Given name: Matr.nr.:
First name:

HAMM-LIPPSTADT
UNIVERSITY OF APPLIED SCIENCES

Microcontroller

Written exam
Study program: Electronic Engineering
Duration: 60 Minutes

15th July 2021

Hints

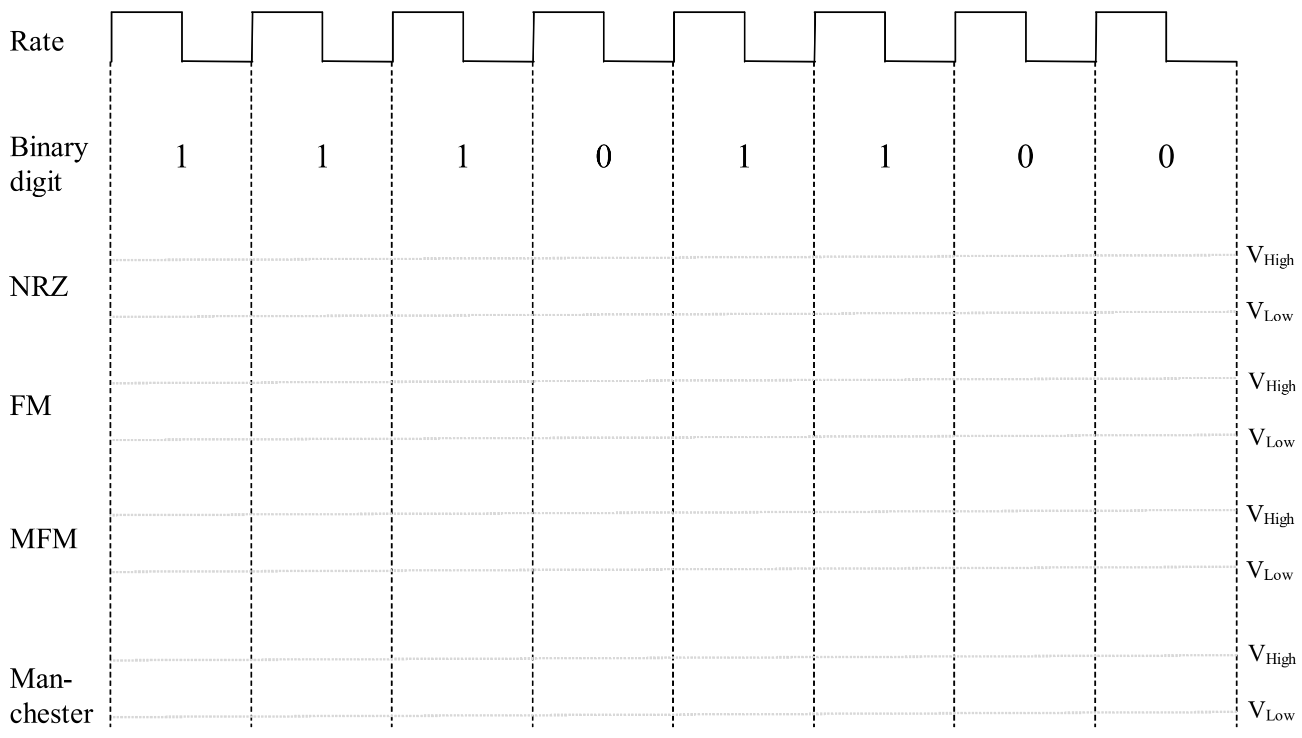
- You are writing an Open Book exam at your home and on your computer.
- You are allowed to use all documents from the lectures.
- You have to program C and C++ - use your development environment.
- Use only the standard C/C++ library functions.
- You have to deliver compilable and runnable code.
- You have to draw UML diagrams - use visual paradigm online.
- You are not allowed to get help from other people.
- By taking the online exam, you agree that you have worked out the solution on your own. Any form of copying of a solution is not allowed. If we find out that solutions are of the same origin, this will be considered as an attempt to cheat (we use plagiarism testing tools), with the corresponding consequences.
- Any form of forwarding the documents/exam is prohibited and will also be considered an attempt to cheat.
- Please note the "health questionnaire" which you have to fill in when handing in the exam.
- The name of the class(es)/file(s) is either given in the assignment e.g. task1.c or is implicit defined by the tasks.
- You have to create the programming results as C source code files or drawings as PDF or JPG files and submit them packed to a ZIP file. Make sure you can create zip files.
- Check where you can find the C-Code files on your computer.
- The submission is done here, via Moodle. Only if Moodle is not operational send the submission file to: pruefungsabgabe@hshl.de.

Good luck!

Task	1	2	3	Sum	Mark
Points	14	26	60	100	

1. Core elements (14 P)

a) Given the following diagram. Specify the transmitted signals based on the given “Rate” and “Binary digit”.



b) Give a schematic overview on the main elements of a microcontroller.

c) Which processors are typically used for microcontrollers?

   a. Give an overview of the characteristics (like: #address bits, cpu frequency, #memory, …)

d) Implement a 2-bit (branch) predictor:

   a. Specifiy it’s state behavior with UML State Machines.

   b. Map the state behavior to C-code. The code must compileable and stored in a C code file named “twoBit.c”.

   c. Map the state behavior to a circuit.

2. UML State Machines and Programming (26 P)

Given the following description of an ATM: “The ATM is initially turned off. After the power is turned on, the ATM switches to the Idle state. In this state the ATM waits for customer interaction. The ATM state changes from Idle to “Serving Customer” when the customer inserts a banking card in the ATM's card reader. On entering the “Serving Customer” state, the entry action “readCard” is performed. Note, that transition from “Serving Customer” state back to the Idle state could be triggered by the cancel event as the customer could cancel transactions at any time. “Serving Customer” state is a composite (hierarchic) state with sequential substates “Customer Authentication” (the authentication via pins should not be modeled) and “Transaction”. “Serving Customer” state has a triggerless transition back to the Idle state after the transaction is finished. The state also has an exit action “ejectCard” which releases customer's card on leaving the “Serving Customer” state, no matter what caused the transition out of the state.”

a) *Identify the states, events (trigger and send events), … of the ATM and model its state behavior with a UML State Machine! (10 P)*

Use e.g. Visual Paradigm Online and export the diagrams to a pdf file. The file names are as follows:

- *state diagram: atmStateDia.pdf*

b) *Map the UML State Machine in a structured way to C code. (10 P). The C code has to be compileable and stored in a file name “atmState.(the appropriate C code ending)”* *c) Given is the following implementation of a "Transaction-"function (6 P).*

Implementation:

```
int balance = 0;

void deposit ( int sum ) {

  int currbalance=balance ; /* read balance */

  ...
  currbalance+=sum ;

  balance=currbalance ; /* write balance */

}
void withdraw ( int sum ) {

  int currbalance=balance ; /* read balance */

  if ( currbalance >0){

    currbalance -= sum ;

  }

 balance=currbalance ; /* write balance */

}
```

And a concrete set of parallel transactions:

ATM1: deposit (200);
ATM2: withdraw (100);
ATM3: deposit (300);

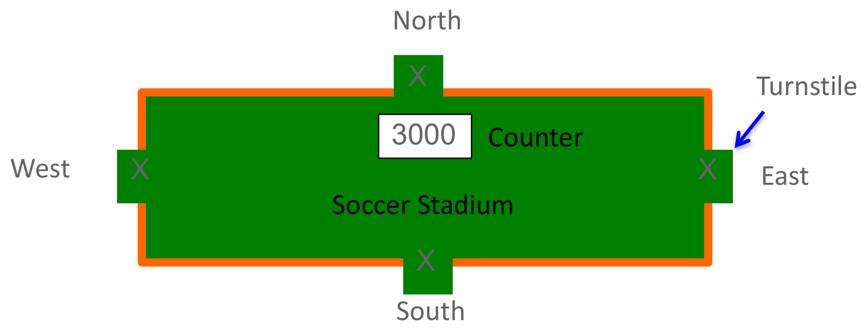
i) Identify critical behavior of the implementation and show two possible different values of the "balance" variable after the three parallel transactions are executed.

ii) Update the implementation by using the concept of mutex. You can use the following definitions and functions:
   a. pthread_mutex_t mutexbalance=PTHREAD_MUTEX_INITIALIZER;
   b. pthread_mutex_lock(…)
   c. pthread_mutex_unlock(…)

and store the code in a file named "transaction.c".

3. Threads, Testing, OpenMP, and Network (60 P)

We consider a soccer stadium that can be entered through four entrances (North, South, West, and East). Visitors/fans can leave the stadium. At each entrance, visitors must pass a turnstile. Each turnstile has a counting device that counts up a common counter.



a) Develop a C program for the described soccer stadium including Pthreads (20 P)!

The code must be compileable and stored in one or many c code files starting with “soccerStadium”.

b) Specify test cases for validation tests and defect tests. Test your system! (15 P). The code must be compileable and stored in one or many code files starting with “soccerStadiumTest”.

c) Implement the soccer stadium example by using OpenMP (10 P). The code must compileable and stored in one or many code files starting with “soccerStadiumOpenMP”.

d) Consider the updated example: all turnstiles and the counter are running on different nodes
   a. Implement a network-based solution by using sockets. The turnstiles send a "new fan" message to the "counter". Only when the counter allows access (as long as less than 3000 fans visit the stadium), the fans are allowed to enter the stadium. (15 P). The code must compileable and stored in one ore many C code files starting with “soccerStadiumNetwork”.