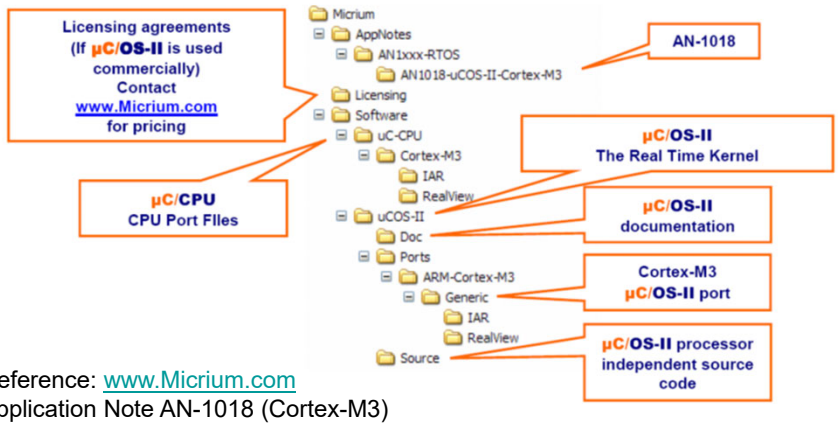
EE445M/ECE380L.12
Embedded and Real-Time Systems/ Real-Time Operating Systems

Lecture 13:
Commercial RTOS,
Final Exam, Review

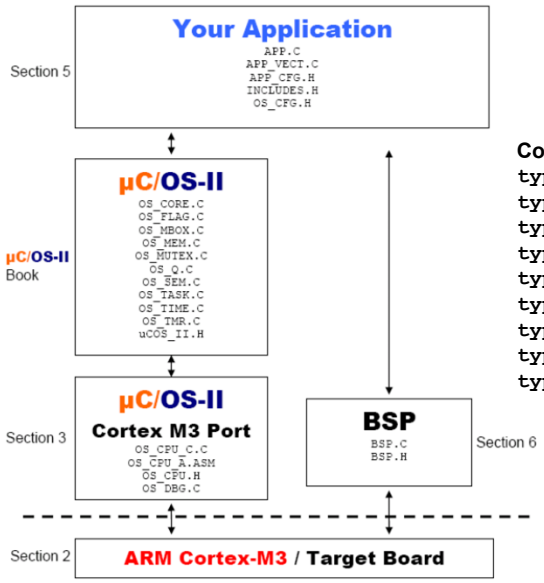
Putting it All Together

- Micrium µCOS-II



Reference: www.Micrium.com
Application Note AN-1018 (Cortex-M3)
µC/OS-II and µC/OS-III by Jean J. Labrosse

µCOS Portability



Compiler independent data types

```
typedef unsigned char  BOOLEAN;
typedef unsigned char  INT8U;
typedef signed   char  INT8S;
typedef unsigned short INT16U;
typedef signed   short INT16S;
typedef unsigned int   INT32U;
typedef signed   int   INT32S;
typedef float          FP32;
typedef double         FP64;
```

3

Hooks

- Run user supplied code at strategic places
- Allows you to
  - Extend the OS
  - Implement debugging
  - Implement performance testing
  - Implement black box recording
- Collect run-time performance data

µCOS Runs with PSP

```
OS_CPU_PendSVHandler
    CPSID   I     ; Prevent interruption during context switch
    MRS     R0, PSP         ; PSP is process stack pointer
    CBZ     R0, OS_CPU_PendSVHandler_nosave  ; Skip save the
                                             ; first time

    SUBS    R0, R0, #0x20   ; Save remaining regs r4-11
                            ; on process stack
    STM     R0, {R4-R11}

    LDR     R1, =OSTCBCur   ; OSTCBCur->OSTCBStkPtr = SP;
    LDR     R1, [R1]
    STR     R0, [R1]        ; R0 is SP of process being
                            ; switched out
    …
```


User Can Hook into µCOS

… (context switch continued)

```
PUSH    {R14}                ; Save LR exc_return value
LDR     R0, =OSTaskSwHook    ; OSTaskSwHook();
BLX     R0
POP     {R14}
```

Many hooks provided:

```
OSInitHookBegin()
OSInitHookEnd()
OSTaskCreateHook()
OSTaskDelHook()
OSTaskIdleHook()
OSTaskStatHook()
OSTaskStkInit()
OSTaskSwHook()
OSTCBInitHook()
OSTimeTickHook()
```

Board Support Package (BSP)

- Hardware abstraction layer (HAL)
  - I/O abstraction for anything OS needs
  - Encapsulate functionality of target hardware
    - Timer initialization
    - ISR Handlers
    - LED control functions
    - Reading switches
    - Setting up the interrupt controller
    - Setting up communication channel
    - CAN, I2C, ADC, DAC, SPI, serial,graphics

```
void LED_Init(void);
void LED_On(CPU_INT08U led_id);
void LED_Off(CPU_INT08U led_id);
void LED_Toggle(CPU_INT08U led_id);
```


µCOS Synchronization

- Message mail box
- Message queue
- Semaphores
- Flags (software events)
  - Groups of flags
  - Names
  - pend/post, and/or

µCOS Mutex

```
/* Description: This function waits for a mutual exclusion semaphore.
Arguments  : pevent  pointer to event control block associated with mutex.
             timeout optional timeout period (in clock ticks).
               If non-zero, your task will wait up to the specified time
               If you specify 0, however, will wait forever for resource
             perr   pointer to where an error message will be deposited.
                    OS_ERR_NONE         successful and your task owns the mutex
                    OS_ERR_TIMEOUT      not available within the 'timeout'.
                    OS_ERR_PEND_ABORT   mutex was aborted.
                    OS_ERR_EVENT_TYPE   If you didn't pass a pointer to a mutex
                    OS_ERR_PEVENT_NULL 'pevent' is a NULL pointer
                    OS_ERR_PEND_ISR     called from an ISR
                    OS_ERR_PIP_LOWER    task priority that owns is HIGHER
                    OS_ERR_PEND_LOCKED called when the scheduler is locked
* Returns    : none
* Note(s)1) The task that owns the Mutex MUST NOT pend on any other event while it
*           owns the mutex.
*        2) You MUST NOT change the priority of the task that owns the mutex
*/
void   OSMutexPend (OS_EVENT *pevent, INT16U timeout, INT8U *perr)
INT8U  OSMutexPost (OS_EVENT *pevent)
```


Other µCOS Features

- Memory manager
- Time delay (sleep)
- Priority resolution table
- Debugger aware

Reference: www.Micrium.com
µC/OS-II and µC/OS-III books by Jean J. Labrosse

Final Exam

- Saturday, May 14, 9am-noon, JGB 2.216
  - Open book, open notes, open web
  - No electronic devices other than laptop
  - No communication, all phones off
- Comprehensive (see Study Guide)
  - Book Chapters 1-9
  - Lectures 1-12
  - Labs 1-7

Lab Topics

- Lab 1
  - Interrupts, Cortex M architecture, FIFOs, serial port, ADC
- Lab 2
  - Real time OS, semaphores, critical sections, synchronization, communication
- Lab 3
  - Priority scheduling, blocking semaphores, debugging
- Lab 4
  - File system, SPI, SD cards
- Lab 5
  - Memory & process management, process loading & linking
- Lab 6
  - Interfaces, CAN/Ethernet, TCP/IP, networking, distributed systems
- Lab 7
  - Applications, abstractions, sensing, actuating and control

Topics Not in Labs

- DMA
- PID and ~~Fuzzy logic~~ control
- Ethernet (vs. CAN), TCP/IP
- Virtual Memory, Paging
- Synchronization & Communication
  - Monitors
  - KPNs
  - Design and implementation of thread flags OS_Wait_Event_Or, OS_Wait_Event_And, OS_Trigger_Event(Thread)

General Topics

- Software development
  - Data flow graphs, call graphs
- I/O
  - Device drivers
  - CPU bound, I/O bound
- Debugging
  - Intrusiveness, stabilization, profiling, dumps, monitors
- SPI/SSI, disk/flash concepts
  - Synchronization, Bandwidth, Protocol
- Networking fundamentals
  - How does CAN/Ethernet and TCP/IP work
- Memory management
  - Virtual memory, paging, page table, TLB

General OS Concepts

- Kernel, Hooks
- Deadlock (detection, prevention)
- Aging, Starvation
- Race condition, Critical Section
- Reentrancy, Mutual exclusion, Atomic
- Bounded waiting
- ROMable, Portability, Scalability
- CPU utilization, latency, jitter
- FIFO queue/pipe implementation, usage

OS Design & Implementation

- Interrupts
  - Arm, enable
  - Protocol, interrupt processing on TM4C123
  - Interrupt priority
- Context switch
  - PSP/MSP, Stack
- Scheduling
  - Round robin, priority based, rate monotonic
  - Linked list, TCB

Synchronization

- Semaphores
  - Binary, counting
  - Spin lock, Blocking
- Mailbox, FIFO
- Monitors
- Path expression
- Implementation, applications
  - Little book of semaphores
  - Study lecture examples, old exams

High-Speed I/O

- Hardware FIFOs
- Dual-port, banked memory
- DMA Concepts
  - DMA controller
  - Cycle steal, Burst
  - Single address, dual address
  - Latency, Bandwidth

File Systems

- SD card interface
  - Bandwidth
  - DMA
- Internal/external fragmentation
- Free space management
- Disk block allocation
  - Contiguous, linked, indexed
  - First fit, best fit, worst fit
  - Directory
- Linked, Indexed, FAT

Memory & Process Management

- Heap, Heap Manager
  - Dynamic memory allocation, malloc/free
- Processes (vs. threads)
  - Creation, termination, PCB
- Loading, linking, relocation
  - ELF files
  - Position-independence, dynamic linking, SVC
- Protection
- Virtual memory
  - Address translation
  - MMU, paging, swapping, TLB

Networking

- Framing and Messaging
- Layering
- CAN & Ethernet concepts
  - Message protocol, arbitration
  - Bandwidth, stuff bits
  - Error detection
- TCP/IP
- IoT, Application-layer protocols

Sensing

- Timed input capture (fundamentals)
- Measuring delay
- Measuring frequency
- Measuring pulse width
- Measuring period
  - Precision
  - Resolution
  - Range

Actuating, Control

- PWM
  - Range, resolution, precision
- Motor interface
  - H bridge
- PID control
  - Show how controller is run within the RTOS
- ~~Fuzzy logic control~~

Old Exams (See Study Guide)

- All Since Spring 2014
  - TM4C, excluding filter questions
- Spring 2013, Spring 2012, Spring 2011
  - All but filters relevant, change LM3S to TM4C
- Spring 2010
  - Change STM32 to LM4C
- Older exam topics no longer covered
  - 9S12 ports, interface, software, paging
  - Memory interfacing, Timing diagrams

http://www.ece.utexas.edu/~valvano/EE345Moldquiz/