Semaphores and FreeRTOS

ENCE464 Embedded Software and Advanced Computing

Course coordinator: Steve Weddell (steve.weddell@canterbury.ac.nz)

Lecturer: Le Yang (le.yang@canterbury.ac.nz)

Department of Electrical and Computer Engineering

Where we’re going today

- **Introduction to semaphores**
- Signaling with semaphores
- Mutex and Queues

Introduction to Semaphores

- Semaphore is like an integer with three differences
  - When a semaphore is created, it can be initialized to *any* integer
    - After that only increment (increase by 1) and decrement (decrease by 1) can be performed via **functions**
  - If the result, after a task decrements a semaphore, is negative, the task blocks itself until another task increments the semaphore
    - The task notifies the scheduler (kernel) that it cannot proceed until an event causes the task to become unblocked
  - When a task increments a semaphore, if there are other tasks waiting, one of the waiting tasks become unblocked

From The Little Book of Semaphores.pdf

Language of Semaphores

- increment(), decrement()

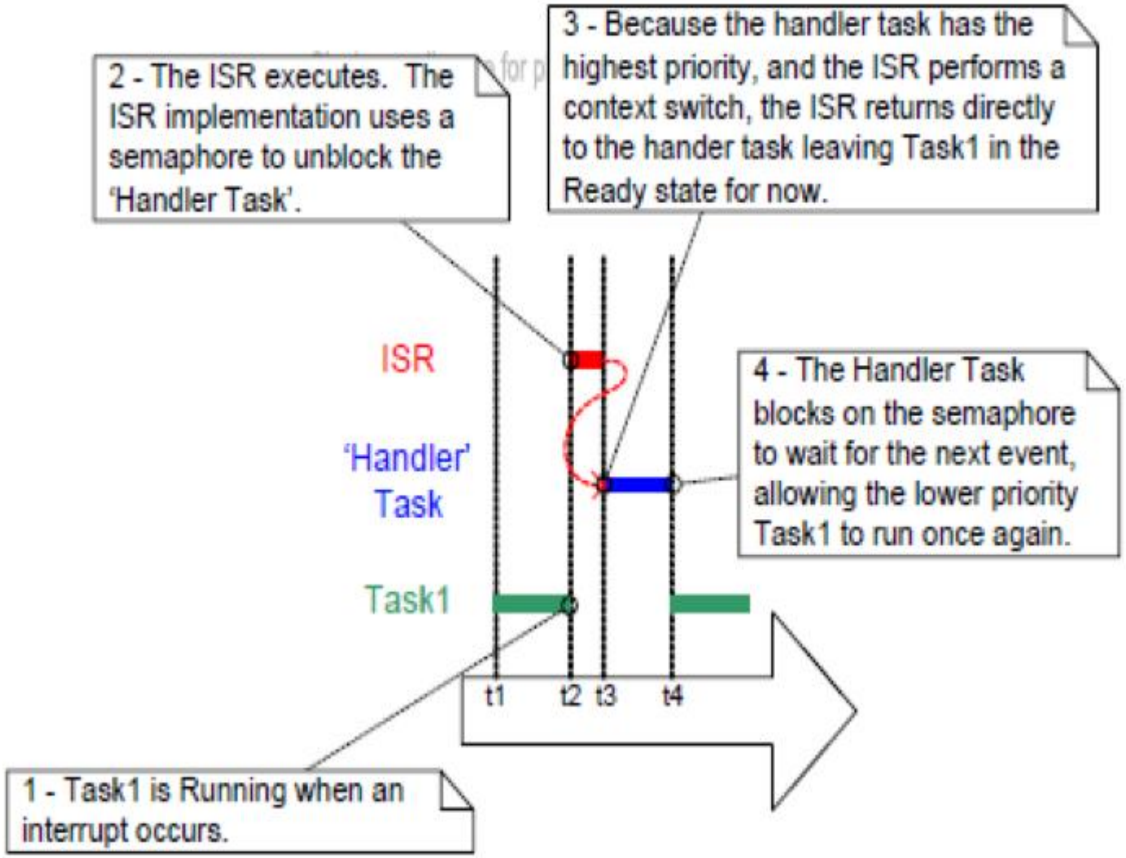
- signal(), wait()

- FreeRTOS
  - xSemaphoreGive(), xSemaphoreTake()
  - xSemaphoreGiveFromISR()

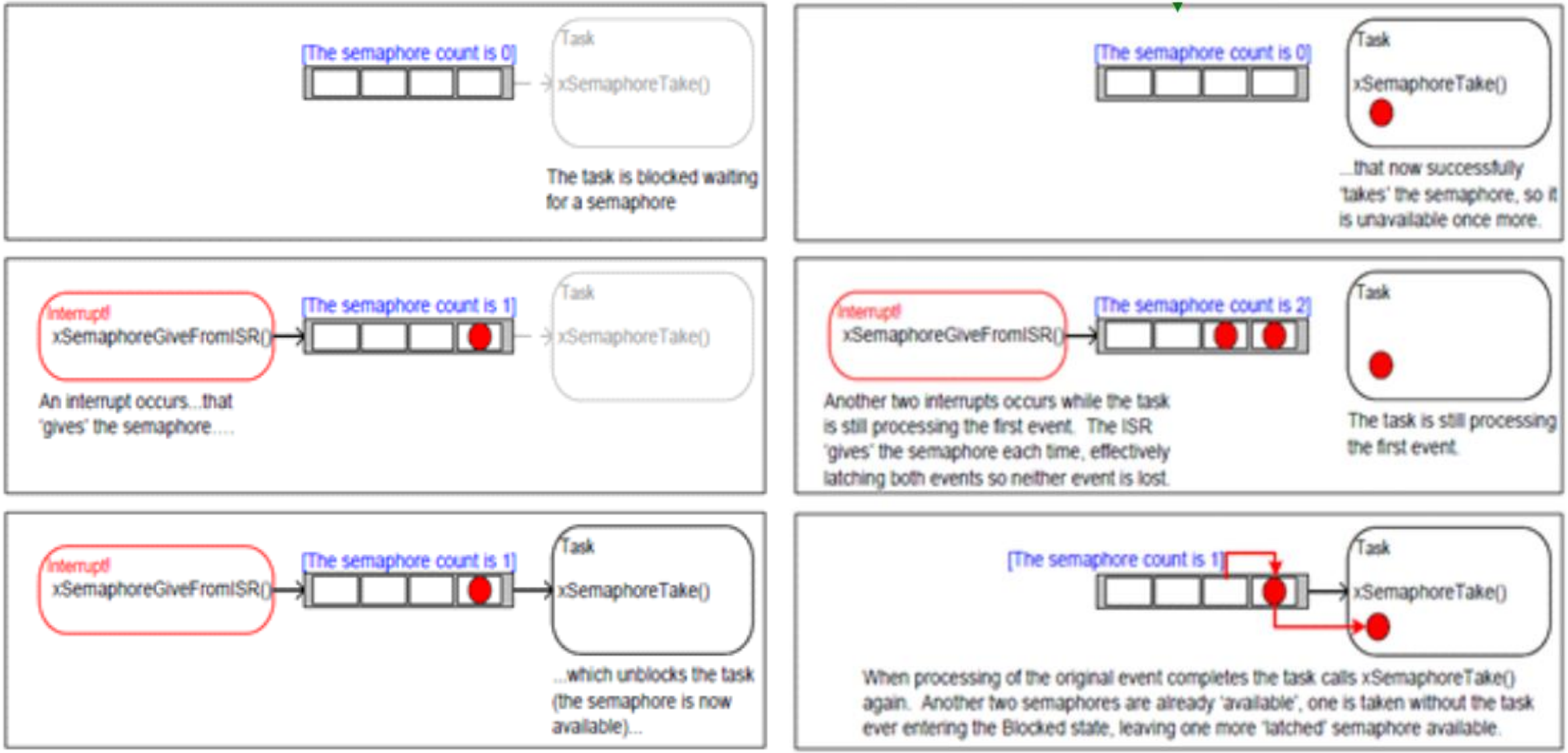
  - xSemaphoreGiveRecursive(), xSemaphoreTakeRecursive()

Example: Deferred Interrupt Structure

- Use a *binary* semaphore to unblock a task each time a particular interrupt occurs
  - Allows the majority of ISR to be realized within the task
  - Interrupt service ‘deferred’ to a ‘handler’ task
- ‘Handler’ task can be set to have high priority so that it can preempt other tasks
  - ISR can include a context switch to ensure that ISR returns to ‘handler’ task
  - Entire processing runs contiguously



Example: Counting Semaphore



Why Using Semaphores?

- Semaphores impose deliberate constraints that can help programmers avoid errors

- Solutions using semaphores are often clean and organized, making it easy to check their correctness

- Semaphores can be implemented efficiently on many systems
  - Solutions using semaphores are portable

Where we’re going today

- Introduction to semaphores
- **Signaling with semaphores**
- Mutex and Queues

Signaling

- Possibly the simplest use of a semaphore
  - A task sends a signal to another task to indicate that an event has occurred
    - Make it possible a section of codes in one task will run before a section of codes in another task
- Example
  - sem initialized to be 0
  - The order of a1 followed by b1 is guaranteed

	Thread A
1	statement a1
2	sem.give()

	Thread B
1	sem.take()
2	statement b1

Signaling with a Semaphore (1)

- Requirement
  - a1 happens before b2
  - b1 happens before a2

	Thread A
1	statement a1
2	statement a2

	Thread B
1	statement b1
2	statement b2

Signaling with a Semaphore (2)

- Is this correct?

	Thread A
1	statement a1
2	bArrived.take()
3	aArrived.give()
4	statement a2

	Thread B
1	statement b1
2	aArrived.take()
3	bArrived.give()
4	statement b2

- Deadlock!

Where we’re going today

- Introduction to semaphores
- Signaling with semaphores
- **Mutex and Queues**

Mutexes

- Mutexes are used to achieve mutual exclusion
  - Control concurrent access to shared resources (e.g., a buffer or a critical variable)
  - Guarantee that only one task accesses the shared variable at a time

- A mutex is like a token that passes from one task to another, allowing one task at a time to proceed

- In order for a task to access a shared variable, it has to obtain mutex first and it releases the mutex when it is done

Queues

- A queue in FreeRTOS holds a finite number of fixed-size data items
  - Maximum number of items a queue can hold is called its ‘length’
  - First In First Out (FIFO) buffer

- Access by multiple tasks
  - Any number of tasks can write to and read from a queue
  - A queue having multiple writers is common
  - A queue having multiple readers is rare

Case Study (1)

- Flight data recorder (FDR)
    - “Blackbox”
    - Record important flight parameters
        - Control and actuator positions
        - Engine information
        - Time of day
        - Minimum 88 parameters under current U.S. federal regulations

        - Each parameter is recorded a few times per second
        - But bursts of data at a much higher frequency are possible

        - Most FDR record approximately 17-25 hours of data in a continuous loop
        - Annual FDR check (readout) is performed annually

Case Study (2)

- Tasks suggested
  - Maintain a real-time clock (F)
  - Monitor signal changes (F)
  - Go into low-power mode on loss of external power (F)
  - Detect unauthorized access attempt (F)

  - Record samples into SRAM buffer
  - Compress and store data in flash memory
  - Maintain status display
  - Monitor and report battery status
  - Dump all data to flash memory
  - Allow download on recovery
  - …