Försättsblad till skriftlig tentamen vid Linköpings Universitet

Cover page for written exam at Linköping University

Datum för tentamen Date of exam	**2010-01-08**
Sal Room	**TER2**
Tid Time	**14:00-18:00**
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	**7** 7 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** klaar@ida.liu.se
Telefon under skrivtid Phone during exam	**013 - 28 21 46**
Besöker salen ca kl. Time of exam visit	**Start + 1h**
Kuradministratör Course administrator	**Gunilla Mellheden** 013 - 28 22 97, gunme@ida.liu.se
Tillåtna hjälpmedel Allowed aids	**Ordlista och enkel miniräknare (+,-,*,/)** Dictionary and simple pocket calculator (+,-,*,/)
Övrigt Other information	*Preliminary graded: U < 50% < 3 < 67% < 4 < 84% < 5* *Grades may be raised or lowered based on overall impression.* ***Precise, explained and clearly motivated assumptions, statements and reasoning raise the impression and may triple the points. Solve at most one assignment per sheet.*** *Results available within 10 working days.* *Read all assignments and estimate how many points you can get before you start.*
Typ av papper Paper to use	**Rutigt, linjerat eller blankt** No preference
Antal anmälda Number of exams	**6**

1. (8p)

a) *Define an embedded system.*
What makes a system embedded? Explain each term you use. (4p)

b) What are important considerations from hardware point of view? (2p)

c) What are important considerations from software point of view? (2p)

2. (3p)

Describe the organization of a foreground-background system. Pictures is appreciated.

3. (3p)

Show how to set (to 1) and also how to clear (to 0) the N:th bit of a 32-bit integer without affecting other bits.

4. (6p)

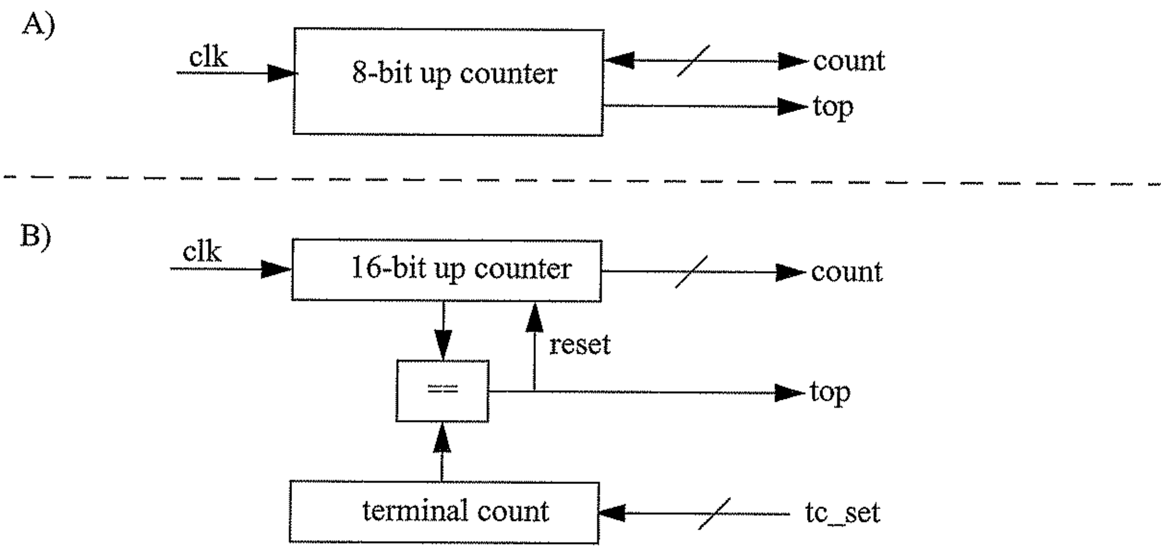
Bit 7 FIFO Error	Bit 6 Transmitter Empty	Bit 5 **T**ransmitter **H**olding **R**egister **E**mpty	Bit 4 Break Detect	Bit 3 Framing Error	Bit 2 Parity Error	Bit 1 Overrun Error	Bit 0 **R**eceive **B**uffer **F**ull

The UART implementing a serial port are controlled using 8 registers. For the purpose of this assignment the constant variable BASE contain base address of the UART control registers. For this assignment the “Line Status Register“ (LSR) at address BASE+5 and “Transmitter Holding Register“ (THR) at (write only) address BASE+0 is important. The meaning of each bit in LSR is outlined in the figure above. THRE indicate that THR is ready to be written. Implement a function capable of sending one null-terminated C-string using polling:

```
void send_string(const char* str);
```

You may assume memory mapped I/O or port based as you like. You may use C or assembler as you like. If you assume existence of any basic functions or instructions, explain what each do.

5. (12p)



The figure above pictures two counters. In A) the *count* register is memory mapped to a read-write address. In B) the *count* register is available on a read-only I/O port, and *tc_set* is available on a write-only port. In both cases you may connect the *top* signal to an other counter or CPU interrupt pin or I/O address/port as you like. The exact port numbers/addresses can be configured by you and is not important for the assignment, it is enough to state the assumptions you make. The *clk* signal can be connected to a fix 1 MHz clock or to the *top* signal. In all questions you want to get a interrupt signal every 1 ms.

a) Using two counters as outlined in A), explain in detail how to set up the system and software to achieve the desired interrupt every 1 ms. Write code to initiate and maintain the counter. Explain any instructions, symbols, abbreviations or assumptions you use. (6p)

b) Using one counter as outlined in B), explain in detail how to set up the system and software to achieve the desired interrupt every 1 ms. Write code to initiate and maintain the counter. Explain any instructions, symbols, abbreviations or assumptions you use. (4p)

c) In B) it is desired to be able to retrieve the currently set *(write-only)* terminal count at all times. Explain how you can achieve this in your software. (2p)

6. (4p)

A certain system will be designed to keep a tank of liquid full and at a certain temperature. The program to control the system have two inputs TEMP_HIGH and LEVEL_LOW. Two output signals should be controlled, OPEN_VALVE to fill more liquid and COOLING_ON to lower the temperature. Design a Finite State Machine to keep the tank filled and at proper temperature. All assumptions must be explained, all possible transitions must be represented and all outputs must be set.

7. (4p)

The code below is given. It read what seems to be temperatures and sound pressure levels from some source, and then calculates some kind of weighted average. Let us assume the exact purpose of the code is unimportant for the assignment, and let us further assume that the code is too slow. The important point now is that the given code is possible to optimize in many ways.

Suggest possible optimizations that improves performance. For full points you need to come up with the best optimization, or a good combination of optimizations. Explain how your suggested optimization(s) increase performance, and explain any drawbacks coming from this optimization(s). The code runs on a 16-bit CPU.
The optimized code is required to be easily to extend in order to handle more types (cases in the switch) even if you remove the switch. Your solution does not have to write code for the optimized version.

```
void hanlde_temp(&avg, temp)
{
  *avg = (*avg + temp) / 2;
}

void hanlde_spl(&avg, spl)
{
  *avg = (*avg * 3 + spl * 2) / 5;
}

int main()
{
  // Weighted averages. Precision of 1 decimal required.
  double avg_temp; // Only values in range 0.0 to 100.0
  double avg_spl;  // Only values in range -50.0 20.0

  while (true)
  {
    int type;
    double data;

    read_data(&type, &read);

    switch(type)
    {
    case 0:
      handle_temp(&avg_temp, read);
      break;
    case 1:
      handle_spl(&avg_spl, read);
      break;
    }
  }
}
```