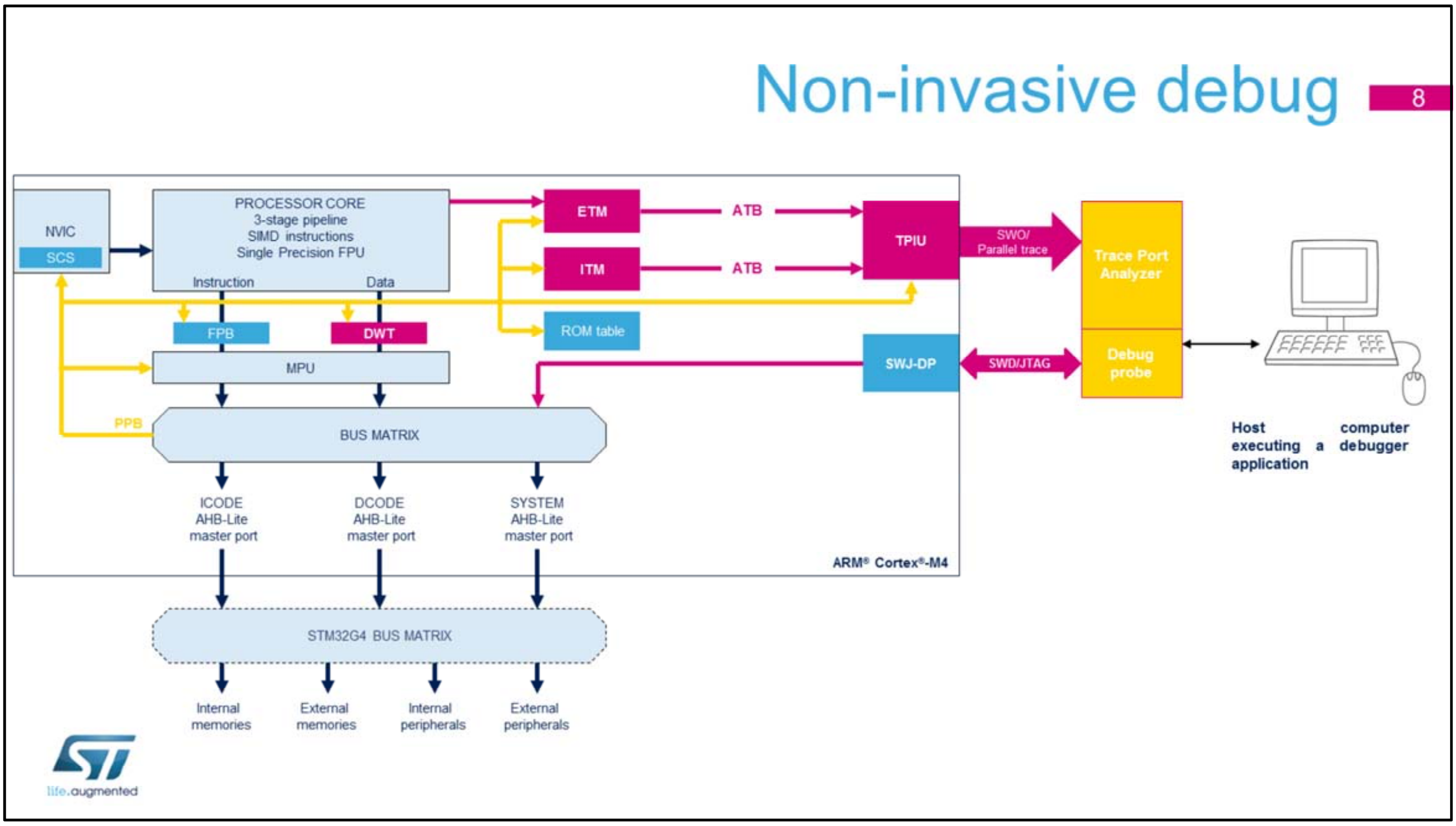
Invasive debug

7

Invasive debug halts the processor when a debug event occurs.

Two units are involved in invasive debug:

- The Flash Patch and Breakpoint (FPB)
- The Data Watchpoint and Trace (DWT).



Non-invasive debug

Non-invasive debug aims to track what happens in the processor, without affecting its performance. It is based on real time trace.

The ETM and ITM modules have the capability of synthesizing trace packets and issuing them over the AMBA Trace Bus (ATB).

The TPIU receives the trace packets and outputs them over an external interface, so that the Trace Port Analyzer (TPA) can capture them.

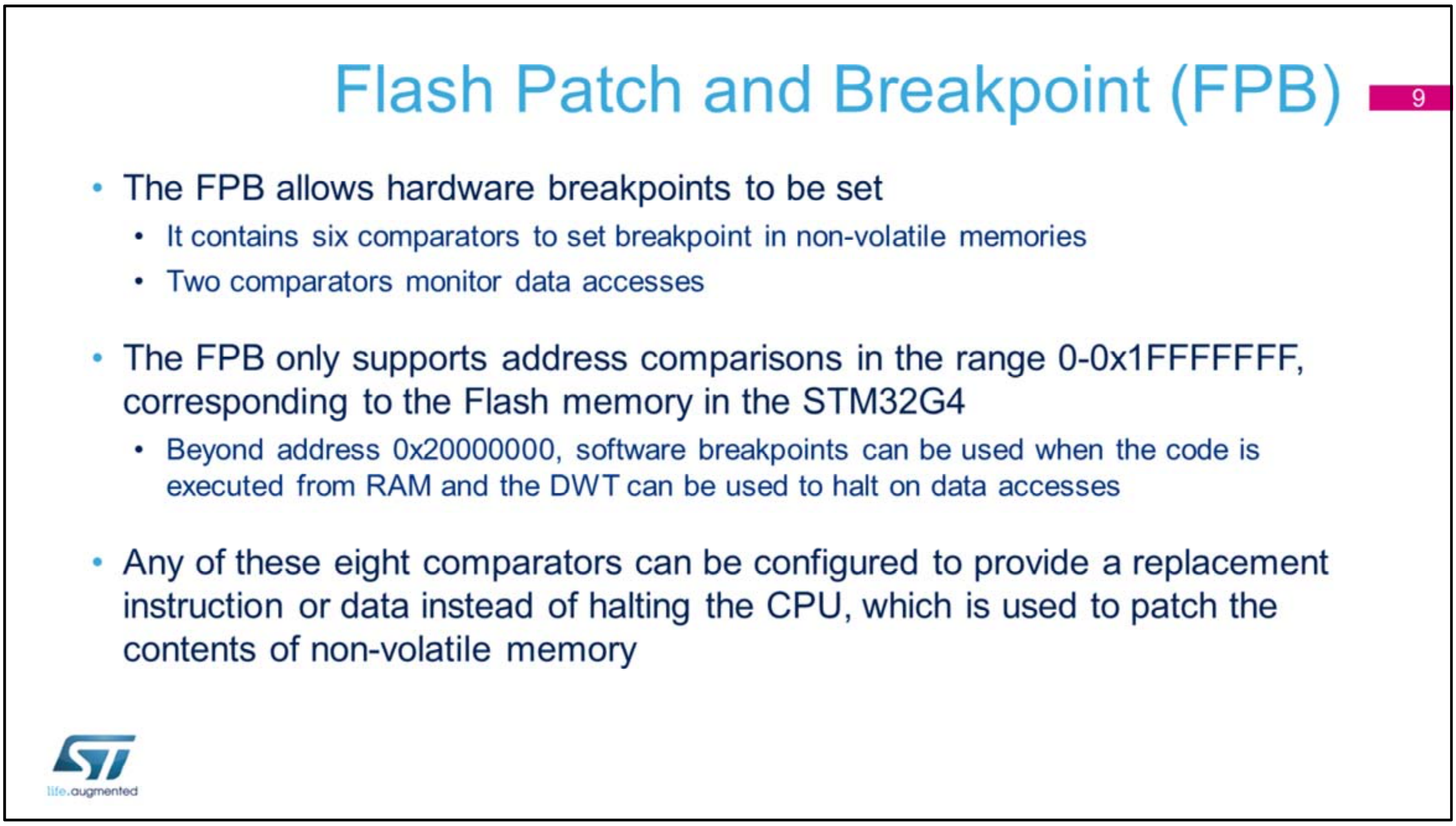
The Trace Port Analyzer is generally contained in the same device as the debug probe.

Two interfaces are supported to transfer trace packets to the TPA:

- Either the asynchronous Single Wire Output (SWO)
- Or the synchronous parallel trace port.

An overflow condition occurs when a contention is detected in the TPIU. It means that the bandwidth of the trace port is not large enough to export all trace packets.

Note that trace packets are timestamped in the ITM and ETM modules.



Flash Patch and Breakpoint (FPB)

- The FPB allows hardware breakpoints to be set
  - It contains six comparators to set breakpoint in non-volatile memories
  - Two comparators monitor data accesses
- The FPB only supports address comparisons in the range 0-0x1FFFFFFF, corresponding to the Flash memory in the STM32G4
  - Beyond address 0x20000000, software breakpoints can be used when the code is executed from RAM and the DWT can be used to halt on data accesses
- Any of these eight comparators can be configured to provide a replacement instruction or data instead of halting the CPU, which is used to patch the contents of non-volatile memory

The Flash Patch and Breakpoint unit (FPB) allows hardware breakpoints to be set.

It contains six comparators to set breakpoint in non-volatile memories. They monitor the instruction fetch address and return a breakpoint instruction when a match is detected. When the breakpoint instruction is executed, the processor halts in Debug mode.

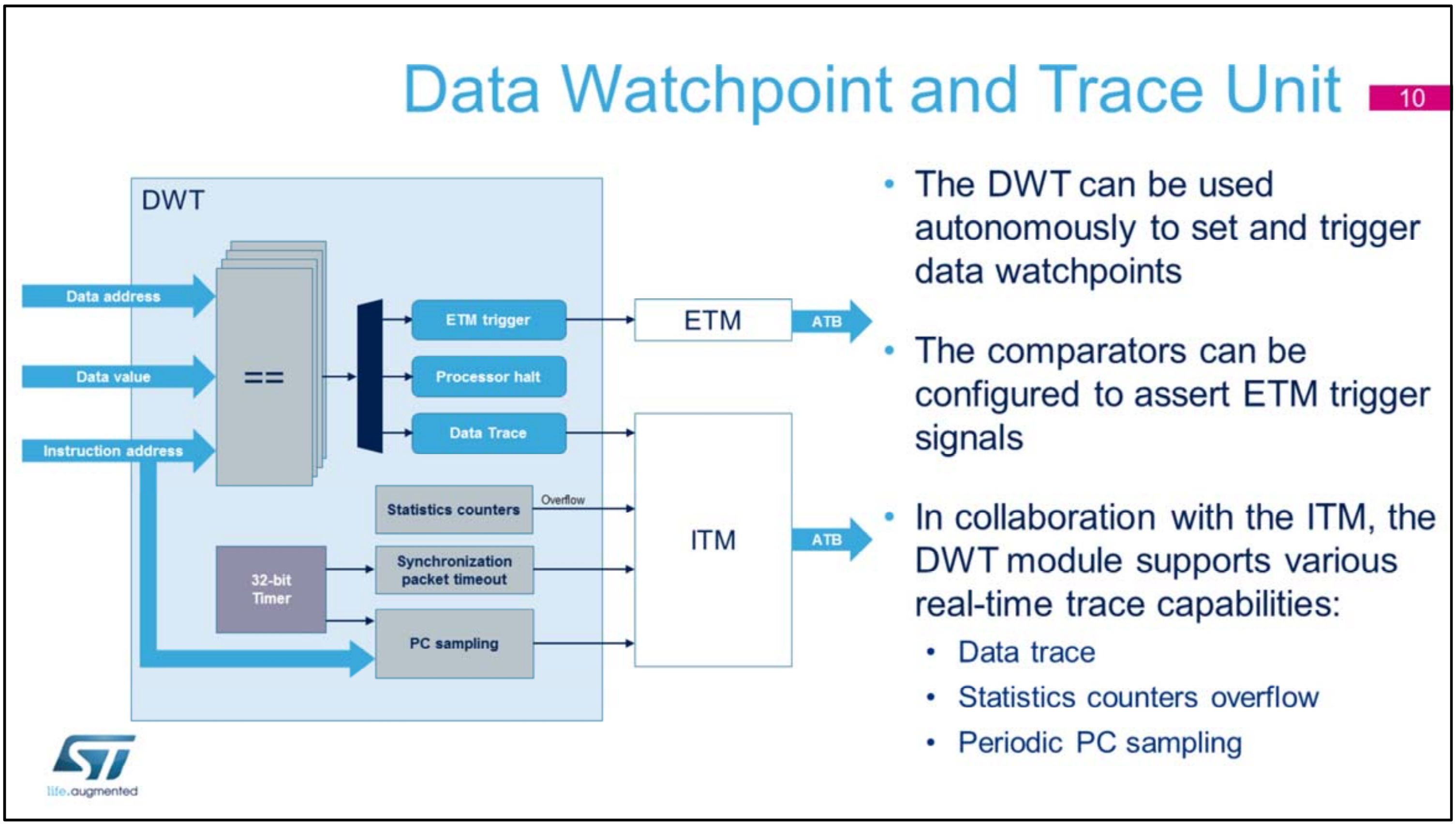
Two comparators monitor data accesses, used to halt the processor when particular constants present in non-volatile memory are accessed.

The FPB only supports address comparisons in the range 0-0x1FFFFFFF, corresponding to the Flash memory in the STM32G4.

Beyond address 0x20000000, software breakpoints can be used when the code is executed from RAM and the DWT can be used to halt on data accesses.

When a software breakpoint is used, the debugger replaces the instruction on which the user wants to stop with a dedicated instruction called BKPT.

Any of these eight comparators can be configured to provide a replacement instruction or constant data instead of halting the CPU, which is used to patch the contents of non-volatile memory.



Data Watchpoint and Trace Unit

- The DWT can be used autonomously to set and trigger data watchpoints
- The comparators can be configured to assert ETM trigger signals
- In collaboration with the ITM, the DWT module supports various real-time trace capabilities:
  - Data trace
  - Statistics counters overflow
  - Periodic PC sampling

The DWT supports more features than just implementing data watchpoints.

The four comparators on the left of the figure detect a match condition on either a data address, or an instruction address.

The comparator number 1 can also be configured to detect a match condition on a data value.

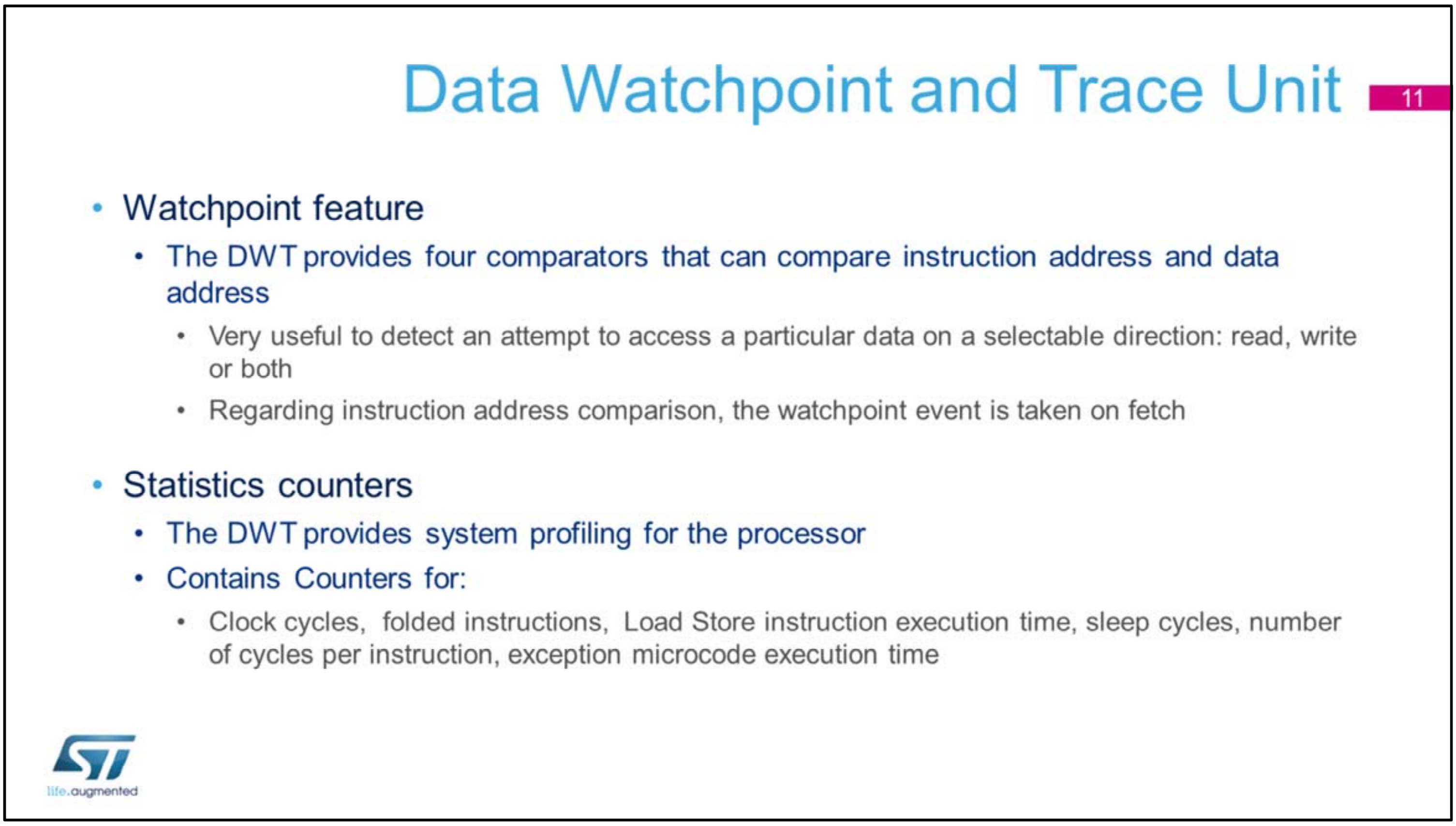
The following actions are supported when a match condition occurs:

- Halting the core, this is the watchpoint feature
- Asserting a trigger signal to the ETM
- Generating a trace packet to report the data access and passing it to the ITM so that it can be exported. The current data and PC value can also be captured to complement the trace information.

Complex conditions, mixing address and data comparisons are programmable.

In addition to the comparison logic, the DWT contains 8-bit statistics counters. When an overflow condition occurs on

any of them, the DWT informs the ITM to issue a trace packet to report this overflow to the debugger.
The DWT is also in charge of triggering the periodic generation of synchronization packets, which are mandatory to maintain the synchronization with the trace port analyzer. This is achieved by a 32-bit timer, which can also be used to trigger the periodic sampling of the PC value, which is passed to the ITM, in order to be stored in a trace packet.



Data Watchpoint and Trace Unit

- Watchpoint feature
  - The DWT provides four comparators that can compare instruction address and data address
    - Very useful to detect an attempt to access a particular data on a selectable direction: read, write or both
    - Regarding instruction address comparison, the watchpoint event is taken on fetch
- Statistics counters
  - The DWT provides system profiling for the processor
  - Contains Counters for:
    - Clock cycles, folded instructions, Load Store instruction execution time, sleep cycles, number of cycles per instruction, exception microcode execution time

The DWT is used to trigger watchpoint events caused by a match between the current data or instruction address and the contents of comparators programmed by the debugger. For address matching, the comparator can use a mask, so it can match a range of addresses.

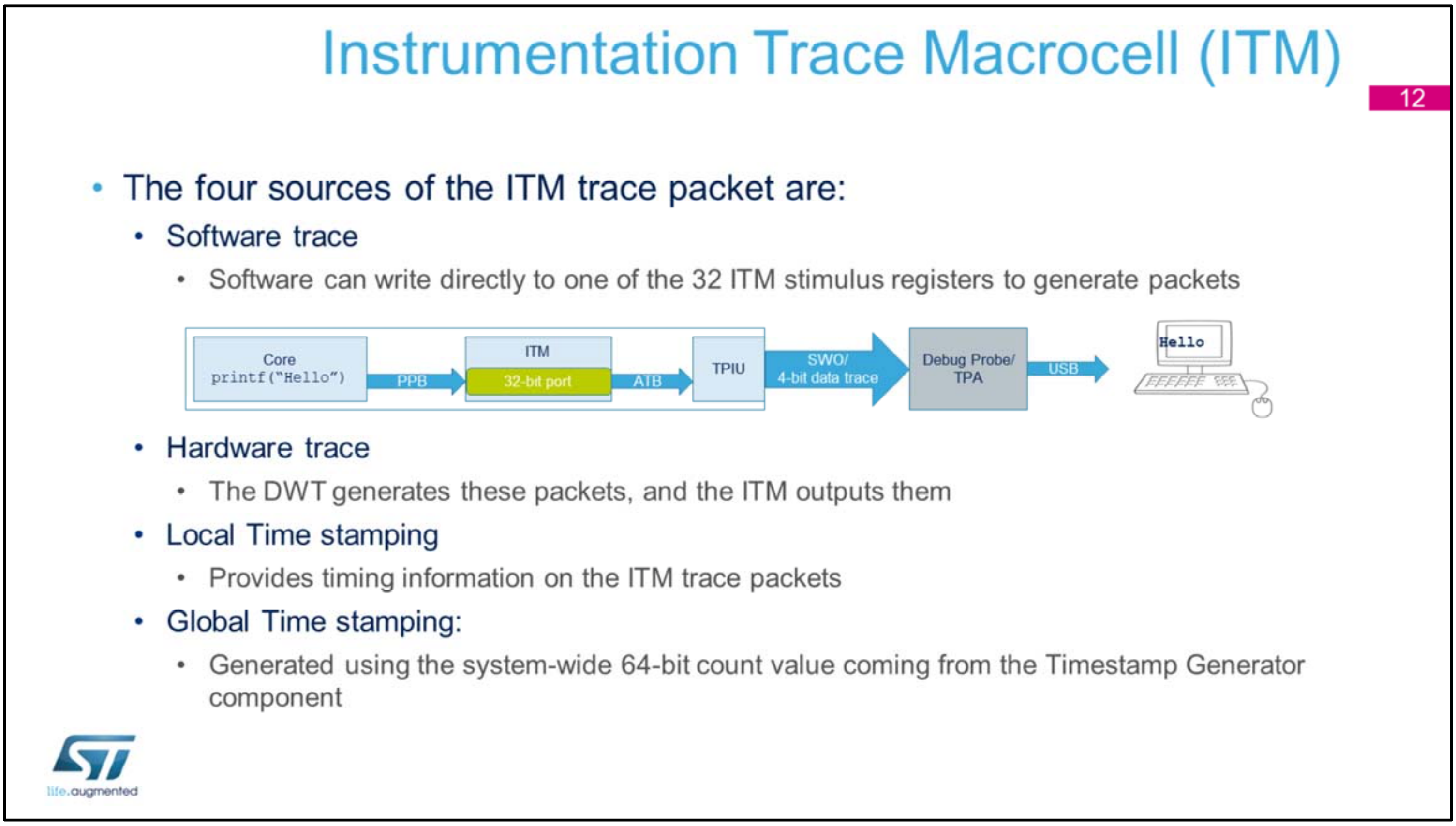
On a successful match, the comparator generates a watchpoint debug event, on either the PC value or the accessed data address.

A match on a data value can be combined with an address match.

The FPB is more appropriate for implementing instruction breakpoints, because the breakpoint event occurs when the instruction is about to enter the execute unit. So if the instruction which has been fetched is discarded due to a taken branch, the breakpoint event does not occur while the watchpoint event occurs.

To read or write a core register, such as R0, the debugger has to halt the core.

The DWT contains statistics counters that facilitate the system profiling for the processor.
For instance, a counter accumulates the number of clocks during which the microcode in charge of exception taking and exiting is active.



Instrumentation Trace Macrocell (ITM)

- The four sources of the ITM trace packet are:
  - Software trace
    - Software can write directly to one of the 32 ITM stimulus registers to generate packets

Core
printf("Hello")
PPB
ITM
32-bit port
ATB
TPIU
SWO/
4-bit data trace
Debug Probe/
TPA
USB
Hello

  - Hardware trace
    - The DWT generates these packets, and the ITM outputs them
  - Local Time stamping
    - Provides timing information on the ITM trace packets
  - Global Time stamping:
    - Generated using the system-wide 64-bit count value coming from the Timestamp Generator component

Two types of trace information are handled by the ITM:

- The hardware trace, which has been described in the DWT module
- The software trace, which enables software to send data that will be displayed in the debugger window.

The data written by software in one of the 32 ITM 32-bit ports is transported in the payload of a trace packet to the host debugger.

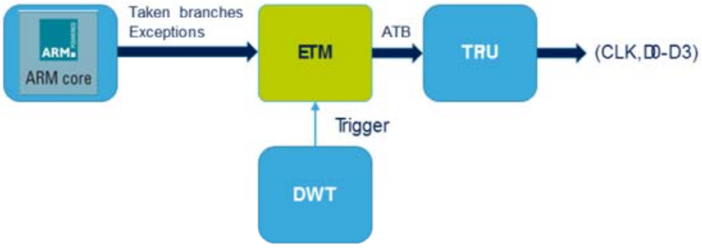
The printf function can be redirected to the ITM ports to facilitate the code instrumentation.

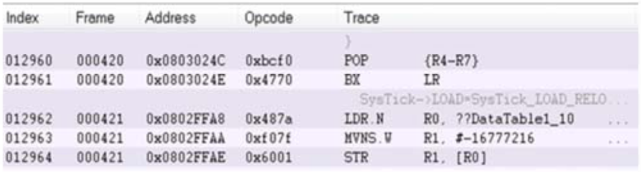
Two types of timestamps are supported by the ITM unit: the local time stamping, which indicates the relative time between two consecutive local timestamp packets and the global time stamping.

The global time stamping comes from a dedicated 48-bit also used by the ETM unit.

Embedded Trace Macrocell (ETM)

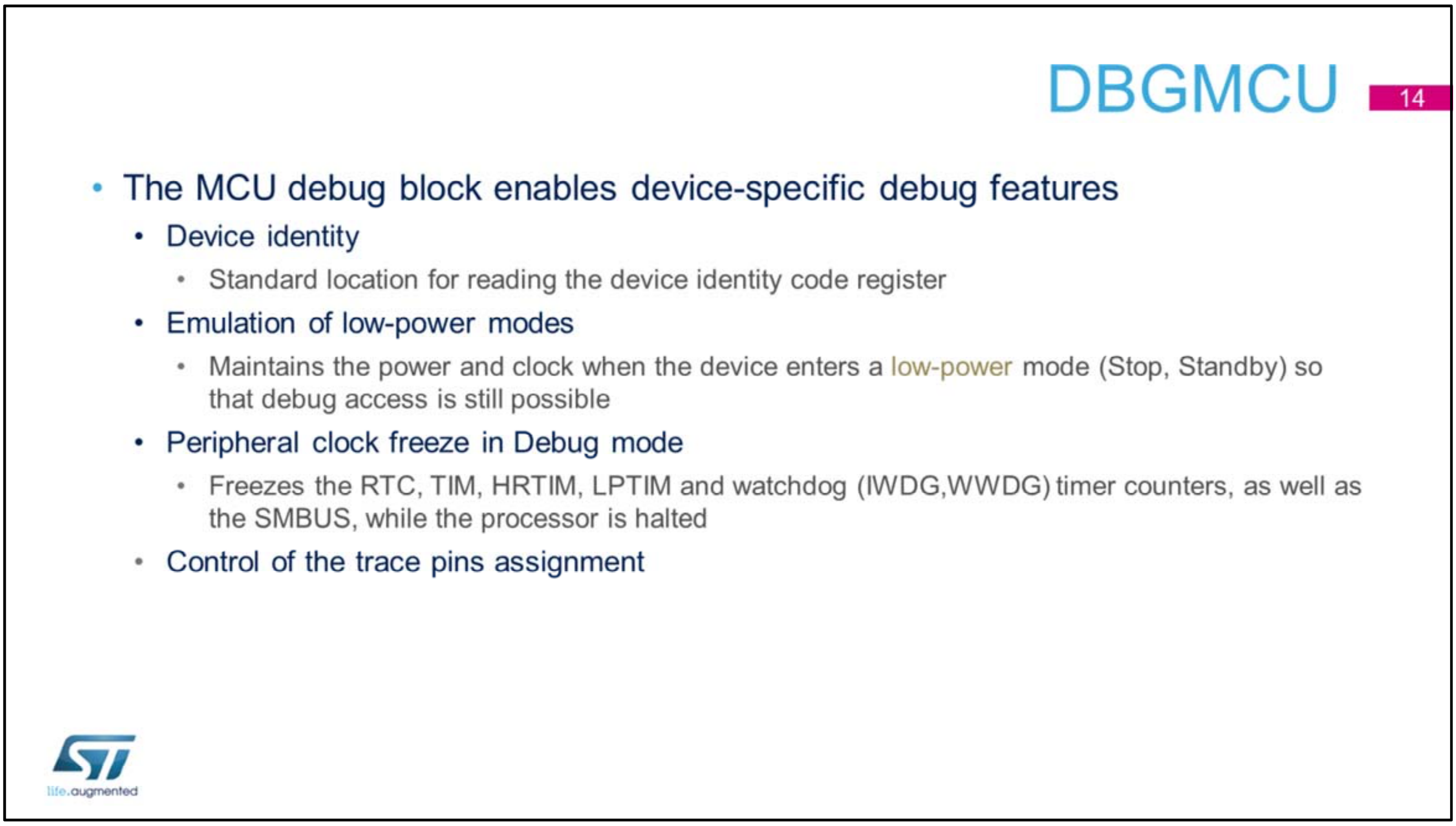
- The ETM is a real-time trace module providing instruction trace of a processor
  - The ETM is used with the TPIU comprising a clock and up to 4 data outputs TRACECLK, and four data outputs, TRACEDATA(3:0)





```
Index   Frame   Address     Opcode   Trace
                                     }
012960  000420  0x0803024C  0xbcf0   POP     {R4-R7}
012961  000420  0x0803024E  0x4770   BX      LR
                                       SysTick->LOAD=SysTick_LOAD_RELO...
012962  000421  0x0802FFA8  0x487a   LDR.N   R0, ??DataTable1_10     ...
012963  000421  0x0802FFAA  0xf07f   MVNS.W  R1, #-16777216          ...
012964  000421  0x0802FFAE  0x6001   STR     R1, [R0]
```

The ETM requires the 4-bit data trace port, because the SWO does not offer enough bandwidth.
Unlike the ITM, no code instrumentation is needed.
Taken branches and exception events are signaled by the Cortex®-M4 core to the ETM units, which converts them into trace packets, which are output onto the ATB port.
When the trace has been captured, the debugger decompresses it to provide a full disassembly, with symbols, of the code that was executed. The debugger also link this back to the original high-level source code, providing you with a visualization of how the code was executed on the target.
The ETM is useful to profile the code executed by the Cortex®-M4 and to investigate complex software bugs.



DBGMCU

- The MCU debug block enables device-specific debug features
  - Device identity
    - Standard location for reading the device identity code register
  - Emulation of low-power modes
    - Maintains the power and clock when the device enters a low-power mode (Stop, Standby) so that debug access is still possible
  - Peripheral clock freeze in Debug mode
    - Freezes the RTC, TIM, HRTIM, LPTIM and watchdog (IWDG,WWDG) timer counters, as well as the SMBUS, while the processor is halted
  - Control of the trace pins assignment

The MCU debug component provides support for:

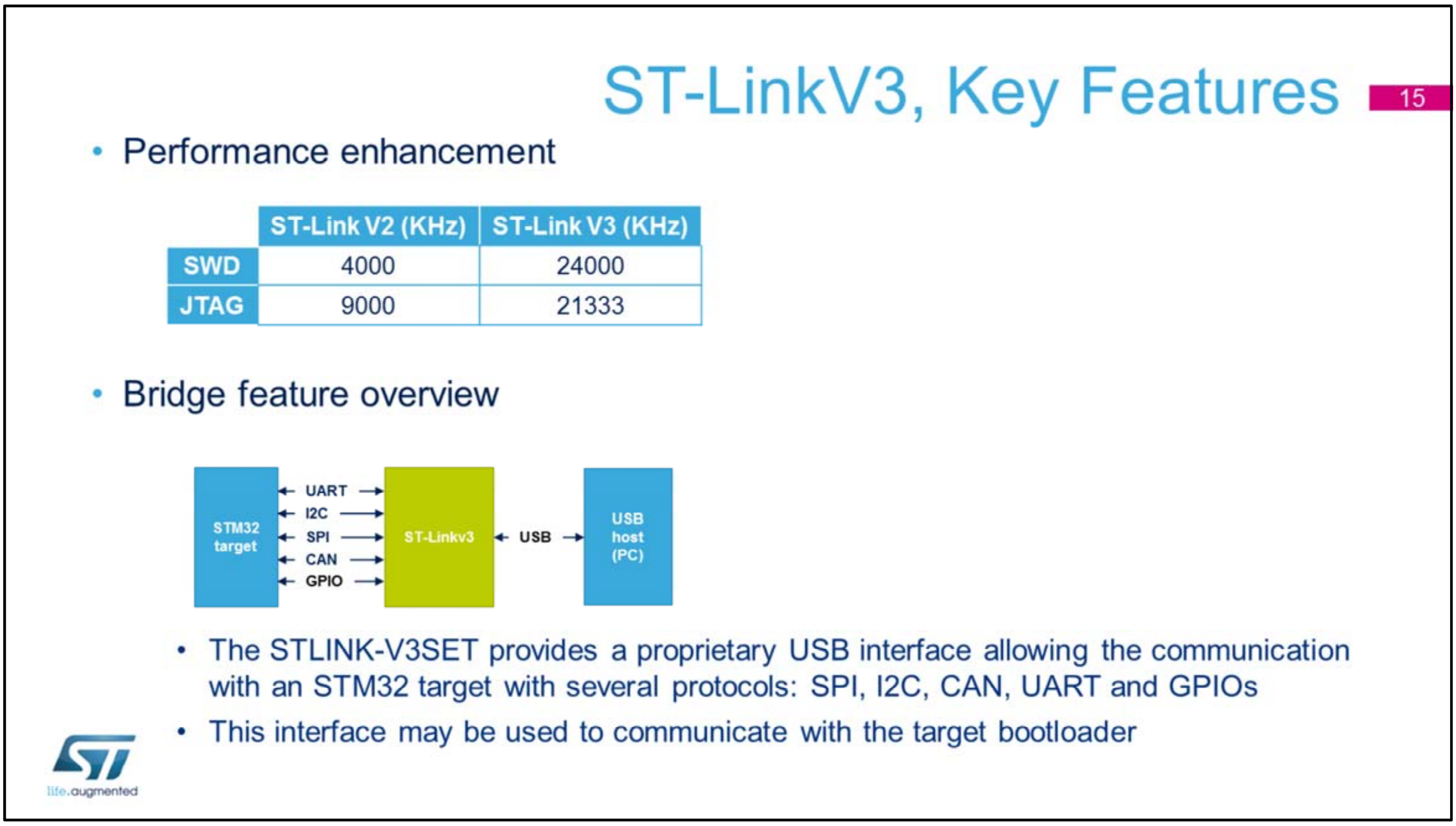
- MCU identification
- Low-power modes
- Clock control for timers, watchdog, I2C and bxCAN during a breakpoint
- Control of the trace pins assignment.

The DBG_IDCODE register provides the device ID and revision codes in STM32 standard format. This code is accessible by the SWJ-DP or by the user software.

Low-power mode emulation means that the debugger connection is not lost when entering a 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 DBGMCU_CR register contains a control field to set configure the trace port: either asynchronous SWO mode or synchronous mode with TRACEDATA size of 1, 2 or 4.



ST-LinkV3, Key Features

- Performance enhancement

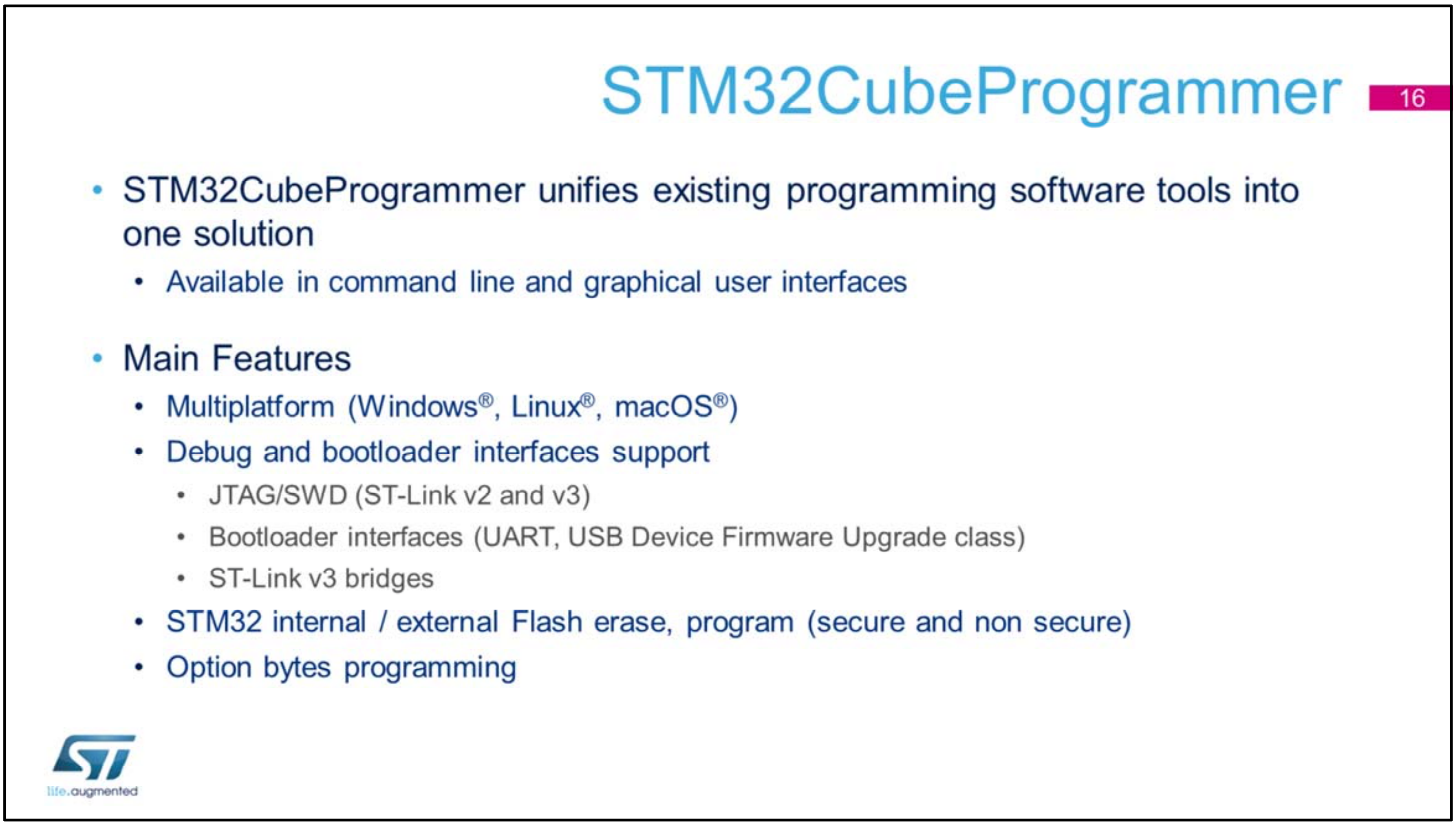
	ST-Link V2 (KHz)	ST-Link V3 (KHz)
SWD	4000	24000
JTAG	9000	21333

- Bridge feature overview

STM32 target
UART
I2C
SPI
CAN
GPIO
ST-Linkv3
USB
USB host (PC)
life.augmented

- The STLINK-V3SET provides a proprietary USB interface allowing the communication with an STM32 target with several protocols: SPI, I2C, CAN, UART and GPIOs
- This interface may be used to communicate with the target bootloader

The ST-Link is a probe developed by ST, that supports SW and JTAG debug protocol and the SWO trace port.
The third generation of ST-Link is available, supporting higher bit-rates for SWD and JTAG.
The ST-Link version 3 can be used to transmit/receive data on the target using UART, I2C, SPI, CAN or GPIO. Only one of these interfaces can be active at the same time.
For instance, a UART of the STM32G4 can be tested by connecting it to the host PC via the ST-Linkv3. Character streams will be transparently transferred over USB.
Furthermore, the ST-Link version 3 supports the download of a user image by using UART or CAN link from a host PC, when the boostrap mode is selected upon STM32G4’s reset deassertion.



STM32CubeProgrammer

16

- STM32CubeProgrammer unifies existing programming software tools into one solution
  - Available in command line and graphical user interfaces
- Main Features
  - Multiplatform (Windows®, Linux®, macOS®)
  - Debug and bootloader interfaces support
    - JTAG/SWD (ST-Link v2 and v3)
    - Bootloader interfaces (UART, USB Device Firmware Upgrade class)
    - ST-Link v3 bridges
  - STM32 internal / external Flash erase, program (secure and non secure)
  - Option bytes programming

STM32CubeProgrammer is an all-in-one multi-OS software tool for programming STM32 microcontrollers.
STM32CubeProgrammer is delivered in GUI (graphical user interface) and CLI (command-line interface) versions.
It provides an easy-to-use and efficient environment for reading, writing and verifying device memory, including option bytes, through both the debug interface (JTAG and SWD) and the bootloader interface (UART and USB).
Its additional features are:

- Multi-OS support: Windows®, Linux®, macOS®
- Debug and bootloader interfaces support
- ST-Link V3 bridges.

STM32CubeProgrammer supports the Device Firmware Upgrade (DFU) USB class designed to download a firmware image to a USB device.