5 Implementing a Simple Signal Generator

5.1 Overview

In this experiment, you implement a simple signal generator. The signal generator will be capbable of outputting a sine wave, a square wave, or a sawtooth wave. It will be capable of output signals whose frequencies range from 1 Hz to 999 Hz in steps of 1 Hz. The frequencies will be as accurate as the board's crystal allows.

The user will operate the signal generator by sending it commands via the UART. The command will be a four character string. The first character will be an 'S', and 'A', or a 'Q' – where 'S' stands for Sine wave, 'A' stands for sAwtooth wave, and 'Q' stands for sQuare wave. The next three characters will be a three digit number. Thus to have the signal generator output a 200 Hz sawtooth wave, the user will type A200.

5.2 Technical Specifications

Have Timer3 clock the UART, and set the baudrate to 115,200. Have the signal output via DAC1. Set the DAC to run from 0 V to V_{ref} in 8-bit mode. The value passed to the DAC by the program should go from near 0 to near 255. The user will not be able to vary the amplitude of the signal being output.

5.3 Some Hints

5.3.1 Setting the Frequency Accurately

In order to make the frequency as accurate as possible, the period of the output should be set using a timer running in autoreload mode.

I found it helpful to work with a cosine sampled 200 times (rather than 256 times as in the lab of Chapter 19). I set a timer in autoreload mode to interrupt every $1/(256 \times 200)$ s, and I had a two-byte "counter" that was updated in the timer's interrupt subroutine (ISR). The most significant byte was allowed from 0 to 199, and the least significant byte varied from 0 to 255. Each time the ADuC841 entered the timer's ISR, it caused the DAC to output the sample whose position in the array of cosine values was equal to the most significant byte of the counter.

In addition, each time the μC entered the ISR, it added the desired sampling frequency to the two byte counter. Whenever the high byte of the counter was 200 or greater, the μC subtracted to 200 from the value (using the `SUBB` command). Please think about why incrementing the two-byte

counter by the sampling frequency works! (It is closely tied to the length of the array of the values of the cosine and the frequency with which the timer interrupt occurs.)

5.4 Reading a Four Character String

When reading the different characters, you may want to define four flags. One flag is raised when you are expecting the first character, one for the second, and so forth.

5.5 Converting the Numbers from ASCII to Binary

When reading in the numbers, note that the format of a decimal digit in ASCII is $0011b_3b_2b_1b_0$B. Thus, `and`ing a character that is supposed to be a number with 00001111B returns the number. Also, please note the 8051's assembly language has a multiply command – `MUL AB`. It should also prove useful.