Lab 5: Timers, Interrupts and DAC

ECSE 444: Microprocessors

Winter 2019

In this lab, you will learn how to use more peripherals on the STM32L475VG MCU. In particular, you will:

- Learn how to use the UART to enable communication with a host workstation.
- Use Matlab to obtain audio data from a recorded voice.
- Use the audio data in your MCU to generate audio on a headset.
- Utilize a timer to periodically generate an interrupt.

Changelog

- 12-Feb Initial release.

Grading

- 10 % Audio data generated by Matlab.
- 10 % Button press and LED operations.
- 30 % DAC output generation.
- 40 % Time source generation by Timer.
- 10 % For downloading data from the serial interface (0 % if you copy/paste the array).

Introduction

In this assignment, your MCU operates as an speaker. You have an array of data—your recorded voice—and this voice should be heard in your headphones (connected to the MCU) when the blue button is pressed. The LED should be ON during the process of generating the audio signal and downloading data from PC, so the user has an indication of the behavior of the system.

Matlab and Audio Data

In this assignment you need a sample of a recorded voice. This voice is your group number (in Hexadecimal, e.g., G21 would be G0x15). So, you must record your voice within one to two seconds while saying your group number. You can use the uploaded Matlab code to generate an array of raw data. This file generates a .csv file (comma separated). Then you can open this file in text editors and copy this data into Keil as an array of data stored into your MCU flash or memory.

To get full points, you need to download this recording onto your MCU from the PC using the MCU's UART. You can use Matlab or other tools; there are many resources online. If you have recorded a longer voice, you can cut it to the duration you prefer. You can use any other tool to record a voice or convert it into an array of raw data.

DAC

The STM32L4 family of MCUs come with two Digital to Analog Converters (DACs). You must use **both** channels to generate stereo sounds. In this task, you will need to use the HAL drivers to initialize the DAC and feed this block from your audio data. You are already familiar with DAC block. More attention should be paid to the voltage level you are applying on the output; it is important that you be able to justify this. Note that you must set zero voltage to the output once your output signal generation is done.

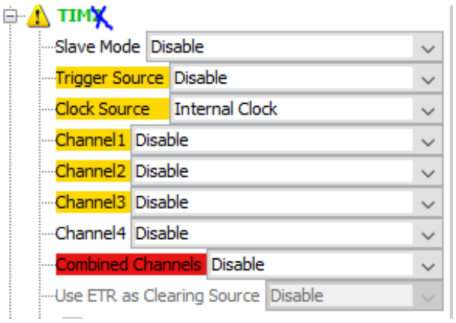
The headphone jack has three wires. One is the common or ground which should be connected to the GND on board. The other two wires should be connected to the DAC outputs. **Note that the metal on the head of jack would be GND. Be careful of short circuits if it touches other pins or components on the board.**

Timer

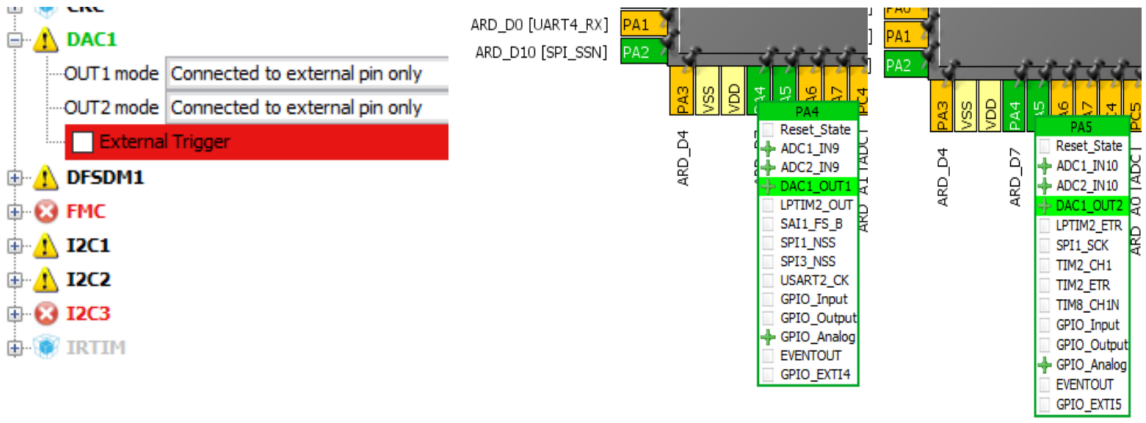
Your MCU has many timers, from basic to advanced modes. In this assignment, your application is the most basic usage of timers which is generating interrupts whenever the counter of the timer reaches the value you set (its *period*). This number is called Period in the settings of the timer. You are free to choose a timer from the list of basic and general-purpose timers. By referring to the Doc-2 (reference manual) you can find the clock source of the timer you selected. Now, you need to set a value for Period to obtain the desired frequency of interrupt calls. At every interrupt call you must write an element of the audio data into your DAC channels. The speed of writing to the DAC block should match the speed of sampled audio data.

You can follow instructions to generate the base project or use the uploaded base project. Note that the uploaded base project contains only the GPIO setting. You must write code to initialize the DAC and Timer. **If you are using CubeMx, select "No" when it asks whether to configure all peripherals by default.**

First, setup your timer's basic operation:

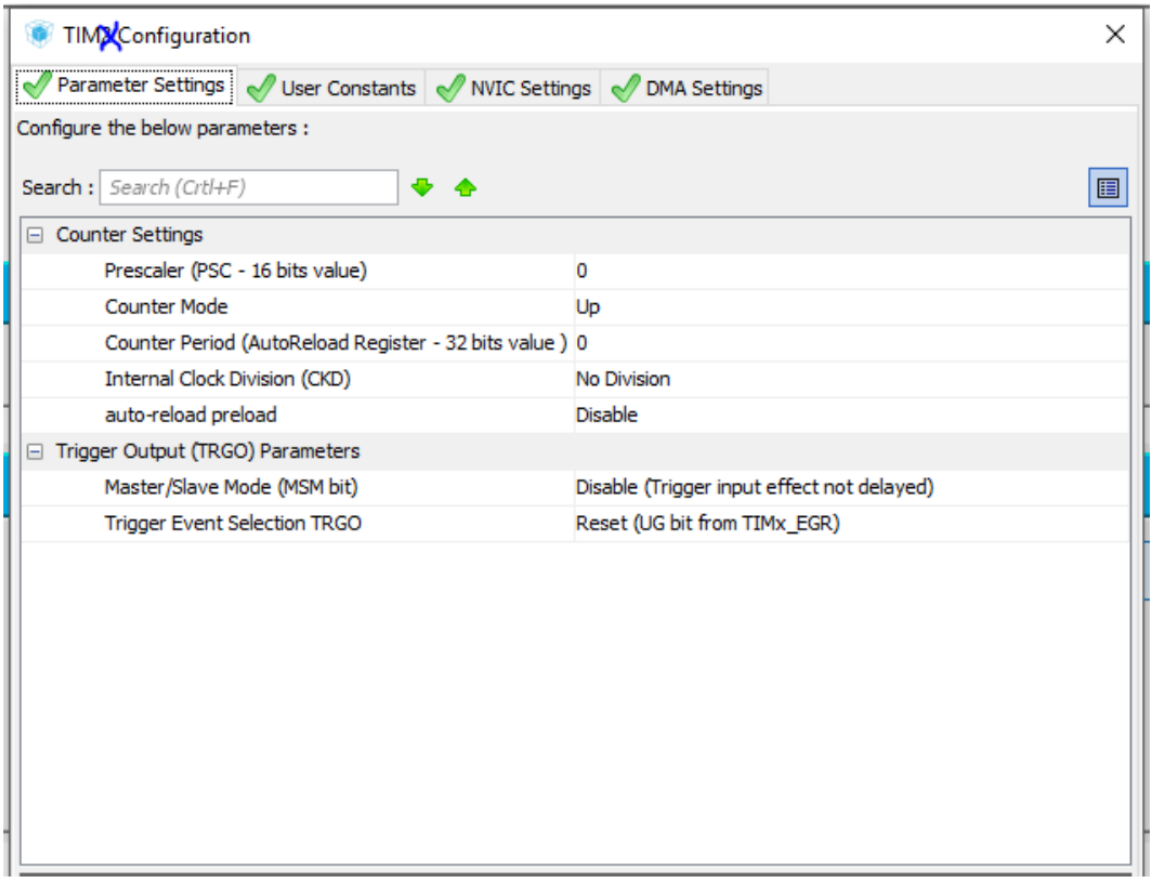


Enable your DAC channels:



Then, ensure your clock-tree is well configured. By default, the HCLK = 80 MHz.

You can set your timer settings (period and prescaler) in this window. They could be changed later in your code. In the NVIC tab, enable the interrupt.



Now, your code is ready to be generated. Start the timer in the interrupt mode and use the timer interrupt handler for outputting your audio signal. The rest is similar to what you have done before.