Real-Time Operating Systems

Terminology

uC/OS-III, The Real-Time Kernel, or a High Performance, Scalable, ROMable, Preemptive, Multitasking Kernel for Microprocessors, Microcontrollers & DSPs, there are a bunch of versions, with and without a board, Hardcover, by Jean J Labrosse

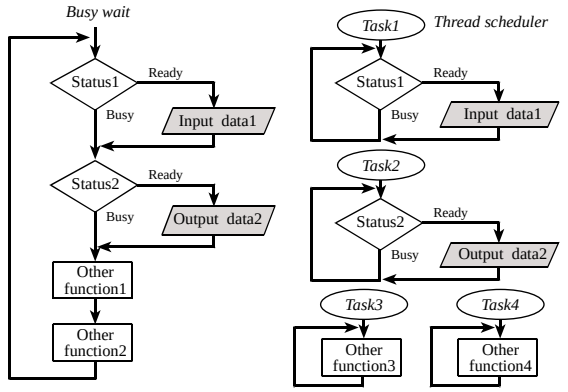
MicroC OS II: The Real Time Kernel, by Jean J. Labrosse , 2002, ISBN 1-5782-0103-9, $72.76

The Definitive Guide to the ARM Cortex-M3 TI, Second Edition, Paperback, Joseph Yiu, $53.95

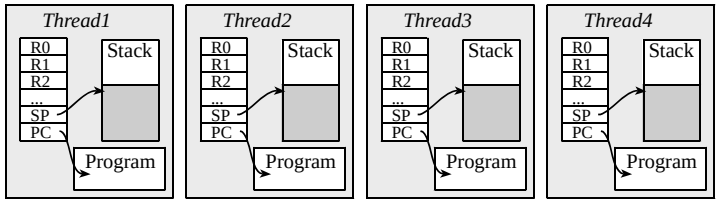
Chapters 5 8 13, **Embedded Microcomputer Systems: Real Time Operating Systems for Arm Cortex M Microcontrollers**, , Jonathan W. Valvano,

Reference McDermott EE382N-4

Single thread vs multithread



What is a thread?



Thread or Task

```
void Producer(void){
unsigned short data;
  data = ADC_In(1);
  if(OS_Fifo_Put(data) == 0){
    DataLost++;
  }
}
```

```
void Consumer(void){
unsigned short data,average;
unsigned long sum;
unsigned short n;
  for(;;){
    sum = 0;
    for(n = 0; n < LENGTH; n++){
      data = OS_Fifo_Get();
      sum = sum + data;
    }
    average = sum/LENGTH;
    OS_MailBox_Send(average);
  }
}
```

```
void Display(void){
unsigned long data,voltage;
  for(;;){
    data = OS_MailBox_Recv();
    voltage = 31*data/64;
    LCD_Message(0,"v(mV) =",voltage);
  }
}
```

Show main, threads in Lab7

Real-time tasks

- Hard real-time
  - Bounded latency
- Soft real-time
  - Execute ASAP
- Not real-time

Thread Classification

- Periodic, execution at regular intervals
  - E.g., ADC, DAC, motor control
  - E.g., Check CO levels
- Aperiodic, execution can not be anticipated
  - Execution is frequent
  - E.g., New position detected as wheel turns
- Sporadic, execution can not be anticipated
  - Execution is infrequent
  - E.g., Faults, errors, catastrophes

Thread Scheduler

- List possible thread states
- List possible scheduling algorithms
  - What?
  - How?
  - Why?
- Performance measures
  - Utilization
  - Latency
  - Bandwidth

When to run scheduler??

Round robin
Weighted round robin
Priority

Static
Dynamic
Deterministic

Priority

- Execute highest priority first
  - Can you have two tasks at same priority?
- Minimize latency on real-time tasks
- Assign a dollar cost for delays
  - Minimize cost

Priority Schedulers

- Earliest deadline first, dynamic
- Earliest slack-time first , dynamic
  - Slack = (time to deadline)-(work left to do)
- Rate monotonic scheduling, static
  - Assign priority based on how often Ti is runs
  - Lower Ti (more frequent) are higher priority

Rate Monotonic Scheduling Theorem

- All n tasks are periodic
  - Priority based on period of T_i
  - Maximum execution time E_i
- No synchronization between tasks
- Execute highest priority task first

$$\sum \frac{E_i}{T_i} \le n\left(2^{1/n} - 1\right) \le \ln(2)$$

Time Management

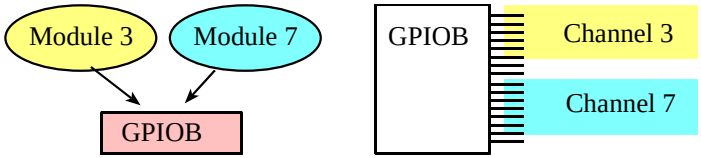
- System time
- Time stamps
  - When did it occur?
  - Performance measures
- Thread sleeping
- Measurements
  - Input capture period -> wheel RPM
  - Input capture PW -> ultrasonic distance

Communication

- Types
  - Data sharing
  - Pipes=FIFO (one to one, buffered, ordered)
  - Mailbox (one to one, unbuffered)
  - Messages (many to many)
- Deadlock
  - prevention, avoidance, detection, recovery
- Performance measures
  - Latency
  - Bandwidth
  - Error rate

Race condition

- Two or more threads access the same global
- At least one access is a write



Race, Critical Sections

- Permanently allocated object
  - Shared variables
  - I/O ports
- Write access changes official copy
- Read access creates two copies
  - Original copy in memory
  - Temporary copy in register
- Nonatomic access, load/store architecture

Reentrant

- Variables in registers, stack
- No nonatomic write sequence
  - Permanently allocated object
  - WR, RMW, WW sequence

Look at programming manual
LDREX STREX
CortexM3Programmer.pdf pg33,71
CortexM3InstructionSet.pdf pg 39

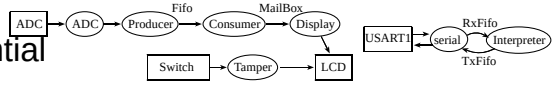
Making the access atomic

- Disable all interrupts
- Lock the scheduler
  - No other foreground threads can run
  - Background ISR will occur
- Mutex semaphore
  - Blocks other threads trying to access info
  - All nonrelated operations not delayed

Measure time with I=1
- Maximum time
- Total time

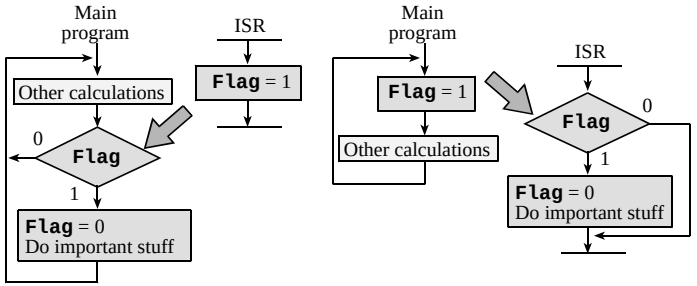
Show code with NestCnt++
If NestCnt-- == 0 then run or don't run scheduler??

Synchronization

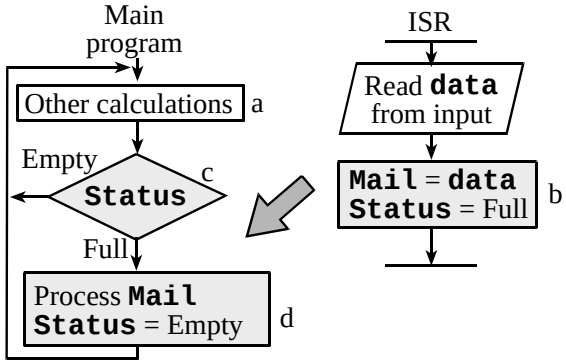


- Sequential
- Fork, spawn, join
- Rendezvous
- Trigger, event flags
  - or, and
  - I/O event (e.g., I/O edge, RX, TX)
  - periodic time (e.g., TATOMIS)
- Sleep

Semaphore



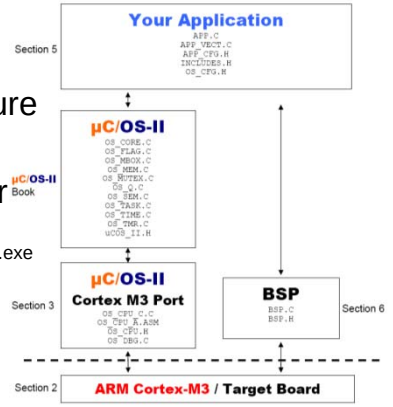
Mailbox



Portability

- Small kernel
- Common structure
- Hardware abstraction layer

Micrium-ARM-uCOS-II-Cortex-M3.exe
Show Micrium directory



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

Additional OS terms

- Run-time configurable
  - Priority, stack size, fifo size, time slice
- Certification
  - Medical, transportation, nuclear, military
- Scalable
  - 10 threads versus 200 threads
- ROMable

Performance measures

- Breakdown Utilization (BU)
  - The percentage of resource utilization below which the RTOS can guarantee that all deadlines will be met. (*shipping company analogy*)
- Normalized Mean Response Time (NMRT)
  - The ratio of the “best case” time interval a task becomes ready to execute and then terminates, and the actual CPU time consumed.
- Guaranteed ratio (GR)
  - For dynamic scheduling, the number of tasks whose deadlines can be guaranteed to be met versus the total number of tasks requesting execution.

Summary

- Threads are executing software
- Synchronization is important
- RTOS has unique requirements
  - Reliability
  - Real-Time
  - Priority
  - Certification
  - Runs in ROM