ECE271: Microcomputer Architecture and Applications — University of Maine

Prelab for Lab #11: Digital to Analog Converter (DAC)

Week of 22 April 2019

Part A – Textbook Readings

1. Read Textbook `Chapter 21` to review Digital/Analog Conversion.

Part B – Prelab assignment

In this lab we will set up the Digital to Analog Converter (DAC) and use it to generate a 440Hz sine wave.

1. Setting up TIM4

In digital audio, a sampling frequency of 44.1kHz is often used for recording and playback. This is because the human ear can hear up to roughly 20kHz, and according to the Nyquist-Shannon sampling theorem you can reconstruct a waveform if you sample at twice the maximum frequency. For example, compact-disc (CD) audio is recorded at 44.1kHz with 16-bit samples.

1. Assume we want to drive the DAC at a frequency of 44.1kHz using the output of timer TIM4. We will use the HSI clock running at 16MHz. Can you find values for `PSC` and `ARR` that will be as close to 44.1kHz as possible (within 5% or less)

$$44.1kHz = f_{sampling} = \frac{f_{HSI}}{(1+PSC)(1+ARR)}$$

`TIM4->PSC` = ________________

`TIM4->ARR` = ________________

2. Generating the 440Hz signal

We are trying to generate a 440Hz signal (music note A) and the DAC is being triggered by TIM4 TRG0 at 44.1kHz.

1. When outputting, we want to increase the angle of the sinewave being output from 0 to 360 degrees over the course of 440Hz. Assuming we are being interrupted at 44.1kHz, how much do we need to increment the angle during each timer interrupt? (Note: info on how to do this can be found in section 21.7 of the textbook).

2. We are not using the floating point unit on the STM32L board, so what technique can we use to handle non-integer (fractional) increments in the angle size?

3. Generating the sine lookup table

Instead of generating the sine value on the fly, we are going to pre-generate the sine values in advance, and look up the values in a lookup table. Shown below is a lookup table you can use for the assignment, this was generated by code similar to Example 21-1 in the textbook.

The code in the textbook is generating a lookup table for use in an assembly language program. Briefly describe how you would modify it so that it would generate the code suitable for C, as shown below.

```
int sine_lookup[91]={
        /* 00 */ 0x800,0x823,0x847,0x86b,0x88e,0x8b2,0x8d6,0x8f9,0x91d,0x940,
        /* 10 */ 0x963,0x986,0x9a9,0x9cc,0x9ef,0xa12,0xa34,0xa56,0xa78,0xa9a,
        /* 20 */ 0xabc,0xadd,0xaff,0xb20,0xb40,0xb61,0xb81,0xba1,0xbc1,0xbe0,
        /* 30 */ 0xc00,0xc1e,0xc3d,0xc5b,0xc79,0xc96,0xcb3,0xcd0,0xcec,0xd08,
        /* 40 */ 0xd24,0xd3f,0xd5a,0xd74,0xd8e,0xda8,0xdc1,0xdd9,0xdf1,0xe09,
        /* 50 */ 0xe20,0xe37,0xe4d,0xe63,0xe78,0xe8d,0xea1,0xeb5,0xec8,0xedb,
        /* 60 */ 0xeed,0xeff,0xf10,0xf20,0xf30,0xf40,0xf4e,0xf5d,0xf6a,0xf77,
        /* 70 */ 0xf84,0xf90,0xf9b,0xfa6,0xfb0,0xfba,0xfc3,0xfcb,0xfd3,0xfda,
        /* 80 */ 0xfe0,0xfe6,0xfec,0xff0,0xff4,0xff8,0xffb,0xffd,0xffe,0xfff,
        /* 90 */ 0xfff,
};
```