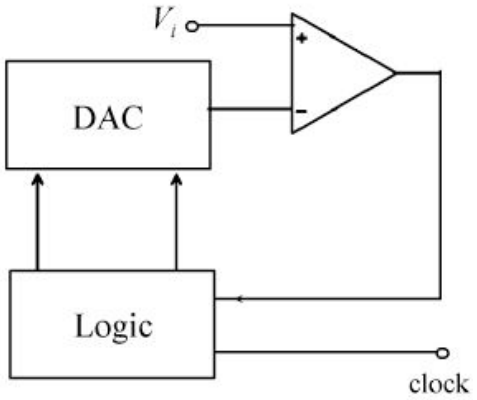
HOMEWORK VII
ECE 3710 — Microcomputer Hardware and Software
Utah State University
Due date: **November 10, 2013**

Problem 1. You are to create a program in C that uses interrupts to produce a dual-tone signal. A dual-tone signal is the signal that results from the combination of two, different frequency sinusoids (the telephone system uses dual-tone sounds for dialing). Assuming `SysClk = 25` MHz and a 64 entry table, use the DAC to produce the dual-tone signal that results from the combination of 1700 Hz and 2200 Hz sinusoids—i.e. output $s = \sin(2\pi 1700t) + \sin(2\pi 2200t)$. The relative error of each of the sinusoids must be less than 1% (relative error = abs($desired - actual$)/$desired$). You need to calculate the table values. Assume that a 10-bit DAC with V_{REF} = 3.3 V is connected to the uC and that the DAC will output a code written to `0x2000C000`. (HINTS: Use two timer ISRs and an index variable for each that indicates the current table entry for the timer. You should then have another timer ISR that combines the entries and outputs them periodically. You can use the NXP board to develop and verify your strategy to create the dual-tone signal [look at it on the logic analyzer, etc] but will probably want to write the timer code for either the LM3S1968 or your board.)

Problem 2. It is possible to build an analogue-to-digital converter using only a comparator and a digital-to-analog converter. This is how it works: connect the signal you wish to convert (V_i) to the non-inverting input of the comparator and the output of the DAC to the inverting input (see the figure below). If V_i is greater than the DAC output, the comparator will output a one (high); if the DAC output is greater than or equal to V_i the comparator will output a zero (low). To find the ADC code then, you have the DAC output code zero and steadily increase the code until the comparator outputs a zero. The DAC code that makes the comparator change from one to zero is taken as the ADC code. Such an ADC is known as a successive approximation ADC.



Write an assembly routine for the LM3S1968 that implements a successive approximation ADC. The ADC code the routine determines should be 1) the same as the LM3S1968's ADC would produce if it were measuring the same input and 2) pushed on the stack before the routine returns. You will need to configure one of LM3S1968 analog comparators (Section 15 of the datasheet). Assume that a 10-bit DAC with V_{REF} = 3.0 V is connected to the -ve input of analog comparator zero and that the DAC will output a code written to `0x2000C000`. Use the simulator to create and debug your code.

Problem 3. (EXTRA CREDIT)The algorithm (starting at DAC code zero and increasing by one) by which the ADC code is determined in problem two is not efficient. A better way to find the code is using binary search. Create an assembly routine that implements a successive approximation ADC using binary search.