Jonathan W. Valvano April 4, 2008, 10:00 to 10:50am

(10) Question 1. Each frame takes 1 ms to transfer. There are 5 buffers in the receiver. So an error occurs 4ms (plus a little time) after the RXF flag is first set.

At time	Buffers full	Status
1ms	One	Four free buffers, RXF flag first set
2ms	Two	Three free buffers, RXF flag still set
3ms	Three	Two free buffers, RXF flag still set
4ms	Four	One free buffer, RXF flag still set
5ms	Five	No free buffer
5.01ms	Five	Stall and/or lost frame

(30) Question 2.

Part a) Binary fixed-point means $\Delta=2^n$. 5V/1024 = 0.005, 1/256 is about 0.004, which is smaller than ADC resolution. Binary resolution of 1/256 = 2^{-8} V is ok. Smaller resolutions are ok. There is a fundamental tradeoff between range (reducing the chance of overflow) and resolution (reducing the chance of drop out).

Part b) For each **n**, need two longs, which is 8 bytes. There is 2K RAM. Assuming we use half the RAM for the buffer. 1024/8 is 128. 128 is the largest possible value for **n**. (if **n** were 256, then all of 2k RAM would used, and there would be no place for a stack or other variables)

Part c) In fact, most high speed FFT implementations use a table lookup for sin and cos, because it is faster. Since **`mmax`** can only be 2, 4, 8, 16, 32, I would use a 33 byte table to convert 2, 4, 8, 16, 32 to 0, 1, 2, 3, 4 respectively.

```
t = table1[mmax];
```

Then I would use a second table of size 5 to convert **`t`** into the corresponding value for **`wtemp`**, a fixed point number

```
Itemp = table2[t];
```

Part d) Since **`Idata`** and **`Itempr`** are the same format, all you need to convert is the **`wr wi`**

```
Itempr = (Iwr*Idata[j-1]-Iwi*Idata[j])/8192;
```

or

```
Itempr = (Iwr*Idata[j-1]-Iwi*Idata[j])>>13;
```

(30) Question 3. Review your EE411 before going on the interview trail.

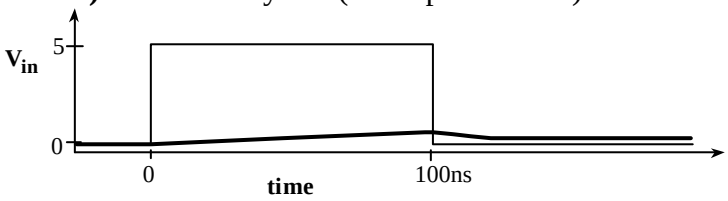
Part a) RC= 1000ns. From 0 to 100ns, $V_{out} = 5\text{-}5e^{-t/RC} = 5\text{-}5e^{-t/1000ns}$

At 100ns, $V_{out} = 5\text{-}5e^{-100ns/1000ns} = 0.48V$

After 100ns, $V_{out} = 0.48e^{-(t-100)/RC} = 0.48e^{-(t-100)/1000ns}$

At 200ns, $V_{out} = 0.48e^{-(200-100)/RC} = 0.48e^{-(100)/1000ns} = 0.43$

Part b) It is basically flat (no response at all)



Part c) Decrease R (shorter fatter wires), decrease C (increase separation between wires), or slow down transmission rate.

(30) Question 4.

Part a)

`Sema4Type NumFree` is the number of free LCD displays, initially 2

- 2 means two free LCD
- 1 means one free LCD
- 0 means no free LCD
- -1 means no free LCD and one blocked thread

`Sema4Type Mutex` is a binary semaphore used to read/modify/write the globals, initially 1

- 1 means the private globals **`LCD1Free`** and **`LCD2Free`** can be accessed
- 0 means the private globals **`LCD1Free`** and **`LCD2Free`** can not be accessed
- -1 means a thread is temporarily blocked thread, waiting for the private globals

`int LCD1Free` is a Boolean true if LCD1 is free, initially 1

`int LCD2Free` is a Boolean true if LCD2 is free, initially 1

Part b) The first thread to call will output on LCD1, the second thread to execute will output on LCD2, the third thread will block on **`OS_Wait(NumFree)`**. Whichever thread finishes first, will set its **`LCD1Free`** or **`LCD2Free`** to one, and call **`OS_Signal(NumFree)`**. This will wakeup thread 3 and it will use the free LCD.

```
void OS_Display(char *string){
  OS_Wait(NumFree);     // will block if no LCDs are free
// only two threads can enter pass here
  OS_Wait(Mutex);       // read modify write to globals
  if(LCD1Free){
    LCD1Free = 0;       // using LCD1
    OS_Signal(Mutex);   // done with globals
    LCD1_OutString(string);
    LCD1Free = 1;       // done with LCD1
  }else{
    LCD2Free = 0;       // using LCD2
    OS_Signal(Mutex);   // done with globals
    LCD2_OutString(string);
    LCD2Free = 1;       // done with LCD2
  }
  OS_Signal(NumFree);   // release resource
}
```