Reexam 1DT106

⚠ Det här är en förhandsvisning av den publicerade versionen av quizet

Startad: 2 dec kl 12.28

Instruktioner för Quiz

Welcome to the re- exam in 1DT106 Programming Embedded Systems. This quiz is available from 08:00 till 13:00 on Wednesday, August 18th 2021. You have until then to answer the questions, and hand in the quiz here on Studium.

Help

In case you have questions regarding the exam, I will be available from 10:00 to 12:00 in the following zoom room:

https://uu-se.zoom.us/j/68643490266 (https://uu-se.zoom.us/j/68643490266)

There is a waiting room for that zoom meeting, and you will be let in one after the other. I will only answer questions regarding the wording or the intent of the questions.

Honour code

This year we have special circumstances, both regarding the course itself, but also regarding the examination. I cannot prevent you from using any material that is available to you both here on Studium, as well as on the internet, and I therefore also do not ask you not to take advantage of that. What I do ask from you is to work on the exam by yourself, and not share any answers with your [illegible]mates who are also taking the exam or haven taken it before. I cannot legally require you to [illegible]re to this honour code, but given the special circumstance, I kindly ask you to be fair towards your colleagues, especially the ones who will write this exam at a later date, and might do so in a more traditional setting again.

Please be aware that I have to report obvious cases of plagiarism to the university, and you might face temporary suspension if these cases are confirmed!

Grading

- 0 - 29 points => U
- 30 - 39 points => 3
- 40 - 49 points => 4

- 50 - 60 points => 5

Fråga 1

3 poäng

During the lecture we discussed multiple constraints we need to keep in mind when developing software for embedded systems. Given the two constraints "weight" and "power consumption", find another constraint, and give examples of how these 3 constraints affect each other/interact with each other.

Redigera Visa Infoga Format Verktyg Tabell

12pt Paragraph

p 0 words

Fråga 2

1 poäng

In the context of testing software, we often talk about coverage criteria regarding our test cases. What do we mean by statement coverage, what do we mean by branch coverage? Write an example for a program in C code (you cannot use the one from the flipped classroom lecture), where statement coverage through test cases is impossible, and explain why.

Redigera Visa Infoga Format Verktyg Tabell

12pt Paragraph

p 0 words

Fråga 3

3 poäng

Quite often when we work with embedded systems, we have to deal with signals transmitted through wires and physical connections. A common problem in those cases is, that whenever these signals change, they cannot do so instantaneously, which often leads to a short period of time, in which the signal is not stable. These problems appear especially when dealing with buttons and switches in the system. What do we mean by the term "debouncing"? Explain two methods of how we can achieve debouncing when dealing with buttons in our system.

Redigera Visa Infoga Format Verktyg Tabell

12pt Paragraph

p 0 words </>

Fråga 4

2 poäng

In the context of realtime operating systems, what do we mean by the term "deferred interrupt handling"? Why is it a good design principle?

Give a short example of a system that might make use of deferred interrupt handling. You can do so textually or with a short C code snippet.

Redigera Visa Infoga Format Verktyg Tabell

12pt Paragraph B I U A T²

p 0 words </>

Fråga 5

2 poäng

Give a short description of the term "embedded system". What makes an embedded system different from a general computing system?

Redigera Visa Infoga Format Verktyg Tabell

12pt Paragraph

p 0 words

Fråga 6

1 poäng

In a typical microcontroller, which two types of memory are present? What are these different types of memory used for?

Fråga 7

4 poäng

In the context of realtime operating systems, what do we mean by the term "mutex"?

Which problems does a mutex solve?

When using a mutex, two common problems might occur:

- Priority inversion
- Deadly embrace

Give a short example (in C code or textually) for both of these problematic cases, and explain what the problem is and how we can solve it. I will not try to compile your C code, so it does not have to be syntactically fully correct. If you want, you can use the Zephyr API we used during the course, otherwise you can use generic API calls like mutex_lock( ) and mutex_unlock( ).

Fråga 8

1 poäng

What is the difference between a "semaphore" and a "mutex"? In wich scenario would you prefer to use a mutex, and in which would you use a semaphore?

Fråga 9

4 poäng

As you have seen during the first flipped classroom lecture, we can use pre and post conditions as a form of contract based programming, where we establish, that if a function fulfils certain pre conditions, we can logically conclude, that the post conditions will follow.

Have a look at this (very naive) implementation of a queue (i.e. ring buffer). You can also think about it as a First-In-First-Out buffer, where the ends are connected.

```c
#define QUEUE_SIZE 10

int queueData[QUEUE_SIZE];
int headPointer = 0;
int tailPointer = 0;

void queue_write(int data)
{
    queueData[headPointer] = data;
    headPointer = (headPointer + 1) % QUEUE_SIZE;
}

int queue_read()
{
    int data = queueData[tailPointer];
    tailPointer = (tailPointer + 1) % QUEUE_SIZE;

    return data;
}
```

As a first step, take either the queue_write or the queue_read function, and use ACSL (ANSI/ISO C Specification Language) to add one pre and one post condition. In a second step, rewrite the function you chose, and use the special assert( ) call to implement both the pre and post condition checking. You have to give the full function code, including the already existing function body.

Hint: The logic of a ring buffer actually requires, that multiple pre conditions are met in order to work properly. It is enough, if you just pick one here. Similarly, you can probably find multiple post conditions for each function. Again it is enough if you just pick one.

Fråga 10

2 poäng

Assume you have a function **foo( )** in your code, which you suspect to be very slow, and therefore a good candidate for optimisation. How can you figure out, if this function really contributes a lot to the runtime of the overall program? List and describe at least 2 different methods.

Once you have verified, that **foo** is indeed slow, name and describe at least 2 common strategies for manually trying to increase the performance of this function!

▶

Fråga 11

4 poäng

In the context of realtime operating systems, (RTOS) what is a reentrant function? What criteria must a function fulfil to be reentrant?

Why are reentrant functions important when designing a system using an RTOS?

Give two examples (in C code) for a reentrant and a non-reentrant function. Your C code does not have to be fully syntactically correct, I will not try and compile your functions.

Fråga 12

2 poäng

Why is reliability a concern when developing embedded systems? What do we mean with by the term safety-critical system? How is reliability different for many types of embedded systems, compared to for example desktop applications? Give some examples!

Fråga 13

4 poäng

When developing embedded software, one often has to make the decision wether to use a realtime operating system (RTOS), or to just write the firmware on top of the vendor specific SDK for the hardware chip used. Compare the two different methods of embedded software development (i.e. "bare-metal" vs RTOS), what advantages and disadvantages do they have. Give two examples, one where using an RTOS is favourable, and one, where not using an RTOS might be advantageous. Outline shortly for each example, why you think that the chosen design is the better fit for the respective problem.

Fråga 14

2 poäng

Many realtime operating systems offer different scheduling algorithms to schedule the tasks/threads defined in your system. Two very common ones are preemptive and non-preemptive priority based scheduling. What is the difference between those two, and why would we sometimes want to favour one over the other and vice versa?

Fråga 15

4 poäng

When talking about inter-task communication in the context of a real-time operating system, which problem can arise by using shared memory (i.e. a global variable or data structure on the heap)? Outline a short example (either textually or with a short snippet of code), that showcases this problem.

Common strategies for solving the issue with shared memory are the following:

- Using cooperative (i.e. non-preemptive) tasks
- Locking interrupts before accessing shared memory

Give a short description of each method, and explain how they solve the aforementioned issue.

(You might be wondering why using a mutex is not mentioned as a method here. There is a separate question about mutexes in this exam, so you don't have to mention them here)

Fråga 16

4 poäng

What do we mean by the term "state-centric state machine"? How can they be implemented (you can write some C/pseudo code or just textually describe the implementation pattern)?

What are common problems with those types of state machines? Give at least one strategy/change we could implement to overcome some of these problems, and mention which/how they solve these problems. If you want you can invent some simple system to show your points.

Fråga 17

2 poäng

One concept, that is very important when using a realtime operating system (RTOS), are threads or tasks (as said during the lecture, for our purposes here these two terms can be seen as synonyms). In your own words, describe what a task/thread is in the context of an RTOS. Give an example of a system (you can be creative) where you would have two independent tasks running on the same microcontroller.

Fråga 18

4 poäng

In the context of a realtime operating system (RTOS), threads/tasks can be in different states. Different RTOSs define different states, but most of them have at least the following:

- Ready
- Running
- Waiting/Blocked
- Terminated

Give a short description of each state, what it is used for, and how we go from and to this state.

Fråga 19

3 poäng

When talking about the efficiency of embedded software, very often we can either pick a solution that is very space efficient, but will increase the execution time, or a solution that favours fast code execution, but might require more memory. Give three examples of such cases, and quickly describe how/why they favour either space or time efficiency.

Fråga 20

3 poäng

In the flipped classroom lecture about testing you saw, that in order to test a function, we usually have to put it into a test harness (a function that "wraps"

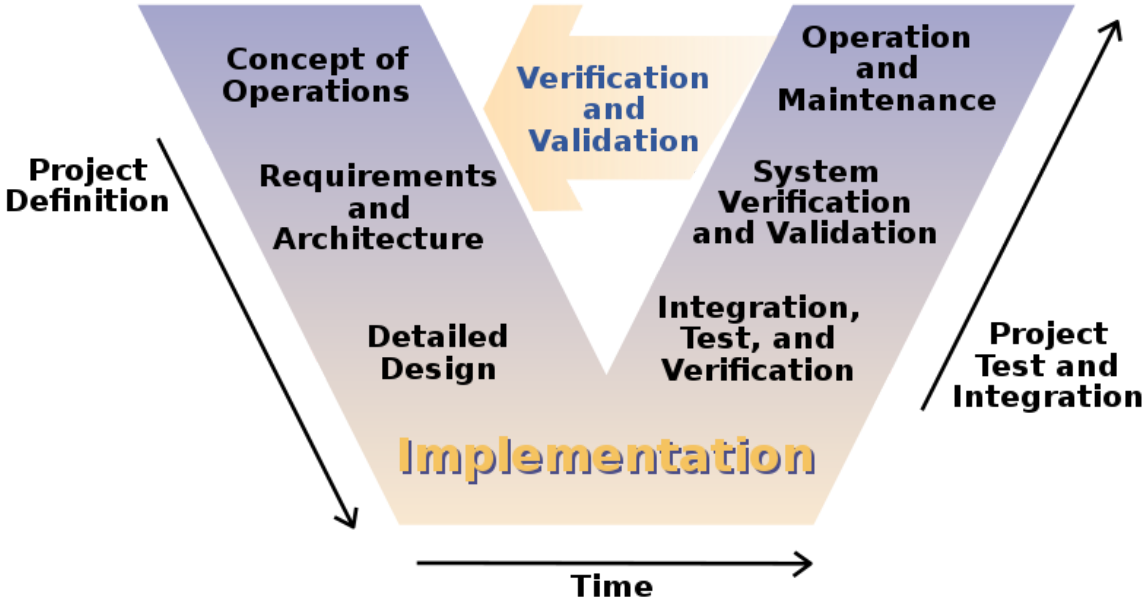
around the function you want to test), which performs any necessary setups, then calls the function with given (probably random) inputs and then usually has an "oracle" to decide, if the function we are testing has returned the correct result. This is quite commonly called "unit testing".

Assume we want to test a function **int min(int * array, int size)**, which takes as inputs a pointer to an array of integers, and the size of the array, and returns the **index** of the **minimum element** in this array. You can assume, that each element in the array is unique (i.e. all elements in the array are different).

Write a test harness for this function, including proper setup (assume you have a function **int random(int minInt, int maxInt)**, which will return a random integer between and including the parameters minInt and maxInt), and then write a proper oracle to figure out, if **min** worked correctly. In case it did, the test harness should return 1, otherwise 0. I will not try and compile your C code, so it does not have to be fully syntactically correct. You do not have to strive for any coverage criteria here, so you can pick values for the random function that make sense for you, but that do not necessarily test out all corner cases. However, the oracle **has to work with arrays of different lengths**!

Fråga 21 **2 poäng**

Below is the well-known V-diagram, which shows the usual phases of an (embedded) software development project. Pick at least 2 phases and describe them in more detail. What happens during those phases? Are there any special tools that are used?



Fråga 22 1 poäng

One C keyword, which is very common when programming embedded systems is "volatile". What does volatile mean, and why do we want to use it?

Fråga 23

2 poäng

What is the difference between the average and the worst case execution time for any given algorithm? Why is this important for the field of embedded systems? How do data-structures affect the run-times of algorithms?

Inte sparad

Lämna in quiz