life.augmented

STM32U0 – DBG
Debug and trace

Hello, and welcome to this presentation of the STM32 debug interface. It covers the debug capabilities offered by STM32U0 devices.

Overview

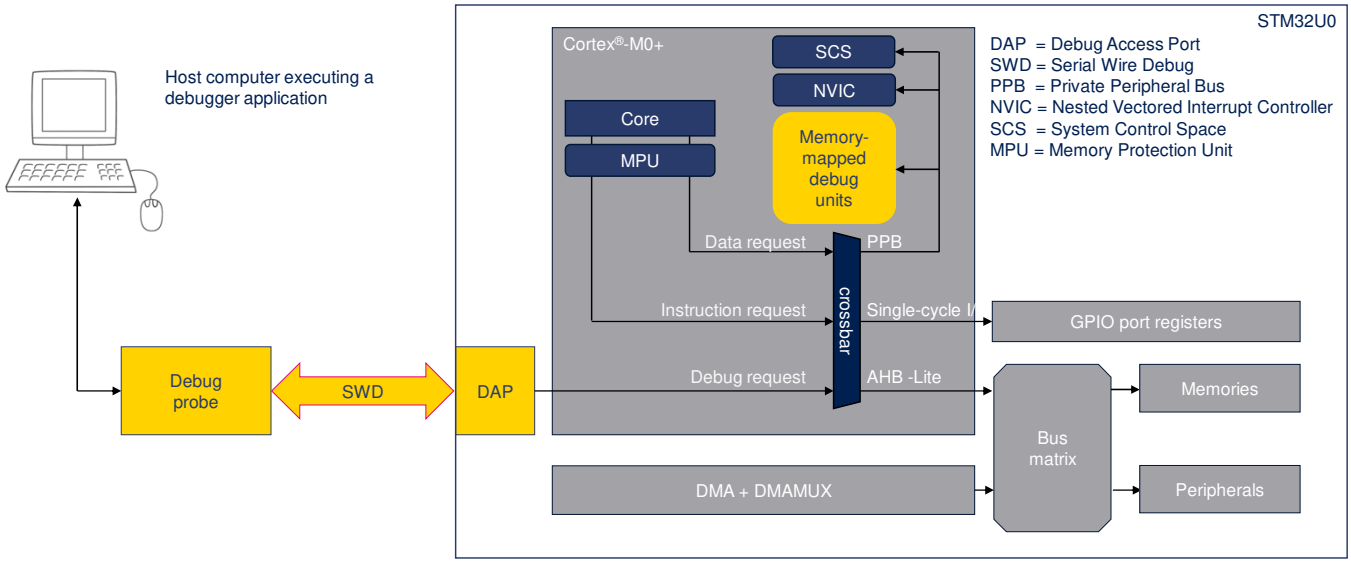
- STM32U0 provides rich support for debug
  - Download programs into RAM or Flash memory
  - Examine memory and register contents
  - Insert breakpoints and halt the processor
  - Run or Single-step through programs
- Based on Arm® CoreSight™ architecture
  - Wide range of compatible tools
  - Serial Wire Debug standard interface to debug probe such as ST Link

life.augmented

2

The STM32U0 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.

Debug architecture



life.augmented

3

The Debug Access Port (DAP) enables an external debug probe to access any memory-mapped resources also accessible from the Cortex®-M0+ core.
Note that the Memory Protection Unit (MPU) does not intercept the requests initiated by the DAP.
The Serial Wire Debug (SWD) protocol used to connect the debug probe to the Cortex®-M0+ relies on 2 wires, and is appropriate for the STM32U0 low pincount packages, starting with 20 pins only.

Debug port

- The debugger accesses the STM32U0 via the SWD debug port
  - Serial wire debug (SWD) port uses only 2 port pins
  - When debug is not required, these debug pins can be reallocated for functional use

	PA13	PA14
SWD pinout	SWDIO	SWCLK
General Purpose IO	GPIO	GPIO

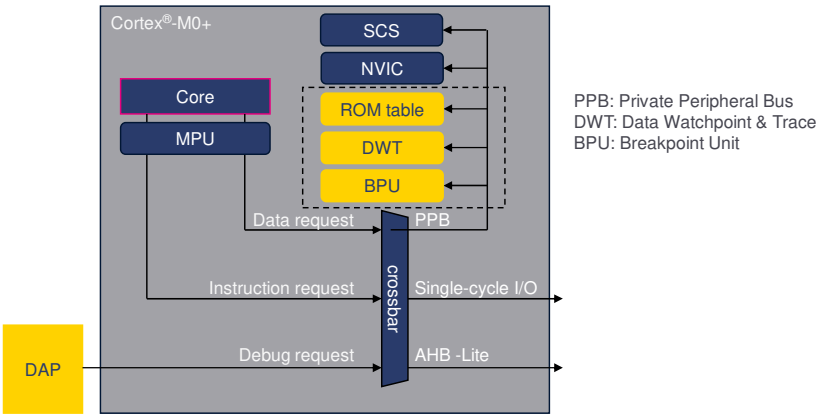
  - Upon reset, these pins are configured as SW debug alternate functions, and the internal pull-up on PA13 pin and the internal pull-down on PA14 pin are activated

life.augmented

The configuration for debug requires that pins PA13 and PA14 be allocated to serial wire debugging (SWDIO and SWCLK respectively).
ST-Link, and most 3rd party debug adaptors (for example, Ulink), support serial wire debug.

Cortex-M memory-mapped debug units

- The Cortex-M cores contain the following debug units:
  - System Control Space (SCS)
  - Data Watchpoint and Trace Unit (DWT)
  - Breakpoint Unit (BPU)
  - ROM table



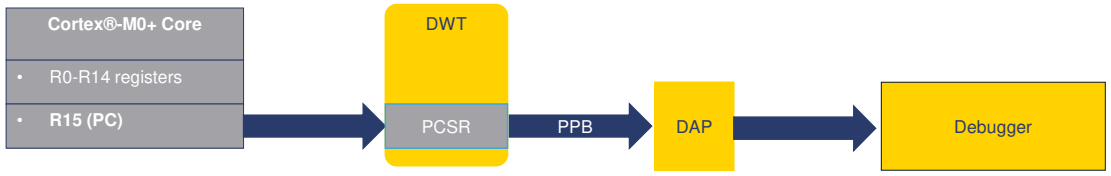
life.augmented

All units involved in the debug process have memory-mapped registers accessible through the Private Peripheral Bus by both the Core and the DAP.
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 Breakpoint Unit connected to the Private Peripheral Bus while the processor is executing instructions.
The ROM table contains pointers to the base addresses of each debug component visible from the Core.
The DAP contains a read-only register pointing to the ROM table. This is used by some debug tools to automatically detect the topology of the CoreSight™ infrastructure in the target.
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.
The other units are described in the following slides.

Data Watchpoint and Trace Unit

- The DWT provides two 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
- The unique trace capability is provided by the PC Sampling Register (PCSR)



life.augmented

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.
The BPU 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. However, the current value of R15, which is the Program Counter, is readable in a memory-mapped register contained in the DWT called the PCSR. The debugger can read the PC value without halting the Cortex®-M0+ Core.

Breakpoint Unit

- The BPU allows hardware breakpoints to be set
  - It contains four comparators which 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
  - The breakpoint unit only supports address comparisons in the range 0-0x1FFFFFFF, corresponding to the Flash memory in the STM32U0
  - Beyond address 0x20000000, software breakpoints can be used when the code is executed from RAM

life.augmented

The STM32U0 supports four hardware breakpoints, used by the debugger to set breakpoint in non-volatile memories.
The breakpoint unit only supports address comparisons in the range 0-0x1FFF_FFFF, corresponding to the STM32U0's Flash memory. Beyond address 0x2000_0000, software breakpoints can be used when the code is executed from RAM.
Regarding software breakpoints, the debugger replaces the instruction on which the user wants to stop with a dedicated instruction called BKPT.

Security

- In order to protect user code and ensure that the debug features cannot be used to alter or compromise the normal operation of the finished product, these features can be disabled

Debug state	Authentication signal state	Description
OPEN	dbgen=1	Debug is enabled
CLOSED	dbgen=0	Debug is disabled

- Debugger access is disabled while the processor is booting from system flash memory

life.augmented

The trace and debug components allow a high degree of access to the processor and system during product development. The debug facilities are automatically disabled when the product life cycle state transitions from OPEN to CLOSED. Independently of life cycle state, the debugger access is also temporarily disabled during the execution of ST proprietary bootloader.

Security

- The product life cycle determines the current debug state

Product life cycle state	Debug state	Authentication signal state	Description
OPEN	OPEN	dbgen=1	Debug is enabled
CLOSED	CLOSED	dbgen=0	Debug is disabled
LOCKED	CLOSED	dbgen=0	Debug is disabled

- When the product is in the field (state = CLOSED or LOCKED), debug is disabled

life.augmented

The product life cycle indicates whether the product is under development or in the field.
When development is in progress, state is OPEN, that enables debug.
When the product is in the field, state is CLOSED or LOCKED, that disables debug.

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, LPTIM, and watchdog (IWDG, WWDG) timer counters, while the processor is halted

life.augmented

10

The DBGMCU is located on the debug APB bus and can be accessed by the debugger via the APB access port. It is also accessible by the processor in the debug APB address space.
The DBG_IDCODE register provides the device ID and revision codes in STM32 standard format. This code is accessible by the serial debug port (two pins) or by 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.

Thank you

life.augmented

Thanks for attending this presentation.