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

Embedded Systems
Final exam. June 5, 2015

Duration: 4 hours. Last revision day: June 26

1 Warming up questions

1. Define a real time system (RTS).

2. Situate the course project DTMF in the previous definition of a RTS.

3. *What are the uses of the keyword STATIC?* 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.

   This simple question is rarely answered completely. Static has three distinct uses in C:

   a) A variable declared static within the body of a function maintains its value between function invocations.

   b) A variable declared static within a module, (but outside the body of a function) is accessible by all functions within that module. It is not accessible by functions within any other module. That is, it is a localized global.

   c) Functions declared static within a module may only be called by other functions within that module. That is, the scope of the function is localized to the module within which it is declared.

   Most candidates get the first part correct. A reasonable number get the second part correct, while a pitiful number understand answer (c). This is a serious weakness in a candidate, since they obviously do not understand the importance and benefits of localizing the scope of both data and code.

   Someone has written the following code in order to increase the value of a counter n each time the TIMER0 ISR of the ARDUINO AVR is called. A digital output is toggled every 255 calls to the ISR. As you can see, this someone has forgotten to declare n. Completely declare the variable n in one of two ways: as in answer (a) or as in answer (b).

```
int main(){
  setup();
  while(1){
    if (n==255){
      PORTD ^= 1 << PD5;//PD5 toggled
      n=0;
    }
  }
}

ISR(TIMER0_COMPA_vect){
  n++;
}
```

   Is the keyword STATIC useful when your code is used just for simulation (code size and code speed aren't important)? Consider the three distinct uses in C.

2 C for embedded systems programming

1. You are using a 16-bit architecture microcontroller and you need a counter from 0 to 255. Which data type would you choose to declare this counter?

2. Do you know the size of a variable declared as *int*? What do you need to know? Is it related to TYPEDEF?

3. What can you say about the PROGMEM atribute?

4. Consider the following C code

```
volatile uint8_t n=0;
int main(void){
    while(true){
    n++;
    }
return 0;
}
```

   and the following part of the assembly code inside the main, generated using the avr-gcc compiler with the optimizations options -Os and -mmcu=atmega328p

```
.L2:
        lds r24,n (load direct from SRAM)
        subi r24,lo8(-(1)) (substract constant from register)
        sts n,r24 (store direct to SRAM)
        rjmp .L2
```

   Discuss the result.

5. Consider the following C code

```
volatile uint8_t n=0;
int main(void){
    while(true){
    n++;
    n++;
    }
return 0;
}
```

   and write the part of the assembly code inside the main generated using the avr-gcc compiler with the optimizations options -Os and -mmcu=atmega328p?

3 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 941 Hz, using 205 samples sampled at 8 kHz, has the value $a = 2 * \cos(2 * \pi * \text{round}(941 * 205/8000)/205) \approx 1.4829$. 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: $a_{scaled} = \text{round}(\text{a} * 256) = \text{round}(379.61) = 380$. So, the absolute error after rounding is slightly greater than 0.1%.
   a) Why do we scale by a power of two (i.e 256, 512, 1024...)? Are there powers of two betters than others when using a microcontroller? and when using an FPGA?
   b) Could we obtain a lower error scaling by 512?
   c) By which number should we scale to get an error lower than 0.001%? Could we code a_{scaled} as a 16-bit signed integer?
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) Code a with the minimum number of bits to achieve the same absolute error that the one obtained when scaling by a factor of 256.
   b) Code $-a$ with b and q used in the previous question.
3. Are the two previous questions related? Hint: is the scale factor related with q?

4 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.

1. Consider two analog voltages at the input of the ADC: 0.5 V and 1 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 1.1 V and a digital value coded with the 8 more significant bits.
   b) A DE0-Nano with a digital value coded with the 8 more significant bits.
2. Consider a DTMF signal that moves between −0.5 V and 0.5 V.
   a) Which analog reference will you select when working with Arduino?
   b) Draw a sketch of the circuit used to add an appropriate DC voltage to this signal *prior* to its connection to the ADC input. Which type of filter are we dealing with? Set the values of the components, and compute the cutoff frequency of the filter.

5 Serial communication

1. Samples of 8 bit, sampled at a sampling frequency F_s, are send from an Arduino to the Raspberry Pi (RasPi) through a 57 600 bps serial connection (USART to USB bridge) using 1-start-bit and 1-stop-bit. What is the maximum F_s you could use?

2. Consider the previous Arduino trying to send samples sampled at $F_s = 8\,\text{kHz}$ through a 57 600 bps serial connection. What happens? Quantify the effect.

3. The measured maximum, mean and minimum time used by the RasPi B + to read L bytes, for any L between 1 and 1024, stored in the system buffer are approximately 500 us, 450 us and 300 us respectively. Now consider the following code (in the next questions ignore the execution time of the *print* and the *time.time()* functions).

```
# -*- coding: utf-8 -*-
import time
import numpy
import serial
# parameters
L=...
# open, wait and clean serial port
ser = serial.Serial('/dev/ttyACM0', baudrate , timeout=1)
time.sleep(2)
ser.flushInput()
while 1:
        t0=time.time()
        x=ser.read(L)
        t1=time.time()
        print t1-t0
```

   a) If, as in the previous question, Arduino is sending samples at $F_s = 8\,\text{kHz}$ through a 57 600 bps serial connection to the RasPi, which is the mean of the printed value $t_1 - t_0$? Let $L = 1$.

   b) Now let $L = 100$.

4. Now we add some code to compute something (e.g. an implementation of the Goertzel algorithm) after reading the serial port.

```
while 1:
        t0=time.time()
        x=ser.read(205)
        compute_something(x)
        t1=time.time()
        print t1-t0
```

   a) Consider an Arduino sending samples at $F_s = 8\,\text{kHz}$ through a 115 200 bps serial connection to the RasPi. The mean of the printed value $t_1 - t_0 = 25.625\,\text{ms}$. Can you determine the execution time of the *compute_something()* function? Can you limit its value?

   b) Repeat the previous question considering that the mean of the printed value $t_1 - t_0 = 27\,\text{ms}$. What percentage of the samples are lost?

   c) Repeat the two previous questions changing the baud rate from 115.2 kbps to 250 kbps.

5. If the operating system of the RasPi makes us wait 100 ms, what is the maximum sampling frequency F_s at which samples could arrive without losing them, considering that the size of the system buffer is 4095 bytes? Consider the best case, i.e. when the buffer is empty.

6 FPGA and VHDL

1. Comment on the main differences between a CPLD and an FPGA.

2. 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?

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

```
process(b)
  begin
    case b is
      when '1' => test<= '1';
      when others => test <= '-'; -- don't care
    end case;
end process;
```

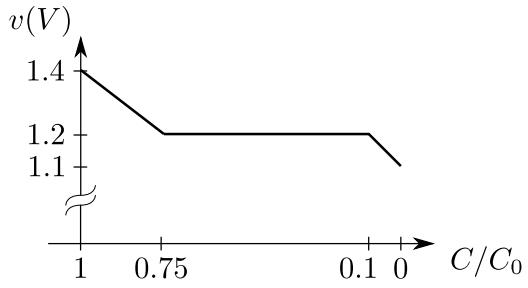
4. In the lab we have seen that the number of multipliers used, to implement one of the band-pass filters based on the Goertzel algorithm, is 6 (two 9-bit multipliers for each one of the three 16-bit multiplications used to compute the power at one frequency). When using 8 filters (for each one of the 8 DTMF frequencies) the total number of multipliers is 48. Could we reduce the numbers of multipliers from 6 to 2 for each filter (a total of 16)? Could we reduce the total number of multipliers to 2? Elaborate the answer to explain when and how this is possible.

7 Power consumption

Consider that the consumption of a RasPi with some peripherals is 5 W. It is powered by a power supply system made by 3 NiMH AA batteries, each one with a nominal voltage of 1.2 V and a capacity $C_0 = 1800$ mAh (for any current discharge, to make it easy), followed by an appropriate DC-DC converter (with a 90% efficiency).

1. Compute the running time, before the batteries are empty, of the RasPi. Let the voltage battery be the nominal one no matter the percentage discharge.

2. Repeat the previous question considering now that the voltage decreases as the remaining capacity C of the battery decreases as is shown below.



8 Choosing the right platform: FPGA, microcontroller or RasPi

1. Which platform would you use to implement a battery of filters at high sampling rate?

2. Which platform would you use to communicate with several peripherals using I2C, ISP... read data from analog and digital inputs and write analog and digital data, all of this with low complexity decision algorithms?

3. Which of these platforms could be used in an application with real time requirements?

4. Considering the course project, comment on the effect that each of the following changes would have on each platform: 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.

9 Goertzel algorithm

Many of you, maybe all of you, have chosen the set of parameters $F_s = 8$ kHz and $N = 205$ to, using the Goertzel algorithm, compute the power of a signal at frequencies $F = \{697, 770, 852, 941, 1209, 1336, 1477, 1633\}$. Give as many details as you can of why this set of parameters is a *suitable* one. Are these parameters bounded? Is there any other *suitable* set of parameters? Some of you have used a different set of frequencies F. Comment on the benefits of this decision.