SILICON LABS

ARM Cortex-M and RTOSs Are Meant for Each Other

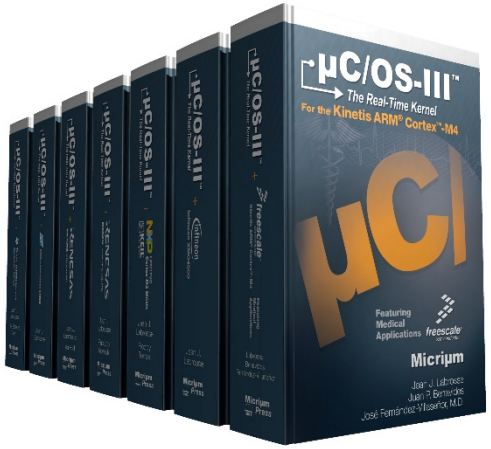
FEBRUARY 2018

JEAN J. LABROSSE

Introduction

Author

µC/OS series of software and books
Numerous articles and blogs



Lecturer

Conferences
Training

Entrepreneur

Micrium founder (acquired by Silicon Labs in 2016)

Embedded Systems Innovator

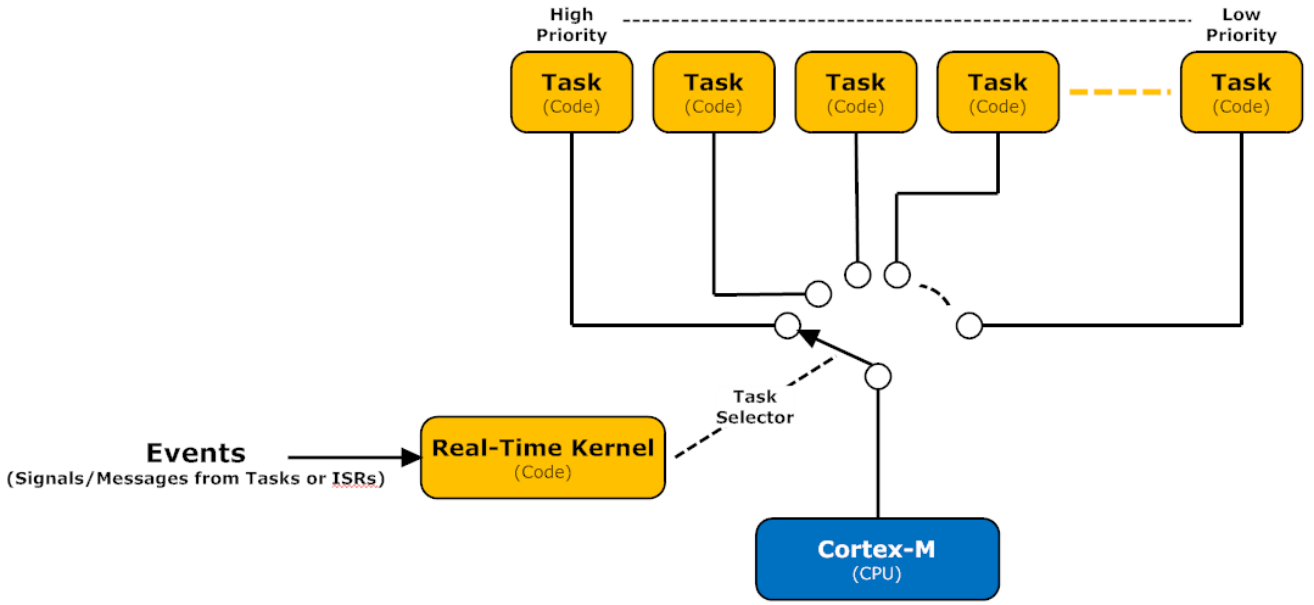
Embedded Computer Design Innovator of the Year award (2015)

Jean.Labrosse@SiLabs.com

www.SiLabs.com/EW2018

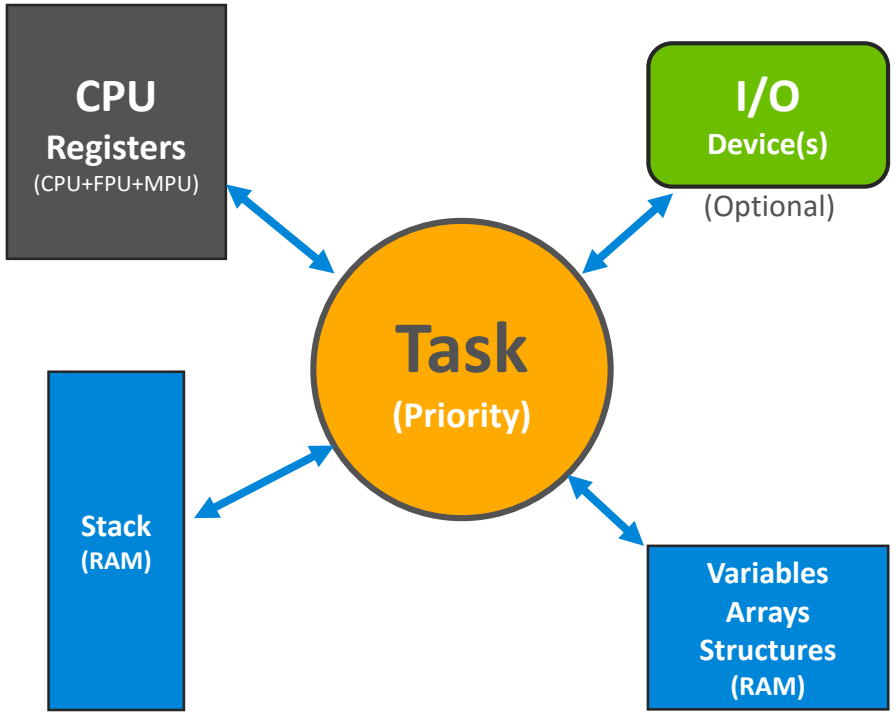
What Is an RTOS? - Multitasking

- Software that manages the time of a CPU
  - Application is split into multiple tasks
  - The RTOS's job is to run the most important task that is ready-to-run

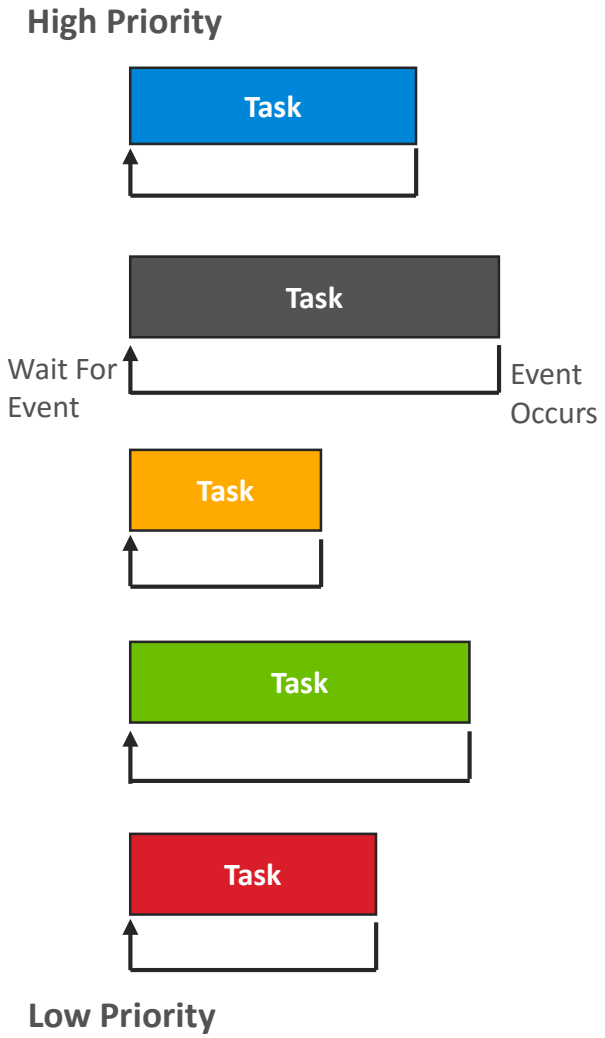


What Is an RTOS? – Tasks

- Each task:
  - Is assigned a *priority* based on its importance
  - Its own set of CPU registers (*Thinks* it has the CPU all to itself)
  - Requires a *stack*
  - Manages its own variables, arrays and structures
  - Possibly manages I/O devices
  - Contains *YOUR* application code

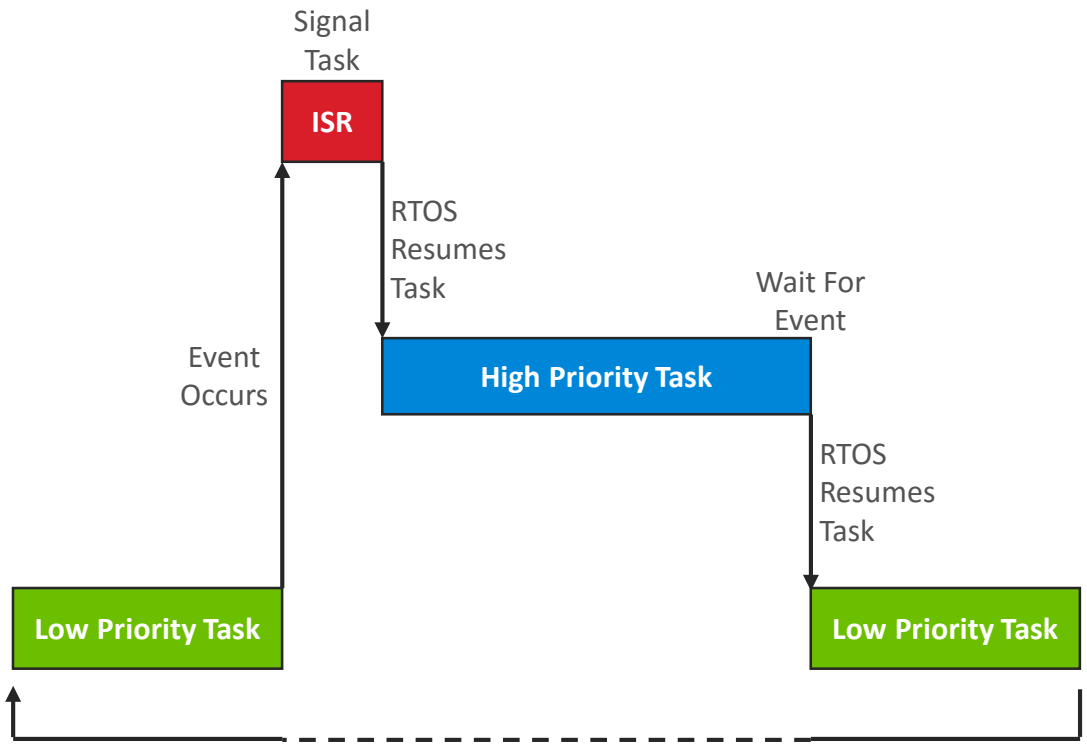


What Is an RTOS? – Event Driven



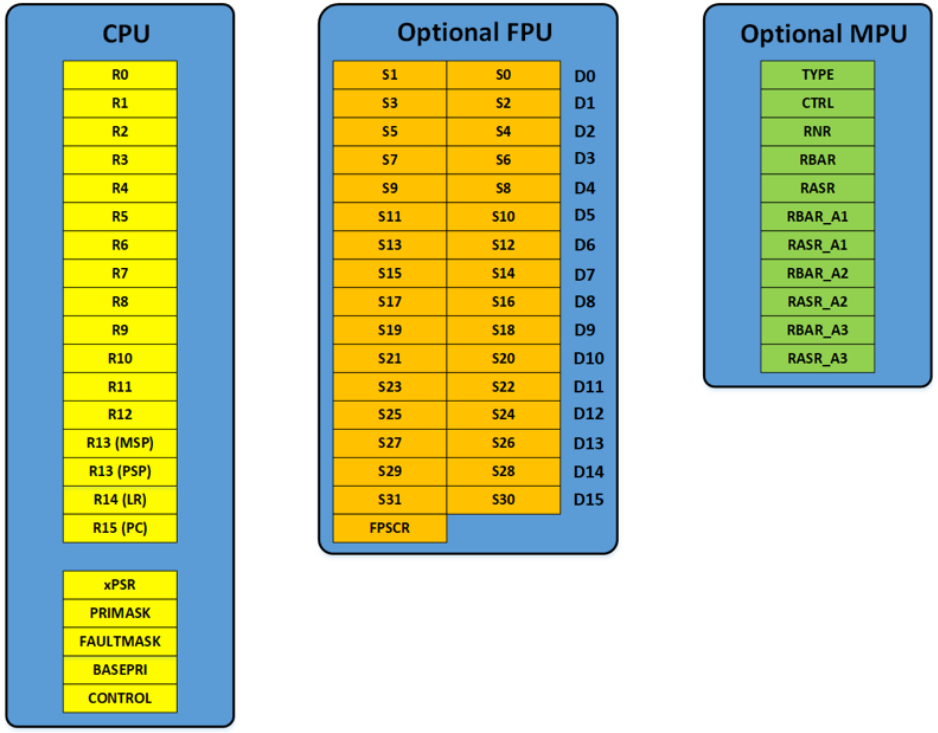

```
void Task (void)
{
  Task initialization;
  while (1) {
    Wait for event to occur;
    Perform task operation;
  }
}
```

What Is an RTOS? – Preemptive Scheduling



ARM – Cortex-M - Introduction

- A family of 32-bit processors
  - Introduced in 2004
  - RISC architecture
- Many derivatives
  - Cortex-M0/M0+ ARMv6-M
  - Cortex-M3 (MPU) ARMv7-M
  - Cortex-M4 (MPU, FPU, DSP) ARMv7-M
  - Cortex-M7 (MPU, FPU, DSP, Cache) ARMv7-M
  - Cortex-M23 (MPU) ARMv8-M
  - Cortex-M33 (MPU, FPU, TrustZone) ARMv8-M
- The most popular CPU architecture in the world!

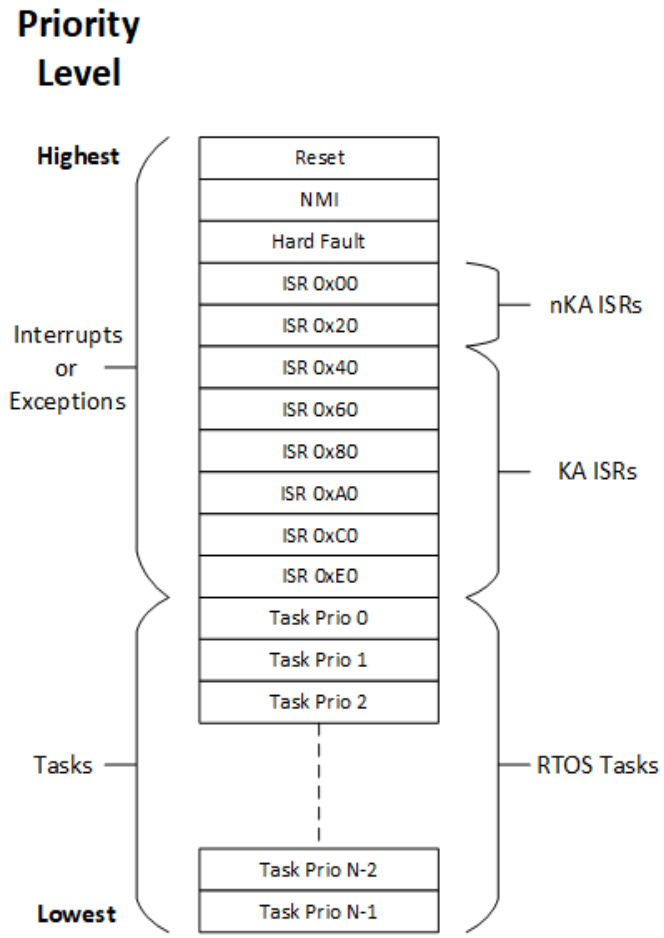


ARM – Cortex-M – Features Summary

- Privileged / non-privileged modes
- Mandatory NVIC
- Support of nKA and KA
- Easy to disable/enable interrupts
- Dedicated stack for ISRs
- Exception handlers / ISRs can be written in C
- Tail chaining
- Dedicated timer for RTOS tick
- Dedicated context switch handler
- CLZ for scheduling
- Low-power mode via WFI
- MPU
- ARMv8-M
  - Stack Limit Registers
  - TrustZone
  - Dual MPU
- CoreSight Debug Port

ARM – Cortex-M - Features

- Privileged / non-privileged modes
  - System code (ISRs, Some Tasks) runs in privileged mode
    - Enable/disable interrupts (both at the CPU and NVIC)
    - Changing MPU (if present) settings
  - Tasks run with limited privileges
- Mandatory NVIC (Nested Vectored Interrupt Controller)
  - Same for all Cortex-Ms
  - Supports up to 256 exception/interrupt sources
  - Each exception/interrupt can be assigned a priority
- Support of nKA and KA
  - nKA (non-Kernel Aware) have **near** immediate response
  - KA (Kernel Aware) interact with tasks

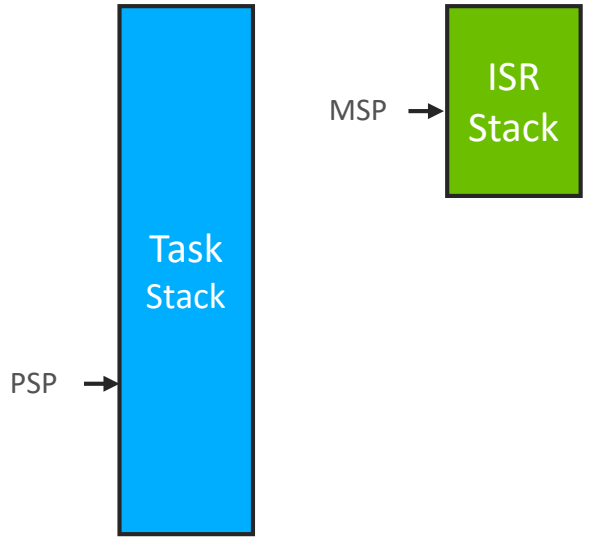


ARM – Cortex-M - Features

- Easy to disable/enable interrupts

```
KA_InterruptDisable:
    Save current BASEPRI;       // Save current interrupt disable level
    Set BASEPRI to 0x40;        // Set nKA boundary
KA_InterruptsEnable:
    Set BASEPRI to saved value; // Restore previously saved interrupt level
```

- Dedicated stack for ISRs
  - Tasks uses the PSP (Process Stack Pointer)
    - Half of the CPU registers are pushed onto the PSP upon entering the ISR
    - Half of the FPU registers are pushed onto the PSP (if an FPU is present)
      - Support of Lazy Stacking
  - ISRs use the MSP (Main Stack Pointer)
    - Avoids having to oversize the stack of each task for ISR code

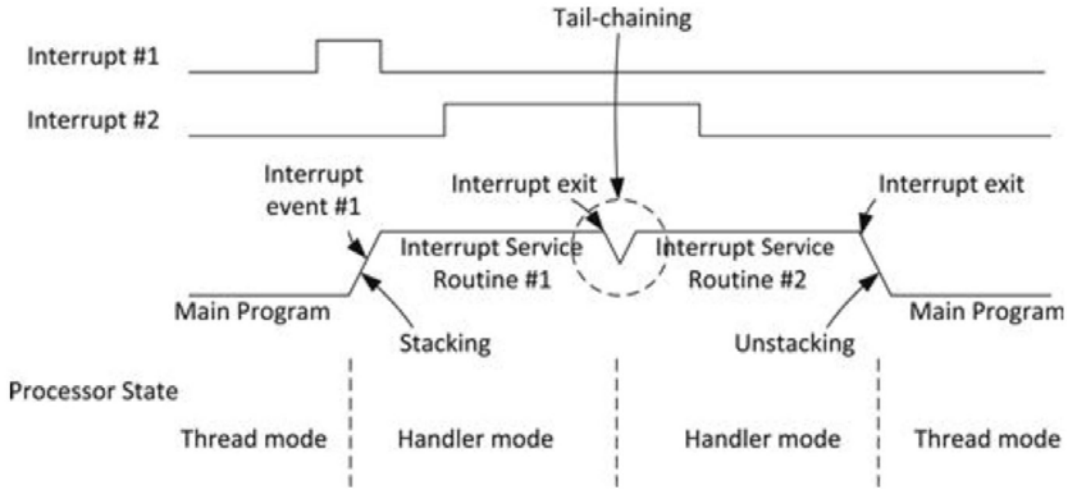


ARM – Cortex-M - Features

- Exception handlers / ISRs can be written in C

```
void  MyISR (void)
{
    // ISR code
}
```

- Tail chaining



- Dedicated timer for RTOS tick (SysTick)
  - 24-bit hardware timer reserved for RTOSs

- Dedicated context switch exception handler (PendSV)
  - Runs at the lowest priority

ARM – Cortex-M - Features

- CLZ for scheduling

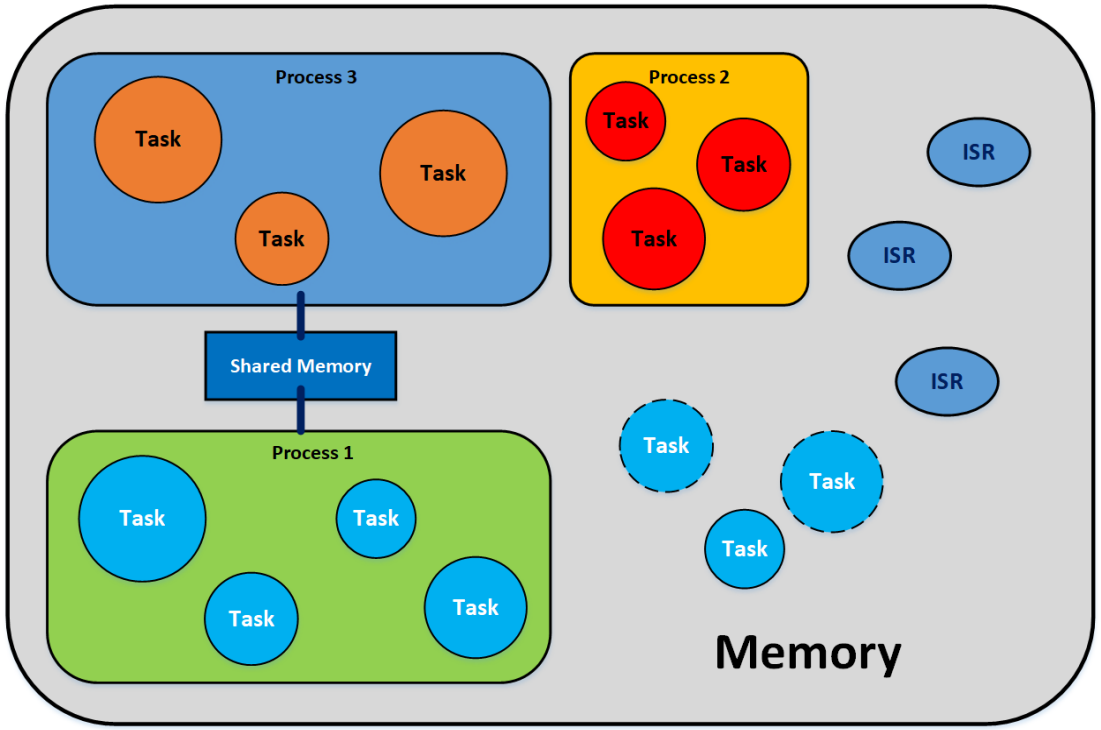
Bitmap of Task Priorities that are Ready-to-Run

- Low-power mode via WFI (Wait-For-Interrupt)

```
void OSIdleTask (void)
{
    while (1) {
        WFI();  // Enter Low Power Mode
    }
}
```

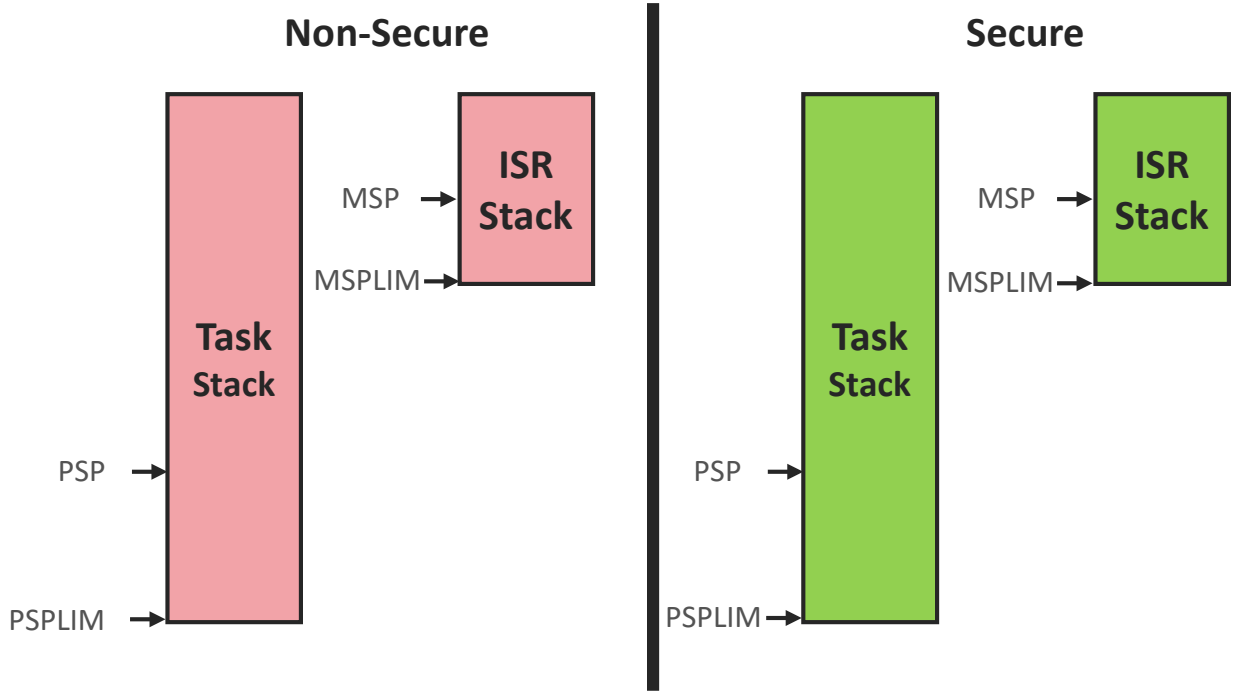
ARM – Cortex-M - MPU

- Process separation
- Prevent tasks from corrupting stack or memory of other tasks
- Prevent unprivileged tasks from accessing peripherals
- Prevent execution out of RAM
  - Prevents code injection attacks
- MPU registers loaded at context switch

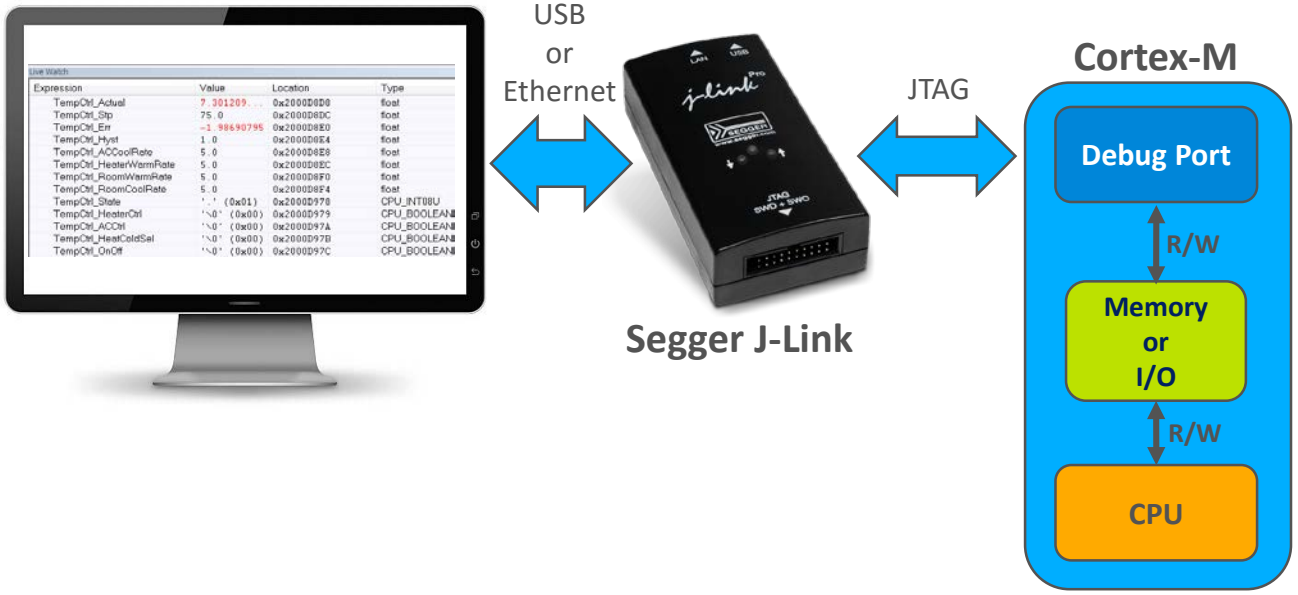


ARM – Cortex-M – ARMv8-M

- TrustZone™
  - Isolation of code/data
  - Improve security
- Stack limit registers
  - Stack overflow detection
- Dual MPU
  - One for non-Secure and one for secure



ARM Cortex-M - Debug Port



- Core debugging:
  - Halting
  - Single stepping
  - Resume
  - Reset
  - Register accesses
- Up to 8 hardware breakpoints
- Up to 4 hardware watchpoints
- Optional ***instruction*** trace
- ***Data*** trace
- Instrumentation trace (printf() like) – 32 channels
- Profiling counters
- PC sampling
- **On-the-fly memory and I/O accesses**
  - Can be a security risk for deployed systems though

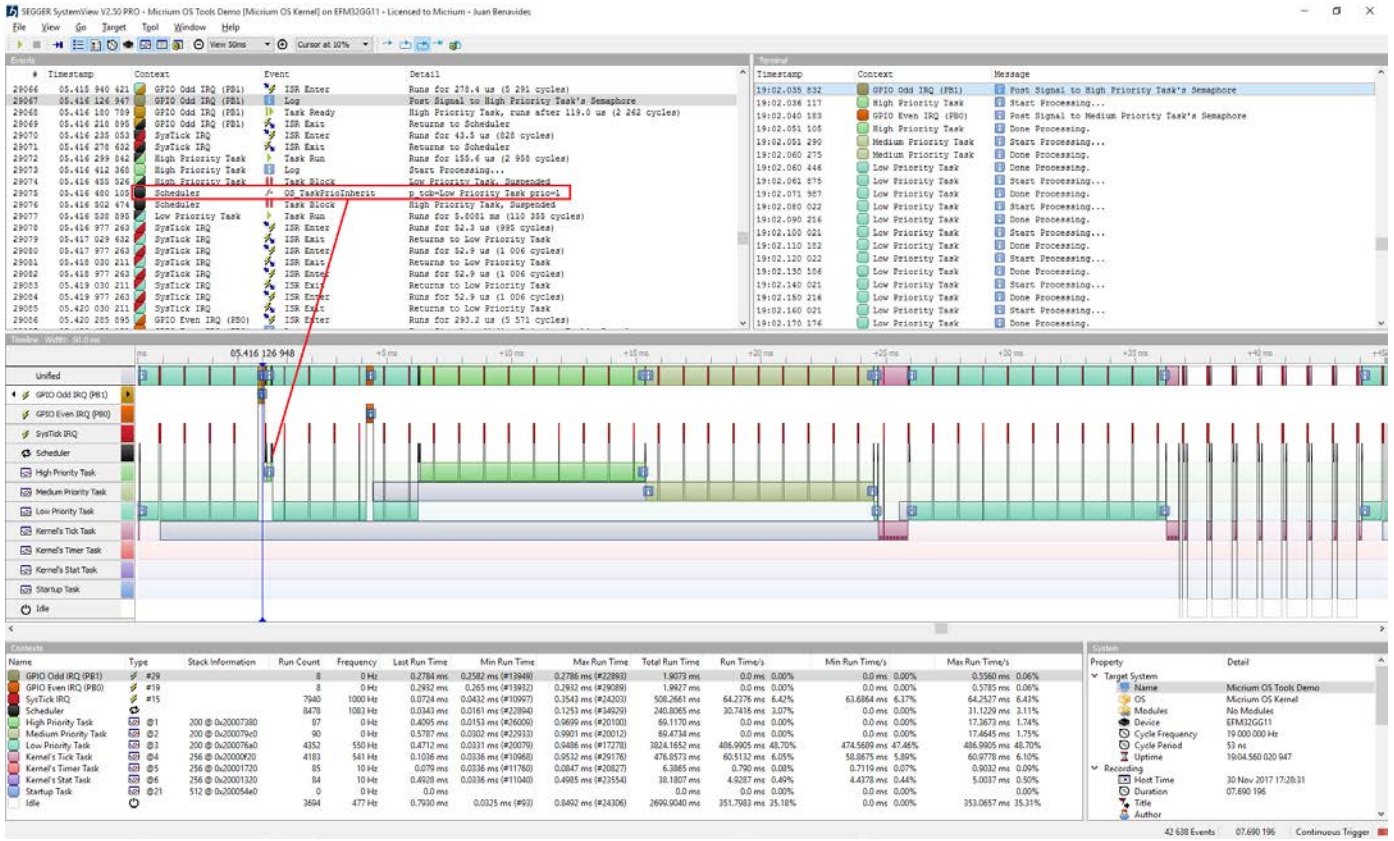
ARM Cortex-M – Debug Port

Micrium's µC/Probe™

- Allows for tools to monitor an embedded system ***non-intrusively***
  - Memory or I/O values can be displayed or changed ***LIVE*** and ***graphically***
- Live RTOS awareness
  - Task state stack usage
  - CPU usage
  - Run counters
  - Interrupt disable time
  - Kernel objects
  - Etc.
  - www.micrium.com

ARM Cortex-M – Debug Port

Segger's SystemView™



- Displays the execution profile of RTOS-based systems
  - Displayed **live**
    - Trigger on any task or ISR
  - Visualizing the execution profile of an application
  - Helps confirm the expected behavior of your system
- Measures CPU usage on a per-task basis
  - Min/Max/Avg task run time
  - Counts the number of task executions
- Display the occurrence of 'events' in your code
- Traces can be saved for post-analysis or record keeping

- www.Segger.com

Conclusion

- ARM's Cortex-M was designed from the get-go to be RTOS friendly
  - Privileged / non-privileged modes
  - Mandatory NVIC
  - Support of nKA and KA
  - Easy to disable/enable interrupts
  - Dedicated stack for ISRs
  - Exception handlers / ISRs can be written in C
  - Tail chaining
  - Dedicated timer for RTOS tick
  - Dedicated context switch handler
  - CLZ for scheduling
  - Low-power mode via WFI
  - MPU
  - ARMv8-M (Stack Limit Registers, TrustZone, Dual MPU)
  - CoreSight debug port
- Once an RTOS is ported to a Cortex-M, the RTOS works on **all** Cortex-M
  - From any manufacturer

Thank you!

SILABS.COM