Försättsblad till skriftlig tentamen vid Linköpings Universitet

Cover page for written exam at Linköping University

Datum för tentamen Date of exam	**2011-05-26**
Sal Room	**KÅRA**
Tid Time	**08:00-12:00**
Kurskod Course code	**TDDI11**
Provkod LADOK code	**TEN1**
Kursnamn/benämning Course name	**Programmering av inbyggda system** Embedded software
Institution Department	**IDA**
Antal uppgifter som ingår i tentamen Number of assignments	**9** 9 assignments for a total of 40 points
Antal sidor på tentamen (inkl. försättsbladet) Number of pages including cover	**4**
Jour/Kursansvarig Responsible/Examiner	**Klas Arvidsson** klas.arvidsson@liu.se
Telefon under skrivtid Phone during exam	**013 - 28 21 46** 013 - 28 21 46
Besöker salen ca kl. Time of exam visit	**Ca 1h efter tentastart** About 1h after exam start
Kursadministratör Course administrator	**Gunilla Mellheden** 013 - 28 22 97, 070 - 597 90 44, gunilla.mellheden@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 within 10 working days.
Typ av papper Paper to use	**Rutigt, linjerat eller blankt** No preference
Antal anmälda Number of exams	**37 (+3 som glömde anmäla i tid)**

1. (5p)

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. (5p)

The spiral model is a development methodology that can be used. *Describe* it and *compare advantages* or *disadvantages* to the waterfall model, and to embedded systems development in particular.

3. (2p)

Embedded development sets many non-functional requirements on the system, such as unit cost or battery life. List four other *common* and *important* such requirements.

4. (4p)

Code can be optimized for several reasons, with several goals.

a) *Mention* three different goals of optimization. (1p)

b) *Explain* at which step during development you may start optimizing code. What are the steps before? What is tried first? (3p)

5. (4p)

An imaginary 32-bit system using interrupt address table located at address 0x1000 receives interrupt signal 8 when executing some code summing an array.

a) The programmer of the system need a function to read the serial port to be executed in response to the interrupt. How can he arrange for the function to be called correctly? (1p)

b) *Explain* how the system (CPU) handle the interrupt from start to end. (3p)

6. (3p)

Describe precisely a situation when two executing threads require mutual exclusion by means of a binary semaphore.

7. (2p)

A bicycle have a wheel with a circumference of 2 m. On the outer edge is a small magnet that will rotate with the wheel. During the time the magnet passes a 10 cm sensor it will induce a voltage that will be detected as a signal by an embedded system. The signal is also high due to noise (sometimes). All noise last less than 2 ms. The system polls this signal at regular intervals to count the number of wheel revolutions. The bike travels at speeds between 3 and 10 m/s.

a) How often shall the signal be polled as an absolute minimum? (1p)

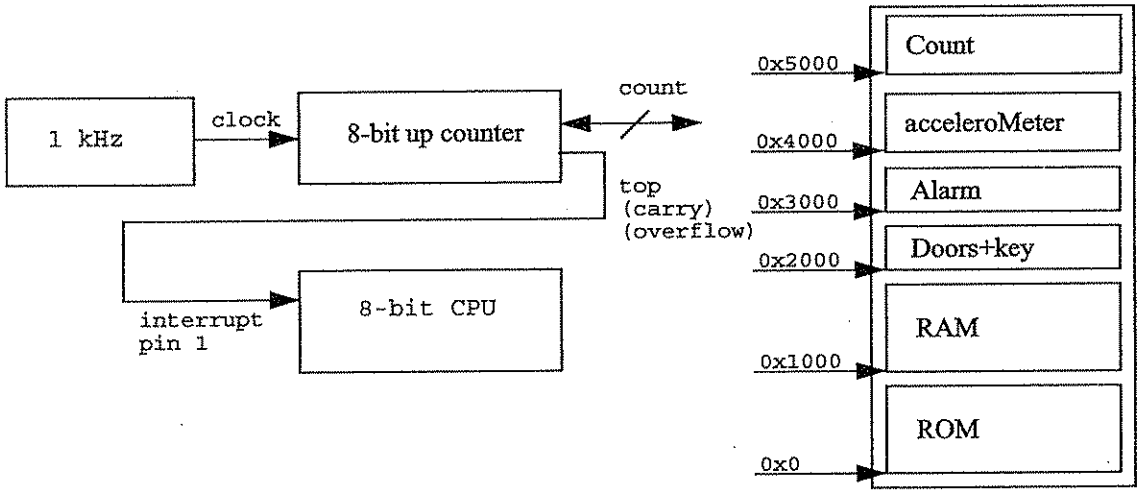
b) Does polling more often give any advantage? (1p)

c) **This question is an option to question 8c). Answer EITHER 7c)** (this question) **OR 8c).** Design a state machine that run every 3 ms and reliably counts each revolution once. Avoid counting noise. Use a 5-state Moore (not Mealy) machine, no variables except "revolution-count", and only the revolution sensor R as input. (5p)

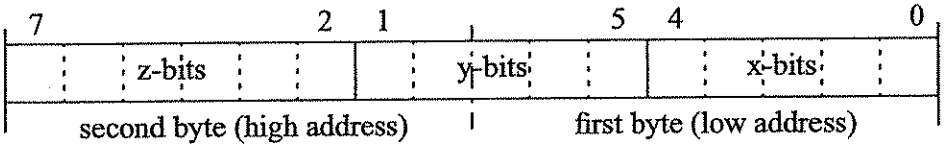
8. (12p)

As we all know from good (?) old movies many car alarms trigger by simply bumping into the car in question. Testimonies also claim that the only way to shut the alarm off again is to enter the car, lock all doors, and then put the key in the ignition.

A layout of components in such alarm-system, and a memory map is shown below.



1) "Count" represent the value on the counter (timer). Overflow trigger an interrupt.
2) "acceleroMeter" consists of two bytes. The low 5 bits of the first byte is a measure of motion in X-direction. Motion in Y-direction is measured in next five bits. The low three Y-bits is the high 3 bits of the first byte, and the high two Y-bits is the low 2 bits in the second byte. The high 6 bits of the second byte measure the motion in Z-direction:



3) "Alarm" is set to 0x01 to turn a red "indicator" diode on, 0x02 to turn sound alarm on, and 0x03 turn both on. 0x00 turn everything off.
4) "Doors+key" allow to check the door status. When all doors are closed and locked this is 0x00. If the correct key is inserted bit 6 it high. If any door is open bit 5 is high. If any of bits 0-4 is high, then the corresponding door is unlocked.

Use the uppercase letter in above names to refer to corresponding address in a short way.

a) Write a C-code interrupt handler for the counter (timer) so that it runs *exactly* every 200ms. (What do you put in "count" to get interrupts every 200ms?) (1p)

b) Implement functions that reads each of the X,Y,Z-values from the accelerometer. (How do you use bit-manipulation (&,|,<<,>>,~,^) in C?) (3p)

c) Design a state-machine (executed every 200ms) for the alarm (*no code - just graph*) (5p):
The alarm-system is *inactive* when any door is unlocked. It is the only time the diode is off.
The alarm-system is *active* once all doors are closed and locked.
When the system is active the sound-alarm is triggered if any X,Y,Z-motion is detected for five (5) consecutive times during 1 second.
The alarm is tuned off after all doors are closed and locked and the key set in the ignition.
Write a macro to simplify any long conditions, so the states and transitions look simple.

d) Write `main` to initialize the system and connect assignment a) and c) to work together. Assume that the function `alarmSM` executes the machine in c) and sets outputs correct. (3p)

9. (3p)

An 8-bit processor have two registers for arithmetic operations, `areg1` and `areg2`. It also have the following assembly operations (the first operand is the destination):

```
mov [ADR] regX  ;; Move from register to memory
mov regX  [ADR] ;; Move from memory to register
add regX  regY  ;; Addition, regX = regX + regY
adc regX  regY  ;; Addition, regX = regX + regY + carry
;; Example
DST EQU  0xA3F8   ;; Constant for destination base address
mov areg1 [DST+2] ;; Move third byte of DST to areg1
```

Base addresses A, B and C each store one 24-bit integer. Show assembly code to perform the *24-bit addition* C = A + B. Assume that `areg1` and `areg2` are free, and that no other registers exist.