Lab 11: Digital to Analog Converter (DAC)
Instructor: Prof. Yifeng Zhu
Spring 2016

Goals

1. Understand basic concepts of DAC conversions
2. Configure DAC resolution and sampling rate
3. Use a timer to periodically trigger DAC conversions
4. Perform fixed-point floating point operations by using only integer instructions
5. Efficiently find the result of a complex function by looking up a table

Pre-Lab Assignment

1. Read Chapter 21 Digital to Analog Conversion of the Textbook
2. Complete the prelab assignment

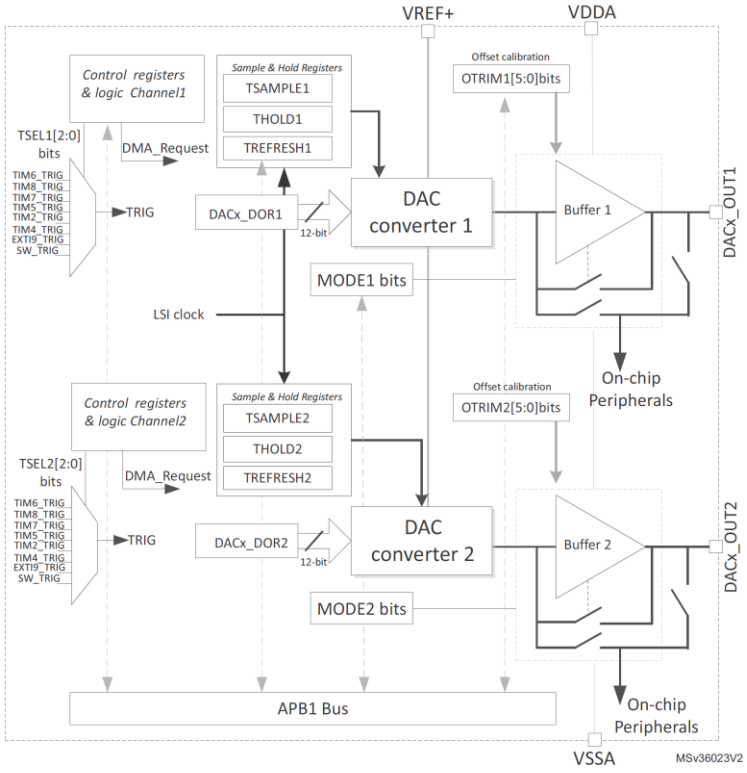
Lab Demo

1. Generate a sinusoidal waveform with a frequency of 440Hz on **PA.5** (DAC1_OUT2) and use an oscilloscope to verify the frequency
2. Something cool (10%)

Post-lab Assignment

1. Write your answer in Readme.md and submit it to the gitlab server.

Introduction to DAC modules



DAC diagram (from *STM32L4 Reference Manual*)

The STM32L4 has two DAC channels: **PA.4** (DAC1_OUT1) and **PA.5** (DAC1_OUT2).

- Each channel has its own converter.
- Each channel has two modes:
    1. the ***normal*** mode. The DAC output can be buffered to increase the output power.
    2. the ***sample and hold*** mode. The sample and hold mode is a low power mode, and it holds the converted voltage on a capacitor. When DAC is not converting, the DAC core is turned off and the DAC output is tri-stated, thus improving the energy efficiency. This mode requires that LSI (Low Speed Internal) is used to drive the DAC core.
- The Discovery Kit extends the pin PA.5 (DAC1_OUT2) but **not** the pin PA.4 (DAC1_OUT1).
- If we need two DAC channels, we can leverage the on-chip OPAMP. Within the processor ship, software can program the OPAMP module and route DAC1_OUT1 to pin PA.3 (OPAMP1_VOUT).

This lab requires you to generate a 440Hz sinusoidal wave and output it on **PA.5** (DAC1_OUT2). You shall configure TIM4 as a master trigger to the DAC converter. When the MMS bits in the TIM4_CR2 register controls the trigger output (TIM4_TRGO).

For example,

- When MMS bits are 010, the TRGO signal has a rising edge each time an update event occurs.
- When MMS bits are 100, 101, 110, and 111, the OC1REF, OC2REF, OC3REF and OC4REF is selected as the TRGO output, respectively.

The following gives an example C code of configuring the TIM4 that uses OC1REF as the TRGO output, and the default MSI clock is used.

Please note that you might need to change the parameters of this program. The following demo program sets the timer trigger as 10 KHz when the timer is driven by 80 MHz.

```
RCC->APB1ENR1 |= RCC_APB1ENR1_TIM4EN; // Enable Clock of Timer 4

TIM4->CR1  &= ~TIM_CR1_CMS;  // Edge-aligned mode
TIM4->CR1  &= ~TIM_CR1_DIR;  // Counting direction: Up Counting

// Master mode selection
TIM4->CR2  &= ~TIM_CR2_MMS;  // Master mode selection
TIM4->CR2  |= TIM_CR2_MMS_2; // 100 = OC1REF as TRGO

TIM4->DIER |= TIM_DIER_TIE;  // Trigger interrupt enable

TIM4->DIER |= TIM_DIER_UIE;  // Update interrupt enable
```

```
// OC1M: Output Compare 1 mode
TIM4->CCMR1 &= ~TIM_CCMR1_OC1M;
TIM4->CCMR1 |= TIM_CCMR1_OC1M_1 | TIM_CCMR1_OC1M_2;  // 0110 = PWM mode 1

// The counter clock frequency (CK_CNT) = fCK_PSC / (PSC[15:0] + 1)
// Timer driving frequency = 80 MHz/(1 + PSC) = 80 MHz/(1+7) = 10MHz
TIM4->PSC  =   7;    // max 65535

// Trigger frequency = 10MHz / (1 + ARR) = 10MHz/1000 = 10KHz
TIM4->ARR  = 999;   // max 65535

TIM4->CCR1 = 500;   // Duty ratio = 50%

TIM4->CCER |= TIM_CCER_CC1E;  //  OC1 signal is output on the corresponding output pin

// Enable timer
TIM4->CR1  |= TIM_CR1_CEN;   // Enable counter
```

ECE 271 Pre-lab
Lab 10: Digital to Analog Conversion (DAC)
Spring 2016

Part 1: Configuration of ADC registers

In Figure 21-8, flowchart of configuring DAC, we need replace "step 2" of "Configure DAC" as the following for STM32L4 board:

"2. Modify DAC_MCR register, and configure the DAC Channel 2 (DAC_OUT2, PA.5) as normal mode connected to external pin with buffer enabled"

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	DAC_CR	Res.	CEN2	DMAUDRIE2	DMAEN2	MAMP2[3:0]				WAVE2[2:0]		TSEL2 [2:0]			TEN2	Res.	EN2	Res.	CEN1	DMAUDRIE1	DMAEN1	MAMP1[3:0]				WAVE1[2:0]		TSEL1 [2:0]			TEN1	Res.	EN1
	Value																																
0x04	DAC_ SWTRGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SWTRIG2	SWTRIG1
	Value																																
0x2C	DAC_DOR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DACC1DOR[11:0]											
	Value																																
0x30	DAC_DOR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DACC2DOR[11:0]											
	Value																																
0x34	DAC_SR	BWST2	CAL_FLAG2	DMAUDR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BWST1	CAL_FLAG1	DMAUDR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Value																																
0x38	DAC_CCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OTRIM2[4:0]					Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OTRIM1[4:0]				
	Value																																
0x3C	DAC_MCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MODE2 [2:0]			Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MODE1 [2:0]		
	Value																																
0x40	DAC_SHSR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TSAMPLE1[9:0]									
	Value																																
0x44	DAC_SHSR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TSAMPLE2[9:0]									
	Value																																
0x48	DAC_SHHR	Res.	Res.	Res.	Res.	Res.	Res.	THOLD2[9:0]										Res.	Res.	Res.	Res.	Res.	Res.	THOLD1[9:0]									
	Value																																
0x4C	DAC_SHRR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TREFRESH2[7:0]								Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TREFRESH1[7:0]							
	Value																																

Part 2: Submit your answer in *readme.md*.

In digital audio a common sampling frequency is 44,100Hz, i.e, 44.1kHz. That means an analog audio signal is recorded as 44100 digital values per second. Human ears can hear up to 20,000Hz. According to the Nyquist–Shannon sampling theorem, the sampling frequency must be at least twice of the maximum frequency of signals audible to human ears. Most Compact Discs (CD) are recorded with this rate.

(1) Assuming an audio is recorded at a rate of **44,100** Hz, and the DAC is driven by the Timer trigger output (TGRO). What is the time interval of between two consecutive triggers? If the timer is driven by the HSI clock (16MHz), how do you set up the values of **TIM4->PSC** and **TIM4->ARR**? Show your calculation.

TIM4->PSC = _____
TIM4->ARR = _____

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

(2) Assuming we are to generate a sinusoidal waveform of 440Hz (music tone A) and the DAC converter is triggered by TIM4 TRGO with a frequency of 44100Hz.

a. How many DAC outputs we should produce during one cycle of sinusoidal waveform?

b. The angle of the sine function should increase from 0° to 360° to complete one cycle in the sinusoidal waveform. In order to achieve that, how many degrees the angle variable should be increased in ***TIM4_IRQHandler*** each time?

c. Suppose we only use integer arithmetic. If the degree to be increased is not an integer, what can you do to get around of this issue?

ECE 271 Post-lab
Lab 10: Digital to Analog Conversion (DAC)
Spring 2016

Submit your answer in *readme.md*.

1. Assume we are required to generate a sinusoidal waveform of 293.665 Hz (music tone D) and the DAC converter is triggered by TIM4 TRGO with a frequency of 44,100Hz.
   a. How many DAC outputs should we produce during one cycle of sinusoidal waveform?

   b. The angle of the sine function should increase from 0° to 360° in order to complete one cycle of the sinusoidal waveform. How many degrees should the angle variable be increased in timer interrupt handler each time?

2. The math library in C provides a function called "sin" to calculate the sine of a radian angle. The sin function is declared as follow: `double sin(double x)`. However, in this lab, we cannot call the sin function in the interrupt handler. Explain why? (Hints: FPU only provides single-precision floating-point operations. All double-precision operations are carried out by software.)