arm Education

L.EEC025 - Fundamentals of Signal Processing (FunSP)

2022/2023 – 1st semester

Week03, 26 Sep 2022

Objectives:

-getting started with the DSP Education kit (2nd part)

- **generating sinusoids from a LUT**
- **viewing program output**

DSP Education Kit

LAB 2

LUT sinusoid generation and viewing Program Output

Issue 1.0

Contents

1 Introduction

1.1 Lab overview

The STM32F746G Discovery board is a low-cost development platform featuring a 212 MHz Arm Cortex-M7 floating-point processor. It connects to a host PC via a USB A to mini-b cable and uses the ST-LINK/V2 in-circuit programming and debugging tool. The Keil MDK-Arm development environment, running on the host PC, enables software written in C to be compiled, linked, and downloaded to run on the STM32F746G Discovery board. Real-time audio I/O is provided by a Wolfson WM8994 codec included on the board.

This laboratory exercise introduces the use of the STM32F746G Discovery board and several of the procedures and techniques that will be used in subsequent laboratory exercises.

2 Requirements

To carry out this lab, you will need:

- An STM32F746G Discovery board
- A PC running Keil MDK-Arm
- MATLAB
- An oscilloscope
- Suitable connecting cables
- An audio frequency signal generator
- Optional: External microphone, although you can also use the microphones on the board

2.1.1 STM32F746G Discovery board

An overview of the STM32F746G Discovery board can be found in the Getting Started Guide (previous PL class).

3 Basic Digital Signal Processing System

A basic DSP system that is suitable for processing audio frequency signals comprises a digital signal processor and analogue interfaces as shown in Figure 1. The STM32F746G Discovery board provides such a system, using a Cortex-M7 floating point processor and a WM8994 codec.

The term codec refers to the *coding* of analogue waveforms as digital signals and the *decoding* of digital signals as analogue waveforms. The WM8994 codec performs both the Analogue to Digital Conversion (ADC) and Digital to Analogue Conversion (DAC) functions shown in Figure 1.

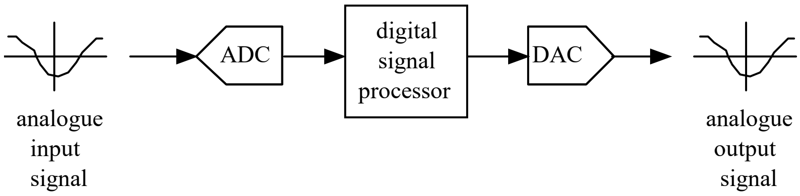

Figure 1: Basic digital signal processing system

Program code may be developed, downloaded, and run on the STM32F746G Discovery board using the *Keil MDK-Arm* integrated development environment. You will not be required to write C programs from scratch, but you will learn how to compile, link, download, and run the example programs provided, and in some cases, make minor modifications to their source files.

You will learn how to use a subset of the features provided by MDK-Arm in order to do this (using the full capabilities of MDK-Arm is beyond the scope of this set of laboratory exercises). The emphasis of this set of laboratory exercises is on the digital signal processing concepts implemented by the programs.

Most of the example programs are quite short, and this is typical of real-time DSP applications. Compared with applications written for general purpose microprocessor systems, DSP applications are more concerned with the efficient implementation of relatively simple algorithms. In this context, efficiency refers to speed of execution and the use of resources such as memory.

The examples in this document introduce some of the features of *MDK-Arm* and the STM32F746G Discovery board. In addition, you will learn how to use *MATLAB in* order to analyze audio signals.

4 Real-Time Sine Wave Generation

4.1 Program operation

Consider the C source file `stm32f7_sine_lut_intr_FunSP.c`, shown in the code snippet below, which is a modified version of the `stm32f7_sine_lut_intr.c` already tested in the previous PL class. The `stm32f7_sine_lut_intr_FunSP.c` source file is available on the Moodle platform.

```
// stm32f7_sine_lut_intr_FunSP.c

#include "stm32f7_wm8994_init.h"
#include "stm32f7_display.h"

#define SOURCE_FILE_NAME "stm32f7_sine_lut_intr_FunSP.c"
#define LOOPLENGTH 8

extern int16_t rx_sample_L;
extern int16_t rx_sample_R;
extern int16_t tx_sample_L;
extern int16_t tx_sample_R;

int16_t sine_table[LOOPLENGTH] = {0, 7071, 10000, 7071, 0, -7071, -10000, -7071};
int16_t sine_ptr_L = 0; int16_t sine_ptr_R = LOOPLENGTH/4;  // pointers into
lookup table

void BSP_AUDIO_SAI_Interrupt_CallBack()
{
// when we arrive at this interrupt service routine (callback)
// the most recent input sample values are (already) in global variables
// rx_sample_L and rx_sample_R
// this routine should write new output sample values in
// global variables tx_sample_L and tx_sample_R

  BSP_LED_On(LED1);
  tx_sample_L = sine_table[sine_ptr_L];
  tx_sample_R = sine_table[sine_ptr_R];
  sine_ptr_L = (sine_ptr_L+1)%LOOPLENGTH;
  sine_ptr_R = (sine_ptr_R+1)%LOOPLENGTH;
  tx_sample_R *= tx_sample_L;
  BSP_LED_Off(LED1);
  return;
}

int main(void)
{
  stm32f7_wm8994_init(AUDIO_FREQUENCY_8K,
                      IO_METHOD_INTR,
                      INPUT_DEVICE_INPUT_LINE_1,
                      OUTPUT_DEVICE_HEADPHONE,
                      WM8994_HP_OUT_ANALOG_GAIN_0DB,
```

```
                      WM8994_LINE_IN_GAIN_0DB,
                      WM8994_DMIC_GAIN_9DB,
                      SOURCE_FILE_NAME,
                      GRAPH);
  plotSamples(sine_table, LOOPLENGTH, 32);
  while(1){}
}
```

This code generates two output sinewaves that are different in their frequencies and amplitudes.

Question 1 [3pt / 10]: Compare the two C source files and explain: what are the analytical expressions that describe the two sinusoids that can be observed on the output LEFT and RIGHT channels ?

Question 2 [2pt / 10]: Copy the new `stm32f7_sine_lut_intr_FunSP.c` source file to the source code directory:

C:\uvision\Keil\STM32F7xx_DFP\2.9.0\Projects\STM32746G-Discovery\Examples\DSP Education Kit\Src

and proceed, as explained in Section 4.2 of guide of the previous PL class, to replace in the Project, the existing `main()` source file, by the current one (`stm32f7_sine_lut_intr_FunSP.c`), and then proceed to compile the new code (i.e., to build the Project) and to download it to the STM32F7 Kit (by starting the debug session and then pressing "Run").

When you take the two output analog LEFT and RIGHT channels of the STM32F7 kit to the inputs CHAN1 and CHAN2 of the oscilloscope, do you observe the waves you expect (as in Question 1) ? If not, why?

4.2 Viewing program output using MATLAB (sinusoid)

To view your program output in Matlab, you can first store the output values into a file and then use Matlab to load the values from the saved file.

`stm32f7_sine_lut_buf_intr.c` shows how to store the output values, it is very similar to program `stm32f7_sine_lut_intr.c`, but it also stores the most recent `BUFFER_LENGTH` number of output values in the array `buffer`. Array `buffer` is of type `float32_t` for compatibility with the *MATLAB* function that will be used to view its contents.

To save the program output into a file and view them in Matlab, follow these steps:

1. Run the program and press the user button to start the program.
2. Halt it by clicking on the ***Stop*** toolbar button in the MDK_Arm debugger.

3. Type the variable name **`buffer`** as the ***Address*** in the debugger's ***Memory 1*** window. Right-click on the ***Memory 1*** window and set the displayed data type to ***Decimal*** and ***Float*** as shown in Fig. 2.

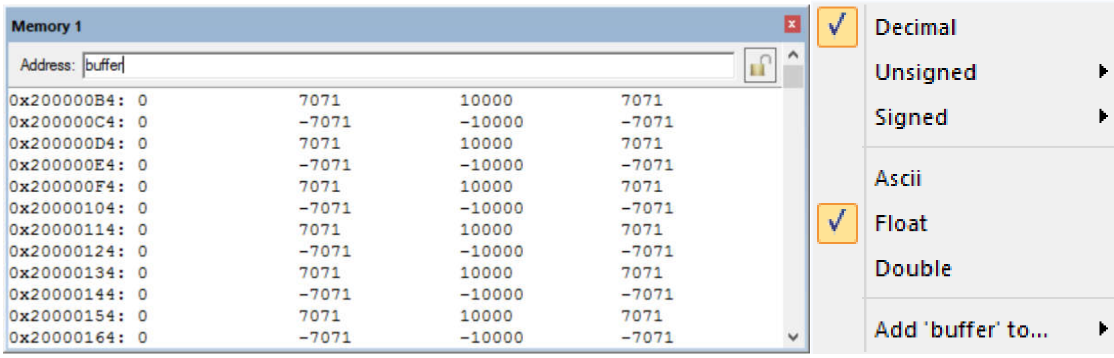

Figure 2: Memory 1 window showing the contents of array `buffer`

The start address of array `buffer` will be displayed in the top left-hand corner of the window.

4. Use the following command at the prompt in the debugger's ***Command*** window to save the contents of the **`buffer`** array to a file in your project folder.

   `SAVE <filename> <start address>, <end address>`

   The end address should be the start address plus 0×190 (bytes) representing 100 32-bit sample values. For example,

   `SAVE sinusoid.dat 0x200000B4, 0x20000244`

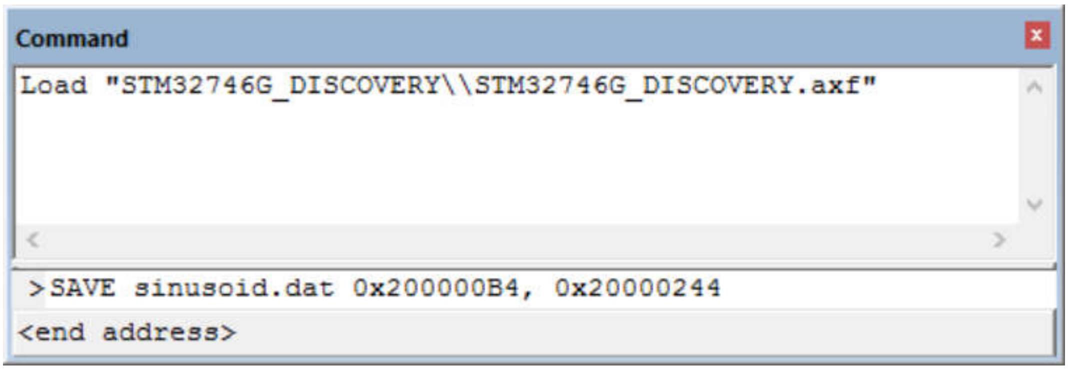

Figure 3: Saving data to file in MDK-Arm

5. Launch *MATLAB* and run the *MATLAB* function `stm32f7_logfft.m` (provided with the DSP Education Kit in **General_Matlab_Files**) to obtain a graphical representation of the contents of the buffer. The *MATLAB* function will require you to input some information, such as the saved `.dat` filename (full path) and sampling frequency.

NOTE: Matlab function `stm32f7_logfft.m` is available on Moodle. You also need function `hexsingle2num.m` which should be co-located with `stm32f7_logfft.m` .

Question 3 [3pt / 10]: When you view the contents of `sinusoid.dat` using the Matlab command file `stm32f7_logfft.m`, you observe a "discontinuity" similar to that illustrated in Fig. 4. How do you explain that "discontinuity" ? Indicate one possible alternative value for `BUFFER_LENGTH` that is closest to 100 and that avoids that "discontinuity".

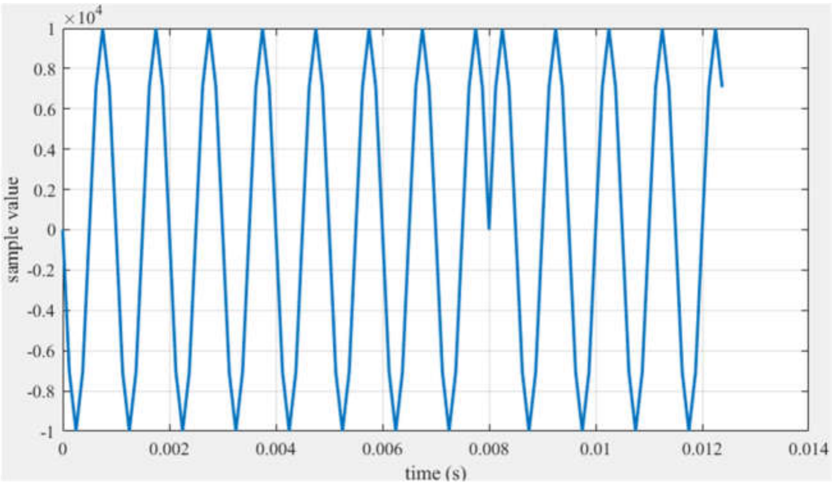

Figure 4: Saving data to file in MDK-Arm

4.3 Viewing program output using MATLAB (noise)

Repeat 4.2 but this time using the provided C file named `stm32f7_sine_lut_buf_intr_FunSP.c`, which you should copy to the `Examples\DSP Education Kit\Src` directory (in case it is not yet there). This code is a modified version of the previous code in the sense that it calls function `prand()` that generates pseudorandom sample values using the Park-Miller algorithm (a random number generator). Thus, the left channel outputs a sinusoid whereas the right channel outputs noise.

```
// stm32f7_sine_lut_buf_intr_FunSP.c

#include "stm32f7_wm8994_init.h"
#include "stm32f7_display.h"

#define SOURCE_FILE_NAME "stm32f7_sine_lut_buf_intr_FunSP.c"
#define LOOPLENGTH 8
#define BUFFER_LENGTH 1000 // was 100

extern int16_t rx_sample_L;
extern int16_t rx_sample_R;
extern int16_t tx_sample_L;
extern int16_t tx_sample_R;

int16_t sine_table[LOOPLENGTH] = {0, 7071, 10000, 7071, 0, -7071, -10000, -7071};
int16_t sine_ptr = 0;  // pointer into lookup table
```

```
float32_t buffer[BUFFER_LENGTH];
int16_t buf_ptr = 0;   // pointer into buffer

void BSP_AUDIO_SAI_Interrupt_CallBack()
{
// when we arrive at this interrupt service routine (callback)
// the most recent input sample values are (already) in global variables
// rx_sample_L and rx_sample_R
// this routine should write new output sample values in
// global variables tx_sample_L and tx_sample_R

  tx_sample_L = sine_table[sine_ptr];
  tx_sample_R = prand();
      //tx_sample_R = tx_sample_L;
  buffer[buf_ptr] = tx_sample_R;
      // buffer[buf_ptr] = tx_sample_L;
      sine_ptr = (sine_ptr+1)%LOOPLENGTH;
      buf_ptr = (buf_ptr+1)%BUFFER_LENGTH;

  BSP_LED_Toggle(LED1);

  return;
}


int main(void)
{
  stm32f7_wm8994_init(AUDIO_FREQUENCY_8K,
                      IO_METHOD_INTR,
                      INPUT_DEVICE_DIGITAL_MICROPHONE_2,
                      OUTPUT_DEVICE_HEADPHONE,
                      WM8994_HP_OUT_ANALOG_GAIN_6DB,
                      WM8994_LINE_IN_GAIN_0DB,
                      WM8994_DMIC_GAIN_9DB,
                      SOURCE_FILE_NAME,
                      GRAPH);
      plotSamples(sine_table, LOOPLENGTH, 32);
  while(1){}
}
```

Replace the code file in Section 4.2 by this new file (`stm32f7_sine_lut_intr_FunSP.c`) and build and run the program again.

Repeat the above procedure to save the contents of array **buffer** to a file (`noise.dat`) in your project folder.

> **SAVE <filename> <start address>, <end address>**
>
> The end address should be the start address plus 0xFA0 (bytes) representing 1000 32-bit sample values. For example,
>
> `SAVE noise.dat 0x200000B4, 0x20001054`

Question 4 [2pt / 10]: Proceed to repeat the above file analysis by running the *MATLAB* function `stm32f7_logfft.m` this time with `noise.dat`. Does the observed waveform look like one would expect ?

The following is to be carried **out of the class** (and is not considered for grading): how would you characterize the noise that is generated by the DSP Education kit pseudorandom generator ? Adapt the Matlab code that is suggested in Problem 1 of this week's Extra Exercises to check its auto-correlation function and its PDF (Probability Density Function).

5 Conclusions

At the end of this exercise, you should have become familiar with a simple procedure to import to Matlab data that is generated in the DSP Education Kit. This will be used subsequent lab exercises.

6 Additional References

Link to Board information and resources:

https://www.st.com/en/evaluation-tools/32f746gdiscovery.html#overview