Departament de Disseny i Programació de Sistemes Electrònics
UNIVERSITAT POLITÈCNICA DE CATALUNYA

Embedded Systems
Final exam. June 6, 2014

Duration: 4 hours. Last revision day: June 27

1 Warming up questions

1. Define an embedded system.
2. Situate the course project DTMF in the previous definition of an embedded system.
3. *What does the keyword VOLATILE mean? Give three different examples of its use.* This is one of the questions of a document entitled *Best Questions for Embedded Programmers.* Next you have an explanation given by Nigel Jones, an interviewer who tells us what should be answered and what not.

   A volatile variable is one that can change unexpectedly. Consequently, the compiler can make no assumptions about the value of the variable. In particular, the optimizer must be careful to reload the variable every time it is used instead of holding a copy in a register. Examples of volatile variables are:

   a) Hardware registers in peripherals (e.g., status registers)

   b) Non-stack variables referenced within an interrupt service routine.

   c) Variables shared by multiple tasks in a multi-threaded application.

   If a candidate does not know the answer to this question, they aren't hired. I consider this the most fundamental question that distinguishes between a 'C programmer' and an 'embedded systems programmer'. Embedded folks deal with hardware, interrupts, RTOSes, and the like. All of these require volatile variables. Failure to understand the concept of volatile will lead to disaster. On the (dubious) assumption that the interviewee gets this question correct, I like to probe a little deeper, to see if they really understand the full significance of volatile. In particular, I'll ask them the following:

   a) *Can a parameter be both const and volatile? Explain your answer.*

   b) Can a pointer be volatile? Explain your answer.

   c) What is wrong with the following function?

   ```
   int square(volatile int *ptr)
     {return *ptr * *ptr;}
   ```

   The answers are as follows:

   a) Yes. An example is a read only status register. It is volatile because it can change unexpectedly. It is const because the program should not attempt to modify it.

   b) Yes. Although this is not very common. An example is when an interrupt service routine modifies a pointer to a buffer.

   c) This one is wicked. The intent of the code is to return the square...

   In short, what does the VOLATILE keyword mean?

   Is it useful when your code is used just for simulation (code size and code speed aren't important)?

2 Approximating a real number with a finite number of bits

1. The coefficient that appears in the Goertzel algorithm to detect the power of a signal at 697 Hz, using a sampling frequency of 8 kHz, has the value $a = 2 * \cos(2 * \pi * 697/8000) \approx 1.7077$. During the course we have scaled a by a factor of 256 and defined the result as a 16-bit signed integer. Scaling this way we have a smaller error: $a_{scaled} = round(a * 256) = round(437.18) = 437$. So, the error after rounding is less than 0.04%.
   a) Why do we scale by a power of two (i.e 256, 512, 1024...)? Are there powers of two betters than others?
   b) Could we obtain a lower error scaling by 512? and by 1024?
   c) By which number should we scale to get an error lower than 0.01%?
   d) Which is the maximum scale factor we could use?
2. Now consider that a is coded using a fixed point coding, with a total of b bits and q fractional bits (signed fractional coding).
   a) Choose the better number of q fractional bits to code a with a total of $b = 4$ bits. How is it coded?
   b) Code $-a$ with b and q used in the previous question.
   c) Repeat the two previous questions with a total of $b = 10$ bits.
3. Are the two previous questions related? Hint: is the scale factor related with q?

3 Analog to digital converter

During the project course we have taken samples of a signal that contains a DTMF signaling. When working with Arduino we have used the 10-bit ADC of the AVR microcontroller with a range that goes from ground to an analog reference that can be selected between 1.1 V, 5 V and an external voltage (for example 3.3 V available on the same board). When working with DE0-Nano we have used the 12-bit ADC, external to the FPGA, with a range that goes from ground to 3.3 V.

1. Consider two analog voltages at the input of the ADC: 2.5 V and 1.65 V. Compute the digital value (in decimal) at the output of the ADC in the following conditions:
   a) An Arduino working with an analog reference of 5 V and a digital value coded with the 8 more significant bits. Repeat for all 10 bits.
   b) An Arduino working with an analog reference of 3.3 V. a digital value coded with the 8 more significant bits. Repeat for all 10 bits.
   c) A DE0-Nano with a digital value coded with the 8 more significant bits. Repeat for all 12 bits.
2. Consider a DTMF signal that moves between −1 V and 1 V.
   a) Draw a sketch of the circuit used to add an appropriate DC voltage to this signal *prior* to its connection to the ADC input. Give approximate values to the components.
   b) If working with an Arduino, which analog reference will you select?

4 Raspberry Pi

1. Samples of 8 bit, sampled at 8 kHz, are send from an Arduino to the Raspberry Pi (RasPi) through a serial connection (USART to USB bridge) using 1-start-bit and 2-stop-bit. What transmission rate should you use in your connection?

2. Repeat the previous question considering 12-bit samples.

3. The following code is used to measure the maximum, mean and minimum time used by the RasPi to read L bytes. The results for $L = 1$ are 700 us, 350 us and 300 us respectively.

```
# -*- coding: utf-8 -*-
# May 13, 2014 by jb

import time
import numpy
import serial

# parameters
N=100
L=1

# open, wait and clean serial port
ser = serial.Serial('/dev/ttyACM0', 115200, timeout=1)
time.sleep(2)
ser.flushInput()

n=0
Dt=numpy.zeros((N,1))
while n<N:
        # to ensure buffer has enough bytes
        a=0
        while a<L:
                a=ser.inWaiting()
        t0=time.time()
        x=ser.read(L)
        t1=time.time()
        ser.flushInput()
        ddt=t1-t0
        Dt[n,0]=ddt
        n=n+1
ser.close()
print "\r\nmax : "+str(max(Dt))
print "mean : "+str(numpy.mean(Dt))
print "min : "+str(min(Dt))+"\r\n"
print str(L)
```

   a) How many time, *reading time*, is needed to read a DTMF-window made up of 205 samples, one at a time, sampled at 8 kHz ?

   b) What percentage of the signal is lost?

4. Executing the previous code with $L = 205$, i.e. reading blocks of 205 samples at a time, gives the following results: 650 us, 400 us and 330 us.

   a) Compute again the *reading time* that is needed to read a DTMF-window made up of 205 samples, 205 at a time. Are we losing samples?

b) Now consider the *decision time* that is needed to make computations with these 205 samples, once they have been read, in order to decide the DTMF-tone received. How many time is left to make these computations before a new DTMF-window is read?

5. In the following consider that the *decision time* is 15 ms.
   a) What happens if occasionally the operative system of the RasPi makes us wait 100 ms to read a DTMF-window? Are we losing samples if the memory buffer is empty? And if the memory buffer is at 90% of its capacity? Remember that the memory buffer has a capacity of 4 kB.
   b) Compute the maximum periodicity at which this 100 ms-wait can appear without losing samples.
6. During the course project we have found that the total execution (of a DTMF-window made up of 205 samples sampled at 8 kHz) of some codes written in Python is approximately 26 ms. As it lasts more than a DTMF-window, after how many time the 4 kB memory buffer is full? Could a code written in C solve this problem?

5 FPGA and VHDL

1. When compiling this code

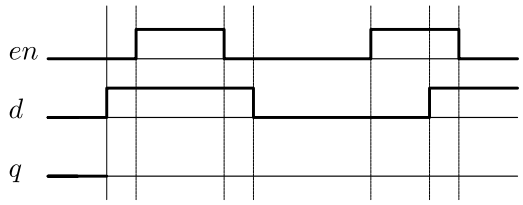
```
process(b)
  begin
    case b is
      when '1' => test<= '1';
      when others => null;
    end case;
end process;
```

we get this warning: "*inferring latch(es) for signal or variable 'test' which holds its previous value...*". Why?

2. Will we obtain the same warning when compiling this other code?

```
process(clk)
  begin
    if rising_edge(clk) then
      case b is
         when '1' => test<= '1';
         when others => null;
      end case;
    end if;
end process;
```

3. Complete the digital waveform depicted bellow for each of the following processes.



```
process(d,en)
  begin
    if en='1' then
      q<=d;
    end if;
end process;
```

```
process(d)
  begin
    if en='1' then
      q<=d;
    end if;
end process;
```

```
process(en)
  begin
    if en='1' then
      q<=d;
    end if;
end process;
```

```
process(en)
  begin
      q<=d;
end process;
```

4. A possible description of a D Flip-flop is:

```
process(clk)
  begin
    if rising_edge(clk) then
      if reset='1' then q<='0';
      elsif set='1' then q<='1';
      else q<=d;
      end if;
    end if;
end process;
```

   a) Are the set and reset signals synchronous or asynchronous?

   b) What happens if the set and reset signals are high activated at the same time?

   c) If they are synchronous/asynchronous write the code to make them asynchronous/synchronous.

5. In the lab we have reduced the number of multipliers used to implement the Goertzel algorithm. Comment on when and how this is possible.

6 Power consumption

Consider that the consumption of a RasPi with no peripherals (just a LAN connection) is 2 W, and the consumption of an ATmega328P AVR microcontroller is the one given in the table bellow. Both are powered by a power supply system made by 2 NiMH AA batteries, each one with a nominal voltage of 1.2 V and a capacity of 2000 mAh (for any current discharge, to make it easy), followed by an appropriate DC-DC converter (with a 100% efficiency, to make it easy).

28.2.4 ATmega328P DC Characteristics

T_A = -40°C to 85°C, V_{CC} = 1.8V to 5.5V (unless otherwise noted)

Symbol	Parameter	Condition	Min.	Typ.[2]	Max.	Units
I_{CC}	Power Supply Current[1]	Active 1 MHz, V_{CC} = 2V		0.3	0.5	mA
		Active 4 MHz, V_{CC} = 3V		1.7	2.5	mA
		Active 8 MHz, V_{CC} = 5V		5.2	9	mA
		Idle 1 MHz, V_{CC} = 2V		0.04	0.15	mA
		Idle 4 MHz, V_{CC} = 3V		0.3	0.7	mA
		Idle 8 MHz, V_{CC} = 5V		1.2	2.7	mA
	Power-save mode[3][4]	32 kHz TOSC enabled, V_{CC} = 1.8V		0.8	1.6	µA
		32 kHz TOSC enabled, V_{CC} = 3V		0.9	2.6	µA
	Power-down mode[3]	WDT enabled, V_{CC} = 3V		4.2	8	µA
		WDT disabled, V_{CC} = 3V		0.1	2	µA

Notes:
1. Values with "Minimizing Power Consumption" enabled (0xFF).
2. Typical values at 25°C. Maximum values are test limits in production.
3. The current consumption values include input leakage current.
4. Maximum values are characterized values and not test limits in production.

1. Compute the running time, before the batteries are empty, of the RasPi.
2. Compute the running time of the AVR microcontroller in normal operation (active mode) in the following conditions:
   a) Normal operation (active mode) With an 8 MHz clock and $V_{cc} = 5$ V.
   b) Normal operation (active mode) With a 4 MHz clock and $V_{cc} = 3$ V.
   c) Normal operation (active mode) With a 1 MHz clock and $V_{cc} = 2$ V.
3. Repeat the previous question considering that 80% of the time the AVR microcontroller goes into one of the sleep modes (to facilitate computations consider that the consumption during sleep mode is zero).
4. Comment on the use of a RasPi instead of a microcontroller in your Systems Engineering course project (bed-head inclination).

7 Choosing the right platform: FPGA, microcontroller or RasPi

1. Briefly discuss the advantages and disadvantages of each platform to carry out the course project DTMF.
2. Considering the course project, what will be worst: increasing the sampling frequency from 8 kHz to 16 kHz or increasing the number of frequencies (filters) to be detected from 8 to 16. Comment on the effect this changes would have on each platform.

8 Final question

If you know very well a topic that hasn't been asked yet this is your opportunity. You can briefly talk about one of these topics: CMOS, rate monotonic scheduling, debouncing, electrical noise and interferences, fixed-point arithmetic, DTMF codification, Goertzel algorithm...