Part I

Evaluation Page Part B

Name: ______________________________ Signature: ______________________________

1 Evaluation

This part of the exam has 161 questions, with a total of 427 points and 6 bonus points.

Part	Max. Points	Scored Points
A	60	
B	180	
	Total	

2 Evaluation Part B

Page	Points	Bonus Points	Score
5	7	1	
6	12	0	
7	4	0	
8	9	0	
9	10	0	
10	10	0	
11	9	0	
12	9	0	
13	7	0	
14	7	0	
15	10	0	
16	9	0	
17	11	0	
18	11	0	
19	14	0	
20	2	0	
21	2	1	
22	10	0	
23	6	0	
24	10	0	
25	4	0	
26	8	0	
27	5	0	
28	16	0	
29	4	0	
30	5	0	
Total:	211	2	

Page	Points	Bonus Points	Score
31	5	2	
32	4	0	
33	6	0	
34	4	0	
35	7	0	
36	15	0	
37	13	0	
38	11	0	
39	9	0	
40	8	0	
41	10	0	
42	11	0	
43	6	0	
44	12	0	
45	9	0	
46	11	0	
47	12	0	
48	6	0	
49	6	0	
50	3	0	
51	12	0	
52	3	0	
53	9	0	
54	3	0	
55	18	0	
56	3	0	
57	0	2	
Total:	216	4	

Part II
Rules

Answer the questions within the space provided. If you do not have enough space, you can use the backside of the sheet. In that case clearly indicate that your answer continues on the backside.

3 Supporting Materials

This is an examination in writing, without the usage of any electronic devices, except a scientific pocket calculator. No restriction on the model of calculator that may be used, but no device with communication capability shall be accepted as a calculator. All other electronic devices are prohibited. Writing paper is available, writing instruments (pencil, pens, etc) have to be organized by the student.

- **Part A**: Without any supporting material, with calculator.
- **Part B**: With a self written summary (format A4, 8 sheets or up to 16 pages), with calculator

4 Procedure

1. Duration:
   Part A: 1 hour = 60 minutes = 60 points.
   (short break)
   Part B: 3 hours = 180 minutes = 180 points.
2. Sign the first page in the provided space. With this you certify that you are only using permitted support material and you are complying to the rules.
3. Write your name on any detached or additional paper sheets. Sheets without a name will not be evaluated.
4. Use the provided paper for your solutions. Use the provided space in the forms and tables. If needed use scratch paper. Document your way to your solution as appropriate.
5. Each question has a defined number of maximum points associated.
6. If a question is unclear, make reasonable assumptions. Document your assumptions and provide a rationale.
7. Write clearly and legibly. Unclear or multiple solutions will not be evaluated.
8. There is a short break between part A and B. You have to sign into a list for a needed break during the examination parts. Only one person can leave the room for a short time.
9. If something is unclear, ask your supervisor in the room.

5 Time Management

Read first all questions. Make sure you distribute your available time to all the questions. To reduce disturbance, ask questions in the first 15 minutes of the exam period.

6 Multiple-Choice Questions

1. Try to answer all questions if possible. If you are not sure, choose the answer which seems the best one.

2. For the questions of type ○: Choose **exactly one** option with ⊗ (or $\surd$), which you think is the best match. With a correct answer you get the given number of points for that question.

3. For the questions of type ±: After a question or possibly incomplete sentence there are four answers or extensions. Evaluate each of them if they are true or false and mark them accordingly with '+' (true) or '-' (false). Independent if the question is formulated grammatically in singular or plural, it is possible that **0, 1, 2, 3, 4** of the choices are true. For three correct answers out of four you receive half of the points.

4. Wrong answers will have no penalty. Each question which has no answer is treated like a wrong answers and will be evaluated with zero points.

5. If you are changing your mind: cross out your old answer and clearly mark which answer is the new one.

May Dilbert be with you! ☺

Question 1 .... 1 Point (Bonus)

This could be your own bonus question ☺.

1. ____________________

Question 2 .... Points: [2]

In version control systems you are asked to provide a 'commit message': explain why you should provide such a message:

....

....

....

....

....

....

Question 3 .... Points: [2]

In our drivers we implemented `Deinit()` functions, for example for the shell:

```
void SHELL_Deinit(void);
```

Explain the purpose of such Deinit() functions:

....

....

....

....

....

....

Question 4 .... Points: [5]

(a) The boards we used (Robot and FRDM) have different number of LEDs available. Describe your strategy how you can write a common LED driver for all these boards (supporting multiple platforms with a single driver): [3]

....

....

....

....

....

....

....

(b) Describe with your approach how you can deal with a board which has no LED's at all: [2]

....

....

....

....

....

....

Question 5 .... Points: [10]

Given the following interfaces:

```
bool KEY_Get(void); /* return FALSE if key is pressed, TRUE otherwise */
void WAIT_Waitms(uint16_t ms); /* realtime waiting for the given
    milliseconds */

typedef enum {
  NO_KEY,     /* no key pressed */
  SHORT_KEY,  /* short key press */
  LONG_KEY    /* long key press */
} KEY_State;

KEY_State GetKey(void);
```

Implement the function `GetKey()` without the usage of interrupts, with following requirements:

1. Return `NO_KEY` if the key is not pressed.
2. If the key is pressed, debounce it for 50 ms.
3. If the key is pressed for less or equal than 500 ms, return `SHORT_KEY`.
4. For a short key press or long key press detection, the function shall not block longer than needed.
5. If the key is pressed for more than 500 ms, return `LONG_KEY`.

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

Question 6 .... Points: [7]

In INTRO we used a command line shell module and implementation.

(a) What's the advantage of using a shell? [1]

....

....

....

....

(b) List three different USB *device* classes: [1]

....

....

....

(c) USB uses 'non-return-to-zero', so it inserts a zero marker after 6 consecutive one bits. Why? [2]

Question 6 continuous on the next page...

....

....

....

....

....

....

(d) The USB bus protocol uses a frequency of 12 MHz for the bits. Why did we had to configure the microcontroller for 24 MHz operation? [2]

....

....

....

....

(e) What is special with the OTG USB class, compared to host or device classes? [1]

....

....

....

....

Question 7 .... Points: [4]

Explain the difference between a windowed and a non-windowed Watch Dog Timer, and what kind of failures can be catched only with a windowed Watch Dog Timer.

....

....

....

....

....

....

....

Question 8 .... Points: [8]

You are using GIT as VCS. Explain what the following actions mean. Use '*remote repository*', '*local repository*', '*index*' and '*working directory*' in your answers.

(a) clone [2]

. . . .

. . . .

. . . .

. . . .

(b) add [2]

. . . .

. . . .

. . . .

. . . .

(c) push/sync [2]

. . . .

. . . .

. . . .

. . . .

(d) commit [2]

. . . .

. . . .

. . . .

. . . .

Question 9 . . . . Points: [4]

Given the following Watchdog State Machine:

```
void wdt_a(void)
  if (state != 0x5555) {
    HALT;
  }
  state += 0x1111;
}

void wdt_b(void)
  if (state != 0x8888) {
    HALT;
  }
  KickDog();
  if (state != 0x8888) { /* Question: why? */
    HALT;
  }
  state = 0;
}
```

```
void main(void){
  for(;;) {
    state = 0x5555;
    wdt_a();
    ...
    state += 0x2222;
    wdt_b();
  }
}
```

Explain why that second check in `wdt_b()` on `(state != 0x8888)` is necessary:

....

....

....

....

....

....

....

Question 10...Points: [3]

List three different clock sources which can be used for the microcontroller we used in INTRO:

....

....

....

Question 11...Points: [4]

You are using a MEMS accelerometer. This accelerometer should provide the value 0x7FFF (32767) for 0 g, and 0xCFFF (53247) for 1 g. But when you read from the sensor, it reports a value of 33000 for 0 g, and a value of 50000 for 1 g. Determine offset error and gain error factor at 1g:

....

....

....

....

....

....

....

Question 12...Points: [3]

We used in our shell parser the following `xatoi()` interface:

```
byte UTIL1_xatoi(const unsigned char **str, long *res);
```

Explain why it is using `**str` and *not* `*str`:

....

....

....

....

....

....

Question 13 .... Points: [3]

In our labs we used the RTOS with a tick timer period of 10 ms. List advantages and disadvantages for changing it to a period of 1 ms:

....

....

....

....

....

Question 14 .... Points: [3]

Below is an extract of a header file:

```
/* drv.h */
#ifdef DRV_H_
#define DRV_H_
/* header file content follows here... */
#endif
```

Using the above header file, you find out that things are not working properly. Identify the problem.

....

....

....

....

Question 15 .... Points: [3]

Document for following code with doxygen (what the function does, parameter(s)):

```
void LED_On(LED_Set Leds);
```

....

....

....

....

....

....

Question 16 .... **Points: [3]**

FreeRTOS task definitions are using the macro `portTASK_FUNCTION`:

```
static portTASK_FUNCTION(RemoteTask, pvParameters) {
...
}
```

Explain the reason and advantage of using the macro `portTASK_FUNCTION`:

....

....

....

....

....

....

Question 17 .... **Points: [3]**

List advantages and disadvantages of using Processor Expert for a project we did in INTRO:

....

....

....

....

....

....

Question 18 .... **Points: [3]**

The microcontrollers we used in INTRO are using a table for the vectors: The vector table contains the address of the vector function. Explain what will happen during power-on reset if the whole vector table would be filled with zero (0x00) bytes:

Reached: ______
from 9 points

. . . .

. . . .

. . . .

. . . .

. . . .

. . . .

Question 19 . . . . Points: [3]

Explain the fundamental differences between using `EnterCritical()` ... `ExitCritial()` and `xSemaphoreTake()` ... `xSemaphoreGive()` with respect to interrupts:

. . . .

. . . .

. . . .

. . . .

. . . .

. . . .

Question 20 . . . . Points: [4]

On the FRDM board we used an interrupt to detect button presses on the joystick shield.

(a) Why do we not know *which* button was pressed from the interrupt source? [1]

. . . .

. . . .

. . . .

. . . .

(b) How can you know which button is pressed? [1]

. . . .

. . . .

. . . .

. . . .

(c) We get an interrupt for a button. But when we check the pin status in the debouncing state machine we see that this pin is not pressed. What could be the most likely problem? [2]

. . . .

. . . .

. . . .

. . . .

Question 21 . . . . Points: [4]

Figure 1 shows three different circuits how to connect a switch to a microcontroller. Rate them either 'good', 'fair' and 'poor' with a short explanation.

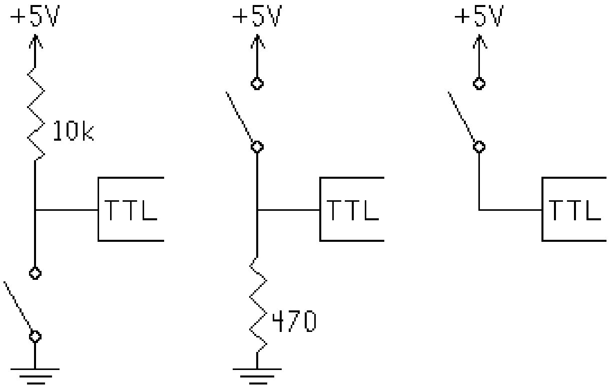

Figure 1: Three Switches

. . . .

. . . .

. . . .

. . . .

. . . .

. . . .

. . . .

Question 22 . . . . Points: [3]

The following `#define` might cause a problem using it:

```
#define INCREMENT(a)    a+1
```

Explain the problem and provide a solution how to fix it:

....

....

....

....

....

....

Question 23..**Points: [3]**

Explain why FreeRTOS is using 'number of addressable units' as parameters to create a task, and not 'number of bytes':

....

....

....

....

....

....

Question 24..**Points: [4]**

An RTOS is using following interface for a 'wait' API call:

```
void vTaskDelay(portTickType xTicksToDelay);
```

(a) Explain why FreeRTOS is using 'number ticks' as parameters to the delay routines and not 'number of milliseconds': [2]

....

....

....

....

....

(b) You need to wait for 200 milliseconds with `vTaskDelay()`, but you do not want to depend on the frequency of the tick timer. How can you do this? [2]

....

....

....

....

....

Question 25..**Points: [3]**

Evaluate the following statements about interrupt system for the FRDM (ARM Cortex M0+ core).

± I cannot change the interrupt priorities of the Cortex M0+ interrupts with positive interrupt vector numbers.

± Multiple interrupts can have the same priority on the Cortex-M0+.

± I cannot change the interrupt priority of the negative interrupt vector numbers.

± The ARM-Cortex M0+ core has only main priorities and no sub-priorities.

Question 26 .... Points: [2]

List two advantages of using an optical quadrature encoder compared to a mechanical one:

....

....

....

....

Question 27 .... Points: [2]

List the 4 specific requirement for a realtime system:

....

....

....

....

Question 28 .... Points: [3]

Explain the fundamental difference between a hard and a soft realtime system:

....

....

....

....

Question 29 .... Points: [2]

Explain the advantages and disadvantages of using Processor Expert for a system we have developed in INTRO:

....

....

....

....

Question 30 .... **Points: [2]**

Explain reasons why you would use a VCS:

....

....

....

....

Question 31 .... **Points: [2]**

Explain reasons when you should use a VCS:

....

....

....

....

Question 32 .... **Points: [2]**

Explain disadvantages and advantages using macros:

....

....

....

....

Question 33 .... **Points: [2]**

Explain why synchronization between systems is needed:

....

....

....

....

Question 34 .... **Points: [2]**

List three different synchronization methods:

....

....

....

Question 35 .... **Points: [1]**

List at least two different output formats produced by doxygen:

....

....

....

Question 36 .... Points: [1]

List a compelling reason why it makes sense to document an API in the sources with doxygen:

....

....

....

Question 37 .... Points: [2]

Explain a case where a synchronization between two systems does not make any sense:

....

....

....

....

Question 38 .... Points: [2]

Explain reasons why we used a Shell in INTRO:

....

....

....

....

Question 39 .... Points: [2]

List reasons why you would use polling instead of interrupts for checking the state of a key:

....

....

....

....

Question 40 .... Points: [2]

List key benefits of using open source operating system like FreeRTOS:

....

....

....

....

Question 41 .... Points: [2]

Explain the difference between a pre-emptive and a cooperative multitasking:

. . . .

. . . .

. . . .

. . . .

Question 42 . . . . Points: [2]

What are stdin, stdout and stderr?

. . . .

. . . .

. . . .

. . . .

Question 43 . . . . Points: [3]

Given in figure 2 a switch on a board: If the switch is **not** pressed, PTA5 will have a logical **HIGH** voltage level. How can this work?

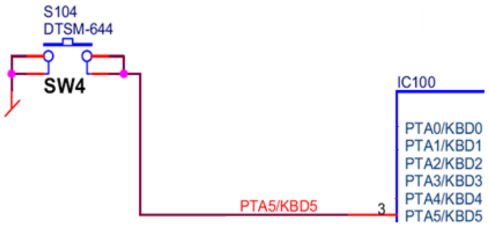

Figure 2: SRB Switch

. . . .

. . . .

. . . .

. . . .

Question 44 . . . . Points: [9]

FreeRTOS is used in priority based preemptive mode. Tick timer is set up for 10 ms, and the processor is running at maximum speed. Task `T3` has priority 3 (highest priority), task `T2` has priority 2 and task `T1` has priority 1. The tasks have been created with `xTaskCreate()` before time t=0 ms, and the scheduler is started with `vTaskStartScheduler()` at the time t=0 ms. Draw in Figure 3 a timing diagram for the execution of the two tasks for the first 100 ms (after t0 = 0 ms). Use a bar to indicate when a task is running. You can ignore the overhead in the task loop and the overhead in the RTOS/scheduler itself.

```
static portTASK_FUNCTION(T3, pvParameters) { /* priority 3 task */
  portTickType xLastTime = xTaskGetTickCount();
  for(;;) { /* task time is 5 ms including overhead */
```

```
    DoWorkFor5ms(); /* this needs 5 ms */
    vTaskDelayUntil(&xLastTime, 25/portTICK_RATE_MS);
  } /* loop forever */
}

static portTASK_FUNCTION(T2, pvParameters) {  /* priority 2 task */
  for(;;) { /* task time is 10 ms including overhead */
    DoWorkFor10ms(); /* this takes 10 ms */
    vTaskDelay(30/portTICK_RATE_MS);
  } /* loop forever */
}

static portTASK_FUNCTION(T1, pvParameters) {  /* priority 1 task */
  for(;;) { /* task time is 2 ms including overhead */
    DoWorkFor2ms(); /* this needs 2 ms */
    vTaskDelay(20/portTICK_RATE_MS);
  } /* loop forever */
}
```

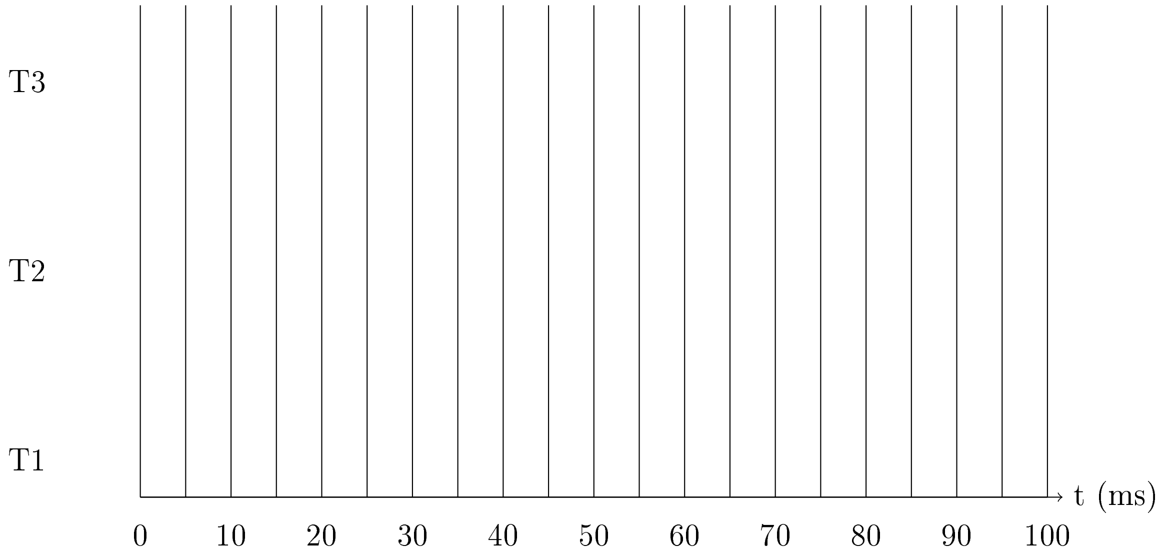

Figure 3: Task Timing

Question 45 .... Points: [2]

Evaluate following statements for the ANSI `strcmp()` and `strncmp()` library functions as used in the INTRO program:

± `strcmp()` has one additional parameter compared to `strncmp()`.

± `strcmp()` and `strncmp()` return both zero, if the two argument strings are equal.

± Using `strncmp()` allow to compare two strings up to a given offset.

± The parameter passing for `strncmp()` is usually more efficient than the parameter passing for `strcmp()`.

Question 46 .... 1 Point (Bonus)

List a compelling reason why normally only debounced switches should be used at an interrupt generating input pin only:

....

....

....

Question 47 .... Points: [2]

Determine the value of variable `j` after calling the function `foo()`:

```
#define SUM(a,b) (a)+b
#define MUL(a,b) a*b
#define CALL(a)  bar(a)

int bar(int i) {
  return MUL(i,SUM(3,MUL(5,10)));
}
int j;

void foo(void) {
  j = CALL(SUM(10,2));
}
```

47. ____________________

Question 48 .... Points: [10]

Consider following semaphore definition:

A semaphore `S` denotes an integer variable, which is accessed through the following two atomic operations:

```
#define DOWN(S)     while(S<=0); /* do nothing */ \
                    S = S-1;
#define UP(S)       S = S+1;
```

Using the two operations, it is possible for a program to implement a critical section or a mutex, as in the following example:

```
Semaphore S = 1; /* initialize semaphore to 1 */
```

Each process or task is able to protect its critical section:

```
while(1) {
  DOWN(S);
  /* in critical section */
  UP(S);
}
```

Question 48 continuous on the next page...

(a) Is the implementation with `UP()` and `DOWN()` using a 'busy waiting'? Justify your answer. [2]

. . . .

. . . .

. . . .

. . . .

. . . .

. . . .

(b) The tasks are using a priority based scheduling. Is it possible that the above solution leads to a priority inversion problem? Provide an rationale for your answer and provide an example why. [2]

. . . .

. . . .

. . . .

. . . .

. . . .

. . . .

(c) Implement in pseudo code the tasks A, B, C, D and E. Use `UP()` and `DOWN()` on the needed semaphores to implement following scenario: After both A and B have finished their work, task C can start. After task C has finish its work, both D and E can start. [6]

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

Total: 10

Question 49..Points: [4]

Explain the *Priority Inversion* problem and illustrate it with an example:

....

....

....

....

....

....

Question 50..Points: [2]

We are using a tick timer with a period of 10ms a preemptive RTOS (FreeRTOS). This tick timer is as well used for the Trigger module as implemented in INTRO. Describe the consequences for the application and RTOS if this tick timer period gets changed from 10 ms to 200 ms, with the example of our INTRO application at the end of the semester.

. . . .

. . . .

. . . .

. . . .

. . . .

. . . .

Question 51 . . . . Points: [10]

A device is connected to a microcontroller as in Figure 4. It is using a data bus (PTB: Data) and two control signals (PTA0: status; PTA1: control). The protocol to send an 8bit data byte is illustrated in Figure 4. Implement the protocol using a *Gadfly* synchronization method.

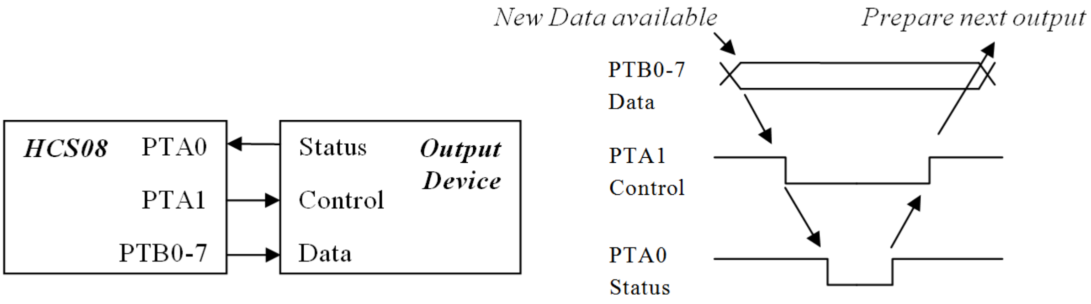

Figure 4: Output Device

Note: use pseudo code as `SetPA0asInput()`*, but make sure with comments and naming to make clear what it does*

(a) Implement the function `Init()` which initializes the ports and any other required variables. [5]

. . . .

. . . .

. . . .

. . . .

. . . .

. . . .

. . . .

. . . .

. . . .

. . . .

. . . .

. . . .

(b) Implement the function `Send(char)` which transmits a single byte. [5]

....

....

....

....

....

....

....

....

....

....

....

....

Total: 10

Question 52...Points: [4]

(a) Determine for the 8bit (binary reflected) Gray code $\mathbf{0x47}_g$ the corresponding 8bit Binary code$_b$: [2]

(a) ______________________

(b) Determine for the 8bit (binary reflected) Gray code $\mathbf{0x74}_g$ Code the corresponding 8bit Binary code$_b$: [2]

(b) ______________________

Total: 4

Question 53...Points: [6]

Consider following ANSI-C implementation:

```
typedef enum LED_Set {
  LED_0 = 1,
  LED_1 = 2,
  LED_2 = 4,
  LED_3 = 8
} LED_Set;
```

Alternatively it can be written as:

```
#define LED_0  1
#define LED_1  2
#define LED_2  4
#define LED_3  8
```

Question 53 continuous on the next page...

(a) List pros and cons for using enumerations for above example: [2]

. . . .

. . . .

. . . .

. . . .

. . . .

. . . .

. . . .

. . . .

(b) List pros and cons for using #define's for above example: [2]

. . . .

. . . .

. . . .

. . . .

. . . .

. . . .

. . . .

. . . .

(c) Explain the reason why the values 1, 2, 4, 8 are used (and not 0, 1, 2, 3): [2]

. . . .

. . . .

. . . .

. . . .

. . . .

. . . .

. . . .

. . . .

Question 54. . . . . Points: [2]

Which three critical hardware settings do you have to configure on the FRDM board to communicate using an RS-232-to-USB connection. Use the shell we used in INTRO as example.

. . . .

. . . .

. . . .

. . . .

. . . .

. . . .

Reached: ______
from 8 points

Question 55 .... Points: [5]

In this example FreeRTOS is used in priority based pre-emptive mode. Tick timer is set up to 10 ms, and the processor is running at maximum speed. The two LED's are switched off at the beginning. Task `T1` has priority 3, and task `T2` has priority 2. `T1` has to work for 15 ms, and `T2` for 2 ms (the overhead within the `for` loop can be ignored). Both tasks have been created with `xTaskCreate()` right after each other, and the scheduler has been started with `vTaskStartScheduler()` at the time t=0 ms.

```
static portTASK_FUNCTION(T1, pvParameters) { /* priority 3 task */
  portTickType xLastTime = xTaskGetTickCount();
  LED1_Off();
  for(;;) { /* task time is 15 ms including overhead */
    DoWorkFor15ms();
    LED1_Neg();
    vTaskDelayUntil(&xLastTime, 25/portTICK_RATE_MS);
  } /* loop forever */
}

static portTASK_FUNCTION(T2, pvParameters) {  /* priority 2 task */
  portTickType xLastTime = xTaskGetTickCount();
  LED2_Off();
  for(;;) { /* task time is 2 ms including overhead */
    LED2_Neg();
    DoWorkFor2ms();
    vTaskDelayUntil(&xLastTime, 30/portTICK_RATE_MS);
  } /* loop forever */
```

Draw in Figure 5 a timing diagram for the execution of the two tasks for the first 100 ms (after t0=0ms). Indicate the state of both LED's (LED1 and LED2). Use a bar to indicate when a task is running, and a bar to indicate the time when a LED is on. You can ignore the overhead in the task loop and the overhead in the RTOS/scheduler itself.

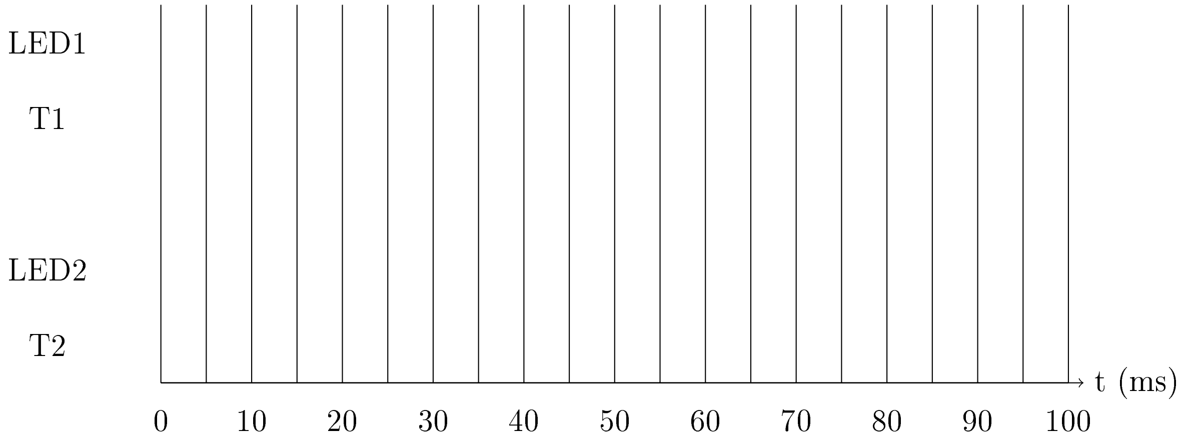

Figure 5: Task Timing

Question 56 .... Points: [2]

Explain the difference between `xQueueReceive()` and `xQueuePeek()` in FreeRTOS:

....

....

....

....

....

....

Question 57 .... Points: [12]

Given a system using *Priority Ceiling* and three tasks (J_1 (Priority 1, highest priority), J_2 (priority 2) and J_3 (priority 3)) and two semaphore S_1 and S_2. The tasks are getting a lock of a semaphore with LS_x, and return the lock with US_x. The tasks are using the semaphores according to following timing and sequence:

- J_1 = {10ms, LS_1, 10ms, US_1, 10ms}: *total 30ms task time.*
- J_2 = {10ms, LS_2, 10ms, LS_1, 10ms, US_1, 10ms, US_2, 10ms}: *total 50ms task time.*
- J_3 = {10ms, LS_1, 30ms, US_1, 10ms}: *total 50ms task time.*

(a) Calculate the *Priority Ceiling* for S_1: [2]

(a) ____________________

(b) Calculate the *Priority Ceiling* for S_2: [2]

(b) ____________________

(c) The delta between t_n and t_{n+1} in Figure 6 is 10 ms. Following times are defined: [8]

- J_3 starts at time t_0.
- J_2 starts at time t_2.
- J_1 starts at time t_4.

Complete and extend the timing diagram in Figure 6 indicating the task execution times. Mark with an LS_x the request for a semaphore x, and with US_x the release of a semaphore.

Question 58 .... Points: [2]

The PWM signal used to drive the INTRO motor is 'low active'. Explain what this means.

....

....

....

....

....

....

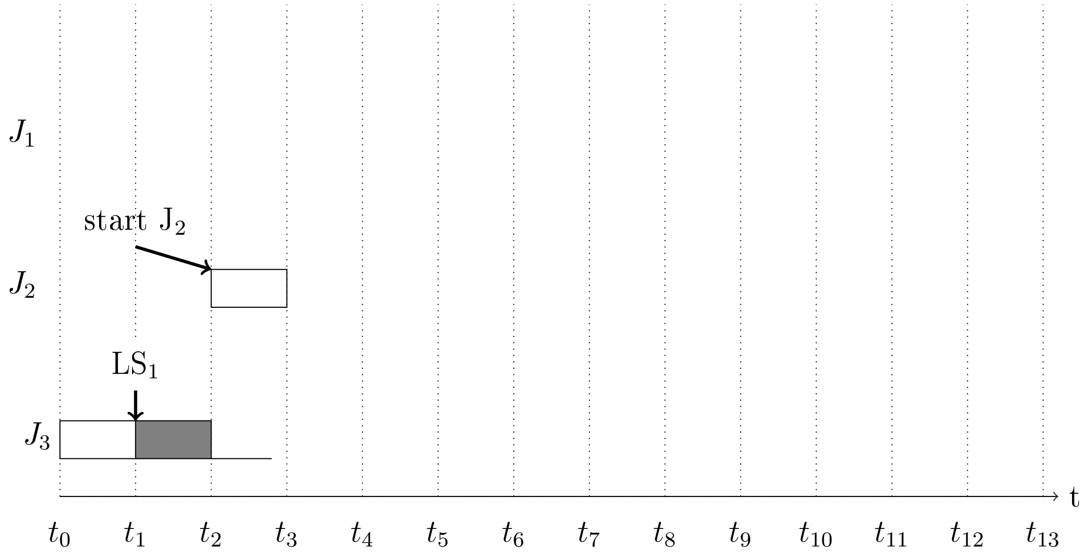

Figure 6: PCP

Question 59 .... **Points: [1]**

The following code

```
s = k0*e + k1*(e-eprev)/Ta;
eprev = e;
```

Implements in software a closed loop control using the following method:

○ PD
○ ID
○ PI
○ P
○ PID

Question 60 .... **Points: [2]**

Consider the H-Bridge we used for our INTRO lab. You have to implement a software which is able to drive the motor, from -100% (full speed backward) to +100% (full speed forward). Explain how you realize this with the signals you have available.

....

....

....

....

Question 61 .... **Points: [2]**

(a) Determine for the 8bit **Binary Code 0x31**$_b$ the corresponding 8bit (binary reflected) Gray code$_g$: [1]

(a) ________________

(b) Determine for the 8bit (Binary Reflected) **Gray Code 0x8F**$_g$ the corresponding 8bit Binary Code$_b$: [1]

(b) ____________________

Question 62 . . . . Points: [1]

Given the pseudo code for a closed loop control system:

```
esum += e;
s = a*e + b*Ta*esum;
```

This software controller implements a

- ○ PD controller
- ○ ID controller
- ○ PI controller
- ○ P Controller
- ○ PID Controller

Question 63 . . . . Points: [1]

For the digital encoder (using C1 and C2 signals) and the motor (as used in the INTRO lab) following applies:

- ± The quadrature encoder produces with C1 and C2 a 4 bit Gray encoded signal.
- ± In case C1 is not working or not available (only C2 is producing a signal), then we still can determine the motor direction.
- ± With minimal four consecutive codes it is possible to determine the motor direction.
- ± Using the two signals C1 and C2 it is possible to determine the absolute motor position.

Question 64 . . . . Points: [2]

Given following program to debounce a switch on a board. `RawKeyPressed()` returns 0 for if the switch is pressed. The function `DebounceSwitch()` gets called periodically every 3 ms.

```
bool DebounceSwitch(void) {
  static word16 State = 0;
  State = (State<<1) | !RawKeyPressed() | 0xF800;
  if(State==0xFC00) return TRUE;
  return FALSE;
}
```

Determine the switch release debouncing time in milliseconds which is used to debounce the switch.

64. ____________________

Question 65 .... Points: [3]

Processor Expert components are using the concept of *Method*, *Property* and *Event*. Give examples of this concept using the FreeRTOS component.

(a) 3 FreeRTOS examples for *Methods*: [1]

....

....

....

(b) 3 FreeRTOS examples for *Properties*: [1]

....

....

....

(c) 3 FreeRTOS examples for *Events*: [1]

....

....

....

Question 66 .... Points: [1]

You connect a push button which is *not* debounced to interrupt input pin, using an external pull-up resistor. You configure the pin to generate an interrupt whenever there is a falling edge. What are the things you have to consider?

....

....

....

....

....

....

Question 67 .... Points: [1]

Explain why there is the word 'quadrature' used to denote a quadrature encoder/decoder:

....

....

....

Question 68 .... 2 Points (Bonus)

You are implementing a state machine which performs a debouncing. You want to use the same state machine driver for different keys at the same, and depending on the application you want the state machine calling different things. What are you using to implement the *user/application defined* function calls which are evaluated at run time?

Reached: ______
from 5 points

....

....

....

....

....

....

....

Question 69 .... Points: [8]

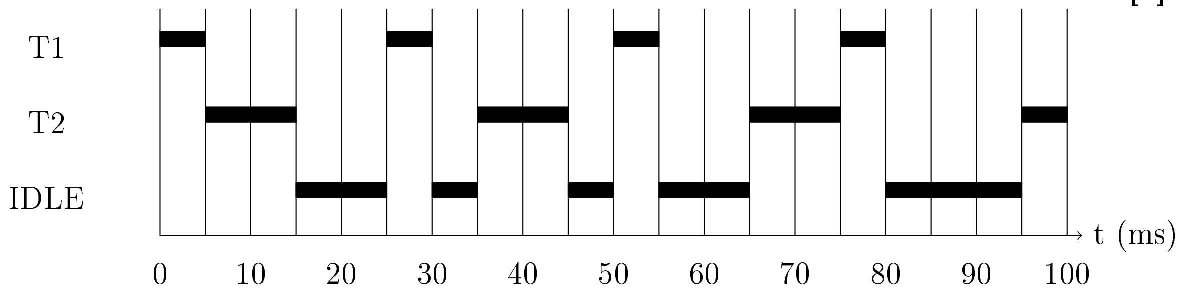

Figure 7: FreeRTOS Task Schedule

Given the the FreeRTOS task scheduling pattern as shown in Figure 7 for the first 100 milliseconds. The RTOS is in preemptive scheduling mode. The RTOS IDLE task is running with priority `tskIDLE_PRIORITY`. The tasks T1 and T2 are implemented as following:

```
static portTASK_FUNCTION(T1, pvParameters) {
  for(;;) {
    DoWorkT1();
    vTaskDelay(delayT1);
  } /* loop forever */
}

static portTASK_FUNCTION(T2, pvParameters) {
  for(;;) {
    DoWorkT2();
    vTaskDelay(delayT2);
  } /* loop forever */
}
```

(a) Determine the value of `delayT1` and `delayT2` in *milliseconds*: [2]

(a) ____________________

(b) Each task does some work which is *not* using any blocking RTOS calls. Determine the time of work for `DoWorkT2()` and `DoWorkT1()` in *milliseconds*: [2]

(b) ____________________

(c) What is the tick timer period of the RTOS? Explain the reasoning behind your answer. [2]

....

....

....

....

....

....

(d) Could it be possible that T1, T2 and IDLE share the same priority level? Justify your answer. [2]

....

....

....

....

Question 70 .... Points: [2]

Given the following enumeration type:

```
typedef enum { RED, BLUE, GREEN, YELLOW, BLACK } ColorsEnum;
```

For the compiler we used in INTRO, the following applies for ANSI-C (with default compiler settings):

- ○ sizeof(ColorsEnum)==1
- ○ sizeof(ColorsEnum)==2
- ○ sizeof(ColorsEnum)==4
- ○ sizeof(ColorsEnum)==5
- ○ sizeof(ColorsEnum)==sizeof(int)

Question 71 .... Points: [4]

Given following declaration of a trigger:

```
#define TRG_LAST       4
typedef void (*TRG_Callback)(void*);

typedef struct TriggerDesc {
  uint16_t triggerTicks;
  TRG_Callback callback;
} TriggerDesc;

static TriggerDesc TriggerList[TRG_LAST];
```

The triggers are initialized with following function:

```
void Init(void) {
  static volatile uint8_t i;

  for(i=sizeof(TriggerList)/sizeof(TriggerDesc);i>0;i--) {
    TriggerList[i].triggerTicks = 0;
    TriggerList[i].callback = NULL;
  }
}
```

(a) Evaluate the following statements: [2]

± The function `Init()` will never finish.

± Every element of `Triggerlist` is initialized with with the same values.

± Using `static volatile` for the variable `i` ensures that the function `Init()` is reentrant.

± Using `volatile` for the variable `i` ensures that the function `Init()` is reentrant.

(b) Using above implementation, you notice that your application is working in a strange way, it even crashes after a while. Explain the reason: [2]

. . . .

. . . .

. . . .

. . . .

. . . .

. . . .

. . . .

. . . .

. . . .

Question 72 . . . . Points: [5]

Given following implementation of an event module:

```
#define GET_EVENT(event) \
  (bool)(EVNT_Events[(event)/8]&(0x80>>((uint8_t)((event)%8))))
#define CLR_EVENT(event) \
  EVNT_Events[(event)/8] &= ~(0x80>>((uint8_t)((event)%8)))

static uint8_t EVNT_Events[((EVNT_NOF_EVENTS-1)/8)+1];

bool EVNT_GetEvent(EVNT_Handle event) {
  bool isSet;
  EnterCritical();
  isSet = GET_EVENT(event);
  ExitCritical();
  return isSet;
}
```

```
void EVNT_ClearEvent(EVNT_Handle event) {
  EnterCritical();
  CLR_EVENT(event);
  ExitCritical();
}

uint8_t EVNT_CheckEvents(void) {
  uint8_t i;

  EnterCritical();
  for(i=0; i<sizeof(EVNT_Events)/sizeof(EVNT_Events[0]); i++) {
    if (EVNT_GetEvent(i)) {
      EVNT_ClearEvent(i);
      break;
    }
  }
  ExitCritical();
  return i; /* return the event which was set */
}
```

(a) Using that implementation and calling `EVNT_CheckEvents()`, you find out that your periodic interrupt timer does not work any more. Explain the problem and how to fix it: [3]

....

....

....

....

....

....

(b) After you have fixed the previous problem, the application still does not work correctly. Somehow `EVNT_CheckEvents()` does not properly return events which have been set. Explain the problem and how to fix it: [2]

....

....

....

....

....

....

....

Question 73 .... Points: [2]

Evaluate following statements in the context the RTOS startup, as used in the INTRO lab (FreeRTOS):

± All interrupts are disabled during the (ANSI) startup code.

± Global interrupts are enabled at the entry point of `main()`.

± Interrupts for RTOS timers are disabled at the entry point of `main()`.

± Interrupts for RTOS timers get enabled during startup of the scheduler.

Question 74 .... Points: [3]

Explain why it is necessary to implement a timeout using the delta-time approach for measuring the speed with a quadrature encoder:

....

....

....

....

Question 75 .... Points: [3]

If you can choose between the delta-time and delta-pos method for quadrature decoding for fast (many steps) state changes, which one would you prefer and why?

....

....

....

....

Question 76 .... Points: [3]

If you can choose between the delta-time and delta-pos method for quadrature decoding for slow (few steps) state changes, which one would you prefer and why?

....

....

....

....

Question 77 .... Points: [3]

Explain why a tacho implementation as we have implemented is not an exact measurement, but rather a speed estimation?

....

....

....

....

Question 78 .... Points: [3]

Name a data structure which can be used to efficiently implement a delta-pos algorithm for estimating a a tachometer.

....

....

....

....

Question 79 .... Points: [3]

What is the difference between a Mutex and a normal semaphore in FreeRTOS?

....

....

....

....

Question 80 .... Points: [3]

How many processes and priorities are needed to demonstrate a Priority Inversion?

....

....

....

....

Question 81 .... Points: [3]

Given the two Semaphores `S1` and `S2` and the three processes `PL` (high priority (priority 3)), `PM` (medium priority (priority 2)) and `PL` (low priority (priority 1)). Semaphore `S1` is used by `PL` and `PM`, `S2` is used by `PL` and `PH`. Determine the *Priority Ceiling* of `S1` and `S2`.

....

....

....

Question 82 .... Points: [3]

If you use a Semaphore in FreeRTOS from an interrupt service routine (ISR), is there something special to consider?

....

....

....

Question 83 .... Points: [1]

In which state is a process in FreeRTOS when it is waiting on a Semaphore?

. . . .

. . . .

. . . .

Question 84 . . . . Points: [3]

Using a network stack like RNet, which things do you have to initialize?

. . . .

. . . .

. . . .

Question 85 . . . . Points: [2]

RNet uses four different layers in the stack. List the layers with their acronyms.

. . . .

. . . .

. . . .

Question 86 . . . . Points: [1]

What is a CRC and for what it is used for?

. . . .

. . . .

. . . .

Question 87 . . . . Points: [1]

What is a so called *Ad-Hoc Network*?

. . . .

. . . .

. . . .

Question 88 . . . . Points: [2]

List pros and cons of an *Ad-Hoc Network*.

. . . .

. . . .

. . . .

Question 89 . . . . Points: [2]

List the pins of the HC-SR04 Ultrasonic distance sensor and for what they are used.

. . . .

. . . .

. . . .

Reached: ______
from 11 points

Question 90 .... Points: [2]

What happens if the HC-SR04 Ultrasonic distance sensor does not measure an echo (no object detected)?

....

....

....

Question 91 .... Points: [2]

Using the HC-SR04 Ultrasonic distance sensor, you are using for the speed of sound 58 cm/μ. Why is the result not correct? Are there other factors influencing the result?

....

....

....

Question 92 .... Points: [1]

What is the purpose of a radio transceiver?

....

....

....

Question 93 .... Points: [1]

Name three software components/layers of the RNet communication stack we used.

....

....

....

Question 94 .... Points: [1]

What does *Payload Packaging* mean?

....

....

....

Question 95 .... Points: [1]

You need to drive a H-Bridge IC with a PWM signal. What fundamental parameter you have to check in the data sheet of the IC?

....

....

....

Question 96 .... Points: [1]

How can the power dissipation of the H-Bridge for of a PWM controlled device be reduced?

. . . .

. . . .

. . . .

Question 97 . . . . Points: [1]

If a closed control loop controller is using a PWM as output signal, what should be considered?

. . . .

. . . .

. . . .

Question 98 . . . . Points: [1]

What is the common purpose of using a Queue or a Semaphore/Mutex?

. . . .

. . . .

. . . .

Question 99 . . . . Points: [1]

`Lock()` and `Unlock()` are functions used for which programming primitive or design pattern?

. . . .

. . . .

. . . .

Question 100 . . . . Points: [1]

The speed of a motor is proportional to the voltage applied to the motor. So why is it necessary to use a closed control loop?

. . . .

. . . .

. . . .

Question 101 . . . . Points: [2]

You have to implement a position controller (closed loop)? Which type do you choose: PD or PI?

. . . .

. . . .

. . . .

Question 102 . . . . Points: [2]

For our Sumo robot, identify the following parts of a closed loop control system: Controller, Actuator, Sensor, System.

. . . .

. . . .

. . . .

Question 103 . . . . Points: [2]

What is the purpose of the *Anti-Windup*?

. . . .

. . . .

. . . .

Question 104 . . . . Points: [2]

Why is it better to use 'industrial qualified' SD cards instead of using 'consumer' ones?

. . . .

. . . .

. . . .

Question 105 . . . . Points: [1]

Which fundamental values have to be calibrated for an accelerometer sensor as we have used it?

. . . .

. . . .

. . . .

Question 106 . . . . Points: [1]

Provide an implementation way where you can store calibration data?

. . . .

. . . .

. . . .

Question 107 . . . . Points: [3]

What do you have to consider if you are programming the internal FLASH memory at run time?

. . . .

. . . .

. . . .

Question 108 . . . . Points: [1]

Why did we use a digital interface to the reflectance sensor and not an analog one?

. . . .

. . . .

. . . .

Reached: ______
from 10 points

Question 109 .... Points: [1]

Even if all sensors of the reflectance sensor are exposed to the same conditions, the measurement will differ from sensor to sensor. Why?

....

....

....

Question 110 .... Points: [1]

The reflectance sensor array has two red LED's. What's the purpose of it?

....

....

....

Question 111 .... Points: [1]

How could you modify the reflectance sensor array hardware to minimize the power consumption?

....

....

....

Question 112 .... Points: [3]

List typical external errors for the reflectance sensor array we have used, and how the errors can be reduced or limited.

....

....

....

Question 113 .... Points: [3]

List several functions or macros which are used in FreeRTOS to control task execution.

....

....

....

Question 114 .... Points: [1]

If several tasks are pending, which task gets executed first by a priority-based preemptive scheduler?

....

....

....

Question 115 .... Points: [1]

Why could be a call to `vTaskEndScheduler()` make sense?

....

....

....

Question 116..Points: [1]

List the four memory allocation schemes in FreeRTOS.

....

....

....

Question 117..Points: [1]

What was the purpose of the Shell queue we have implemented in the INTRO application?

....

....

....

Question 118..Points: [1]

What are the two different categories of timeliness?

....

....

....

Question 119..Points: [1]

List three different clock sources for a microcontroller:

....

....

....

Question 120..Points: [1]

In a microcontroller you can have different kinds of clocks, list at least two:

....

....

....

Question 121..Points: [1]

List a fundamental requirement for a realtime operating system:

....

....

....

Question 122..**Points: [1]**

List pros and cons of using an internal reference clock (compared to an external clock generator):

....

....

....

Question 123..**Points: [3]**

List reasons why to use an RTOS:

....

....

....

Question 124..**Points: [2]**

What is the difference between preemptive and non-preemptive scheduling:

....

....

....

Question 125..**Points: [2]**

List the three standard I/O channels used by the Shell:

....

....

....

Question 126..**Points: [1]**

List the four fundamental states of a task in an operating system:

....

....

....

Question 127..**Points: [1]**

What does the abbreviation *HAL* mean?

....

....

....

Question 128..**Points: [2]**

What is the difference between two different vendors offering a microcontroller based on an ARM Cortex M0+?

Reached: ______
from 12 points

....

....

....

Question 129..Points: [2]

How many hardware stack pointers implements an ARM Cortex M0+ microcontroller?

....

....

....

Question 130..Points: [2]

The ARM Cortex M0+ architecture has 16 32bit registers. Are they all freely usable by the application code?

....

....

....

Question 131..Points: [2]

Which Processor Expert generated macros can you use to protect a critical section?

....

....

....

Question 132..Points: [1]

What could a reason to use the `volatile` qualifier for a loop variable?

....

....

....

Question 133..Points: [1]

What is the advantage of using *Gadfly Synchronization* compared to *Interrupt Synchronization*?

....

....

....

Question 134..Points: [1]

What does this mean: *an instruction is atomic*?

....

....

....

Question 135..Points: [1]

What is the fundamental problem of 'shared data' in the context of interrupts?

....

....

....

Question 136..Points: [2]

The implementation of `EnterCritical()` and `ExitCritical()` as we used it does not allow nesting. Why?

....

....

....

Question 137..Points: [2]

Given the following source code:

```
#defineCALC1 (2+5)
#defineCALC2 (5*3)
```

Why is it important to use parenthesis?

....

....

....

Question 138..Points: [2]

List some advantages and disadvantages of using macros:

....

....

....

Question 139..Points: [2]

You need to write a header file `LED.h`. Write it in such a way so *recursive* includes are not possible.

....

....

....

Question 140..Points: [2]

List additional benefits of Doxygen beside of normal source documentation:

....

....

....

Question 141..Points: [1]

You need to use a variable in multiple modules (.c files). Where do you put the *declaration* of it? Make an example.

....

....

....

Question 142..Points: [1]

What is the fundamental difference between the two VCS: Git and SubVersion?

....

....

....

Question 143..Points: [1]

Two engineers are using Git as VCS. Now they change both the same line of code, and both perform a *commit*. What happens?

....

....

....

Question 144..Points: [3]

An engineer is generating the documentation with doxygen from the source files. He decides to put both the documentation and the source files into a VCS. Discuss this approach:

....

....

....

Question 145..Points: [3]

Below is the content of a .gitignore file. Explain what this file does with such a content:

```
*
!*.c
!*.h
!*.gitignore
!dev/
```

....

....

....

Question 146..Points: [3]

For multiple Eclipse projects, you want to use source files in a common folder (e.g.

`c:/mySources/common`) which is *outside* of your Eclipse project. You consider to use '*Virtual Group*', '*Linked Folder*' or '*Linked File*' methods to use the common file(s) in your project. Now you have added a new source file (e.g. `accelerometer.c`) to that common folder. For each of the three above methods, explain what you have to do for *each of your eclipse projects* which want to use the new `accelerometer.c` file.

....

....

....

....

....

....

....

....

....

Question 147...Points: [3]

Given a Quadrature encoder as in Figure 8, generating two signals C0 and C1. Both signals get sampled with a periodic timer interrupt. The disk is turning with a maximum frequency of 100 Hz. Determine the minimal sampling frequency in Hz to guarantee an error free sampling of the two signals C0 and C1.

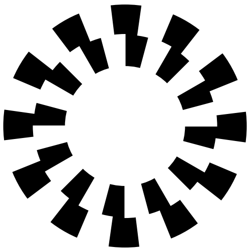

Figure 8: Quad Disk

....

....

....

....

....

....

....

Question 148...Points: [3]

Draw the resulting graph produced by doxygen for following DOT code:

```
\dot
```

```
digraph example_dot_graph {
    node [shape=triangle];
    rankdir=LR;
    A    [label="A"];
    B    [label="B"];
    C    [label="C"];
    A -> A [label="a"];
    A -> B -> C -> A -> C;
    B -> B [label="b"];
}
\enddot
```

....

....

....

....

....

....

....

Question 149..Points: [3]

You are using a wireless transceiver as used in the lab to transmit sensor data values to another system. Describe strategies to reduce the energy consumption for the communication. Limit your answer to the *communication* and *transceiver* only.

....

....

....

....

....

....

....

Question 150..Points: [3]

Calculate for the 16bit **Bindary Code 0x1722**$_b$ the corresponding 16bit (binary reflected) Gray code$_g$:

....

....

....

....

....

....

Question 151..Points: [3]

In Figure 9 are steps transforming a 6bit Gray Code into a 6bit Binary Code. Fill in the gaps:

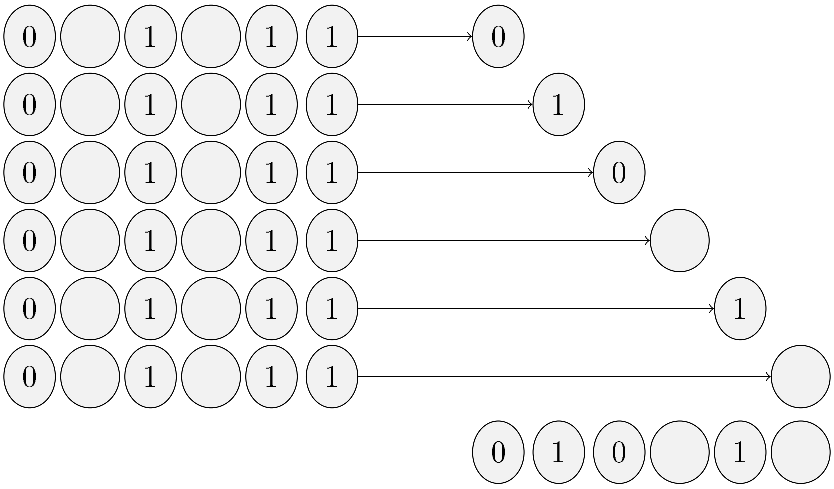

Figure 9: 6bit Gray Code

Question 152..Points: [8]

Below is the implementation of a task which reads and writes to an SCI (Serial Communication Interface). The SCI is using interrupts for RX and TX. As other tasks are using the `Read()` and `Write()` functions as well, it is necessary to use critical sections.

```
void Read(char *buf, size_t bufSize) {
  /* start critical section */
  SCI_Read(buf, bufSize);
  /* end critical section */
}

void Write(char *buf) {
  /* start critical section */
  SCI_Write(&buf[0]);
  /* end critical section */
}

static portTASK_FUNCTION(Task1, pvParameters) {
  unsigned char buf[16];
  (void)pvParameters; /* parameter not used */
  for(;;) {
    /* start critical section */
    Read(buf, sizeof(buf));
    Write(buf, sizeof(buf));
    /* end critical section */
  }
}
```

You identified 3 different ways for implementing the `/* start critical section */` and `/* end critical section */` in above source. Evaluate following statements for the consequences of each implementation for the above source:

(a) With using `DisableAllInterrupts()` and `EnableAllInterrupts()` the following applies: [2]

± It increases the interrupt latency time.

± Is the best solution with respect to RAM/ROM footprint.

± Makes the usage of `SCI_Read()` and `SCI_Write()` reentrant.

± Allows the scheduler to run inside every critical section.

(b) With using FreeRTOS *binary semaphore/mutex* the following applies: [2]

± Calling the semaphore/mutex API might trigger a context switch.

± No context switch will happen within the critical section.

± Only one task at a time will be inside the critical section.

± Interrupts will occur inside the critical section.

(c) With using FreeRTOS *recursive semaphore/mutex* the following applies: [2]

± Calling the semaphore/mutex API might trigger a context switch.

± Multiple tasks might be within the same critical section.

± No context switch will happen within the critical section.

± No interrupts will occur inside the critical section.

(d) From the 3 solutions a), b) and c), select the one you consider as the best one for above source, and explain briefly why: [2]

....

....

....

....

....

....

Question 153 .... Points: [4]

A system using *Priority Ceiling* has four tasks J_1, J_2, J_3 and J_4. The task index denotes the task priority with 4 as the highest priority. Additionally four semaphores S_1, S_2, S_3 and S_4 are used. Task are getting a lock of a semaphore x using LS_x, the lock of a semaphore is released again with US_x. The task and semaphores are used according to following schedule:

- $\mathrm{J}_1 = \{\mathrm{LS}_4, \mathrm{US}_4\}$
- $\mathrm{J}_2 = \{\mathrm{LS}_2, \mathrm{LS}_1, \mathrm{US}_1, \mathrm{US}_2, \mathrm{LS}_3, \mathrm{US}_3\}$
- $\mathrm{J}_3 = \{\mathrm{LS}_1, \mathrm{US}_1, \mathrm{LS}_3, \mathrm{US}_3\}$
- $\mathrm{J}_4 = \{\mathrm{LS}_1, \mathrm{US}_1\}$

Determine the *Priority Ceiling* for each semaphore:
Priority Ceiling for S_1:

153. ______________________

Priority Ceiling for S_2:

153. ______________________

Priority Ceiling for S_3:

153. ______________________

Priority Ceiling for S_4:

153. ______________________

Question 154..Points: [3]

Below is a way to prevent spurious keyboard interrupts on a microcontroller:

```
PTADD = 0x00; /* set port as input */
PTAD = 0xFF; /* write 1's to data port to have defined input values */
PTAPE = 0xFF; /* enable pull ups */
EnableInterrupts;
```

Transform the above code into an implementation using *Gadfly* synchronization:

....

....

....

....

....

....

....

....

....

Question 155..Points: [6]

A preemptive system has 3 tasks T_1, T_2 and T_3 where the task index denotes the task priority, with 1 the lowest priority. The system is using *Priority Inheritance* for the semaphores. Create a timing diagram with the tasks in Figure 10 indicating the execution time for each task. The system has following timing:

- t_0: T_1 and T_2 are ready, T_3 is running,
- t_1: The running task gets suspended
- t_3: The running task gets suspended
- t_4: The running task requests the semaphore
- t_5: T_2 and T_3 get ready
- t_7: The running task requests the semaphore
- t_8: The task having the semaphore releases the semaphore
- t_9: The task having the semaphore releases the semaphore
- t_{10}: The running task terminates
- t_{11}: The running task terminates
- t_{12}: The running task terminates

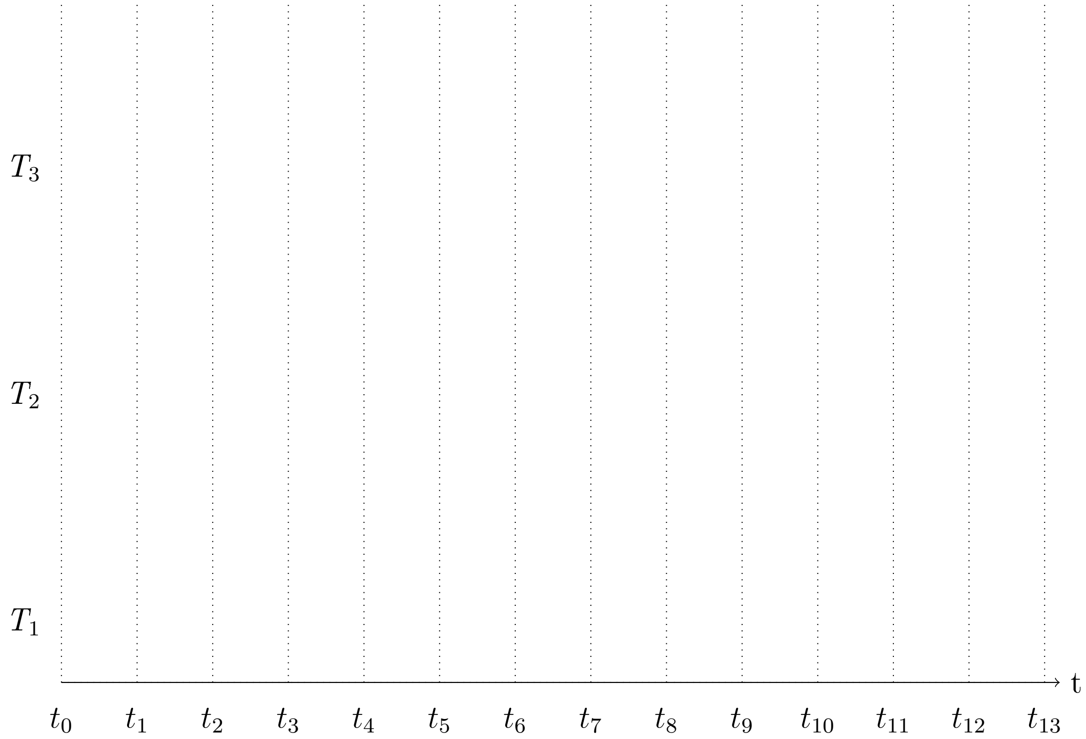

Figure 10: Tasks

Question 156..Points: [3]

Explain for what the *Anti-Windup* is used in a PID control system:

. . . .

. . . .

. . . .

. . . .

. . . .

. . . .

Question 157. . . . . .Points: [3]

Identify the three serious problems in the following source:

```
#define QUEUE_LENGTH 5
#define QUEUE_ITEM_SIZE sizeof(char_t*)

void QUEUE_SendMessage(const char *msg) {
  char *ptr = FRTOS1_pvPortMalloc(UTIL1_strlen(msg));
  UTIL1_strcpy(ptr, msg);
  if (FRTOS1_xQueueSendToBack(queueHandle, ptr, portMAX_DELAY)!=pdPASS)
  {
    for(;;){} /* ups? */
  }
}
```

. . . .

. . . .

. . . .

. . . .

. . . .

. . . .

. . . .

Reached: ______
from 3 points

Question 158 .... Points: [8]

Given a ultrasonic sensor. It has a TRIG input signal and an ECHO output signal. A microcontroller sets the TRIG signal HIGH for 1 ms. Then after some time, it can measure the duration of the ECHO pulse signal which is proportional to the distance measured (Figure 11).

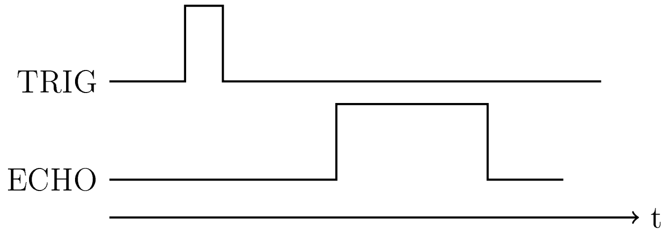

Figure 11: Ultrasonic Module Timing

(a) Implement the generation of the TRIG signal using a *Realtime* synchronization method: [4]

....

....

....

....

....

....

....

....

....

(b) After the TRIG signal generation, the microcontroller has to synchronize to the beginning of the ECHO signal. Implement this using a *Gadfly* synchronization: [4]

....

....

....

....

....

....

....

....

....

Question 159 .... Points: [10]

FreeRTOS is used in priority based preemptive mode. Tick timer is set up for 5 ms, and the processor is running at maximum speed. Task `T3` has priority 3 (highest priority), task T2 has priority 2 and task T1 has priority 1. The tasks have been created with xTaskCreate() before time t=0 ms, and the scheduler is started with vTaskStartScheduler() at the time t=0 ms. Draw in Figure 12 a timing diagram for the execution of the two tasks for the first 100 ms (after t0 = 0 ms). Use a bar to indicate when a task is running. You can ignore the overhead in the task loop and the overhead in the RTOS/scheduler itself.

```
static portTASK_FUNCTION(T3, pvParameters) { /* priority 3 task */
  portTickType xLastTime = xTaskGetTickCount();
  for(;;) { /* task time is 7 ms including overhead */
    DoWorkFor7ms(); /* this needs 7 ms */
    vTaskDelayUntil(&xLastTime, 30/portTICK_RATE_MS);
  } /* loop forever */
}

static portTASK_FUNCTION(T2, pvParameters) {  /* priority 2 task */
  for(;;) { /* task time is 7 ms including overhead */
    DoWorkFor7ms(); /* this takes 7 ms */
    vTaskDelay(30/portTICK_RATE_MS);
  } /* loop forever */
}

static portTASK_FUNCTION(T1, pvParameters) {  /* priority 1 task */
  for(;;) { /* task time is 4 ms including overhead */
    DoWorkFor4ms(); /* this needs 4 ms */
    vTaskDelay(28/portTICK_RATE_MS);
  } /* loop forever */
}
```

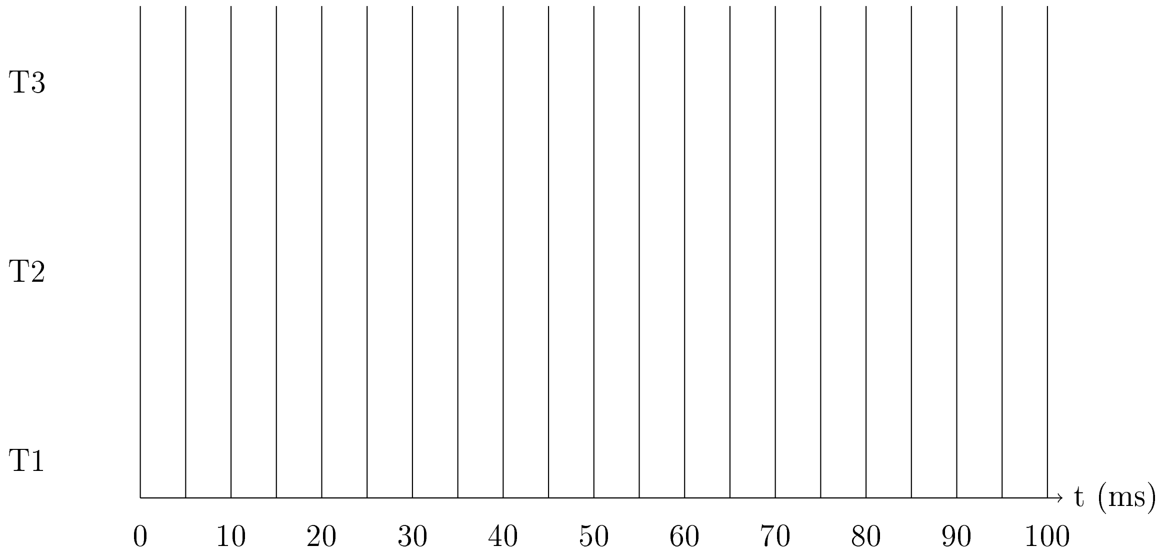

Figure 12: Task Timing

Question 160...Points: [3]
You are working with an Eclipse based project (you can consider the one we used

for INTRO). Which kind of files are you going to put into a VCS? Which ones not? Explain briefly why.

. . . .

. . . .

. . . .

. . . .

. . . .

. . . .

Question 161 . . . . 2 Points (Bonus)

Three men: a project manager, a software engineer, and a hardware engineer are helping out on a project. About midweek they decide to walk up and down the beach during their lunch hour. Halfway up the beach, they stumbled upon a lamp. As they rub the lamp a genie appears and says "Normally I would grant you three wishes, but since there are three of you, I will grant you each one wish."'

The hardware engineer went first. "I would like to spend the rest of my life living in a huge house in Hawaii." The genie granted him his wish and sent him on off to Hawaii.

The software engineer went next. "I would like to spend the rest of my life living on a huge yacht cruising the Mediterranean." The genie granted him his wish and sent him off to the Mediterranean.

Last, but not least, it was the project manager's turn. "And what would your wish be?" asked the genie. The project manager replied:

○ "I would like an expensive Ferrari car."

○ "I would like to win the lottery so I have no money worries."

○ "Send me to the hardware engineer to Hawaii"

○ "Send me to the software engineer on the Mediterranean."

○ "I want them both back after lunch."

Reached: ______
from 0 points