Faculty of Electrical Engineering, Mathematics, and Computer Science
Delft University of Technology

exam – **Embedded Software** – CSE2425

January 26, 2022   13.30 - 15.00

This exam (6 pages) consists of 60 True/False questions.
Your score will be computed as: $max(0, \frac{\#correct}{60} - \frac{1}{2}) \times 2 \times 9 + 1$
It is **not** allowed to consult the book, handouts, or any other notes.

Instructions for filling in the answer sheet:
- You may only use a **B-pencil** so erasures can be applied to correct mistakes.
- Fill in the boxes **completely**.
- Answer **all** questions; there is no penalty for guessing.
- Do not forget to fill in your **Name** and **Student Number**

The following abbreviations are assumed to be known:

- RR (Round Robin)
- RRI (Round Robin with Interrupts)
- FQS (Function Queue Scheduling)
- RTOS (Real-Time Operating System)
- ISR (Interrupt Service Routine)
- UART (Universal Asynchronous Receiver Transmitter)
- ROS (Robotic Operating System)

One system clock tick = 10 ms (unless stated otherwise).

We make use of the following definitions:

```
void delay(int ms) {
   !! do some CPU computation to the number of ms milliseconds
}

void putchar(char c) {
   while (!! UART tx buffer not empty)
       ;

   !! send c to UART tx buffer
}

void puts(char *s) {
   !! write string s using putchar
}
```

1. A defining characteristic of embedded systems is the restricted, or complete lack, of a user interface. true/false

2. An embedded program can be coded as a finite state machine where interrupts trigger state transitions. true/false

3. Several models of computation for embedded systems are described in [Lee:2002].
- Process Networks are primarily used to describe concurrency at the hardware level. true/false

4. Finite State Machines can be coded in VHDL.
- An advantage of doing so is that it results in a fast and predictable process executing on dedicated hardware. true/false

5. Embedded programming is more difficult than "classical" programming because of the event-based programming model. true/false

6. Because embedded software engages the physical world, it has to embrace time and other non-functional properties, which requires the use of interrupt handlers to guarantee responsiveness. true/false

7. The Telegraph System is a prime example of an embedded system as it functions without any input/output devices. true/false

8. The C language is centered around the `int` data type, which is defined to hold 32-bit integral numbers. true/false

9. Valgrind is programming tool that aids memory debugging.
- it does so by executing a program in a safe environment. true/false

10.

```
typedef void *(* resolve)(void *old, void *new);
```

The definition above declares `resolve` as a pointer to a function that takes two void pointers as arguments and returns a void pointer as result. true/false

11.

```
int main(void)
{
   int c;
   statefp state = start;
   while((c = getchar()) != EOF) {
      state = (statefp) (*state)(c);
   }
   return 0;
}
```

The above driver loop for a FSM is interrupt based. true/false

12. Specifying the type of `statefp` is difficult in C because it is recursive and types cannot be referenced before being fully defined. true/false

13. Arrays in C are basically *syntactic sugar* for pointers, and notation may be mixed freely.

```
char hello[] = {'w','o','r','l','d'};
char *ptr = &hello[2];

assert(*ptr == 'o');
```

- the above `assert` holds. true/false

14. When a processor in an embedded system is powered up, interrupts are enabled to meet response-time requirements. true/false

15. To guarantee atomicity critical sections must be disabled. true/false

16. The worst-case latency for servicing an interrupt is a combination of factors, including the longest period of time in which interrupts are disabled. true/false

17. Given is the following RTOS (pseudo) code with priority T1 $>$ T2.

```
void T1(void) {
   while (1) {
      OS_Pend(sem1); // event #1 may unblock any time
      f(1);
      OSTimeDly(1);
   }
}

void T2(void) {
   while (1) {
      OS_Pend(sem2); // event #2 may unblock any time
      f(-1);
      OSTimeDly(3);
   }
}

void f(int i) {
   OS_Pend(mutex);
   counter = counter + i ; // modify some global counter
   OS_Post(mutex);
}
```

This code suffers from a data sharing problem. true/false

18. If the order of events is 1, 2, 1, 2, 1 and they occur within 10 ms from each other, then the final value of the counter will be increased by 1. true/false

19. The function `f()` is reentrant true/false

20. The shared-data problem can be solved through using semaphores. true/false

21. The X32 interrupt controller and CPU use a three-way handshake to change the priority of an interrupt source (i.e. hardware device). true/false

22. Although there are techniques to avoid disabling interrupts, such as alternating buffers or using queues, they are fragile and should only be used if absolutely necessary. true/false

23. An interrupt service routine should restore the context upon entrance. true/false

24.

```
int temp1, temp2;

void isr_buttons(void) // arrive here if a button is pressed
{
   temp1 = X32_PERIPHERALS[PERIPHERAL_TEMP1];
   temp2 = X32_PERIPHERALS[PERIPHERAL_TEMP2];
   ...
}

main() {
   ...
   while (!program_done) {
      X32_display = ((temp1 & 0xff) << 8) | (temp2 & 0xff);
      if (temp1 != temp2) {
         // shutdown plant
      }
   }
}
```

The above pseudo code suffers from the shared-data problem. true/false

25. When detecting a car crash an airbag should not be inflated instantly.
- An RR architecture provides functionality to support such delayed actions. true/false

26. An FQS architecture has a smaller memory footprint than an RTOS as it needs only one stack. true/false

27. The RRI software architecture can be used for implementing an FSM. true/false

28. The software architecture of the Network Simulator NS2 mainly uses the *discrete events* computation model. true/false

29. In the FQS software architecture, using a First-In-First-Out (FIFO) queue effectively leads to RRI. true/false

30. In an RTOS, the system tick can be shortened to fit any required level of accurate timing. true/false

31. In some cases tasks may need to synchronize according to a so-called **multiple-readers / single-writer** pattern. For example, a database server can handle multiple concurrent read actions or one write action. The code below shows how this can be arranged using two semaphores.

Reader	**Writer**

`OS_Pend(lock);`
`if (++cnt == 1) // first reader`
`   OS_Pend(read_or_write);`
`OS_Post(lock);`
`....   // READ action`
`OS_Pend(lock);`
`if (--cnt == 0) // last reader`
`   OS_Post(read_or_write);`
`OS_Post(lock);`
	`OS_Pend(read_or_write);`
	`....   // WRITE action`
	`OS_Post(read_or_write);`

Global variable `cnt` must be initialized to 1. true/false

32. Semaphore `read_or_write` must be initialized to 1. true/false

33. The semaphore `lock` must be initialized to the number of readers. true/false

34. Semaphores can be used for signaling between tasks. true/false

35. Context switching from one task to another is only slightly more expensive than an ordinary function call as the difference is that the stack pointer must be adjusted as well. true/false

36. In the implementation of the `OS_Pend()` primitive, the RTOS first switches the state of the current task to BLOCKED, and then looks for a task in the READY queue.
- if the READY queue is empty the program is deadlocked and may be aborted. true/false

37. Given is the following RTOS (pseudo) code with priority T1 $>$ T2.

```
void T1(void) {
   while (1) {
      OS_Pend(sem1); // event #1 may unblock any time
      f(1);
   }
}

void T2(void) {
   while (1) {
      OS_Pend(sem2); // event #2 may unblock any time
      f(-1);
   }
}

void f(int i) {
   delay(10); // do some computation
   counter = counter + i ; // modify some global counter
   printf("%d\n", counter) ; // print result
}
```

This code suffers from a data sharing problem. true/false

38. If `counter` is set to 15 when event 2 occurs, and event 1 follows 3 ms later, then the first value printed is 14. true/false

39. If the call to `delay` is replaced with `OSTimeDly` the output will be different. true/false

40. To address the shared-data problem, many RTOSs provide communication primitives like queues, mailboxes, and pipes.
- a common pitfall is that they allow pointers to be passed from one task to another. true/false

41. Assume that one system clock tick = 10 ms.
- Calling the function `OSTimeDly(4)` causes a delay of exactly 40 ms. true/false

42. The advantage of pipes over queues is that messages/items can be of variable length. true/false

43. An RTOS usually provides two types of delay functions: polling-based and timer-based.
- polling-based delays are more efficient as other tasks can run while the caller is waiting for the specified time to pass. true/false

44. As the RTOSs is aware of which task is using which semaphore, deadlock can be prevented by delaying the `OS_Pend` operation of the last runnable task. true/false

45. A semaphore S used by task T must be initialized before T is created. true/false

46. The memory footprint of a program grows linearly with the number of tasks. true/false

47. Time-slicing should be avoided in an RTOS because it makes the response time of tasks less predictable. true/false

48. An ISR may not call any RTOS function that might cause a context switch unless the RTOS knows that it is an ISR (and not a task) that is calling. true/false

49. An ISR may not call OS_Pend as that may affect response time and may even cause deadlock! true/false

50. A guiding principle of embedded system design is to only create dynamic tasks when needed, even when the memory footprint is not an issue. true/false

51. The software in ROS is organized in topics. true/false

52. ROS can only run very few nodes simultaneously because resources are limited in embedded systems. true/false

53. `roscore` is the command-line tool to start the ROS master. true/false

54. Bag files can be played back in ROS to the same topic they were recorded from, or even remapped to new topics. true/false

55. The ROS master provides naming and registration services to ROS nodes, and tracks publishers and subscribers to topics and services. true/false

56. Calling the timer interrupt routine directly in the test scaffold on the host machine is preferred. true/false

57. With instruction set simulators, the whole platform can be simulated. true/false

58. Testing on the host machine is useless because most of the embedded code is hardware dependent. true/false

59. A large study of outdoor sensor-network deployments [Beutel:2009] has shown that the one of the most underestimated problems has been the water-proof packaging of the sensor nodes. true/false

60. When debugging code for a distributed sensor network, collecting the (debug) output of the nodes can be arranged in different ways.
- A major advantage of a wireless testbed is that large volumes of (debug) data can be handled. true/false