This document includes solution hints, but **NOT** necessarily the full solution!

Part I
Evaluation Page Part A

Name: _____ Signature: _____

1 Evaluation

This part of the exam has 180 questions, with a total of 310 points and 6 bonus points.

Part	Max. Points	Scored Points
A	60	
B	180	
	Total	

Points:	240-220	219-201	200-182	181-162	161-144	143-0
Grade:	A	B	C	D	E	F
Score:						

2 Evaluation Part A

Page	Points	Bonus Points	Score
5	5	0	
6	7	2	
7	12	0	
8	16	0	
9	5	1	
10	9	0	
11	4	0	
12	10	0	
13	12	0	
14	8	0	
15	6	0	
16	3	0	
17	4	0	
18	4	0	
19	2	0	
20	1	0	
21	2	2	
22	8	0	
23	3	0	
24	2	0	
Total:	123	5	

Page	Points	Bonus Points	Score
26	6	0	
27	5	0	
29	3	0	
30	5	0	
31	2	0	
32	5	0	
33	12	0	
34	7	0	
35	13	0	
36	9	0	
37	13	0	
38	8	0	
39	8	0	
40	6	0	
41	7	0	
42	4	0	
43	8	0	
44	10	0	
45	11	0	
46	10	0	
47	8	0	
48	8	0	
49	7	1	
50	12	0	
Total:	187	1	

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 √), 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 ..... **Points: [1]**

What is a 'recap'?

- √ Learning summary with 5 questions.
- ◯ Sumo robot PCB capacitor.
- ◯ Collection of slides.
- ◯ Line sensor capacitance.
- ◯ Tips for students in next semester.

Question 2 ..... **Points: [1]**

What is a VCS?

- ◯ Variable Capacity System.
- ◯ Volatile Control Status.
- √ Version Control System.
- ◯ Variable Computer Software.
- ◯ Volatile Client Storage.

Question 3 ..... **Points: [1]**

The processor used on the FRDM board is the following:

- √ ARM Cortex-M0+
- ◯ Freescale HCS08
- ◯ MMA8780Q
- ◯ ARM Cortex M4
- ◯ MC13213

Question 4 ..... **Points: [2]**

A directory listing which is under Git control contains the following directories/files:

```
.gitignore
readme.txt
list.txt
src\rotor.c
src\rotor.h
obj\rotor.o
obj\rotor.txt
```

The .gitignore file has following content:

```
/obj
/*.txt
!/r*
```

In above directory listing, ~~strike through~~ the files which are *ignored*.

Solution: not ignored are: .gitignore, readme.txt, src/main.c, src/main.h

Question 5 ..... **Points: [2]**

Explain in a single sentence what each of the following basic VCS actions mean in Git:

(a) Commiting [½]

Solution: Putting a change into the local repository.

(b) Reverting [½]

Solution: Undo a local change.

(c) Pushing [½]

Solution: Moving a local change into the remote repository.

(d) Cloning [½]

Solution: Copy a repository and create a new local one.

Question 6 ..... **Points: [1]**

What is the fundamental difference between SVN and Git?

Solution: SVN: centralized VCS, Git: distributed VCS.

Question 7 ..... **Points: [2]**

Explain the difference between the optimistic and pessimistic approach in a VCS. Explain it with an example.

Solution: Optimistic: assumes that two developers do not work on the same file, so system potentially allows conflicts. Conflicts have to be resolved later. Pessimistic: assumes that conflict will happen, and whenever a developer wants to edit a file, the file gets locked so no conflict can occur.

Question 8 ..... **2 Points (Bonus)**

This could be your bonus question you have submitted... ☺.

8. _____yes or no?_____

Question 9 ..... **Points: [2]**

(a) Provide an example for a *hard* real-time system: [1]

Solution: Air bag, pacemaker or aircraft control system.

(b) Provide an example for a *soft* real-time system: [1]

Solution: Video streamer.

Question 10 ..... **Points: [3]**

List three different problems which can be solved with a WDT:

Solution: Waiting for hardware state, waiting for semaphore/resource, runaway code, stack overflow.

Question 11 ..... **Points: [2]**

(a) Name some benefits implementing a state machine: [1]

Solution: well structured, easy to implement, reusable design pattern.

(b) What should be the first steps when implementing a state machine? [1]

Solution: Define all states with input/output and transitions. Draw a diagram.

Question 12 ..... **Points: [5]**

In INTRO we implemented an 'Events' driver.

(a) Why did we implement it as an array of bits? [1]

Solution: To save RAM.

(b) What is the fundamental disadvantage of such an array of bits? [1]

Solution: Costs runtime performance for bit manipulation.

(c) It implements critical section (e.g. to set an event bit) with `EnterCritical()` and `ExitCritical()`. Under which conditions such a critical section would *not* be required? [2]

Solution: If there is not a possiblity that the operation gets interrupted, or if the operation is atomic.

(d) List reasons why an interrupt service routine *should* use such an Event module: [1]

Solution: Reduce interrupt latency time, only setting a bit and let the main application do the heavy lifting.

Question 13 ..... **Points: [2]**

An RTOS can be either pre-emptive or cooperative: Explain the difference:

Solution: Pre-emptive: Always runs the highest available task. Tasks of identical priority share CPU time Cooperative: Context switches only occur if a task blocks, or explicitly calls yield.

Question 14 ..... **Points: [4]**

 Question 14 continuous on the next page... Reached: _____ from 12 points

(a) In an RTOS, each task can be in one of 5 fundamental states: List them: [2]

Solution: New, Ready, Running, Waiting, Stopped.

(b) What's the purpose of the scheduler in an RTOS? [2]

Solution: To determine which tasks gets executed next to minimize waiting time.

Question 15 ..... Points: [3]

Provide a short definition of the term *Interrupt Latency*, and which factors/aspects are contributing to it:

Solution: Time between the event itself and until the ISR executes. Factors are stopping/finishing the current interrupt, interrupt destination calculation/arbitration, pushing state and diverting to the ISR.

Question 16 ..... Points: [2]

Provide an example of a typical *Reactive System*, and explain why this is a reactive system:

Solution: Airbag, it reacts on external events.

Question 17 ..... Points: [4]

(a) A PWM signal on a H-Bridge is labeled as *low active*. Explain what this means and how this impacts the speed of a DC motor: [2]

Solution: *low active* means that the motor is active when the signal is low. It means for the PWM duty cycle: the longer in the low state, the higher the voltage, the faster the motor turns.

(b) Draw a timing diagram for that PWM signal: the PWM period is 5 ms, and the motor shall at 20% speed. Indicate how many milliseconds the signal is high and low. [2]

Solution: [Timing diagram drawn here, with a frequency of 5 ms and 20% low duty cycle (1 ms low, 4 ms high)].

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

Given the following program:

```
#define ADC_CONFIG (*(volatile uint8_t*)0x123)

static void Interrupt(void) {
  uint8_t i;

  while(ADC_CONFIG & ~0x10);
  for(i=0;i<10;i++) {
    __asm("nop");
  }
```

```
}
```

This program is using

- ◯ Interrupt synchronization.
- ◯ Gadfly synchronization.
- ◯ Realtime synchronization.
- √ Realtime and Gadfly synchronization.
- ◯ No synchronization.

Question 19 ..... Points: [2]

Your Eclipse project stores the make files, object files and the final (binary) application file in a sub folder inside your project. Are you going to store this folder and files in a version control system? Justify your answer:

Solution: No, as the content of this folder is generated. It does not make sense to store derived content in a version control system, as it can be generated from the sources.

Question 20 ..... 1 Point (Bonus)

A tourist walked into a pet shop and was looking at the animals on display. Which monkey is the most expensive one?

- √ The monkey which programs nothing.
- ◯ The monkey which programs in C.
- ◯ The monkey which programs in C++.
- ◯ The monkey which programs in Java and C++.
- ◯ The monkey which programs in C, C++ and Java.

Question 21 ..... Points: [3]

Given the source of a PID control loop implementation. Identify in the source lines for the P, I and D part: mark them clearly and label it with P, I and D. Mark/circle this in the following source listing:

```
#define max 0x33ff
static int32_t old=0, b=0;
void PID_Control(void) {
  int32_t f, s, a;

  v = 0;
  f = should-actual;
  a = f-old;
  old = f;
  v += a/10;
  v += f*35;
  b += f;
  if (b > max) { b = max; }
  v += b/4;
```

```
  setAcuator(v);
}
```

Solution: D, then P, then I with anti-windup.

Question 22 ..... Points: [3]

Processor Expert components are using the concept of *Methods*, *Properties* and *Events*. What would you expect for an ADC (Analog to Digital Converter) component?

(a) 2 typical *Methods* for an ADC component: [1]

Solution: Measure(), SetChannel(), ...

(b) 2 typical *Properties* for of an ADC component: [1]

Solution: Pin, sampling time, channel

(c) 2 typical *Events* for an ADC component: [1]

Solution: OnSamplingStart(), OnConversionEnd(), OnError(), ...

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

Given following variable definition:

```
static char *string = "hello";
```

What is the difference between the two following usages

```
sizeof(string)
```

```
strlen(string)
```

in respect to the result and the expected code generated?

Solution: sizeof gives the size in memory, which is here the size of a pointer (2 or 4 bytes, depending on the machine), while strlen() the length without the zero byte. sizeof() is calculated at compile time (constant), while strlen() is a library routine call.

Question 24 ..... Points: [3]

Given following interface implementation for a DC motor driver:

```
/* motor.h */
#include "LED.h" /* LED interface */
#include "PWM.h" /* PWM interface */

static uint16_t MOT_motorSpeed;

void MOT_Init(void);
/* end of motor.h */
```

Identify three issues with such an interface implementation (issues which could lead to linker/compiler failure, or things which are not considered as good programming style):

Solution: No #ifndef...#define, not necessary includes, static definition in header file.

Question 25 ..... Points: [2]

Given following program:

```
#define DEC(i) {int b=0; i--;}

void main(void) {
  int a = 5, b = 5;
  DEC(a);
  DEC(b);
  printf("a is %d, b is %d\n", a, b);
}
```

What is the output of `printf()`?

Solution: a is 4, b is 5

Question 26 ..... Points: [3]

Given the *Mealy Sequential State Machine* in Figure 1 with five states, two input values and two output values:

(a) The state machine in Figure 1 is in the state **'B'**. Determine the *input sequence* in order to generate the following *output*: [2]
1, 1, 1, 0

Solution: B: input1 output1 -> C
C: input0 output1 -> C
C: input0 output1 -> C
C: input1 output0 -> B
Solution: 1001

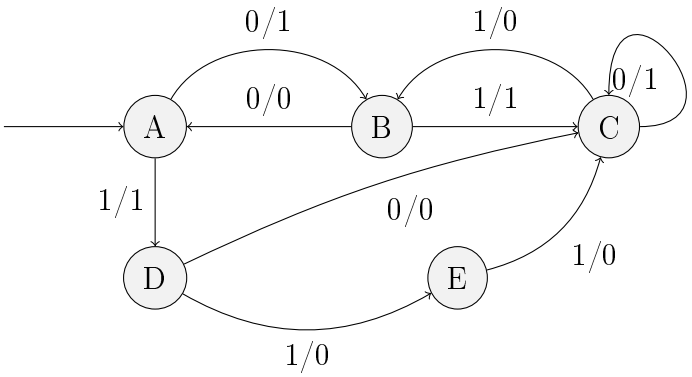

Figure 1: Mealy Sequential State Machine

(b) The State Machine in Figure 1 is not complete and has an undefined transition from one state to another: fix this with a solution in Figure 1. [1]

Solution: Node E needs to have another outgoing arrow with 0/? (e.g. to node E).

Question 27 ..... Points: [2]

Write in C code a single statement which only *sets* bit number 0 in PORTA and let the other bits *untouched* (bit number 0 is the least significant bit):

Solution: PORTA |= (1«0);

Question 28 ..... Points: [4]

(a) Provide an example of a typical *Transforming System*, and explain why this is a Transforming System: [2]

Solution: Network router, it transform packes and distributes them.

(b) Explain why *Optimized Memory Usage* is a typical attribute for a *Transforming System*: [2]

Solution: Such systems transform an input stream into an output stream, and this usually involves larger amount of memory for buffering and transforming. As memory is expensive, such systems need to be optimized for this.

Question 29 ..... Points: [3]

Explain the reason why some processors push all their core registers onto the interrupt stack, and some only push a subset of the registers:

Solution: In order not to increase the interrupt latency in case there are many registers, and as well to reduce the chance for stack overflow.

Question 30 ..... **Points: [3]**

(a) List 2 typical reactive systems: [1]

Solution: Airbag, ABS

(b) List 2 typical interactive systems: [1]

Solution: PDA, ticket selling machine

(c) List 2 typical transformative systems: [1]

Solution: network router, encryption engine

Total: 3

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

List reasons, why a company would *not* allow any interrupt synchronization methods:

Solution: Timing might be difficult to calculate. Everything must be deterministic. Problem with stack consumption, timing problems, missed interrupts. Interrupts might not be acknowledged.

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

List the things a processor has to do in order to jump to an interrupt service routine and to return from it.

Solution: Stop actual instruction (or undo, or finish), calculate new address based on vector, store status on stack, branch to ISR and context switch. Do the same thing in reverse order to return from the ISR.

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

Explain multiple things which affects the interrupt latency time.

Solution: Latency time depends on the speed of the CPU, the amount of data/registers/stack to be changed for the context switch, and the entry time inside the ISR until the ISR routine can start.

Question 34 ..... **Points: [1]**

What does it mean, if somebody says "I have masked the interrupts"?

Solution: The interrupts are disabled.

Question 35 ..... **Points: [2]**

Does the ARM Cortex M0+ support nested interrupts?

35. No

Solution: Yes, an interrupt with lower interrupt priority number (higher interrupt priority number) can be interrupted by an interrupt source with higher priority (lower interrupt priority number).

Question 36 ..... Points: [5]

Answer the questions for following C code, assuming default compiler settings:

```
typedef signed short MyType;
static unsigned char myVar[3];
typedef enum { RED=5, GREEN, YELLOW } Colors;
```

(a) What gives `sizeof(MyType)` for the FRDM board/project: [1]

(a) 2

(b) What gives `sizeof(MyType)` for the Robot board/project? [1]

(b) 2

(c) What gives `sizeof(MyVar)` for the FRDM board/project? [1]

(c) 3

(d) What gives `sizeof(MyVar)` for the Robot board/project? [1]

(d) 3

(e) Which value has `YELLOW`? [1]

(e) 7

Total: 5

Question 37 ..... Points: [3]

Given following source code:

```
uint16_t abcd[16];
uint8_t buf[10];
static uint16_t values[3];
```

(a) Determine the value of following expression: [1]

```
sizeof("abcd")
```

(a) 5

(b) Determine the value of following expression: [1]

```
sizeof(buf)
```

(b) 10

(c) Determine the value of following expression: [1]

```
sizeof(values)
```

(c) 6

Question 38 ..... Points: [3]

You have to implement the graph in Figure 2 with Doxygen/Dot.
Fill the gaps ('_____') in following code:

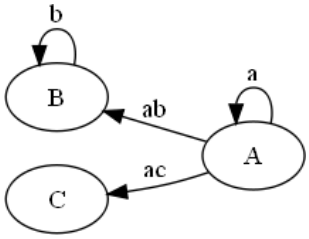

Figure 2: Doxygen ABC

```
\dot
digraph example_dot_graph {
node [];

rankdir = RL ___________;

A __ [label = ''A''] ______;

B__ [label = ''B'' _______];

C__ [label = ''C''________];

A__ -> _A __[label=''a''];

A -> B ____[label=''ab''];

A -> C ____[label=''ac''];

B -> B [label=''b''];______

}

\enddot
```

Question 39 ..... Points: [3]

Consider following doxygen source:

```
\dot
digraph example_dot_graph {
    node [shape=triangle];
```

```
    rankdir=RL;
    A    [style=filled,label="A" ];
    B    [style=filled,label="B" ];
    C    [style=filled,label="C"];
    A -> A [label="a/b"];
    A -> B -> C -> A -> C;
    B -> B [label="b/c"];
}
\enddot
```

This produces the following graph:

⊖ ± Solution is Figure 3.

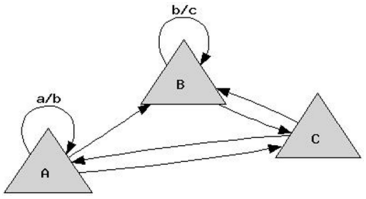

Figure 3: Dot Graph A

⊖ ± Solution is Figure 4.

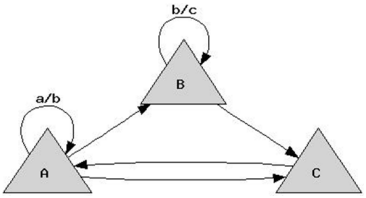

Figure 4: Dot Graph B

⊖ ± Solution is Figure 5.

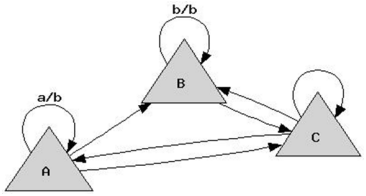

Figure 5: Dot Graph C

⊖ ± Solution is Figure 6.

Question 40 ..... Points: [3]

In eclipse you have different ways how you could reference external files within your project structure:

1. Linked Folder

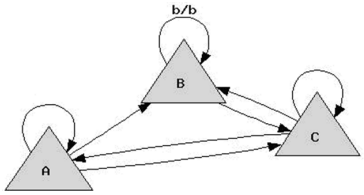

Figure 6: Dot Graph D

2. Linked Files
3. Virtual Group

List pros and cons for each approach:

Solution: Linked folder: Pros: new files in that folder automatically get added to the project. Cons: you get all or nothing.
Linked Files: Pros: You can decide for each remote file if it is included or not. Cons: you need to do this for every file.
Virtual Group: Pro: arbitrary group of files. Cons: No physical folder you can use for the build tools settings.

Question 41 ..... Points: [2]

A hard realtime system or a soft realtime system: which do you consider easier to implement and test? List one pro and one cons for each:

Solution: A hard realtime system is probably harder to implement, but it is easier to test, as if you can make it fail a deadline, it is clear that it fails. A soft realtime system is probably easier to implement as it does not have to stick to hard deadlines, but it will be more difficult to test.

Question 42 ..... Points: [2]

Consider following source:

```
#define MACRO(a,b) a = j \
  =b
```

Write down the text which would be produced by the preprocessor of the compiler, if you call the `MACRO` as following:

```
MACRO(i,5);
```

Solution:

```
MACRO(i,5);
 i = j = 5;
```

Question 43 ..... Points: [3]

Implement a function with doxygen comments which creates a doxygen output as in Figure 7.

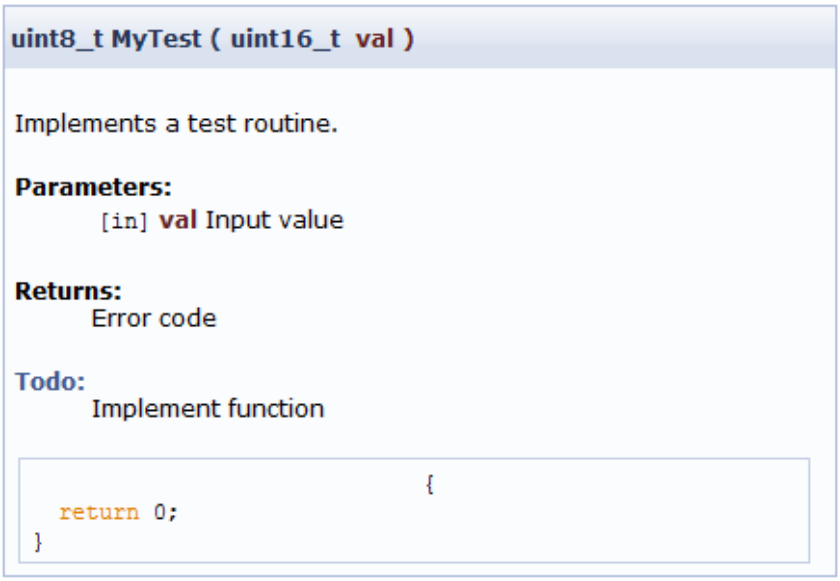

Figure 7: Doxygen for MyTest()

Solution:

```
/*!
 * \brief Implements a test routine
 * \param[in] val Input value
 * \return Error code
 */
uint8_t MyTest(uint16_t val) {
  /*! \todo Implement function */
  return 0;
}
```

Question 44 ..... Points: [1]

Given following C source:

```
#define MACRO(var, mask1, mask2) \
  (var = (var & (~(uint8_t)(mask1))) | (uint8_t)(mask2))
static uint8_t var;

void foo(void) {
  var = 0x22;
  MACRO(var, 16, 0x13);
}
```


Reached: _____
from 4 points

What is the value of `var` after execution of `foo()`?

44. 0x33 oder 51

Solution:

```
var = 0x22;
(var = (var & (~(uint8_t)(16))) | (uint8_t)(0x13));

var = (var & (~(uint8_t)(0x10))) | (uint8_t)(0x13);

==> clear bits in mask1, set bits in mask2
==> clear bit 0x10, set bits 0x13
==> clear has no effect, so it is 0x22 | 0x13 ==> 0x33
```

Question 45 ..... **Points: [1]**

Which sequence is the correct one to configure a keyboard interrupt?

- ◯ Enable Keyboard Interrupts;
  Set Port direction as input;
  Enable Pull-Up Resistors;
  Acknowledge Pending Interrupt;
- ◯ Acknowledge Pending Interrupt;
  Set Port direction register as input;
  Enable Keyboard Interrupts;
  Enable Pull-Up Resistors;
- √ Set Port direction register as input;
  Enable Pull-Up Resistors;
  Acknowledge Pending Interrupt;
  Enable Keyboard Interrupts;
- ◯ Enable Pull-Up Resistors;
  Enable Keyboard Interrupts;
  Acknowledge Pending Interrupt;
  Set Port direction register as input;

Question 46 ..... **Points: [1]**

For the implementation of a driver for an interrupt hardware following has to be considered:

- ⊖ ± Interrupts have to be enabled globally during the driver initialization.
- ⊕ ± The driver shall reset the device interrupt flag during initialization.
- ⊕ ± After a power-on reset, it might be necessary to wait a certain time until the hardware signals have stabilized.
- ⊖ ± The interrupt handler shall be as efficient as possible in order to increase the interrupt latency time.

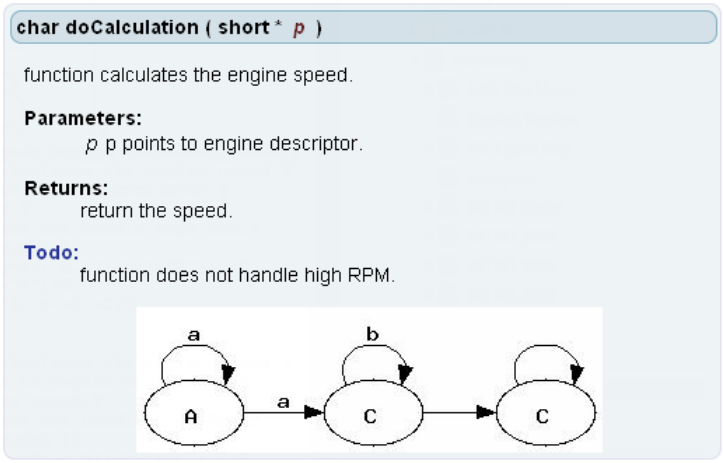

Figure 8: Doxygen Dokumenation

Question 47 ..... Points: [1]

Given the doxygen graph in Figure 8. This figure has been created with following doxygen extract:

○
```
\dot digraph a_graph {
  node[],rankdir=RL; A,B[label="B"],C;
  A->A->B[label="a"];
  B->B[label="b"]; B->C->C;
 }
\enddot
```

√
```
\dot digraph b_graph {
  node[],rankdir=LR; A,B[label="C"],C;
  A->A->B[label="a"];
  B->B[label="b"];  B->C->C;
 }
\enddot
```

○
```
\dot digraph c_graph {
  node[],rankdir=RL; A,B,C;
  A->A->B[label="b"];
  B->B[label="a"]; B->C;
 }
\enddot
```

○
```
\dot digraph d_graph {
  node[],rankdir=LR; A[label="B"],B,C;
  A->A->B[label="b"];
  B->B[label="a"]; B->C;
 }
\enddot
```

○
```
\dot digraph e_graph {
  node[],rankdir=LR; A,B,C[label="B"];
  B->A->B[label="b"];
  B->B[label="a"]; B->C->A;
```

```
}
\enddot
```

Question 48 ..... **2 Points (Bonus)**

And here could be your bonus question. ☺.

Question 49 ..... **Points: [1]**

Given following program:

```
void main(void) {
unsigned char *src=(unsigned char*)0x100, buffer[0x100], i;
  for(i=0;i<100; i++) {
    buffer[i]=*src;
  }
}
```

For the above program, following applies:

- ⊖ ± It reads the values from the address 256 and 512 and stores it in a buffer.
- ⊕ ± It reads 100 times the value at the address 0x100 and stores the values one after each other in a buffer.
- ⊖ ± At termination of the program, the whole buffer is filled with the values from address 0x100.
- ⊕ ± With disabled interrupts, the program behaves in a deterministic way.

Solution: 3rd Answer: only part of the buffer ([0]..[99] is filled with *0x100, but the buffer has the size of [0x100].

Question 50 ..... **Points: [7]**

Given the Mealy Sequential State Machine in Figure 9.

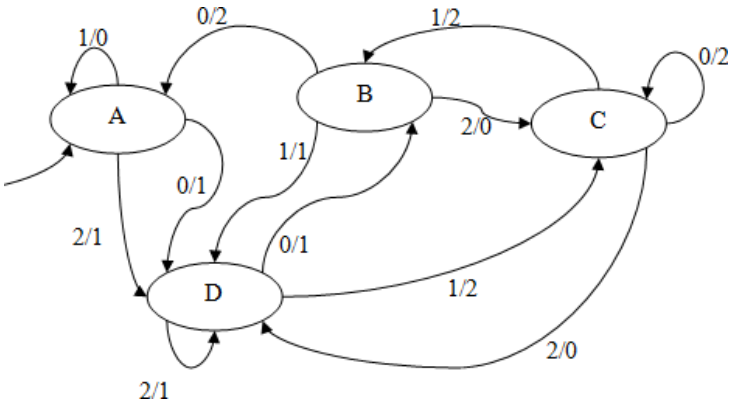

Figure 9: Mealy Machine

(a) The machine in Figure 9 is currently in state 'C'. Determine the output sequence for following input values: 0, 1, 0, 1, 1, 0 [1]

Solution: 2, 2, 2, 0, 0, 1

(b) Given following Mealy program: [6]

```
typedef enum {A=0, B, C, D} States;
void Run(void) {
  char j, i = 0

  for(;;) {
    j = Input();
    Output(tbl[i][j][1]);
    i = tbl[i][j][0];
  }
}
```

To implement the machine in Figure 9, complete the initialization of table `tbl`:

```
const char tbl[4][3][2] = {
```

Solution:

```
{ /*      0       1       2   */
/*A*/  {{D,1}, {A,0}, {D,1}},
/*B*/  {{A,2}, {D,1}, {C,0}},
/*C*/  {{C,2}, {B,2}, {D,0}},
/*D*/  {{B,1}, {C,2}, {D,1}}
};
```

Question 51 ..... Points: [1]

For realtime systems following applies:

⊖ ± Realtime systems have to have reaction times below 1 ms in order to be realtime compliant.

⊕ ± For a realtime system not the average system load matters, but the highest possible system load.

⊖ ± Hard realtime systems are more difficult to verify, because the realtime conditions are not exactly specified.

⊖ ± A system can be a realtime system, if it is using true random number generator for its decision instead of a pseudo random number generator.

Question 52 ..... Points: [1]

Given following program:

```
char buf[0x100];
int i,j;

static void test(void) {
  for(i=0; i<sizeof(buf); i++) {
```

```
    CFG = 0x80; PORTB = 4;
    buf[i] = PORTA;
    PORTB = 0;
  }
}
```

For this program following applies:

⊖ ± Implements an interrupt synchronization.

⊖ ± Implements a gadfly synchronization.

⊖ ± Implements a realtime synchronization.

⊕ ± None of above.

Question 53 ..... Points: [1]

For all reentrant functions in C, following has to apply:

⊖ ± The function shall not be recursive.

⊖ ± The function shall not be called from an ISR.

⊕ ± The access to shared data has to be protected from mutual access.

⊕ ± The function shall not modify itself (self modifying code).

Question 54 ..... Points: [1]

The following program gets compiled for the FRDM board with default compiler options:

```
static char ch@0x10;
void foo(void) {
  static char i, j=4;
  volatile char v;
  v = i;
  v = j;
  ch = v;
}
```

Following applies:

⊖ ± The variables i, j and v are allocated on the stack.

⊕ ± The compiler cannot optimize the two assignments to v because of volatile.

⊖ ± At execution time of foo(), the variable v gets initialized with a value of 4.

⊕ ± After execution of foo(), the memory at address 0x10 will have a value of 4.

Question 55 ..... Points: [1]

For the interrupt system of the ARM Cortex following applies:

 Question 55 continuous on the next page... Reached: _____ from 3 points

⊕ ± The interrupt latency is the sum of execution time of the current instruction, pushing of the registers, calculating the ISR PC address and the branching to the ISR itself.

⊖ ± With 'masking the interrupts' we are enabling the interrupts.

⊕ ± In order for the ISR program to return to the interrupted program, the return address of the interrupted program is stored on the stack by the hardware.

⊕ ± In order to reduce the interrupt latency time, the core can decide not to push all registers on the stack.

Question 56 ..... **Points: [2]**

The diagram in Figure 10 shows an interrupt system with multiple interrupts (IRQ1 and IRQ2) and the corresponding interrupt service routines (ISR) #1 and #2). The lines on the time axis denote the execution time boundaries of the instructions. Following applies:

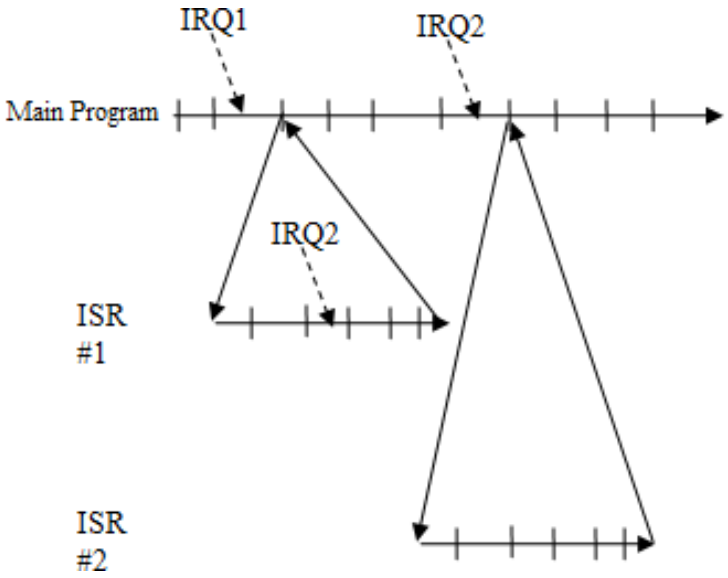

Figure 10: Interrupts

◯ At the beginning of ISR #1 all interrupts get disabled, and at the end of ISR #1 the flag for IRQ1 gets acknowledged.

◯ The main program has at the beginning all interrupts disabled and has the IRQ1 flag acknowledged. After execution of ISR #1 the main program enables all interrupts.

√ ISR #1 turns off all interrupts at the beginning. At the end of ISR #1 it acknowledged the IRQ1 and IRQ2 flag and enables all interrupts again.

◯ At the beginning of ISR #1 the flags for IRQ1 and IRQ2 are acknowledged. All interrupts get disabled at the end of ISR #1.

◯ ISR #1 has not acknowledged the IRQ1 flag. ISR #2 acknowledged the flags for IRQ1 and IRQ2 at the beginning of ISR #2.

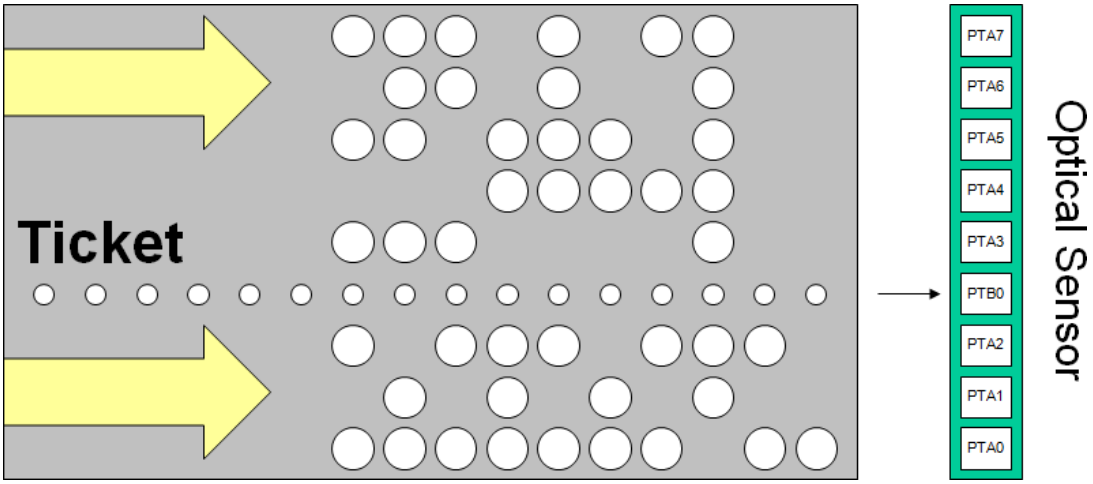

Figure 11: Parking Ticket

Question 57 ..... Points: [5]

A punched paper ticket is used in a parking system. The punched paper ticket is using following format for each data line in Figure11:

- 1 guidance bit (small holes)
- 8 data bits (large holes)

The punched paper tape gets pulled into the machine with constant speed of 50 ms for each data line. The data lines are scanned with an optical sensor, and the sensor digital output is attached to the port of a microcontroller. The state of the sensor/holes is available on the microcontroller PORTA, bit 0 to 7:

- Value of bit is 0: no hole, light does not go through
- Value of bit is 1: hole, light goes through

The state of the guidance hole is available on bit 0 of PORTB. The bit 0 of PORTB is configured to raise an keyboard interrupt on falling edge.

Given following program:

```
extern WaitMs(unsigned int ms); /* wait for the given ms */
unsigned char buffer[16]; /* contains the data read */

void Read(void) {
  uint8_t i;

  for(i=0; i<sizeof(buffer); i++) {
    WaitMs(50);
    buffer[i] = PORTA;
  }
}

interrupt KBI(void) {
  /* Guidance Hole Sensor */
  AcknowledgeKBI();
  DisableInterrupts();
  Read();
```

 Question 57 continuous on the next page... Reached: _____ from 0 points

```
  EnableInterrupts
}

void main(void) {
  for(;;);
}
```

(a) Which synchronization method is used for the detection of **insertion** of the parking ticket? [1]

- ◯ Combination of interrupt and realtime synchronization.
- √ Interrupt synchronization.
- ◯ Realtime synchronization.
- ◯ Combination of gadfly synchronization and realtime synchronization.
- ◯ Gadfly synchronization.

(b) Which synchronization method is used for the synchronization on the **first data hole**? [1]

- ◯ Combination of interrupt and realtime synchronization.
- ◯ Interrupt synchronization.
- √ Realtime synchronization.
- ◯ Combination of gadfly synchronization und realtime synchronization.
- ◯ Gadfly synchronization.

(c) Implement a new function `ReadGadfly()` which does the same as `Read()`, but uses a gadfly synchronization method. [3]

```
void ReadGadfly(void) { ...
```

Solution:

```
void ReadGadfly(void) {
 uint8_t i;

 for(i=0; i<sizeof(buffer); i++) {
   while((PORTB&1)==0);
   buffer[i] = PORTA;
   while((PORTB&1)==1);
 } /* end for */
} /* end ReadGadfly */
```

Total: 5

Question 58 ..... Points: [1]

Given following program:

```
double power(double x, int exp) {
  if (exp<=0) return 1;
```

```
  return(x*power(x, exp-1));
}
```

Evaluate following:

- ⊖ ± In order to have this program reentrant, it is sufficient that `x` and `exp` are variables on a hardware stack.
- ⊕ ± It depends on the compiler and the generated code, if this program is reentrant or not.
- ⊖ ± The program is reentrant if it is called from an interrupt service routine only.
- ⊖ ± The recursive implementation of this program ensures that it is reentrant.

Solution: Notice that the compiler routines for float operations (runtime routines) might not be reentrant. And not every hardware architecture is using a hardware stack for local variables, e.g. some PIC controlles or ST5 do not have a hardware stack (variables are on a software stack which prevent recursion or reentrant code).

Question 59 ..... Points: [5]

Given a system in Table 1 with programs, priorities and timing:

Program	*Main Priority*	*Sub Priority*	*Time*
HP	0	0	5 ms
UP1	1	1	2 μs
UP2	1	2	3 μs
UP3	2	1	5 μs
UP4	2	2	2 μs

Table 1: Interrupt System

The timing required for a context switch is given in table 2, which is illustrated in Figure 12.

Context Switch	*Time*
Total time for the interrupt, switch to a new program and starting execution of the waiting program	1 μs
Total time for the interrupt, switch to the interrupted program, immediate interruption of this program and switching and starting execution of the waiting program	1 μs

Table 2: Context Switch Timing

The interrupt system is using following rules (as used in the lecture):

$$if(MP(s) <= MP(fn)) \rightarrow \; ws = ws \cup s \tag{1}$$

 Question 59 continuous on the next page... Reached: _____ from 5 points

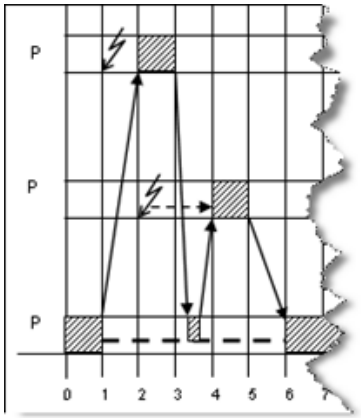

Figure 12: Example Context Switch

$$if(MP(s) > MP(fn)) \rightarrow INT(fn) \tag{2}$$
$$if(MP(s) <= MP(fn)) \rightarrow \ ws = ws \cup s \tag{3}$$
$$if(SP(s) > SP(fn)) \rightarrow \ ws = ws \cup s \tag{4}$$
$$if(ws \neq \{\}) \rightarrow fn(MAX(SP(ws))) \tag{5}$$
$$if(MP(ws) > MP(in)) \rightarrow \ fn = in \rightarrow ws = in \tag{6}$$

The programs run according following information:

1. At the time 0 μs HP starts.
2. At the time 2 μs an interrupt for UP1 is raised.
3. At the time 4 μs an interrupt for UP4 is raised
4. At the time 6 μs an interrupt for UP2 is raised
5. At the time 9 μs an interrupt for UP3 is raised
6. At the time 23 μs an interrupt for UP2 is raised

Show the sequence of programs and interrupts in Figure 13. Use the same notation as in Figure 12 for interrupts (Exception, Pending), program switches, program (aktive, suspended).

Solution: See Figure below.

1. A signal with a higher main priority interrupts always a running program.
2. A signal with same main priority interrupts never.
3. A signal with higher sub priority has to wait for an already started program with same main priority, even if it had been interrupted.
4. If there are multiple signals waiting with the same main priority, then the sub priority decides which one will be handled first. With sub priorities it is possible to influence the sequence of execution and you are not depending on the signal raise time.

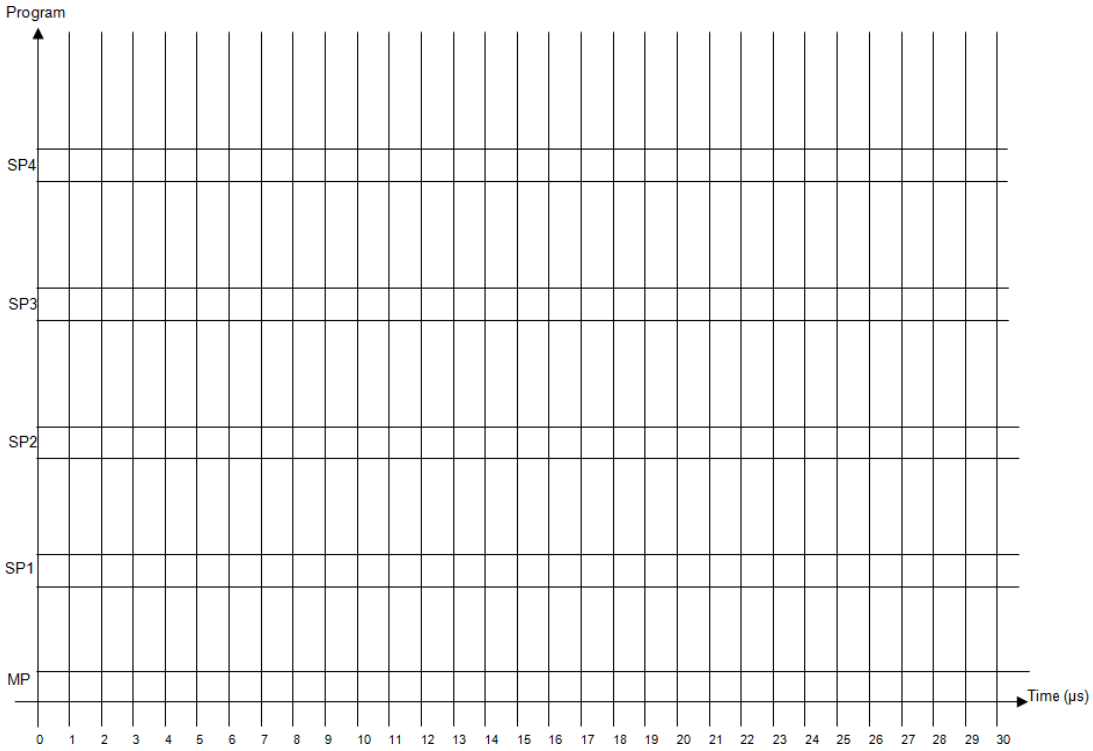

Figure 13: Program Timing

5. If there is a pending signal with a higher main priority than the last interrupted program, then the control goes back to that interrupted program, but gets interrupted directly again.

Question 60 ..... Points: [3]

Consider following implementation:

```
#define FUNC(a,b) i+a+b
int foo(int i, int j) {
  return FUNC(i,j);
}
```

Determin the return value for `foo(5,6);`:

60. 16

Solution:

```
return FUNC(i,j);
return i+i+j;
return 5+5+6;
return 16;
```

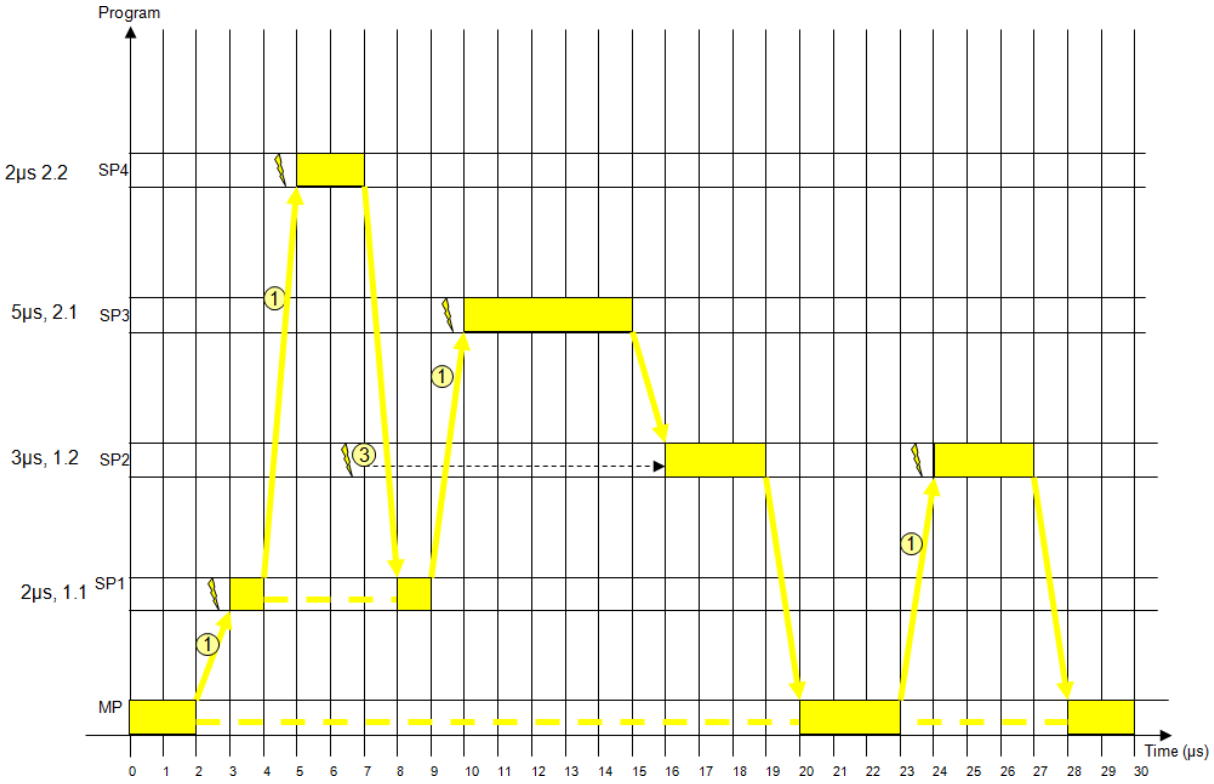

Figure 14: Program Timing (Solution)

Question 61 ..... **Points: [3]**

Given following source code:

```
uint16_t abcd[8];
uint32_t buf[10];
static uint16_t values[2];
```

(a) Determine the value of following expression: [1]

```
sizeof("abcd")
```

(a) 5

(b) Determine the value of following expression: [1]

```
sizeof(buf)
```

(b) 40

(c) Determine the value of following expression: [1]

```
sizeof(values)
```

(c) 4

Question 62 ..... **Points: [2]**

Consider following program:

```
unsigned char data@0x10;
```

```
#define p_data ((unsigned char*)0x10)
void foo(void) {
  data = 10;
  *p_data = 17;
  data++;
}
```

After execution of `foo()`, following applies:

- ⊖ ± data == 11, *p_data == 17, p_data == 10
- ⊖ ± data == 10, *p_data == 18, p_data == 17
- ⊖ ± data == 10, *p_data == 17, p_data == 0x10
- ⊖ ± data == 17, *p_data == 17, p_data == 16
- ⊕ ± data == 18, *p_data == 18, p_data == 16

Question 63 ..... Points: [2]

Consider following program:

```
void delay(void) {
  uint8_t i;
  for(i=0;i<50;i++);
}
```

This program

- ⊖ ± always waits for 50 ms
- ⊕ ± can be optimized by a smart compiler to a function which only contains a `return;` statement
- ⊕ ± will wait for a certain time which is depending on the speed of the micro-controller used
- ⊖ ± will never terminate

Question 64 ..... Points: [2]

Consider following Mealy Sequential State Machine with 3 states and two inputs:

```
typedef enum {A, B, C, D, E, } States;
const char tbl[3][2][2] =
{ {{A,0}, {B,1}},
  {{C,3}, {A,4}},
  {{C,0}, {B,5}}
};

void Run(void) {
  char j, i = 0

  for(;;) {
    j = Input();
    Output(tbl[i][j][1]);
    i = tbl[i][j][0];
```

 Question 64 continuous on the next page... Reached: _____ from 2 points

```
  }
}
```

(a) Given following sequence of `Input()` values: 0, 1, 0, 1, 1, 1. Determine the sequence of `Output()` values: [1]

(a) 0, 1, 3, 5, 4, 1

(b) Draw the corresponding state diagram: [1]

Solution: See diagram.

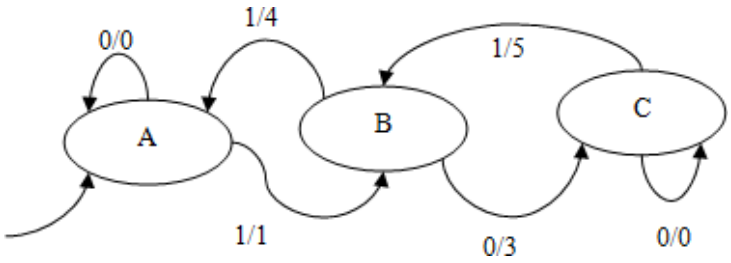

Figure 15: Solution Mealy Diagram Drawing

Total: 2

Question 65 ..... Points: [3]

Consider following program:

```
void main (void) {
  char buf[0x100];
  int i,j;

  PORTB = 0;
  for(i=0; i<sizeof(buf); i++) {
    CFG = 0x80; PORTB = 4;
    while(CFG!=0);
    buf[i] = PORTA;
    PORTB = 0;
  }
}
```

 Question 65 continuous on the next page... Reached: _____ from 5 points

The following applies:

- ⊖ ± It implements an interrupt synchronization.
- ⊕ ± It implements a Gadfly synchronization.
- ⊖ ± It implements a Realtime synchronization.
- ⊖ ± It implements no synchronization.

Question 66 ..... Points: [3]

Evaluate following statements about reentrancy:

- ⊕ ± A function which modifies its own code is not reentrant.
- ⊖ ± A function which calls an interrupt service routine is not reentrant.
- ⊖ ± Recursive functions are always reentrant.
- ⊖ ± Interrupt service routines are always reentrant if they do not call another routine.

Question 67 ..... Points: [3]

If discuss interactive, reactive and transforming systems, then

- ⊕ ± relative short answer times are typical for interactive systems.
- ⊕ ± reactive systems are common in systems which do measurement and control.
- ⊕ ± transforming systems are typically optimized for high throughput.
- ⊕ ± an example for an transforming system could be a network router.

Question 68 ..... Points: [3]

In the context of real time following applies:

- ⊖ ± Realtime means to produce a result as fast as possible.
- ⊖ ± A computer is realtime, if is is able to produce at average system load the correct result as fast as possible.
- ⊖ ± For realtime it is sufficient to have an accurate timing system.
- ⊖ ± An RTOS is required for a realtime system.

Question 69 ..... Points: [3]

For all reentrant functions implemented in C the following applies:

- ⊖ ± A reentrant function shall not be interrupted.
- ⊖ ± Interrupt functions does not have to be reentrant, but all functions called from that interrupt routine.
- ⊖ ± A function which modify itself is reentrant, as long the self modification happens with disable interrupts.
- ⊕ ± On the ARM Cortex-M0+/M4F the usage of local stack variables does not violate reentrancy.

Question 70 ..... **Points: [2]**

Explain in a few words the reasons why a switch (like a button) needs a resistor. Illustrate it with a small drawing.

Solution: Without a resistor, the signal is undetermined or open. It is needed to define the signal to a defined level (low or high) if the switch is not closed.

Question 71 ..... **Points: [2]**

List important points to be considered for the implementation of an ISR:

Solution: As short and as fast as possible. Need to acknowledge the interrupt at the beginning (or additionally at the end. Need to care about reentrancy, and that shared functions are reentrant and shared variables are protected.

Question 72 ..... **Points: [2]**

Explain two different ways how a microcontroller can implement interrupts:

Solution: Using a vector table: the interrupt source gets translated into a vector number which is used as index into a vector table. The other way is that the processor directly jumps to an address and executes the code there (stub based approach).

Question 73 ..... **Points: [1]**

Identify the function of device *U9* in Figure 16?

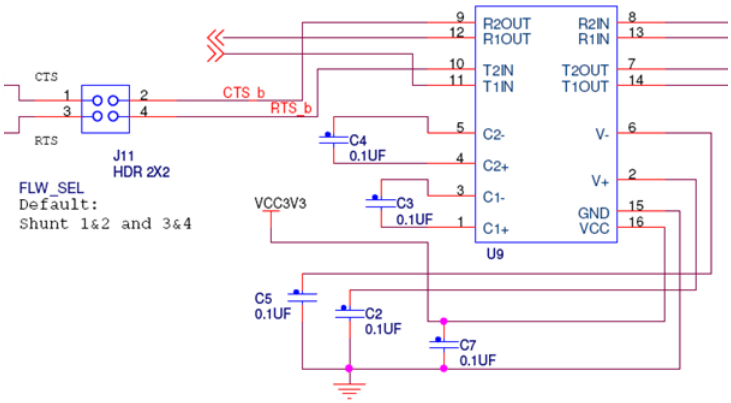

Figure 16: IC U9

- ◯ Motor H-Bridge IC.
- ◯ Quadratur Decoder IC.
- ◯ Analog/Digital Converter IC.
- √ RS-232 Level Shifter IC.
- ◯ Accelerometer IC.

Question 74 ..... **Points: [2]**

What happens if two developers work at the same project but in different files or at different parts in one file and commit on Git?

Solution: Git merge automatically the changes together.

Question 75 ..... **Points: [2]**

Which is the most important difference between SVN and Git?

Solution: SVN: Centralized VCS, Git: Distributed VCS.

Question 76 ..... **Points: [2]**

What happens if two developers work at the same file and at the same part and commit on Git?

Solution: There will be a conflict which cannot be resolved by the VCS. The developers have to solve this problem manually (solving the conflict).

Question 77 ..... **Points: [2]**

What is the meaning of the `.gitignore` file?

Solution: Every file is load in the repository except those are mentioned in the `.gitignore`.

Question 78 ..... **Points: [2]**

Why should you not put the documentation produced by doxygen into the repository?

Solution: It's not necessary, because the documentation is generated out of the code. Apart from that, there would be too many unnecessarily data on the repository.

Question 79 ..... **Points: [1]**

Name one example each for a transforming system, reactive system and interactive system?

Solution: TS: Video encoder, RS: PID Controller, IS: ticket selling machine

Question 80 ..... **Points: [2]**

Why is it necessary to put parenthesis for macros like in the example below?

```
#define CALC1 (2+5)
#define CALC2 (5*3)
```

Solution: To ensure proper usage in other macros or in calculations. 10*CALC1 is not the same as (10*2)+5

Question 81 ..... **Points: [2]**

List advantages and disadvantages using macros:

Solution: Advantages: Faster code, smaller code. Disadvantages: Interface, Encapsulation, Debugging

Question 82 ..... **Points: [2]**

What needs to be present in a header file to avoid recursive inclusions? Give an example.

Solution:

```
#ifndef__LED_H_
#define__LED_H_
/* content of header file */
#endif
```

Question 83 ..... **Points: [2]**

How do you declare global variables? In what kind of file?

Solution: In a header file with *.h extension. Using `extern` for variable declaration, e.g. `extern int LED_global;` I use a good name with prefix because the name is visible in the whole project.

Question 84 ..... **Points: [1]**

Is it possible to include other files than `*.h` with `#include`? Can you give an example?

Solution: Yes, any text files can be included (as long as understood by the compiler). For example I can have an array of hex values in an array and then include it as bitmap.txt.

Question 85 ..... **Points: [1]**

List 4 different kind of output (result) files which can be generated by doxygen:

Solution: HTML, RTF, PDF, LaTeX

Question 86 ..... **Points: [1]**

What is the concept of doxygen, how does it generate the documentation?

Solution: Doxygen extracts the documentation from the source file and from the comments embedded in the files.

Question 87 ..... **Points: [1]**

What is the disadvantage of Gadfly synchronization?

Solution: It blocks further execution.

Question 88 ..... **Points: [2]**

How can you prevent that two interrupts access the same data at the same time?

Solution: Each ISR needs to ensure that the other ISR is not executed. This can be with priorities, or with disabling the interrupt for the other ISR.

Question 89 ..... **Points: [2]**

How can you prevent that two interrupts access the same data at the same time?

Solution: Each ISR needs to ensure that the other ISR is not executed. This can be with priorities, or with disabling the interrupt for the other ISR.

Question 90 ..... **Points: [1]**

What have you to do with unneeded interrupts?

Solution: Disable or mask them, or have a dummy ISR.

Question 91 ..... **Points: [1]**

What is a S19 file?

Solution: Text file with code (encoded as ASCII in a defined format).

Question 92 ..... **Points: [2]**

What kind of two different events do exist?

Solution: Synchronous (periodic timer interrupt, periodic task output) and asynchronous (button pressed, transceiver packet received).

Question 93 ..... **Points: [2]**

What is the purpose of a 'sentinel'?

Solution: Marking the end of a list. Additionally it is used to calculate the size of the array or to compute the number of items.

Question 94 ..... **Points: [2]**

What are the advantages and disadvantages of handling events from the main loop?

Solution: Advantage: simple. Disadvantages: long if-elsif-else/switch in the event handler, order of events has impact about priority, need mutual exclusion for shared data.

Question 95 ..... **Points: [1]**

What kind of processor is used on the FRDM-KL25Z?

Solution: ARM Cortex-M0+

Question 96 ..... **Points: [1]**

List the two different hardware stack pointer present on an ARM Cortex M:

Solution: Main Stack Pointer (MSP) and Process Stack Pointer (PSP).

Question 97 ..... **Points: [1]**

What are the two different categories of timeliness in realtime systems?

Solution: Absolute (e.g. 13:50) or relative (e.g. after 50 ms).

Question 98 ..... **Points: [1]**

What kind of external clock is used on the FRDM-KL25Z?

Solution: 8 MHz crystal oscillator.

Question 99 ..... **Points: [1]**

List 3 different types of synchronization:

Solution: Gadfly, interrupt, realtime.

Question 100 ..... **Points: [1]**

How can the logic level of a pin be enforced?

Solution: Internal (port configuration, pull resistor), external circuit (pull resistor).

Question 101 ..... **Points: [1]**

What is the purpose of pull-up and pull-down resistors for input pins?

Solution: To have defined voltage levels.

Question 102 ..... **Points: [1]**

List at three different state machine design patterns:

Solution: functional, hierarchical, else-if state machine.

Question 103 ..... **Points: [1]**

List at three different ways how to implement a state machine:

Solution: if-elsif-else, switch, table

Question 104 ..... **Points: [1]**

What determines the output in a Mealy Sequential State Machine?

Solution: determined by current state and current output.

Question 105 ..... **Points: [1]**

What is the main advantage of using the *Trigger* module?

Solution: Handle multiple 'interrupt like things' with just one timer.

Question 106 ..... **Points: [1]**

You are using the *Trigger* module with a 10 ms timer interrupt. Now you want to trigger something in 50 ms. Is this possible?

Solution: yes.

Question 107 ..... **Points: [1]**

Why is it important to debounce a mechanical switch?

Solution: To avoid multiple interrupts, and to have reached a stable state.

Question 108 ..... **Points: [1]**

List two ways to debounce a mechanical switch:

Solution: hardware (R-C) or software (timer, delay).

Question 109 ..... **Points: [1]**

What is the advantage of using USB as a virtual UART serial connection (OpenSDA and USB CDC) over a direct USB connection to a USB port of the MCU ?

Solution: USB is a much more complex protocol, and therefore, has a bigger overhead. If the USB performance is not needed, a simpler UART connection is more efficient. At the Freedom board, there is an extra MCU to convert the USB to a UART protocol. This means that there is an extra MCU to take over the USB overhead.

Question 110 ..... **Points: [1]**

What is the meaning of an asynchronous serial protocol ?

Solution: Serial means that it sends the data bits after bits on a single data line (as a sample Rx or Tx line). Asynchronous means that there is no clock supported to read the data. The start of the data has to be detected by the protocol (start bits, stop bits).

Question 111 ..... **Points: [1]**

As which device class does the Freedom board enumerate at the PC, and what is the purpose of it?

Solution: CDC (Communication Device Class), the PC opens a virtual COM port for the device. This enables to have a connection, which acts like a UART connection. Furthermore, it also enumerates as Mass Storage Device, this mode can be used to update the firmware.

Question 112 ..... **Points: [1]**

What's the command parser table?

Solution: It's a list with function pointers. In this list the parser choses the method which is going to be executed.

Question 113 ..... **Points: [1]**

What's the difference between Memory Scheme 1 and 2 in FreeRTOS?

Solution: Memory Scheme 1 only allocates Memory. It's not possible to delete Tasks. With Scheme 2 it's possible to free space and reuse. There would be another Scheme where you can merge freed blocks.

Question 114 ..... **Points: [1]**

Provide a good example how FreeRTOS queues can be used between multiple tasks:

Solution: To send messages between tasks.

Question 115 ..... **Points: [1]**

What is the difference between `xQueueReceive()` and `xQueuePeek()` in FreeRTOS?

Solution: `xQueuePeek()` only checks if there is an item in the queue. `xQueueReceive()` does remove the item.

Question 116 ..... **Points: [1]**

What are the H-bridges of the motors needed for?

Solution: They are used to set the direction of the engine by simply twisting the two wires of the motor.

Question 117 ..... **Points: [1]**

Why can the PWM channels not run with different frequencies?

Solution: There is only one Timer for all the different PWM-channels and therefore the frequency can only set for this common timer. The different channels only have a separate value register which all compare with the same common timer.

Question 118 ..... **Points: [1]**

On the SUMO robot, why can you not use the coast mode of the engines when stopping?

Solution: The coast modus of the engines is only possible if "Mode 0" is used on the motor drivers. On the SUMO robot, this pin is not routet to the FRDM board and just hardware wise set to HIGH (Mode 1).

Question 119 ..... **Points: [1]**

How can you calibrate the offset of the accelerometer?

Solution: Measure the accelerometer value for a known value, e.g. 0g or 1g. Compare it with the value it should report. The difference is the offset correction.

Question 120 ..... **Points: [1]**

Can you tell two typical interfaces for a accelerometer and say how they are implemented?

Solution: Typical interfaces are analog with ADC and digital with SPI/I^2C.

Question 121 ..... **Points: [1]**

What would be good reasons to use industrial SD cards?

Solution: More write cycles, durable, defined read/write times.

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

What can we do to ensure the stored NVMC data are not corrupted?

Solution: Add a checksum for the blocks.

Question 123 ..... **Points: [1]**

What's the difference between a binary Code and the Grey Code? And what is the advantage of the Grey Code?

Solution: The Binary Code several bit changes between the counting steps are possible whereas the Grey Code changes only one bit between each step. It's possible to build the Grey Code recursive and it is permutable.

Question 124 ..... **Points: [1]**

What are the three possibilities to do the data acquisition?

Solution: Interrupts, Sampling, input capture, dedicated quadrature peripheral/IC

Question 125 ..... **Points: [1]**

Why do we use the sampling method and not the interrupt method for our robot?

Solution: The sampling method causes a constant system load whereas the interrupt method causes a speed dependent system load. It's easier to handle a constant system load to test a system.

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

What happens with the system if our robot drives to fast to handle the encoder signals? Compare the interrupt vs. the sampling methods.

Solution: If you use the interrupt method the system freezes (to many interrupts). If you use the sampling method the system still works normally, but can't recognize every encoder step (errors).

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

What are the input and output signals of `QuadCounter.c`?

Solution: The input signals are the two hardware signals and the output signals are the number of errors and the current position/steps.

Question 128 ..... **Points: [1]**

Why is there a NULL pointer at the end of the `CmdParserTable`-Array?

Solution: It is used as a sentinel, so the code can iterate through the table until there is a NULL entry.

Question 129 ..... **Points: [1]**

In the shell the command line parser compares strings with `sizeof("cmpString")-1`. Explain why this -1 is necessary:

Solution: sizeof() returns the size of the string including zero byte, but we want to compare the string without it.

Question 130 ..... **Points: [1]**

Which settings do you have to configure for the shell communication between computer and device?

Solution: baudrate, number of data bits, stopbit, parity

Question 131 ..... **Points: [1]**

Which ANSI-C keyword can you use to prevent loop optimization (and others) in the compiler?

Solution: volatile

Question 132 ..... **Points: [1]**

Why is it necessary to use synchronization between two systems?

Solution: Because they operate with different speeds.

Question 133 ..... **Points: [1]**

Why do have functions which are called both from an interrupt and the main program to be reentrant?

Solution: Because an interrupt can happen any time, it must be ensured that there is no data corruption.

Question 134 ..... **Points: [1]**

Does every microcontroller implement nested interrupts?

Solution: No, for example the HCS08 does not have nested interrupts.

Question 135 ..... **Points: [1]**

Which three interrupts have predefined interrupt priorities on the ARM Cortex-M0+ and cannot be changed?

Solution: Reset, NMI, HardFault

Question 136 ..... **Points: [1]**

On the ARM Cortex-M0+, can a HardFault interrupted by another interrupt?

Solution: Yes, e.g. by a Reset or NMI as they have numerically lower interrupt priorities.

Question 137 ..... **Points: [2]**

Using a quadrature counter: Provide guidelines when you would use the delta-time and when the delta-pos approach:

Solution: Measure the delta-time (duration of period) if frequency is low. Use counting the steps or periods if time intervals are short.

Question 138 ..... **Points: [2]**

Explain the advantage of using a ring buffer with quadrature steps for estimating the speed:

Solution: The speed can be estimated more frequently than the measurement interval. Additionally it allows an averaging of the speed. With the ring buffer a configurable and dynamic time span for the estimate can be used.

Question 139 ..... **Points: [3]**

Why is it necessary to use an anti-windup for a PID?

Solution: Is the system control value (e.g. PWM for the motor) limited in range, then the integral part can go out of this range. Then the integral sum will increase without having an additional impact on the control value, and delays the impact of the integral on the control value. Additionally the limit avoids a numerical overflow of the integral sum.

Question 140 ..... **Points: [1]**

You measure the maximum speed of your quadrature encoder signal, and you measure a quadrature step every 100 μs. Determine the sampling period needed:

Solution: Nyquist/Shannon: need to sample it with at least 50 μs.

Question 141 ..... **Points: [1]**

In your robot application, from where do you call the PID control loop?

Solution: There are several options: one way is to create a drive task and call the PID from there. Or to use a periodic interrupt to call the PID.

Question 142 ..... **Points: [2]**

Why is it not possible to directly measure the output signal of the optical quadrature encoder we have used?

Solution: The signal is an analog sinus-like signal, without a 50%-50% high-low signal distribution. To get a clean quadrature signal a DAC with comparator devices are used.

Question 143 ..... **Points: [3]**

Discuss the pros and cons of using either sampling or interrupt method for a quadrature signal:

Solution: Usually sampling is usually prefered as it will create a constant system load. However, this creates a high load of the system even if the wheel is not moving or only slowly moving. Using interrupts can cause problems if there are too many interrupts. It would allow a low system load if the wheel is not moving or slowly moving.

Question 144 ..... **Points: [2]**

What fundamental problem exists for absolute position encoders, and how can it be solved:

Solution: Because of mechanical tolerances, multiple bits can change from one step to another. The solution is to use a Gray code/encoder, as with this only one bit changes from one sector to another. In addition the Gray code is cyclic and is therefore ideal for wheel position measurement

Question 145 ..... **Points: [2]**

Can you list the main features of the MCP4728?

Solution: 12bit DA-Converter, includes an EEPROM to store DAC values and settings, I^2C bus and protocol, 4 DAC output signals.

Question 146 ..... **Points: [2]**

What are the special things or attributes of the Gray code?

Solution: Hamming distance of 1 (only one bit changes), the code has permutation (every code only occurs once), it is cyclic (last and first code confirm to the rules) and recursive (codes of lower order are embedded in code of higher order), and it is simple to transform a binary code into a Gray code.

Question 147 ..... **Points: [2]**

If the robot moves with a speed of 1 m/s, and you measure the reflectance sensor with 100 Hz, what would be an estimated way distance over the white line until you detect the white sumo line in the application?

Solution: If robo is moving with 1 m/s and we measure with 100 Hz, then the robot will move in average 2 cm until the motors get stopped.

Question 148 ..... **Points: [2]**

The reflectance sensor has two red LEDs to indicate if the sensor is on. For the red LEDs there is a 1K Ohm resistor in series to limit the current through the LED. But why is there another 220 Ohm resistor in series to the LED with that 1K Ohm resistor?

Solution: To limit the current through the photo diodes.

Question 149 ..... **Points: [2]**

What is the advantage of the capacitive discharge circuit used for the reflectance sensor?

Solution: This generates a digital signal we can measure, and we do not need an A/D converter pin.

Question 150 ..... **Points: [1]**

List three typical requirements for an RTOS:

Solution: Predictability, precise timing, speed.

Question 151 ..... **Points: [1]**

List three reasons why to use an RTOS:

Solution: Running multiple things in parallel, using RTOS services (queues, semaphores, mutex, ...), scalability of application.

Question 152 ..... **Points: [1]**

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

Solution: Preemptive: the RTOS distributes the processing time, tasks get suspended by the RTOS. Non-preemptive: the task are cooperative, they pas the control back to the kernel.

Question 153 ..... **Points: [1]**

What is the advantage of scheduling with an RTOS?

Solution: To maximize the CPU utilization among different tasks.

Question 154 ..... **Points: [1]**

What is the difference between an RTOS and a normal OS?

Solution: The RTOS needs to adhere to strict timing and needs to produce output with given timing constraints.

Question 155 ..... **Points: [1]**

List three states of the debounce state machine we have used:

Solution: IDLE, PRESSED, RELEASE.

Question 156 ..... **Points: [1]**

List two different solutions to debounce a push button:

Solution: Software (state machine, time delay/low pass filter) or hardware (capacitor, Schmitt-Trigger.

Question 157 ..... **Points: [1]**

Explain why debouncing is necessary:

Solution: Bouncing is a mechanical problem. To process the state of a bouncing push button, it needs first to stabilize.

Question 158 ..... **Points: [1]**

Explain briefly how to add support for inter-clicks (press one button, then add another button, then release one of the buttons) in the debouncing state machine we have used:

Solution: Add additional events to message an inter-click. Extend the state machine so it either continues or uses different states in the state machine.

Question 159 ..... **Points: [3]**

Explain the principle of 'fast decay' and 'slow decay' motor stopping for a full H-Bridge:

Solution: The fast decay principle is to revert the current from the previous movement. E.g. if the motor is turning forward, to put current into the H-Bridge to in reverse order (turning it backward). This will bring the energy stored in the inductor down fast. So the 'fast' is about how fast the current reaches zero. With 'slow decay' either the lower half or upper half of the transistors are on, allowing the inductive current to flow back. Because this takes longer than with the 'fast' method, this is called 'slow decay'. With 'fast decay', the motor coasts down the speed, while with 'slow decay' the H-Bridge using an active break. Hint: .

Question 160 ..... **Points: [1]**

You are using a 100% duty cycle PWN. Does this mean your motor is at maximum speed?

Solution: No, this depends if the signal is LOW or HIGH active.

Question 161 ..... **Points: [1]**

While driving your H-Bridge with a PWM to drive a DC motor, you hear an audible noise. What could you do to fix the problem?

Solution: Increase the PWM frequency.

Question 162 ..... **Points: [1]**

What is the difference between a full and a half-H Bridge?

Solution: A half-H Bridge has two transistors, while a full H-Bridge has 4 transistors. With a full H-Bridge the direction of a DC motor can be changed.

Question 163 ..... **Points: [1]**

You decide to use a 'common' folder for the INTRO project. Which files do you place into that folder?

Solution: Files which can be used for multiple projects, like files of a library.

Question 164 ..... **Points: [1]**

You decide *not* to use a 'common' folder for the INTRO project. What does this mean for your project?

Solution: A lot of duplicated files, maintenance problem could occur as I need to change files in different places.

Question 165 ..... **Points: [1]**

Name three reasons why a project should be carefully structured with folders?

Solution: easier to read, re-usability, easier to understand.

Question 166 ..... **Points: [1]**

How can you direct the compiler settings to go up one directory in the folder structure in Eclipse?

Solution: Either with '..//' or e.g. with '{PARENT-1 PROJECT_LOC}

Question 167 ..... **Points: [1]**

Why should you use relative paths in your project and not absolute path settings?

Solution: Because with sharing the project the paths might be different on another machine.

Question 168 ..... **Points: [1]**

List a disadvantage of using macros:

Solution: To debug it a preprocessor listing needs to be generated (-E for gcc).

Question 169 ..... **Points: [1]**

What's the difference between using ".." or <..> for includes?

Solution: With double quotes 'user' header files are included, and with <..> library header files. The compiler uses two different search settings for user and library header files.

Question 170 ..... **Points: [1]**

How can you define an index or starting page with doxygen?

Solution: Using the
mainpage command.

Question 171 ..... **1 Point (Bonus)**

Who is Kevin?

Solution: ????

Question 172 ..... **Points: [1]**

Explain the reason why microcontroller pins have names like TSI0_CH11/PTB18-/TPM2_CH0:

Solution: The processor is using muxing: a single pin can have different purposes, like as touch sensing pin, as a input or output port pin or as a timer channel pin.

Question 173 ..... **Points: [1]**

If using pin muxing, what do you have to consider?

Solution: That it is only possible to use the pin in one mode at a given time. And that muxing a pin might have impacts and side effects on other pins, e.g. certain functions are not possible any more.

Question 174 ..... **Points: [2]**

Why is using a function like `CLS1_SendString()` better than using `printf()`?

Solution: printf() is not a save function and can cause a stack overflow, and is subject of security issues. Additionally it needs a lot of code space and stack/RAM.

Question 175 ..... **Points: [1]**

Why needs a string with 5 characters 6 bytes in memory, and not 5?

Solution: Because the string is terminated with zero byte.

Question 176 ..... **Points: [1]**

Inside your timer interrupt service routine you are using

```
static int counter = 0;
```

Now you remove the `static`. What is the effect?

Solution: The variable now gets initialized with zero at every interrupt.

Question 177 ..... **Points: [2]**

You consider to handle the event bits set from the main loop, instead of using a check/clear in several places in your application. Discuss the pros and cons of this approach:

Solution: Pros: events are all handled in a single place and centralized. Complexity is simple. Cons: can be a long switch/if-else-if, event handling depends on frequency of main loop. Cannot handle events independently/concurrently.

Question 178 ..... **Points: [3]**

Can you give reasons why the KL25Z128 bus clock is limited to 24 MHz, while the K22FX512 can run a bus clock of 60 MHz?

Solution: Maximum core clock of KL25Z is 48 MHz, while the K22 can run up to 120 MHz. The K22 core is a Cortex-M4F and it can run at a higher speed than the Cortex-M0+. Additionally, the bus clock can run at maximum half of the speed of the core clock.

Question 179 ..... **Points: [2]**

What are the pros and cons of using an external clock vs. internal clock?

Solution: Pros: more accurate, higher overall speed possible. Cons: more costs, more PCB space needed.

Question 180 ..... **Points: [3]**

Briefly explain the purpose of CPU clock, Bus Clock and System Clock:

 Question 180 continuous on the next page... Reached: _____ from 12 points

Solution: The CPU is clocked by the CPU clock, the bus clock is used for data transfer and to access memory, and the system clock is used to clock the peripherals.