arm Education

L.EEC025 - Fundamentals of Signal Processing (FunSP)

2023/2024 – 1st semester

Week03, 25 Sep 2023

Objectives:

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

- **generating sinusoids from a LUT**

DSP Education Kit

LAB 2

LUT-based sinusoid generation

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

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 ***cod****ing* of analogue waveforms as digital signals and the ***dec****oding* 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.

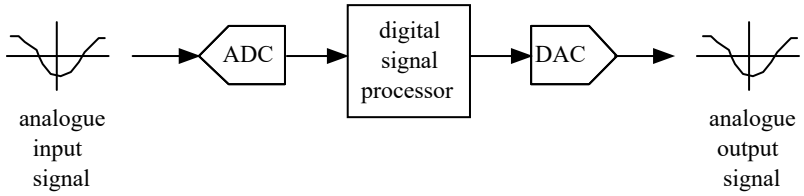

Figure 1: Basic digital signal processing system

Program code may be developed, downloaded, and run on the STM32F746G Discovery board using the *Keil MDK-Arm* integrated development environment (**IDE**). 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
float temp;

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;
  temp = tx_sample_L * tx_sample_R / 10000.0;
  // tx_sample_R = (int16_t)temp;
  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){}
}
```

Question 1 [2pt / 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 analogue 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?

Now, uncomment the following code line:

```
  // tx_sample_R = (int16_t)temp;
```

such that it becomes:

```
  tx_sample_R = (int16_t)temp;
```

Question 3 [2pt / 10 **]**: Explain analytically the impact of this instruction and write the analytical expressions that describe the two sinusoids that should be observed on the output LEFT and RIGHT channels.

Question 4 [2pt / 10 **]**: Now, proceed to recompile the modified code, to download the corresponding object code to the STM32F7 Kit, and to run it in real-time.

When you take the two output analogue 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 3) ? If not, why?

Question 5 [2pt / 10 **]**: Consider the following code modifications and answer the following questions.

Comment again the above code line such that it becomes:

```
  // tx_sample_R = (int16_t)temp;
```

now, modify two lines in the above code such that they become:

```
#define LOOPLENGTH 6  // was 8
```

…

```
int16_t sine_table[LOOPLENGTH] = {0,  8660,  8660,  0,  -8660,  -8660};
```

…

```
  int16_t sine_ptr_R = 3;
```

What waveforms do you expect to observe on the two DSK32F7 analogue outputs ? How should they relate to each other ? What do you expect their frequency to be ?

Now, proceed to recompile the modified code (as just indicated), to download the corresponding object code to the STM32F7 Kit, and to run it in real-time. When you take the two output analogue LEFT and RIGHT channels of the STM32F7 kit to the inputs CHAN1 and CHAN2 of the oscilloscope, do you observe the expected waveforms and frequency ? If not, why?

5 Conclusions

At the end of this exercise, you should have become familiar with a simple procedure to generate sinusoids with different amplitudes and frequencies using a lookup table.

6 Additional References

Link to Board information and resources:

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