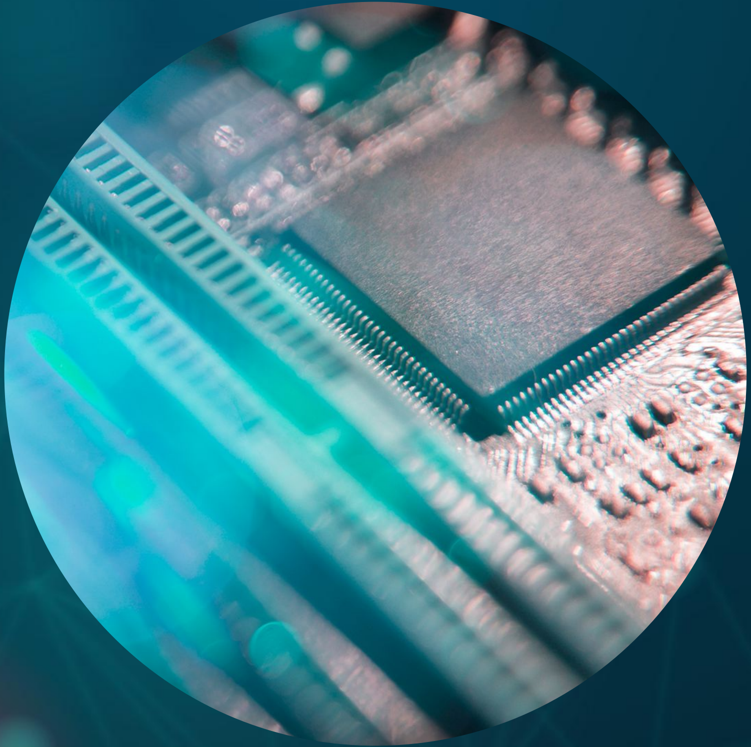
ARM Based Microcontroller

Audible Signal

Lecture 10



Advanced
RISC
Machines

IMT SCHOOL

Introduction 01

Audible Signal

Introduction

This slide discusses a very interesting topic, it's about human voice, its recording and storing, then playing it using the microcontroller and our DAC.
Our world is full of sounds. The wind, birds, traffic, buzz of voices, discussions, radio, music, sirens. The soundscape is an essential part of our environment, but not all sounds are natural anymore. Our technology causes noise and we are surrounded by devices designed to create artificial sound.
One of the keys to our wellbeing is to master the soundscape. Listening to streamed music, always and everywhere, is perhaps what most of us think about when you say audio technology. But the possibilities are wider than that, far wider.

The audio devices are becoming even smaller and will soon be integrated in our clothes. Earbuds support performing artists and technology can improve wellbeing significantly for hearing impaired people.

Introduction

Frequency range : -

The division of the entire frequency spectrum into individual areas reserved for specific applications or techniques is called the frequency domain. The entire frequency spectrum consists of sound waves, electromagnetic waves and light waves. A rough classification characterizes the range above the sound waves between 30 kHz and 300 MHz as high frequency, the frequency range between 300 MHz and 300 GHz as microwaves and the overlying range between 300 GHz and 400 THz as infrared, followed by the visible light.

Frequency scale : -

The frequencies are divided into kHz (kilo-hertz, 1 kilo-hertz = 1000 hertz). Physically, sounds are sound waves. The unit frequency gives the number of vibrations of a sound wave / sec. The deepest test tone the human hearing organ can hear has 125 vibrations / sec. (= 0.125 kHz) and is a low buzz. The highest test tone the human hearing organ can hear vibrates 12000 times per second (= 12 kHz) and is a shrill, high tone, this 12 kHz can be up to 20 kHz for a healthy young person. With the tone scale from 0.125 kHz (125 Hz) to 12 kHz (12000 Hz), the audio metric covers the overall average sonic sensing capability of the human auditory organ. The human hearing organ can thushear a very extensive frequency range (large pitch).

IMT SCHOOL

Human Voice Frequency Range

Female voice :-

1. Frequency range covers fairly up to 350 Hz to 17 KHz.
2. Its fundamental frequency is 350 Hz to 3 KHz and Harmonics is 3 KHz to 17 KHz.

Male voice : -

1. Covers a Frequency range of 100Hz to 8 KHz.
2. The fundamental is 100 Hz to 900 Hz and Harmonics is 900 Hz to 8 KHz

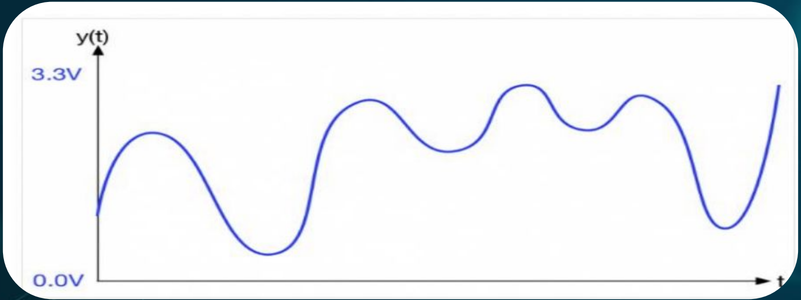
Digitalization 02

Audible Signal

Sampling

Sample and Hold :-

As we can see, the analog signal can take any value y(t) between 0v and 3.3v. Furthermore, you can measure the voltage at any time t. So we can deduce that the analog signal has no breaks in either dimension, so we say this signal is continuous.



But as we deal with a digital microcontroller, we can't measure y(t) for all possible values of t, even for a fixed duration, this would be still an infinite set of data.

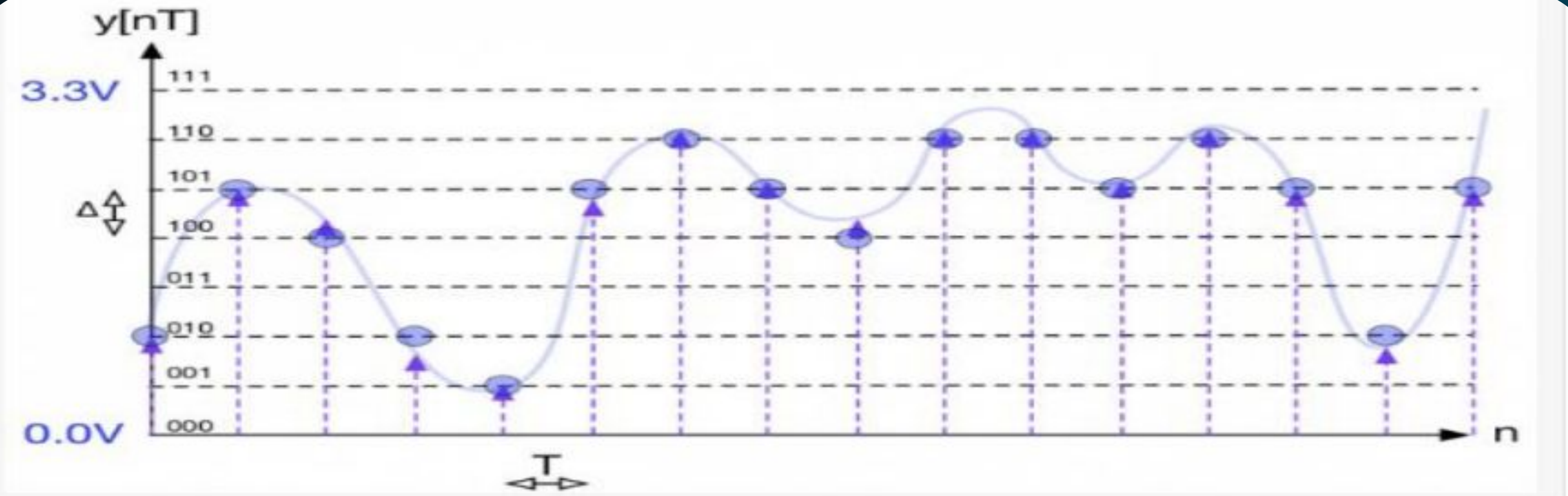
So we must divide this continuous signal into a finite number of samples at regular intervals, so that our microcontroller can measure. This process is known as sampling and the output signal is said to be discrete

Quantization

Like time t, as we said the voltage value y(t) can take any value between 2 limits 0v and 3.3v (in our example), so again there is theoretically an infinite number of possibilities and sets of values. And for any digital computer or microcontroller there is no data type that can hold a value of infinite precision as all numbers are ultimately represented in binary, so we must evenly divide the values along the voltage axis.

In other words each flat region in the sampled signal is "rounded-off" to the nearest member of a set of discrete values (e.g., nearest integer) This process is called "Quantization". For the example shown in figure 10.5, we chose to divide the range into 8 levels to encode the value in 3 bits. So we have the value of binary (000) represents 0v, to know the value of other levels we need to calculate the difference between 2 successive steps, this is as the Step = Reference voltage / (2^n) Where reference voltage is the maximum
voltage the ADC can convert, n is the number of bits of the digital representation.

Quantization



Audio Sampling

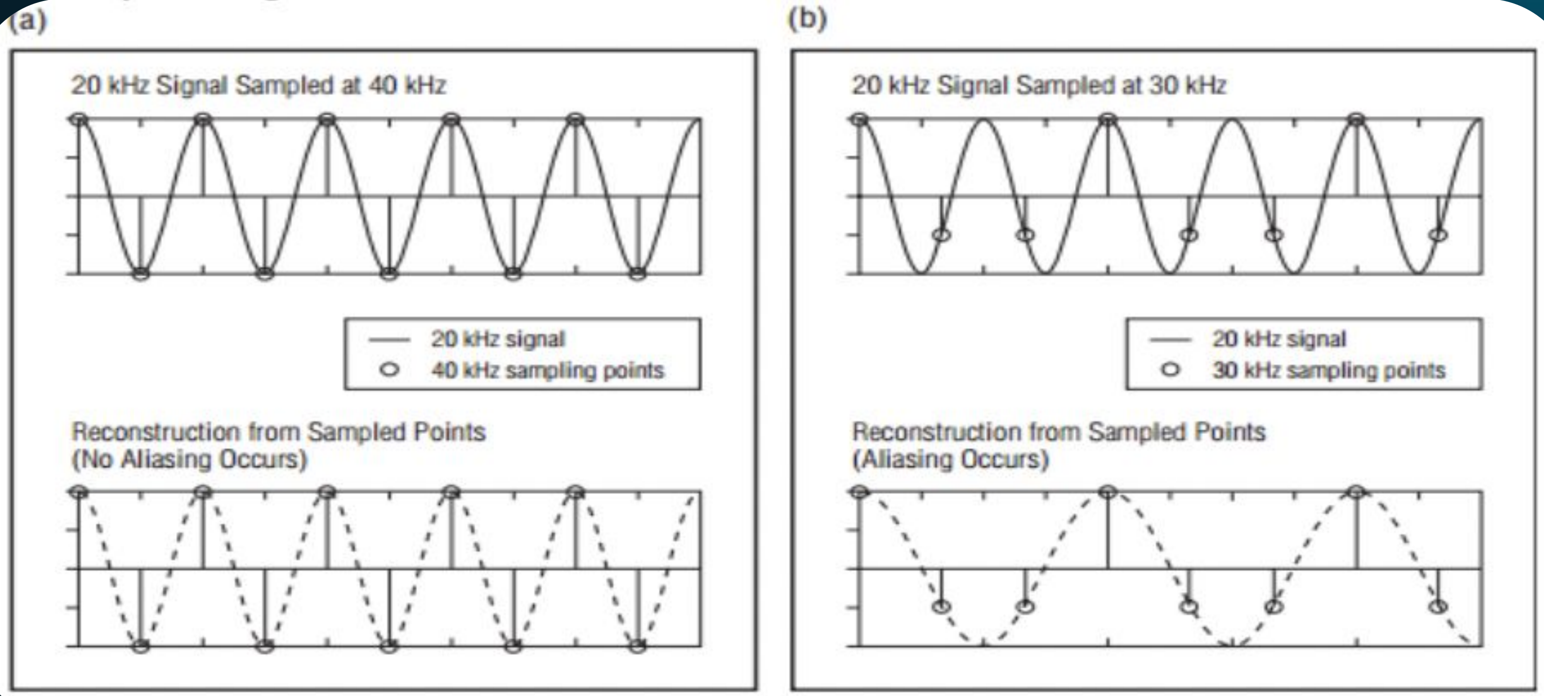
All A/D and D/A conversions should obey the Shannon-Nyquist sampling theorem, as we discussed before, in order for it to be reconstructed in the eventual D/A conversion.

Sampling below the Nyquist sampling rate will introduce aliases, which are low frequency “ghost” images of those frequencies that fall above the Nyquist frequency.

If we take a sound signal that is band-limited to 0–20 kHz, and sample it at 2 × 20 kHz = 40 kHz, then the Nyquist Theorem assures us that the original signal can be

reconstructed perfectly without any signal loss. However, sampling this 0–20 kHz band-limited signal at anything less than 40 kHz will introduce distortions due to aliasing.

Audio Sampling



(a)
20 kHz Signal Sampled at 40 kHz
20 kHz signal
40 kHz sampling points
Reconstruction from Sampled Points
(No Aliasing Occurs)
(b)
20 kHz Signal Sampled at 30 kHz
20 kHz signal
30 kHz sampling points
Reconstruction from Sampled Points
(Aliasing Occurs)

Audacity 03

Audible Signal

Audacity

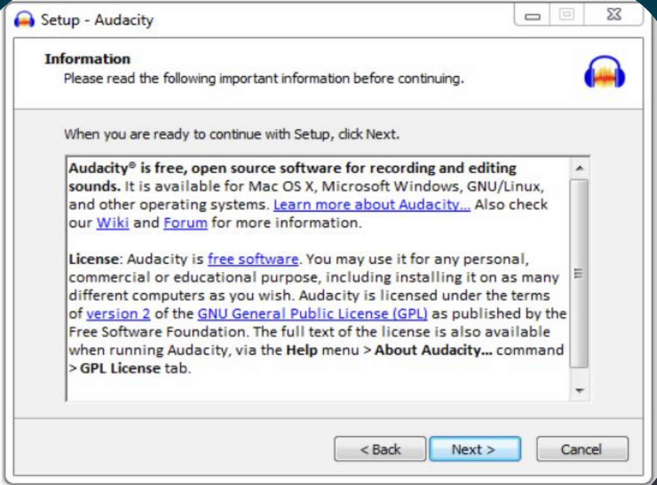
Audacity is an easy-to-use, multi-track audio editor and recorder for Windows, macOS, GNU/Linux and other operating systems.

Developed by a group of volunteers as open source. We have chosen Audacity among other alternatives as it is a free and open source software.

We will use Audacity software to sample our audio file, choose our suitable configuration options, generate the output file that will be used to store our audio in our microcontroller flash memory.

Audacity Setup

1. Double click on the installation executable file , I.e. audacity-win-2.2.2.exe
2. Allow installing the software by clicking yes on the user account control if it asks.
3. Select setup language
4. Click next on the welcome window
5. After reading the license info, click next
6. Choose the destination location to install then click next.
7. Click next on the additional options window
8. Click install on the "Ready to install window"
9. Wait until setup is complete
10. Click next on the after installation window
11. Click finish to exit installation and launch Audacity


Setup - Audacity
Information
Please read the following important information before continuing.
< Back
Next >
Cancel

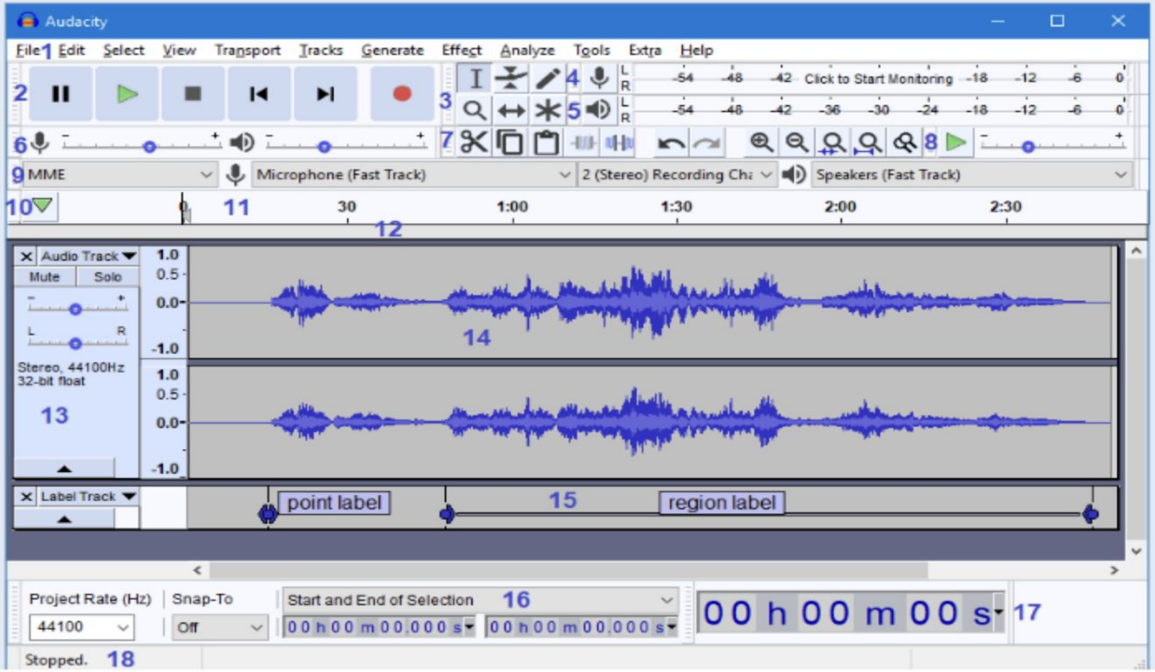
When you are ready to continue with Setup, click Next.

Audacity® is free, open source software for recording and editing sounds. It is available for Mac OS X, Microsoft Windows, GNU/Linux, and other operating systems. Learn more about Audacity... Also check our Wiki and Forum for more information.

License: Audacity is free software. You may use it for any personal, commercial or educational purpose, including installing it on as many different computers as you wish. Audacity is licensed under the terms of version 2 of the GNU General Public License (GPL) as published by the Free Software Foundation. The full text of the license is also available when running Audacity, via the **Help** menu > **About Audacity...** command > **GPL License** tab.

Audacity Stating Window

1 Menu Bar 2 Transport Toolbar 3 Tools Toolbar 4 Recording Meter Toolbar 5 Playback Meter Toolbar 6 Mixer Toolbar
7 Edit Toolbar 8 Play-at-Speed Toolbar 9 Device Toolbar 10 Unpinned Play/Recording Head 11 Timeline 12 Scrub Ruler
13 Track Control Panel 14 Audio Track 15 Label Track 16 Selection Toolbar 17 Time Toolbar 18 Status Bar



Prepare Audio File

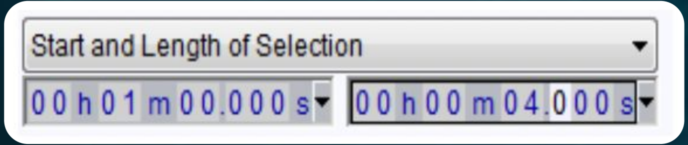
1- Adding the audio file :-

Firstly we need an audio file to work on, as we said we can record our voice using Audacity, so this can be our audio file source, or we can use any audio file we have like a song or a speech, this choice will be let to you, choose whatever the audio file you want then add it to Audacity using drag and drop or through File -> open After adding the file audio track will be shown at the audio area in the Audacity window.

2- Select and cut :-

Recall the we decided at the end of point 11.4 that we would choose a rate of 8000 sample/sec for our audio files, and for out 8-bit DAC the sample size is 8 bits, so every one second of our audio will consume 8000 Byte size, recall also that our microcontroller's flash memory total size is 64KB, this will be used to store the whole program, some data and our audio, so we don't have the luxury of producing a multiple-minute audio file, we will make a 4-second audio file. So here we select a small part of the audio file to work on.

This can be done easily by choosing the "Start and Length of Selection" from the selection toolbar



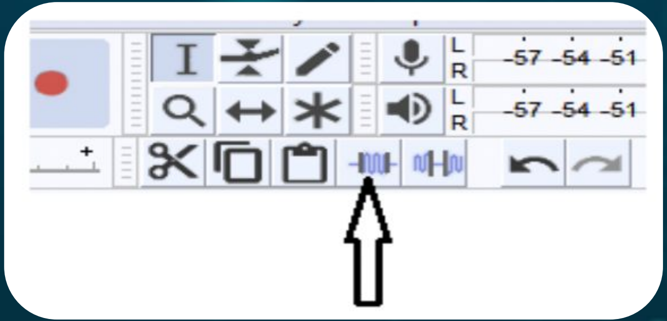
Start and Length of Selection

00 h 01 m 00.000 s   00 h 00 m 04.000 s

Prepare Audio File

Then you can choose the starting point in minutes and seconds from the left part and the length to be 4 seconds from the right part.

Now our 4-second audio is selected, we need to remove the whole other audio except our selection, this can be done by pressing on the "Trim audio outside selection" tool from the toolbar, or simply by pressing "Ctrl+T".



3- Define Project Rate :-

This is the sampling frequency of the audio file,
we would choose 8 KHz sampling frequency,
this can be set by the project rate configuration at the bottom-left most of the window,
set the rate to 8000 Hz.

Prepare Audio File

4- Choose Mono Or Stereo:-

Stereo systems are capable of creating the impression of sound source localization. Sound source localization refers to the human ability to locate the position of a sound source within a space. Mono playback systems use one speaker and can only produce a two dimensional image consisting of height and depth. Two speakers are required to create the directional timing differences that your brain needs to perceive width.

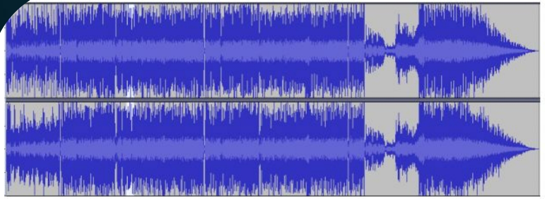


Figure 11.11 A stereo audio file

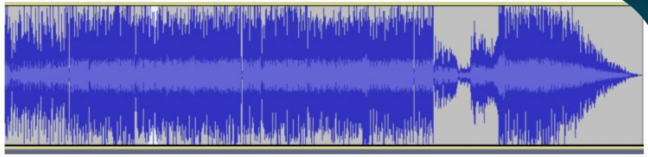
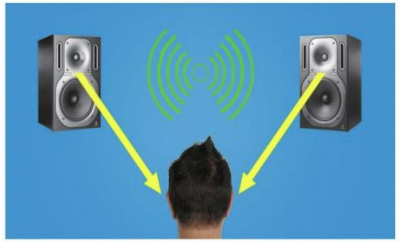
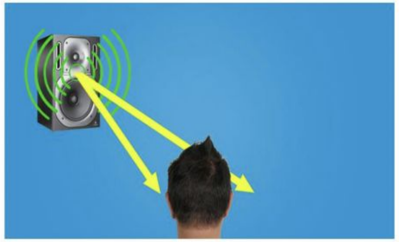


Figure 11.12 A mono audio file

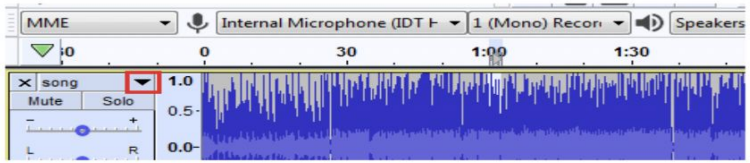




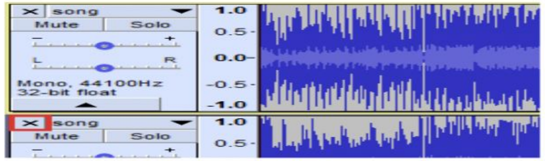
Prepare Audio File

Of course we can deduce that for a better audio listening experience we would choose stereo signal, but for a smaller size audio we would use the mono signal.
We will change our signal into mono because our audio amplifier on the KIT is a mono audio amplifier, but you can try the stereo if you have your own stereo audio amplifier.
Probably the audio file you open in Audacity is stereo, so if you want a stereo audio leave it as it is, but if you want a mono audio file follow.

1. Click the down arrow on the track to open the menu as shown in.



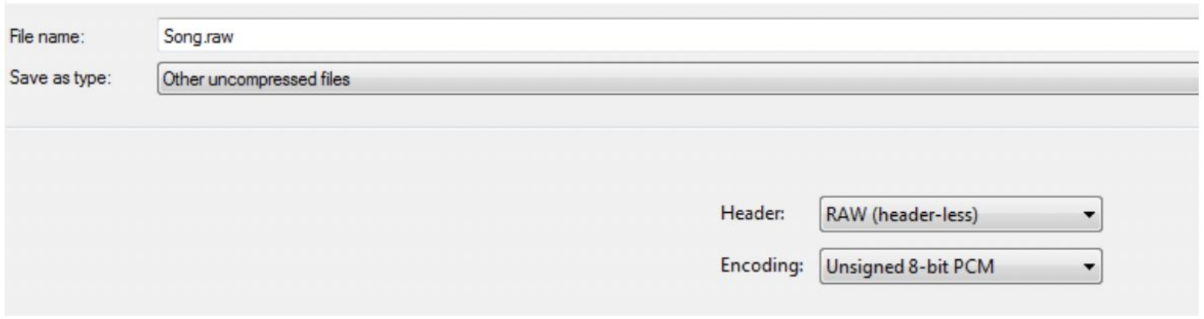
2. In the menu that drops down, select ***Split Stereo to Mono***.
3. Click X to delete one of the two track



Prepare Audio File

5- Export the output :-

From File window -> Export -> Export selected audio
In the Save as type bar, choose: Other uncompressed files
In the header bar at the bottom, choose: RAW (header-less)
In the Encoding bar at the bottom, choose: Unsigned 8-bit PCM

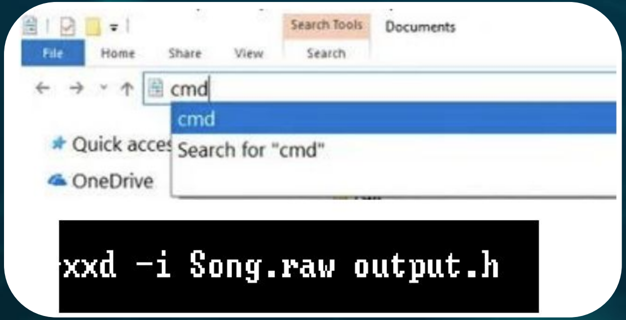


File name: Song.raw
Save as type: Other uncompressed files
Header: RAW (header-less)
Encoding: Unsigned 8-bit PCM

Now we have a ".raw" file as an output, its size is 32KB, now it is ready for the next step

Prepare Audio File

Here we are interested in converting our ".raw" file into an array of the digital representation of each sample, this array that will be stored into our microcontroller memory and played back via our DAC and speaker. To achieve this mission we will use a Linux-tool called "xxd", which can run on windows also. You can find the xxd.exe file in the flash memory accompanied with IMT Kit or you can download it from the internet. xxd creates a hex dump of a given file or standard input. It can also convert a hex dump back to its original binary form. It allows the transmission of binary data in a `mail-safe' ASCII representation, but has the advantage of decoding to standard output. Moreover, it can be used to perform binary file patching. Don't open the xxd.exe or try to install, just copy it to the same location of the ".raw" file, so that the 2 files are in the same folder. Open the command window at this folder, this can be done by multiple ways, the easiest way is by writing "cmd" in the folder path at the top of the page then press



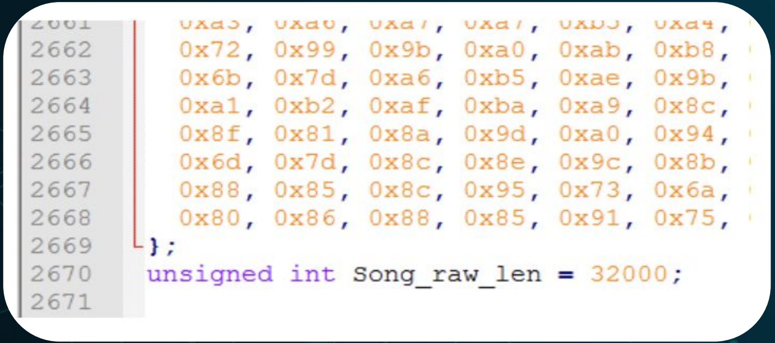
```
xxd -i Song.raw output.h
```

Now we have the audio array, where each element in the array represents the digital value a sample of our audio, these samples are now ready to play back by our DAC.

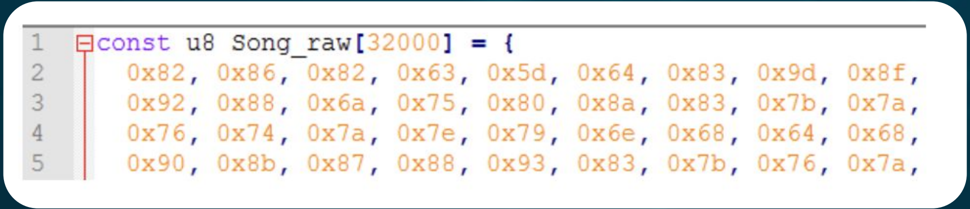
Save the Audio in Flash

Target of this point is to store the output array we have in the microcontroller flash memory, but how can we do this?

1. Get the size of the array from the last line at the most bottom of the file, if you followed the same previous steps, you would probably find it about 32000.



```
    0xa5, 0xa6, 0xa7, 0xa7, 0xb5, 0xa4,
    0x72, 0x99, 0x9b, 0xa0, 0xab, 0xb8,
    0x6b, 0x7d, 0xa6, 0xb5, 0xae, 0x9b,
    0xa1, 0xb2, 0xaf, 0xba, 0xa9, 0x8c,
    0x8f, 0x81, 0x8a, 0x9d, 0xa0, 0x94,
    0x6d, 0x7d, 0x8c, 0x8e, 0x9c, 0x8b,
    0x88, 0x85, 0x8c, 0x95, 0x73, 0x6a,
    0x80, 0x86, 0x88, 0x85, 0x91, 0x75,
};
unsigned int Song_raw_len = 32000;
```



```
const u8 Song_raw[32000] = {
    0x82, 0x86, 0x82, 0x63, 0x5d, 0x64, 0x83, 0x9d, 0x8f,
    0x92, 0x88, 0x6a, 0x75, 0x80, 0x8a, 0x83, 0x7b, 0x7a,
    0x76, 0x74, 0x7a, 0x7e, 0x79, 0x6e, 0x68, 0x64, 0x68,
    0x90, 0x8b, 0x87, 0x88, 0x93, 0x83, 0x7b, 0x76, 0x7a,
```

2. Copy this size, then delete the variable definition line.

3. Goto the first line of the file then redefine the array as constant unsigned char (or u8 as we used to use in our standard types) with the size we've just got.

Audible Signal Reconstruction

In the previous points we've chosen to make the audio rate to be of 8000 Hz, this means that the sampling tool takes 8000 sample per second, which can be interpreted to get a sample every 125 microseconds. But why we should be interested now in this info?

The answer is because we are now trying to reconstruct the sampled signal and play it back, so we should play it back with the same sampling frequency to keep the sound characteristics especially the sound sharpness as it is, otherwise the output signal wouldn't be as the input one.

Simple execution of this requirement is using a timer, the timer generates an interrupt every 125 microseconds, and in the ISR we send a sample to the DAC.

IMT
SCHOOL

STM32
Is AWESOME

IMT SCHOOL

THANKS!

Do you have any questions?

www.imtschool.com

www.facebook.com/imaketechnologyschool/