Hello, and welcome to this presentation of the STM32 debug interface. It covers the debug capabilities offered by STM32L4 devices.

Overview

- STM32L4 provides on-chip debug support
  - MCU programming
  - Application debugging
  - Code analysis

Application benefits

- Basic debugging features
- Advanced features (Embedded Trace Macrocell) quickly identify malfunctioning code
- Coverage and profiling features

life.augmented

The debug interface of STM32 products provides an access to MCU internal resources.
This interface is used to program the MCU and debug applications using basic debug features.
In addition to the basic debugging features, applications benefit from the trace capability used to quickly identify possible malfunctioning parts of the application and to create coverage and profiling reports used for application tests, optimizations and certifications.

Key features

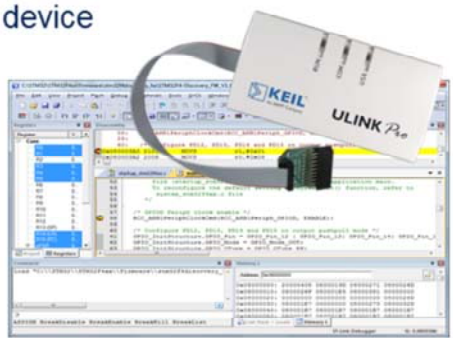
- Basic debugging features
  - JTAG and serial wire debug ports
  - Flexible pin assignment
  - Programming of the device memories through AHB access port
  - Memory & peripheral access
  - Run control – breakpoints, code stepping
  - Peripheral freeze
- Supported by all debugging tools

The STM32L4 offers several basic debugging features which are supported by all hardware and software debug tool sets. Debugging hardware can interface with the STM32L4 through the 5-wire standard JTAG interface or the 2-wire serial wire debug port. The debugging toolset can directly access the AHB access port, an AHB master able to access all internal data buses, or directly read or write to all registers and memories, including programming the Flash memory.
The user can control the execution of the application through breakpoints and code stepping. When the program reaches a breakpoint, internal peripherals like timers can be frozen in their current state or can be left running.

Key features

4

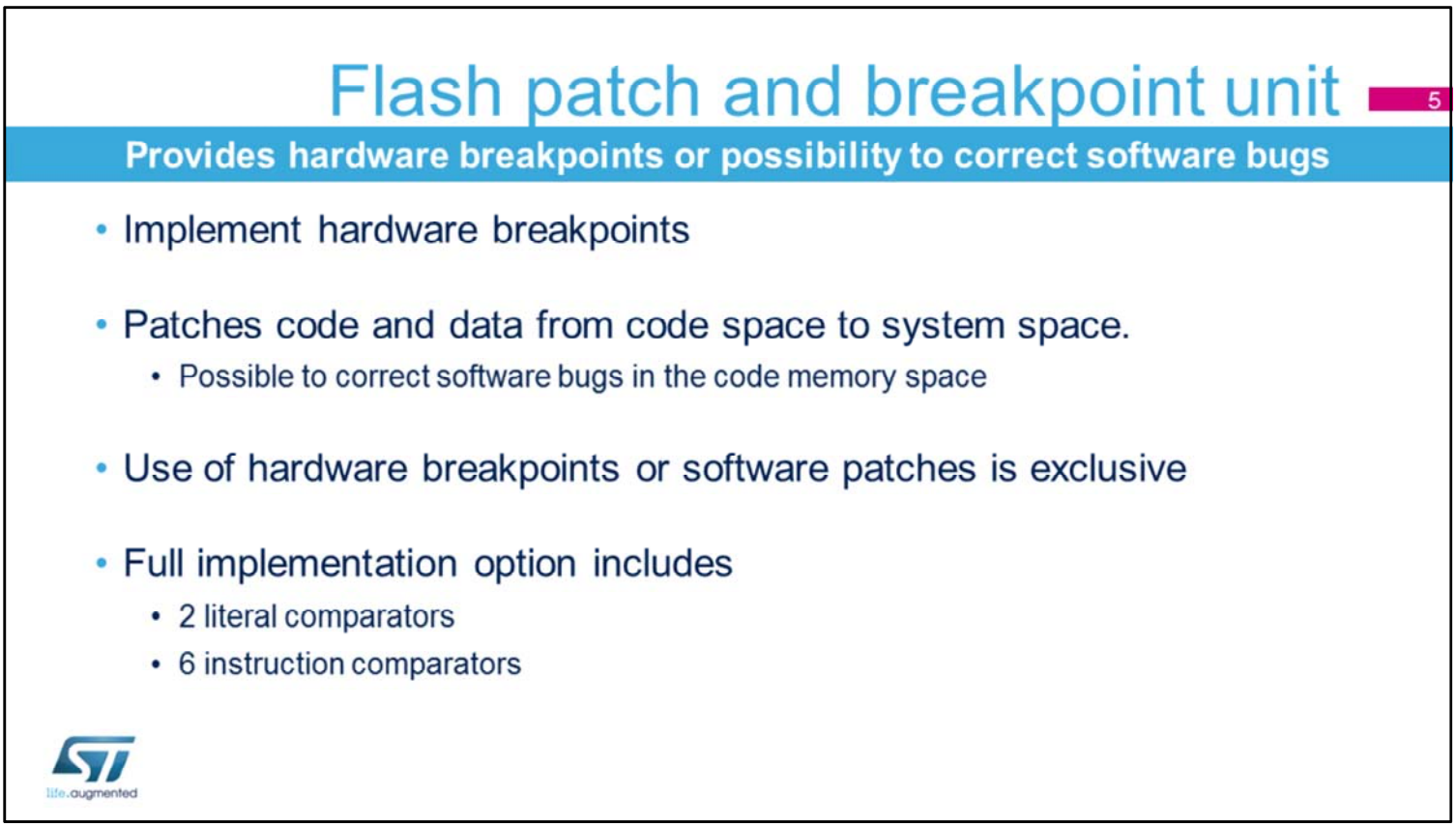
- Advanced debugging features
  - Requires special support by debugging tools and target device
  - Data watch point trigger
  - Instrumentation trace macrocell
  - Flash patch breakpoints
  - Trace point unit interface *
  - Embedded trace macrocell *
  - Boundary scan (on JTAG only)



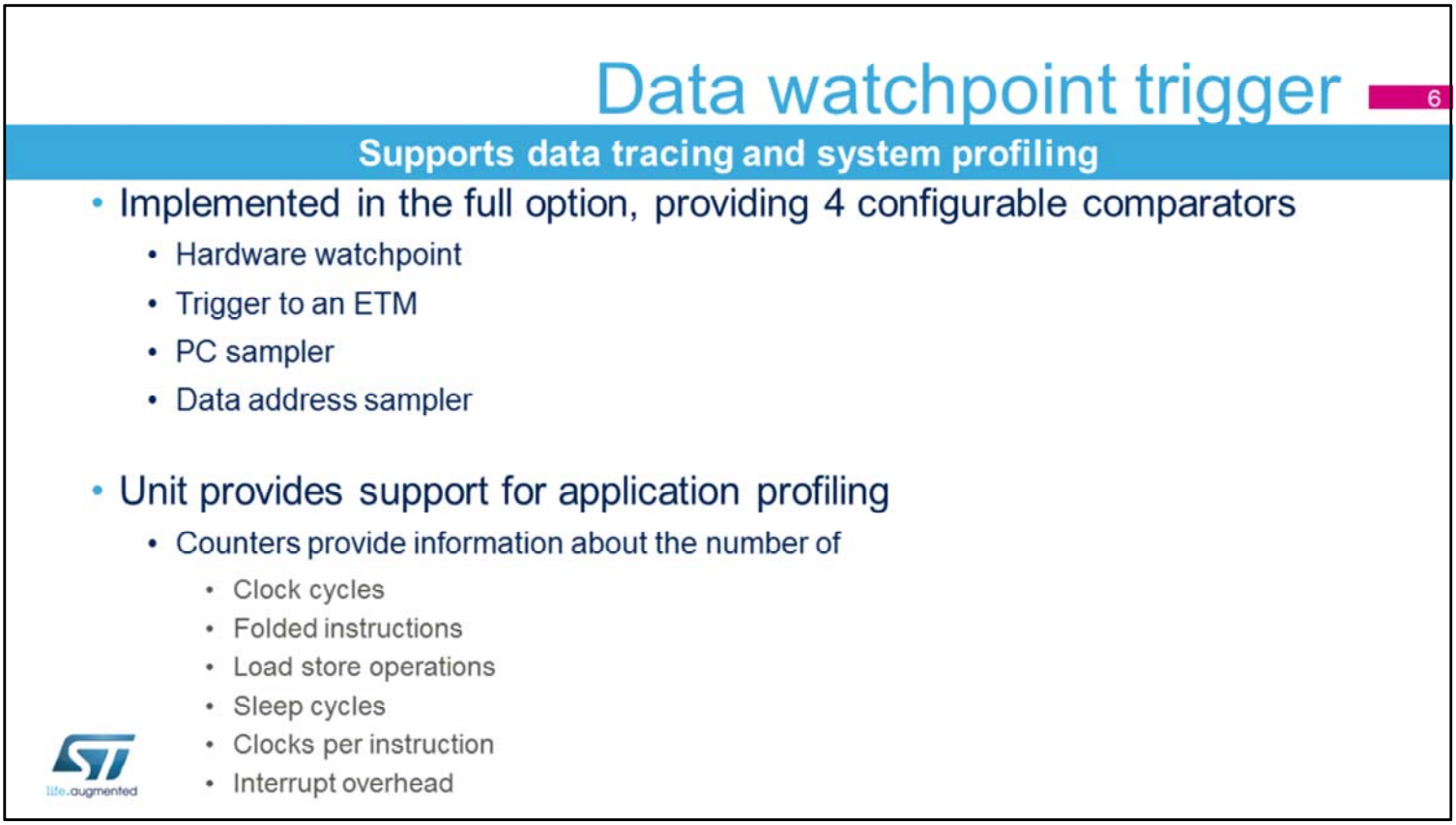
- * These features are only available on large packages where the corresponding pins are available

life.augmented

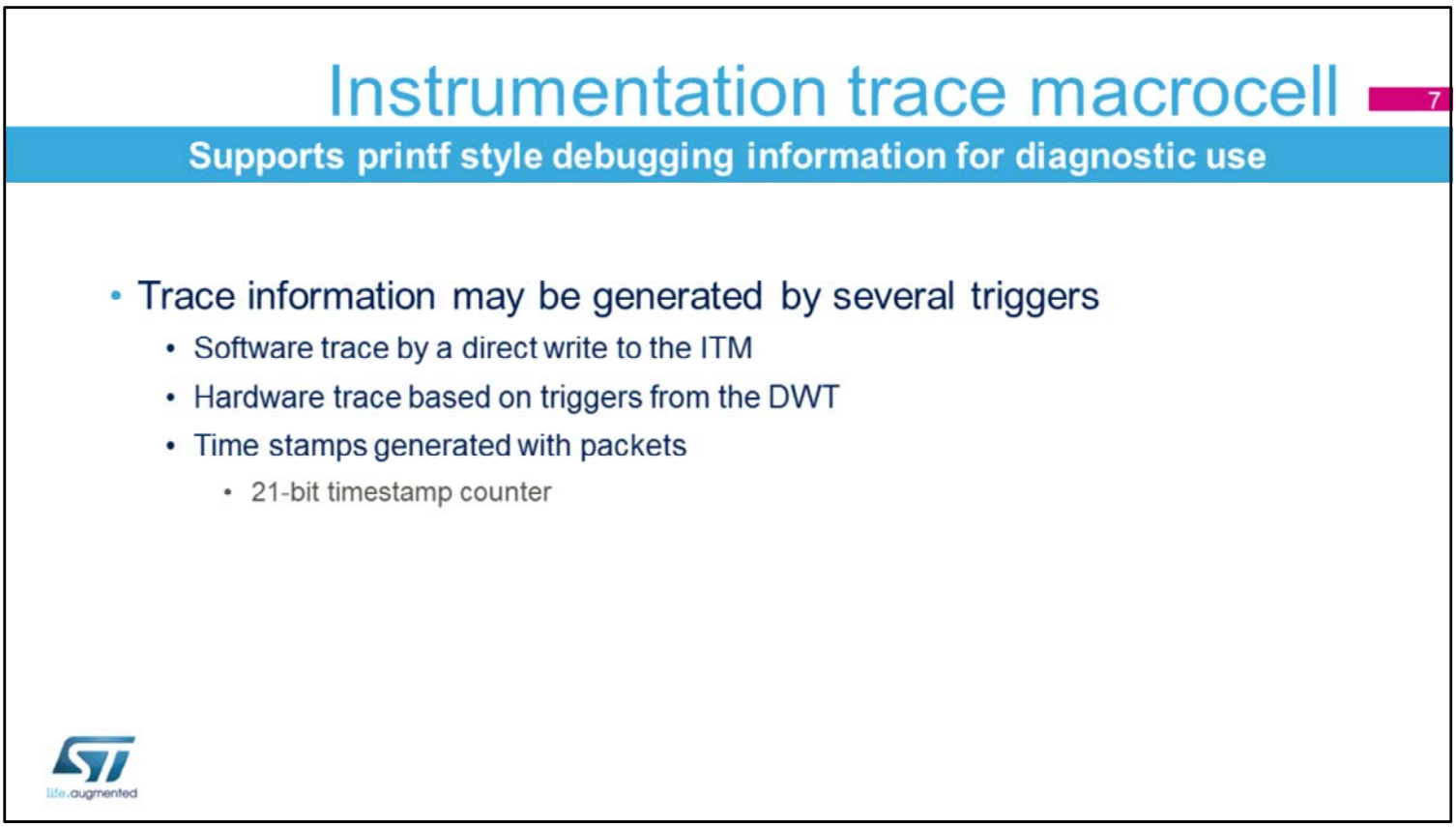
The STM32L4 also supports more advanced debugging features. These features require additional support by the debugging toolset – both hardware and software, as well as support on the MCU side for features requiring additional pins like the embedded trace macrocell (ETM) or the trace point unit interface (TPUI).



The Flash patch and breakpoint unit provides hardware breakpoints for debugging the application or possibly correcting software bugs in the code memory space. The full implementation option offers two literal comparators and six instruction comparators.



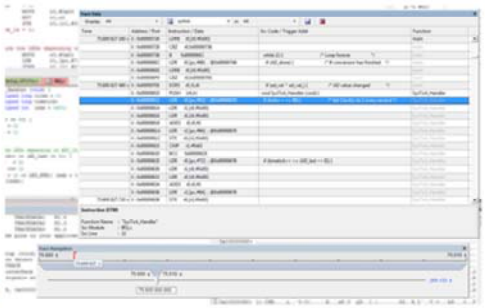
The embedded data watchpoint trigger provides four comparators configurable as a hardware watchpoint, ETM trigger, PC sampler or data address sampler. It provides the necessary information for data tracing and system profiling analysis, for which it embeds counters for counting the number of clock cycles, load and store operations, sleep cycles, clocks per instruction and also information about interrupt overhead. It can also generate reports about the application profile.



The instrumentation trace macrocell (ITM) supports printf style debugging information for diagnostics. Packets can be invoked by software – a direct write to the ITM or by hardware – triggered from the data watchpoint trigger (DWT).
It also provides a timestamp from the 21-bit counter.

Embedded trace macrocell

- **Supports reconstruction of the program execution flow**
- Data traced through DWT or ITM
- Instructions traced through ETM
- Trace data stored in the debug tool or sent to the PC



Application benefits

- Quick identification of malfunctioning firmware thanks to reconstruction of the program flow
- Code coverage and profiling reports for tests and certifications.

life.augmented

The embedded trace macrocell (ETM) provides information about the execution flow of the application by tracing data through the DWT or ITM and tracing instructions through the ETM. This information is then sent to the debugger host for processing.
This information allows the debugger to completely reconstruct the execution flow. It is very useful to quickly identify bugs and also generate code coverage and profiling reports, which are used for test purposes and certifications.

Trace port interface unit

9

Format and send on-chip trace data from ITM and ETM

- Asynchronous mode
  - 1 extra pin is needed
  - Only in Serial-wire debug port mode
- Synchronous mode
  - 2 to 6 extra pins needed depending on the trace port width
  - Offers better data throughput
- Pins are not assigned after reset
  - Debugger host is responsible for their configuration

life.augmented

The trace port interface unit formats information from the on-chip trace units – ITM and ETM – and sends them to the debugger host.
It supports Asynchronous mode with one pin used for communication in Single-wire mode, or Synchronous mode with up to 5 pins working in both JTAG and Single-wire modes. Synchronous mode provides better data throughput.
After a device reset, these pins are not assigned and must be configured by the debugger host.

Flexible SWJ-DP pin assignment

Unused debug pins may be used by applications such as GPIO

- After reset, all serial-wire JTAG debug port (SWJ-DP) pins are assigned to the debug interface
  - Does not include trace outputs
  - Debug pins are connected by internal pull-up or pull-down resistors after reset
- Some pins may be released by the application for standard GPIO functionality
  - By reconfiguring the alternate function register

Available debug ports	PA13	PA14	PA15	PB3	PB4
Full SWJ	X	X	X	X	X
Full SWJ without NJRST	X	X	X	X	
JTAG-DP disabled, SW-DP enabled	X	X			
Both JTAG-DP and SW-DP disabled					

life.augmented

The STM32L4 supports flexible debug pin assignments. After a reset, all five debug pins are assigned to the debug interface. The application may reconfigure them back for GPIO operation and release these pins for application use.
Please note that these debug pins have internal pull-up or pull-down resistors enabled after a reset to prevent any uncontrolled IO levels on these pins.

Low-power modes

Mode	I/O Description
Run	Active.
Sleep	Active. DBG_SLEEP must be previously set by debugger.
Low-power run	Active.
Low-power sleep	Active. DBG_SLEEP must be previously set by debugger.
Stop 0/Stop 1	Active. DBG_STOP must be previously set by debugger.
Stop 2	Active. DBG_STOP must be previously set by debugger.
Standby	Active. DBG_STANDBY must be previously set by debugger.

life.augmented

The debug interface operates in all low-power modes. For Sleep, Stop and Standby modes, related bits must be configured in the DBGMCU_CR register in order to prevent the clock and regulators from stopping when entering a low-power mode.

References

- For more details, please refer to the following sources:
  - Cortex®-M4 r0p1 technical reference manual
    - search for “Cortex®-M4 technical reference manual” at http://infocenter.arm.com
  - ARM® debug interface
  - ARM® CoreSight design kit, revision r0p1 - Technical reference manual

life.augmented

For more technical details, visit the infocenter.arm.com website.