Försättsblad till tentamen vid Linköpings Universitet

Cover page for exam at Linköping University

Datum för tentamen Date of exam	**2012-08-13**
Sal Room	**SU12 - 18**
Tid Time	**08:00 - 12:00**
Kurskod Course code	**TDDI11**
Provkod LADOK code	**DAT1**
Kursnamn/benämning Course name	**Programmering av inbyggda system** Embedded software
Institution Department	**IDA**
Antal uppgifter som ingår i tentamen Number of assignments	**8 uppgifter om totalt 40 poäng** 8 assignments for a total of 40 points
Antal sidor på tentamen (inkl. försättsbladet) Number of pages including cover	**5**
Jour/Kursansvarig Responsible/Examiner	**Klas Arvidsson** klas.arvidsson@liu.se
Telefon under skrivtid Phone during exam	**013 - 28 24 90** 013 - 28 24 90
Besöker salen ca kl. Time of exam visit	**På begäran. Använd kommunikationsklienten först.** On request. Use the communication client if you have questions.
Kursadministratör Course administrator	**Madeleine Häger Dahlqvist** 013 - 28 23 60, madeleine.hager.dahlqvist@liu.se
Tillåtna hjälpmedel Allowed aids	**Ordlista och enkel miniräknare (+,-,*,/)** Dictionary and simple pocket calculator (+,-,*,/)
Övrigt Other information	*Precise, explained and clearly motivated assumptions, statements and reasoning raise the impression and are required for highest score.* ***Solve at most one assignment per sheet.*** ***If in doubt, state clearly your interpretation of the question, your assumptions, and answer according to that.*** Preliminary graded: U < 50% < 3 < 67% < 4 < 84% < 5 Grades may be raised or lowered based on overall impression. Results available after (within) 10 working days.
Typ av papper Paper to use	**Rutigt, linjerat eller blankt** No preference
Antal anmälda Number of exams	**26**

General theory part (19p)

Right-click the desktop to start a *terminal* and the *communication client*.

Copy *"solutions.txt"* from the read-only *given_files* folder and open your read-write copy:

```
cd ; cp given_files/solutions.txt solutions.txt
emacs solutions.txt &
```

Write your answers in the file "solutions.txt". Please follow the given format and save often. Solve all assignments you can answer fast first. Do not spend more than 1 hour on this part. Get started on the practical part early.

1. Definition of embedded system (3p)

a) Describe four criteria that are said to distinguish an embedded system. We say it's an embedded system when some or all of them is true.

b) Give an example of an embedded application where at least two of the distinguishing criteria are found. Motivate.

2. This and that (5p)

a) Embedded development sets many non-functional requirements on the system. Two such are NRE-cost and time-to-market. What is the meaning of those two, and why are they important? (2p)

b) A semaphore have two important operations. Different implementations use various names on the operations such as *wait* / *signal*, *P* / *V*, *down* / *up* or as in lab 5: *pend* / *post*. Regardless of the name the functionality is mostly identical. What does each of the two functions do (give full specification in all situations)? (2p)

c) Explain the meaning of a *memory map*. (1p)

3. Assembly (3p)

A certain CPU have a memory bus with 32-bit addresses and an IO-bus with 16-bit ports. Assume the registers A, B, C, D and that the instruction format: INSTRUCTION *Operand1 Operand2*. The five available instructions are:

```
MOV To   From          |     IO  Dest Source
AND Acc  Mask          |     OR  Acc  Mask
JMP To   Condition     |
```

All operands can be one of the registers A, B, C, D. *To* and *From* can also be a hexadecimal address. *Dest* and *Source* can also be a hexadecimal port number. *Mask* can be a hexadecimal number. *Acc* must be a register and serves as both source and destination of the instruction (*Acc* = *Acc* & *Mask*). *Condition* is a register and the jump is taken if the value is non-zero.

A sensor have a status register connected to port 0x1234 and a data register connected to port 0x5678. Bit position 3 of the status register indicate when new data can be read from the data register. Use polling to read one value from the sensor and store in register A. Write code using the five given assembly instructions only. (3p)

4. Optimization (4p)

At some point in the development process you may need to optimize code.

a) At which point in the development process do you start to worry about optimizations? (1p)

b) A data table lookup is said to be able to improve performance. Explain the meaning of a table lookup and how it can improve the calculation of a slow expression such as *5*sin(x)*cos(x)/tan(2*x/3)*, where x is a floating point angle. Describe one drawback of the method. (3p)

5. Development methods (4p)

a) Name two development processes and describe a major benefit of one over the other. (2p)

b) Describe three important goals of a development process. (2p)

Practical implementation (computer part) (21p)

Right-click the desktop to start *emacs*, a *terminal* and the *communication client*.

To get started on the practical part you need to copy some files:

```
cd; cp given_files/Makefile Makefile
cd; cp given_files/*.c /home/student_tilde/
```

To compile and run the "Safe" system you do:

```
cd; gmake main ; ./main
```

To compile and run the "ultimate-number" assignment you do:

```
cd; gmake ultimate-number ; ./ultimate-number
```

To run the ultimate-number reference solution you do:

```
cd; ./given_files/ultimate-number-solution
```

In case of doubts, use the communication client!

("If I do like ... this problem ... will appear ... do I have to solve that too?")

In case of system trouble or trouble getting the given files to work, raise your hand!

If you are unable to solve one assignment, explain that and implement a simpler replacement for reduced points. Make sure to comment what your code is intended to do in order to get credits despite flaws.

6. Bit manipulation (6p)

Bertram invents a new (not so good) way of storing numbers. Each number uses 64-bits. The most significant 5 bits determine the number of bits in the following exponent, and the remaining bits belong to the mantissa. Complete the C-functions in "ultimate-number.c" that get the number of bits in the exponent (~1p), read the value of the exponent (~2p), and can update the value of the exponent (~3p). Bits other than the exponent may not be modified in any case.

7. State Machine drawing (7p)

a) Draw the state machine implemented in function "Safe" in `main.c` (3p)

b) Is it Mealy or Moore? Motivate your answer. (1p)

c) To show you understand the machine, explain how to open the safe. (1p)

d) The machine is called directly from the keypad interrupt handler. This is often discouraged. Explain a problem that may occur if the state machine is also called at other places in the code. (2p)

8. Hotel Safe (8p)

Hotel rooms are regarded as a bad place to store valuables. Many hotels solve this by providing a safe for your smaller size valuables. Such safe have a door and a keypad. It is operated as follows: You open the door and put your valuables in. Then you close the door. Type a three-digit combination on the keypad and finish with #. The safe will now lock. To unlock the safe again you must press the same three-digit combination. The safe unlocks immediately on the third correct digit (no need to press #) and you can open the door. In some cases your valuables are actually still there. If you do not press anything for 10 seconds you must start over, both when locking and unlocking. If you press more than three digits before you press the locking key (#) the three last digits are used as unlocking code.

You are to program a particular safe to behave like this. The system have one output signal `B` to control the door lock, and one input signal `A` to provide key events. The keypad is connected to a interrupt pin. Any keypress cause the interrupt handler to execute. Setting `B` to 1 cause the safe motor to lock the door, setting it to 2 will open the lock again. The engine will automatically stop when the lock is fully opened or closed, so you do not have to reset B to 0 again. It is however recommended to reset B to 0 anyway since future models will have a different engine that must be turned off by the software.

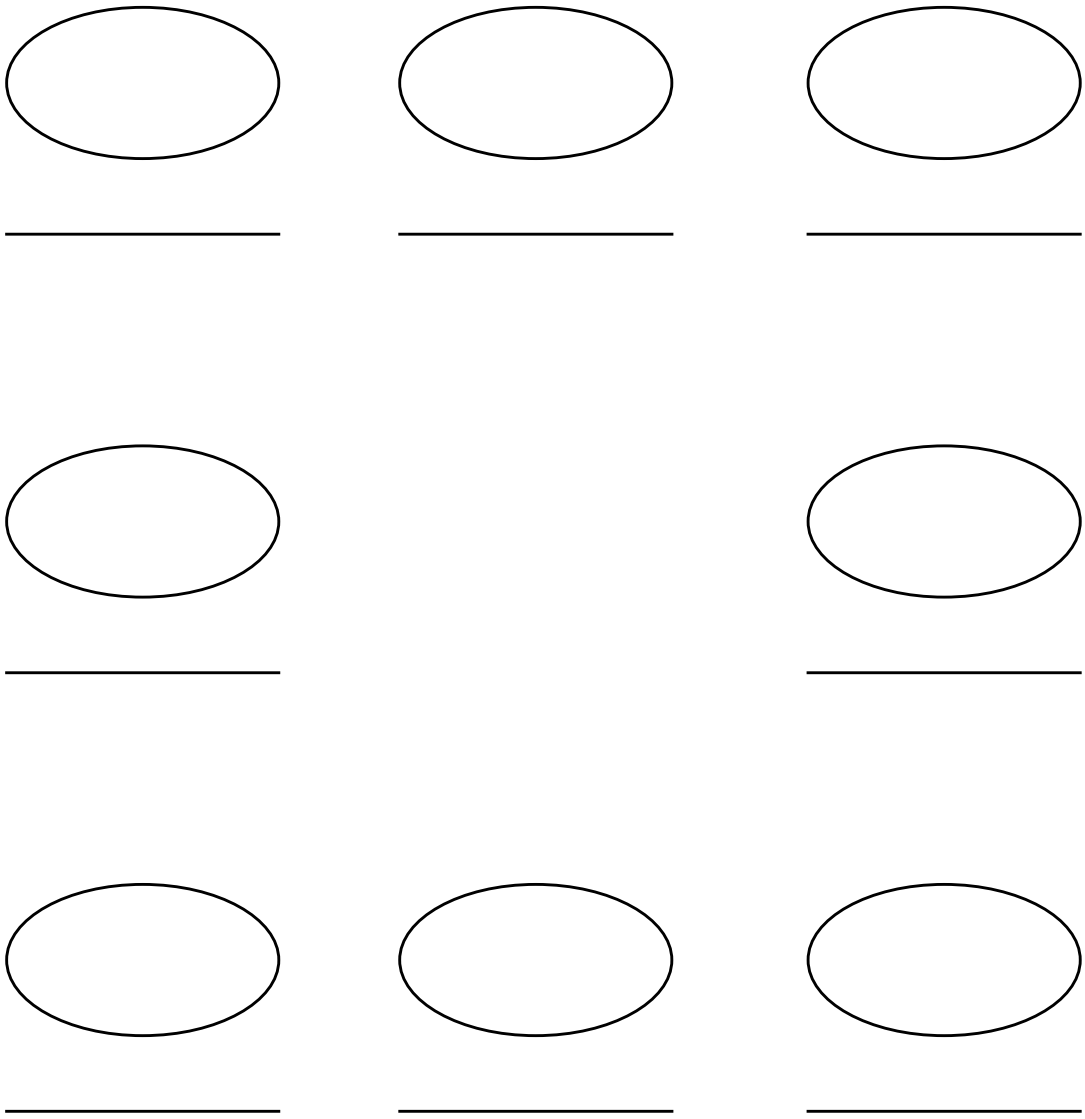
Design and implement a state-machine application to control the safe. You are ***not*** required to draw the state-machine diagram. Please think about question 7d in your implementation. You ***are*** required to explain how your solution works, and motivate why you solved it that way.

(Most real-world hotel safes actually use four-digit pin-codes, but the fourth digit do not add any value to this assignment.)

When you are completely done with all assignments you right-click the desktop and select "Finish exam". Click OK when asked to confirm, your files are saved even if the session is not. You get to a "relogin" page and should select "Quit" and confirm that. When you are back at the normal blue login screen you can leave.

State machine drawing template sheet

Write state names in circle. Write action(s) on line(s) below circle. Draw transitions and state condition for each transition as needed. You do not have to use all circles. You may add additional states. You may ask for a new template.



Inputs to this state machine:_____

Outputs to this state machine:_____

Period this state machine run at:_____