SES Lab 2:
STM32 Audio Recording

John Thompson

Electronics and Electrical Engineering Discipline

Week 9

Week 9: Audio Recording

This week we will record speech as follows:

- Build our audio amplifier circuit on the breadboard
- Digitise the amplified signal using the STM32 and print the results to the computer screen
- We will post-process the samples in MATLAB to convert them into WAV format so you can listen to them

We will make use of the ADC function from **Week 7 Materials**

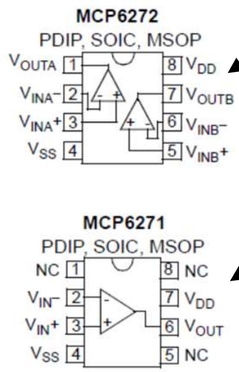
- Use it to check the DC levels in your circuit are OK
- Expand this code to sample the speech at 8 kHz and print to your computer screen

Building your Circuit on Breadboard

- Before you build your circuit on breadboard, draw out your circuit carefully in your lab book
- Connect up your circuit correctly, especially:
  - Electret Microphone must be connected the right way up
  - Do not forget to provide power to the op-amp chip (0V/3.3V) !!!
- **Test op-amp DC levels by disconnecting microphone**

Source: makezine.com

The leg with **green connectors** to the case should connect towards ground

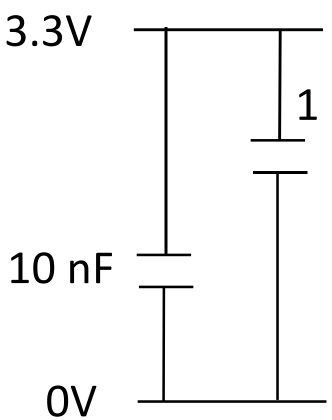


Check the pinout of your op-amp chip carefully: **MCP6272&1** on the datasheet

Amplifier Breadboard Circuits

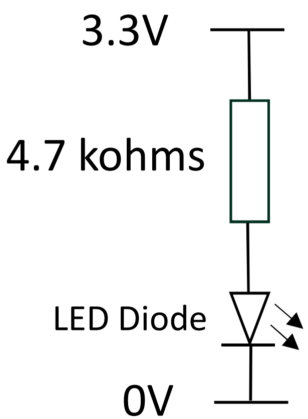
Connect up three simple additional circuits on the breadboard shown below:

Decoupling Capacitors



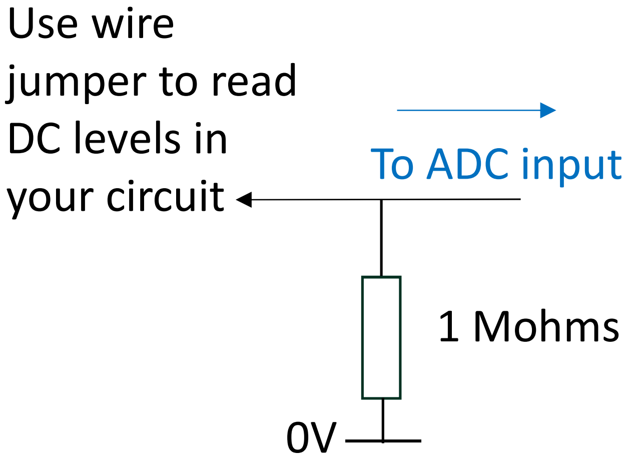
Reduce noise arising from digital circuits on STM32 Nucleo board

Week 7: Power Indicator Circuit:



This simple circuit allows the **LED** to glow showing 3.3V is indeed connected!

Week 7: ADC Measuring Circuit:



This circuit allows us to test the voltage measurement by the ADC

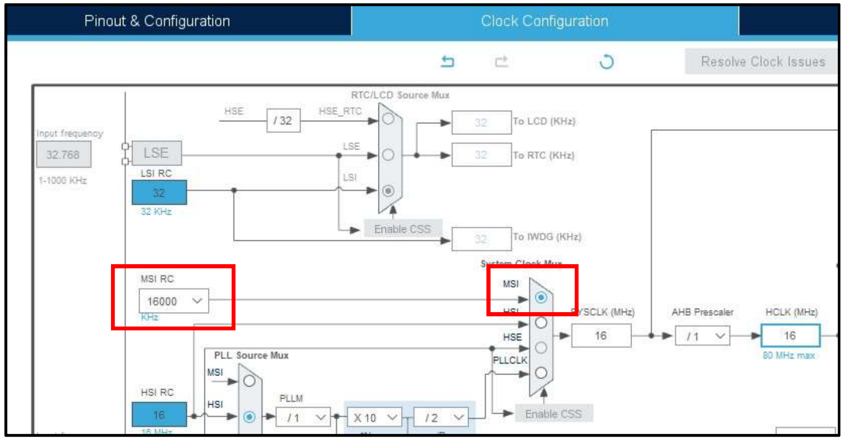
Now Try Question 1 in **Worksheet 1**

Reading Audio Samples

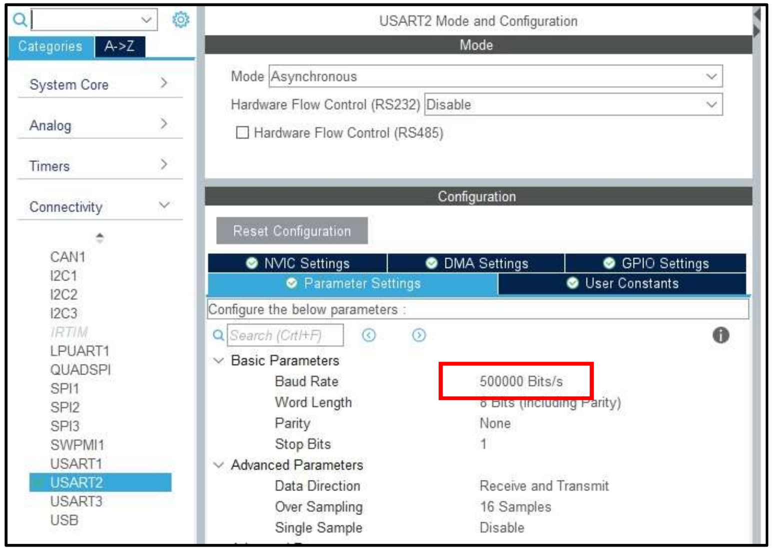
- Next we will look at how we can expand on our previous code projects to do the following:
  1. Sample audio signals at 8 kHz using the STM32 ADC
  2. Print this data from the Nucleo board to the Putty terminal
  3. Post-process the data in MATLAB so you can create a WAV file to listen to!
- We will need to use a counter **TIM16** in the STM32 microprocessor to keep track of time to know when to sample
- We will then store 3 seconds worth of samples in the STM32
- Once the samples are collected, we will print them to Putty
- This approach seems the most reliable way to collect data:
  - The **USART2** interface can be periodically quite slow, so it cannot reliably print data samples to the screen between samples

Setting Up the Project

- In Cube IDE, select File → New → STM32 Project from Existing STM32CubeMX Configuration File (.ioc)
- Select the **ioc** file for your **Oscilloscope** project from Week 7 Lab
- We will make a couple of adjustments as follows:



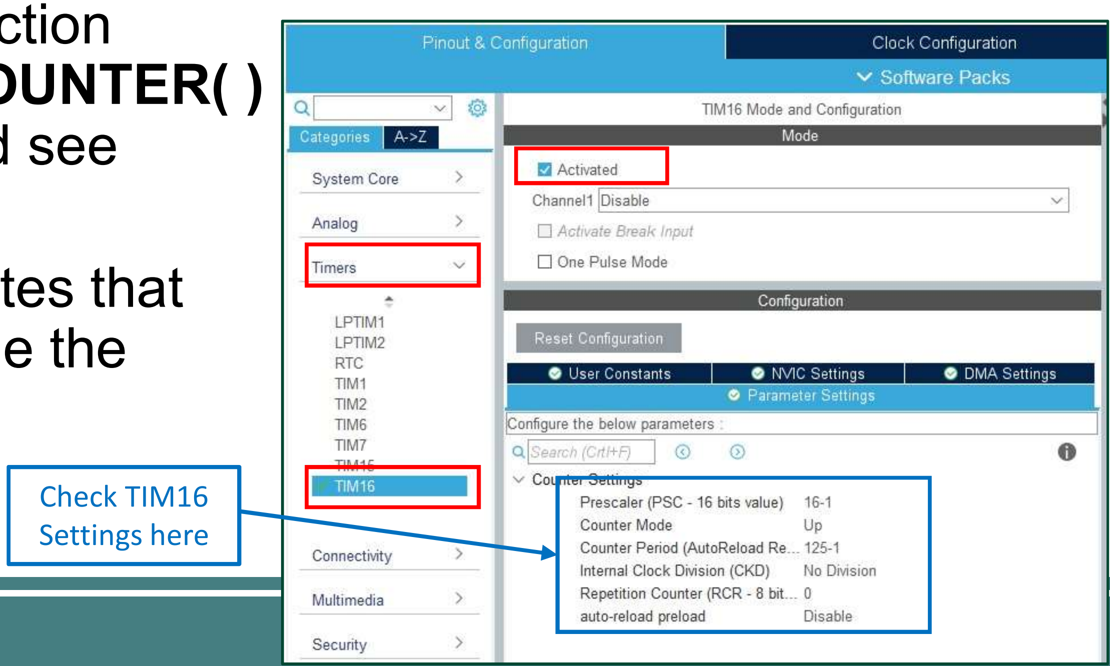
We should now set the MSI clock frequency to **16,000 kHz** and ensure the MSI clock option is still selected



This allows us to set a baud rate of **500,000 bits/sec** on USART2 to print data samples to Putty more rapidly!

Using Timers

- The STM32 chip has a number of timers that are connected to the main clock and we can read their values in C
- We will use timer TIM16 with the following settings:
  - **Prescaler:** 16-1 (= 15) → This means that the counter counts up in steps of one at a rate 1/16 times the main clock: 16/16 = **1 MHz**
  - **Counter Period:** 125-1 (=124) → This means the counter counts up from 0 to 124 and back to 0 again (i.e. a period of **125 microseconds**)
- In C we can use a function **__HAL_TIM_GET_COUNTER( )** to check the value and see when it reaches 124
- When it does, it indicates that we are ready to sample the audio signal



Audio Samples Coding

We need to add these items to your C file to make it work:

1. Include *string.h/stdio.h* as before and define a constant SAMP – number of samples to store

2. Define a string ‘buf’ of type **uint8_t**, a counter k and an array of samples ‘dat’ of type **uint16_t**

3. Before we enter the while(1) loop, we also need to start the TIM16 counter to count with the function **HAL_TIM_Base_Start( )**

We will need some extra code in the while(1) loop to read the data and print the samples…

1.

```
/* USER CODE BEGIN PD */

#define SAMP 24000

/* USER CODE END PD */
```

2.

```
int main(void)
{
  /* USER CODE BEGIN 1 */

        int k;
        uint8_t buf[40];
        uint16_t dat[SAMP];

  /* USER CODE END 1 */
```

3.

```
  /* USER CODE BEGIN 2 */

  HAL_TIM_Base_Start(&htim16);

  /* USER CODE END 2 */
```

Collecting and Printing Samples

- Inside the **while(1)** loop, we will firstly use a for loop to collect ‘SAMP’ data samples (3 seconds worth of audio) then print it out to Putty

The while loop waits for counter to reach **125 micro-secs** and then collects and stores a data sample

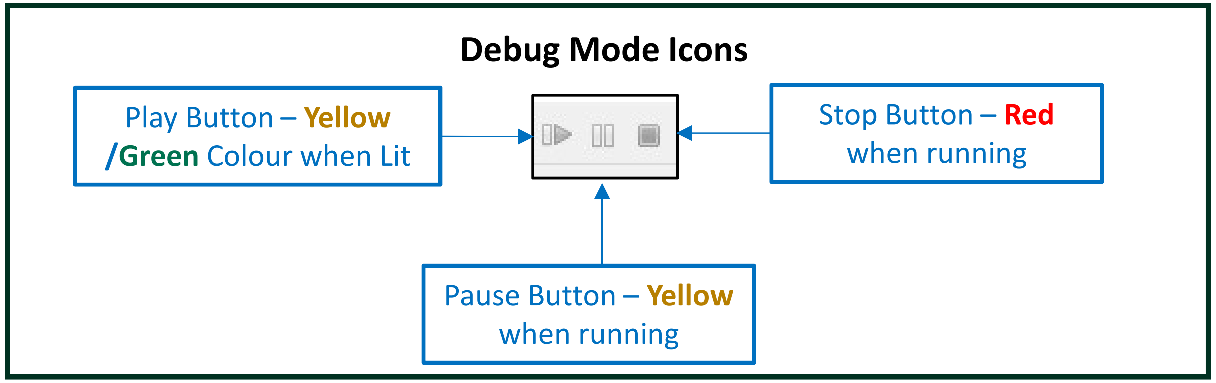
Once the data samples are collected, they are printed to the Putty terminal with a delay of **1 millisecond** between samples:

```c
  while (1)
  {
      // for loop to read samples into memory

      for(k=0;k<SAMP;k++)
          {
              while((__HAL_TIM_GET_COUNTER(&htim16))<124);
              HAL_ADC_Start(&hadc1);
              HAL_ADC_PollForConversion(&hadc1, HAL_MAX_DELAY);
              dat[k] = HAL_ADC_GetValue(&hadc1);
          }

      // for loop to print samples to computer screen with 1 msec pause between samples

      for(k=0;k<SAMP;k++)
          {
              sprintf((char*)buf, "%d\r\n",dat[k]);
              HAL_UART_Transmit(&huart2, buf, strlen((char*)buf), HAL_MAX_DELAY);
              HAL_Delay(1);
          }

      // blink LED here to show we are ready for next recording?

      /* USER CODE END WHILE */

    /* USER CODE BEGIN 3 */
  }
  /* USER CODE END 3 */
}
```

- At the end of the loop, use **HAL_GPIO_TogglePin(GPIOB, GPIO_PIN_13)** to flash LED4 as you wish with to show more samples are about to be collected

Saving Audio Samples

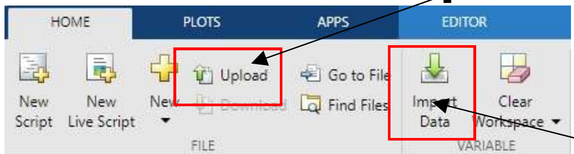
- You will need to run **Putty** to print out and save the data samples to a file
  - We have used here a higher baud rate of **500,000 bits/sec** (5 followed by five zeros) so make sure this is set in Putty!
  - Read the Putty instructions again, especially the section "Saving Data from Putty Terminal Window" on Page 3 – this explains the key steps
- Putty should be run before you press play in Cube IDE debug mode
- Press pause on your code in debug mode and make sure the Putty printout has stopped before pressing "X" to save the file



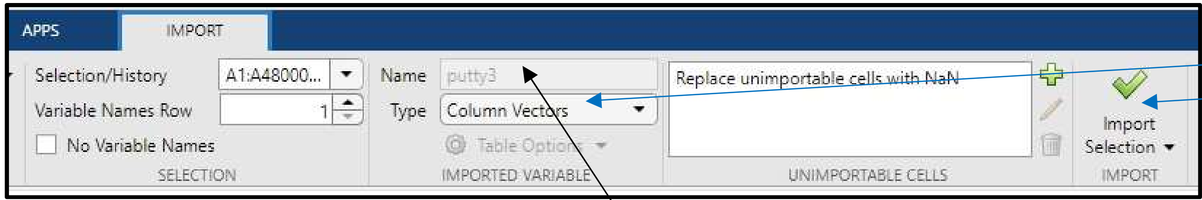
MATLAB Post-Processing

- The datafile that you save from putty can be imported into MATLAB to convert into a WAV file that you can listen to:

1. MATLAB online has an **Upload** option on **Home Screen**



2. Then import the file using the "Import Data" option



Data Import: Select Column Vectors & then "Import Selection

3. The samples (.e.g. **putty3**) are in the range 0-4095 but you need to convert to a data vector **dat** in the range -1 to 1

4. Finally you can use the audiowrite function to save and download your file, e.g for sample rate 8000 Hz:

   audiowrite('data.wav',dat, 8000);

Now Try Questions 2 and 3 in **Worksheet 1**