

HOMEWORK ASSIGNMENT #7

Due Friday, May 8th, 2009

1. (10 points) Serial Data Frames

Draw the 11-bit (8 data, even parity, 1 stop) frames for the asynchronous transmission of the ASCII character 'A' and the ASCII bell character (BEL).

Note: Clearly indicate bit intervals in the waveform.

2. (20 points) Dumb Terminal using the UART

Write a program that will configure the ADuC7026 UART for operation at 38,400 baud, 8 data bits, no parity, 2 stop bits, using pins P1.0 and P1.1.

Configure so that UART reception will cause an IRQ. All other UART interrupts will be turned off (For this problem we will ignore error conditions, and handle transmit complete through polling).

In this problem we will implement a dumb terminal. In Keil μ Vision under: **View**→**Serial Window**→**UART #1** you can get what is like a **very** dumb terminal. Everything you type in this window is sent to the UART at the configured baud rate. Everything you send from the UART will show up in this terminal.

However, what you type, does not automatically show in the terminal. So one thing we need to do is every time we receive a character we need to echo it back. This will be done in the IRQ ISR. Read the received character and transmit it out (write to COMTX).

However, we want to make a prompt to. You should define a prompt string using DCB in the flash AREA...

```
DCD "\nDumb Terminal>\0"
```

Note the newline character at the beginning, and the fact that it is a null terminated ASCII string.

You will need to write a routine (*displayPrompt*) in your main source file to display this prompt. It should read the characters one by one and send them over the UART. After each character is written to COMTX you have to wait for it to actually get transmitted. You can do this by monitoring the TEMT bit of the COMSTA0 register. Do not send the null termination character \0.

Your main routine should initialize the UART as mentioned above. Then it should call the *displayPrompt* routine once. Then it should enter an infinite loop doing nothing, because everything else will be handled by the UART receive interrupt.

The UART receive interrupt will grab the received character and compare it to a carriage return. If it is a carriage return then it should call *displayPrompt*. If it is not a carriage return it should simply echo it to terminal (write it to COMTX).

Important: Submit your program source code files *teamX_prob2_main.s* and *teamX_prob2_exceptions.s* using the homework #7 dropbox in Learn@UW. Also, submit a **paper copy** of *teamX_prob2_main.s* and *teamX_prob2_exceptions.s* with the rest of the assignment.

3. (10 points) INL of a A2D

On the website there is a Excel file called A2D_INL.xls. It contains 17 measurements of a 14-bit A2D converter. Using these measurements find the “best fit” INL of this A2D. Report the answer in whole LSB’s (this is for a datasheet so should round up to be conservative). All work should be done in the .xls file itself. Simply turn in your renamed and completed .xls file when you are done. **Hint:** There are functions called SLOPE and INTERCEPT in Excel that can be used to get the best fit coefficients for a set of data.

Important: Submit your .xls file *teamX_A2D_INL.xls* using the homework #7 dropbox in Learn@UW.

4. (10 points) ADC System Design

A. Assume that a system has to measure an input signal that can vary between 0mV and 400mV. If you are using an ADC with a voltage reference of 2.5V (ADC nominal input range of 0V to Vref) to measure the signal directly, how many bits of resolution do you need to have in order to know the value of the input with a maximum error of 1mV? (Assume that the ADC and voltage reference are ideal.) Then, consider the case where you add an ideal amplifier with a voltage gain of 6 between the signal source and the ADC. What ADC resolution is now required to be able to measure the input with a maximum error of 1mV? Provide calculations to support your answers.

5. (20 points) Sinusoidal Waveform Generation

Using the timer1 interrupt mapped to IRQ exception mode, you are to generate a sinusoidal output on DAC0 that could be used as an audio alarm signal. DAC0 should only be written to once during each timer interrupt. The sinusoidal values must be generated with a single 16-point look-up table that contains samples for a complete sinusoid cycle, and you must use linear interpolation between LUT points. (If you only used the LUT values to create a sinusoid, you should have 16 samples per cycle.) The ADuC7026 DAC has 12-bit resolution, and generates an unsigned unipolar output between 0V and a selectable reference voltage. (Note that the DAC0 output value is controlled by the lower twelve bits of the upper half-word of DAC1DAT, but you should use word transfers to write DAC0DAT for compatibility with the ADuC7026 hardware.)

The DAC0 output will be determined by inputs P4.4 as follows;

P4.4	DAC0 Waveform Output
0	400Hz at 1.25Vpp
1	100Hz at 2.5Vpp

The timer interrupt frequency must be set to 6.4 kHz ($32\text{kHz}/5=6.4\text{kHz}$). You may not be able to get the exact frequencies, but should be as close as you can. All code to control the DAC output signal should be in the timer ISR. The main program should simply sit in a spin loop once DAC is configured and the timer interrupt is enabled.

Configure DAC0CON to use the internal voltage reference (2.5V), and to update the DAC on every HCLK. The signals that you generate should always have an average value of 1.25V, with a peak-to-peak amplitude as determined by the inputs. (Do not include the average value in your sine LUT.)

To interpolate between available points in the LUT, you can either write a division subroutine, or figure out how to do it without requiring division. You are expected to make the better choice! **Think** about the numbers I am giving you to work with (what is $6.4\text{kHz}/100\text{Hz}$). How can I interpolate between 2 numbers if I know the interpolation granularity is a power of 2 fraction? Don't make this hard!

You can use the Keil debugger's Logic Analyzer function to view both the DAC0DAT contents and the DAC0 pin voltage over time. If your code is working correctly, you will see the sinusoid displayed there.

Important: Submit your program source code file *teamX_prob5_main.s* & *teamX_prob5_exceptions.s* using the homework #7 dropbox in Learn@UW. Also, submit paper copies.