B39SB 2019 Lab Exercise #2:   
ANALOGUE INPUT AND OUTPUT

# Overview

The examples in this exercise concern the characteristics of the WM8731 codec (analogue to digital converter (ADC) and digital to analogue converter (DAC)) used on the FM4 board. The effect of sampling rate on the bandwidth of a digital signal processing system is examined and the phenomenon of aliasing is demonstrated.

# Details

## Hardware

To carry out this exercise you will need a Cypress FM4 board, an oscilloscope, an audio frequency signal generator, a PC running *Keil MDK-ARM* and M*ATLAB*, and suitable connecting cables.

### FM4 Starter Kit

The *FM4 Starter Kit* includes a WM8731 stereo audio codec, which is accessed via I2C (for control) and I2S (for data) interfaces. Analogue input and output signals are accessible via three-pole 3.5mm jack sockets (LINE IN, MIC\_IN and HEADPHONES\_OUT).

As configured for these exercises, the WM8731 converts an analogue input signal into 16-bit signed integer sample values and the DAC converts 16-bit signed integer sample values into an analogue output signal.

### SAMPLING and aliasing – generating sinusoids of arbitrary frequency

Consider program fm4\_sine\_intr.c, listed in figure 1. This program generates sinusoidal analogue output waveforms via the WM8731 codec using calls to the function sin\_f32(). The program uses interrupt-based i/o and its sampling rate is set to 8 kHz.

Suppose that the output sinusoid frequency is set to 1 kHz (frequency = 1000.0). As described in exercise #1, at each sampling instant the program statements in interrupt service routine function PRGCRC\_I2S\_IRQHandler() are executed. Here, the value of the variable theta is updated and a sample value equal to the sine of that angle (theta) is computed using function arm\_sin\_f32() and written to the DAC. The DAC *reconstructs* a continuous-time signal from the discrete-time sample values that are written to it and in this case, the reconstructed signal is a sinusoidal waveform with a frequency of 1 kHz, as shown in figure 2.

// fm4\_sine\_intr.c

#include "fm4\_wm8731\_init.h"

#define SAMPLING\_FREQ 8000

float32\_t frequency = 1000.0;

float32\_t amplitude = 10000.0;

float32\_t theta\_increment;

float32\_t theta = 0.0;

void PRGCRC\_I2S\_IRQHandler(void)

{

union WM8731\_data sample;

sample.uint32bit = i2s\_rx();

gpio\_set(DIAGNOSTIC\_PIN, HIGH);

theta\_increment = 2\*PI\*frequency/SAMPLING\_FREQ;

theta += theta\_increment;

if (theta > 2\*PI) theta -= 2\*PI;

// sample.uint16bit[LEFT] = (int16\_t)(amplitude\*sin(theta));

// sample.uint16bit[LEFT] = (int16\_t)(amplitude\*sinf(theta));

sample.uint16bit[LEFT] = (int16\_t)(amplitude\*arm\_sin\_f32(theta));

sample.uint16bit[RIGHT] = sample.uint16bit[LEFT];

gpio\_set(DIAGNOSTIC\_PIN, LOW);

i2s\_tx(sample.uint32bit);

NVIC\_ClearPendingIRQ(PRGCRC\_I2S\_IRQn);

}

int main(void)

{

fm4\_wm8731\_init (FS\_8000\_HZ, WM8731\_MIC\_IN,IO\_METHOD\_INTR,

WM8731\_HP\_OUT\_GAIN\_0\_DB, WM8731\_LINE\_IN\_GAIN\_0\_DB);

while(1){}

}

Figure 1: Listing of program fm4\_sine\_intr.c.

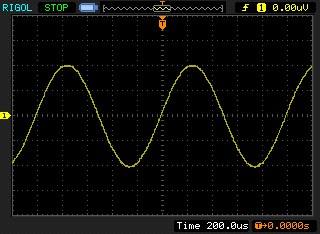


Figure 2: Analogue output waveform generated using program fm4\_sine\_intr.c with value of variable frequency equal to 1000.0.

Change the value of the variable frequency in program fm4\_sine\_intr.c according to table 1 and record the frequency of the output signal you observe on an oscilloscope connected to the right channel of HP OUT on the FM4 board.

|  |  |
| --- | --- |
| Value assigned to variable frequency | Frequency of analogue output signal (Hz) |
| 1500 |  |
| 2573 |  |
| 7000 |  |
| 3500 |  |
| 4500 |  |

**Table 1: Frequency of analogue output signal for different values of variable frequency in program fm4\_sine\_intr.c.**

**(5 marks)**

Your results should be consistent with the following two observations.

1. Using program fm4\_sine\_intr.c it is possible to generate continuous-time sinusoidal waveforms of arbitrary frequency, e.g. 2573 Hz. The program makes it far easier to change the frequency of the waveform than was the case using pre-computed sample values in program fm4\_sine8\_intr.c in exercise #1.
2. Using a sampling rate of 8 kHz, the WM8731 DAC is incapable of generating a sinusoidal output signal with a frequency greater than 4 kHz

You can compare the computational costs of using alternative function calls sin(), arm\_sin\_f32() and sinf() by commenting out those not used (and rebuilding the project) and using an oscilloscope to measure the duration of the pulses on GPIO pin P10 (DIAGNOSTIC\_PIN).

### square wave generation using the WM8731 Codec

Consider the sample sequence {10000, 10000, 10000, 10000, -10000, -10000, -10000, -10000}. These are regularly-spaced samples of a square wave with a period of 1ms, i.e. a frequency of 1 kHz. Edit program fm4\_sine8\_intr.c, replacing the sample values representing a sinusoid in array sine\_table with the sequence given above. That is, replace the relevant line in the source file with

sine\_table[LOOPLENGTH] = {10000, 10000, 10000, 10000, -10000, -10000, -10000, -10000};

Run the program and sketch the resultant output waveform (viewed using an oscilloscope) on the axes of figure 3. Write down or draw answers the following questions in the spaces provided.

Sketch the analogue output waveform on the axes of figure 3.

****

Figure 3: Analogue output waveform generated using program fm4\_sine8\_intr.c using samples of a 1 kHz square wave.

**(4 marks)**

How would you describe the analogue output WAVEFORM?

**(4 marks)**

Is it a square wave?

**(2 marks)**

Explain why The output waveform has this shape?

**(6 marks)** for considering the frequency-domain characteristics of the DAC reconstructing the waveform from discrete samples and the frequency components present in a) a perfect square wave, and b) in the output waveform observed.

### Step and impulse responses of the WM8731 DAC Reconstruction filter

Program fm4\_square\_intr.c repeatedly writes a data sequence comprising 32 consecutive values of 10000 followed by 32 consecutive values of -10000 to the DAC. These are samples of a 125 Hz square wave and the frequency content of these samples has considerable similarity to the frequency content of a continuous-time 125 Hz square wave. However, when the samples are written to the DAC, the resultant continuous-time output waveform is not square. Run the program and look closely at the output waveform using an oscilloscope.

Sketch the analogue output waveform seen on the oscilloscope on the axes of figure 4 and, from this, deduce the impulse response of the reconstruction filter.

Sketch what you think the impulse response of the reconstruction filter is on the axes of figure 5 and explain how you deduced this in the space below that figure.



Figure 4: Analogue output waveform generated using program fm4\_square\_intr.c.

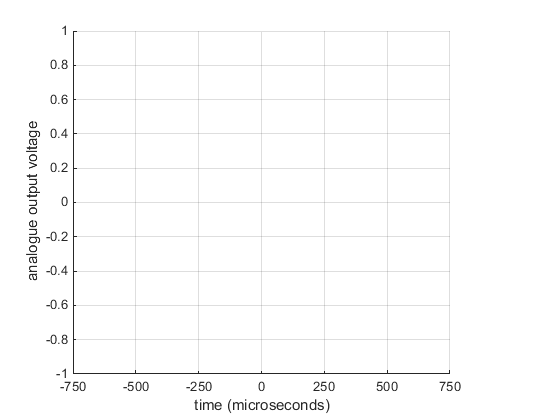


Figure 5: Impulse response of WM8731 DAC reconstruction filter, deduced from waveform shown in figure 4.

**(8 marks**) for the sketch of the output waveform showing clearly its period and symmetrical ‘ringing’ either side of transitions.

**(6 marks)** for a close to sinc-type waveform centred on t = 0 with zero crossings spaced 125 us apart.

**(10 marks)** for explaining that each transition in a square wave may (approximately) be viewed as a step. An impulse is the time derivative of a +ve step. Therefore, the impulse response of the DAC is approximated by the time derivative of one of the +ve transitions in the output waveform of Figure 4.

You can view the *actual* impulse response of the WM8731 DAC using program fm4\_dimpulse\_intr.c or you can use the d/dt function (if available) on an oscilloscope to differentiate the waveform generated using program fm4\_square\_intr.c.

Viewed using the FFT function of an oscilloscope, we can see that the analogue output waveform generated by program fm4\_square\_intr.c comprises a substantially similar pattern of harmonically-related frequency components to that of a 125 Hz square wave, but only up to a frequency of 4 kHz. Higher frequency components (that contribute to the sharper edges of a square wave) are missing. In figure 6, it is apparent that **output from the DAC is negligible at frequencies greater than 4 kHz**.

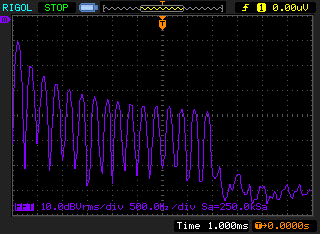


Figure 6: Magnitudes of the frequency components present in the analogue waveform generated using program fm4\_square\_intr.c.

### Magnitude frequency response of the WM8731 DAC Reconstruction filter

You can get a further idea of the magnitude frequency response of the DAC using program fm4\_prbs\_intr.c. This program uses function prbs() to generate a pseudo random binary sequence which, in theory, contains a complete range of different frequency components at equal magnitudes. When this sequence is written to the DAC, the frequency content of the reconstructed analogue output signal reflects the frequency response of the reconstruction filter. Run the program and look at the analogue output signal using the FFT function of an oscilloscope.

Sketch the magnitude frequency response you observe on the axes of figure 7.



Figure 7: Magnitude frequency response of WM8731 DAC reconstruction filter demonstrated using program fm4\_prbs\_intr.c.

**(4 marks)** for showing a curve that is flat over frequency range 0 to 4 kHz and then rolls off steeply.

Run program fm4\_prbs\_intr.c again, having changed the sampling frequency to 48 kHz.

fm4\_wm8731\_init(FS\_48000\_HZ, WM8731\_MIC\_IN,IO\_METHOD\_INTR,

WM8731\_HP\_OUT\_GAIN\_0\_DB, WM8731\_LINE\_IN\_GAIN\_0\_DB);

and note the bandwidth of the noise signal generated.

So far we have seen that **the WM8731 DAC cannot generate signal components having frequencies greater than half its sampling frequency**. This is true no matter howwe produce output sample sequences. It follows that it is inadvisable to allow analogue input signal components having frequencies greater than half the sampling frequency to be sampled. This can be achieved by passing analogue input signals through a low-pass *antialiasing* filter prior to sampling by the ADC. An oversampling digital antialiasing filter with characteristics similar to those of the reconstruction filter in the DAC is built in to the ADC in the WM8731 codec.

### STEP REsponse of the WM8731 antialiasing filter

In order to investigate the step response of the antialiasing filter in the WM8731, connect a signal generator to the left channel of the LINE IN socket on the fm4 board. Use a signal generator to give a square wave output of frequency 200 Hz and amplitude 0.3 V. Run program fm4\_loop\_buf\_intr.c, noting that the output signal from HP OUT is not a perfect square wave. Halt the program. In order to view the 128 most recent input sample values read from the ADC, save these to a file, using the command

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

where start address is that of array lbuffer, and end address is equal to (start address + 0x200), and plot the contents of that file using the MATLAB function fm4\_plot\_real(). You should see something similar to the display shown in figure 9 (bar graph display to emphasise discrete nature of sample values). Figure 8 shows the square wave input signal that produced the display of figure 9 and also the corresponding analogue output signal. The variations in the values of the input samples read from the ADC are due to the low-pass characteristic of the antialiasing filter. Compare figure 9 with the step response of the reconstruction filter that you sketched in figure 5.

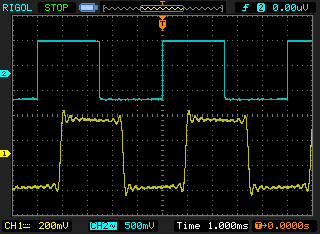


Figure 8: Analogue input and output waveforms observed using program fm4\_loop\_buf\_intr.c.

Figure 9: Input sample values read from the ADC using program fm4\_loop\_buf\_intr.c when the analogue input signal is a 200 Hz square wave.

### Magnitude frequency response of the WM8731 Antialiasing filter

The low pass characteristic of the WM8731 antialiasing filter can further be demonstrated, again using program fm4\_loop\_buf\_intr.c. Adjust the signal generator to give a sinusoidal output. Run program fm4\_loop\_buf\_intr.c for a few seconds, and plot the contents of array lbuffer (the 128 most recent input samples read from the ADC) using MATLAB just as in the previous example. Alternatively, just watch the updated contents of lbuffer in a *Memory* window to get an idea of the magnitudes of the samples read from the ADC. Repeat this procedure for a number of different sinusoid frequencies. You should find that for frequencies above 4 kHz the output of the analogue to digital converter, stored in lbuffer, is effectively zero.

### Estimating WM8731 codec bandwidth using an adaptive filter

Another way of observing the limited bandwidth of the codec is to measure its magnitude frequency response using program fm4\_sysid\_CMSIS\_intr.c. You need not understand exactly how program fm4\_sysid\_CMSIS\_intr.c works in order to use it. Effectively, it identifies the characteristics of the path between its discrete-time output and its discrete-time input (points A and B in figure 10) using an adaptive FIR filter.

Connect HEADPHONE OUT to LINE IN on the audio card using a 3.5mm jack to 3.5mm jack cable. The signal path that will be identified by program fm4\_sysid\_CMSIS\_intr.c comprises the series combination of the digital to analogue and analogue to digital converters.

FM4 Starter Kit

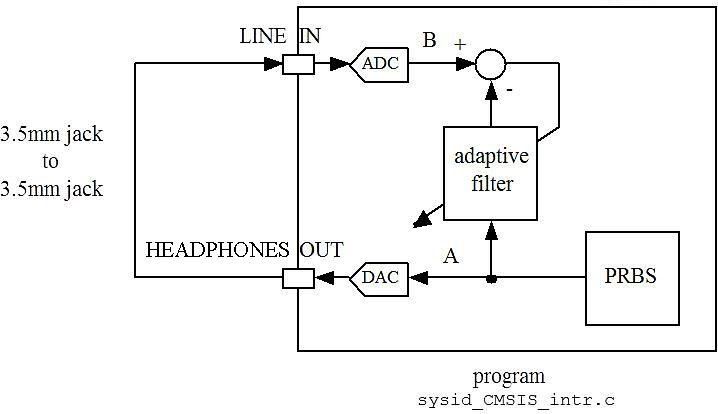


Figure 10: Connection diagram for WM8731 codec bandwidth identification using program fm4\_sysid\_CMSIS\_intr.c.

Run program fm4\_sysid\_CMSIS\_intr.c for several seconds and then halt it. Save the values of the adaptive filter coefficients firCoeffs32 to a file using the command

SAVE <filename.dat> <start address>, <end address>

where start address is that of array firCoeffs32, and end address is equal to (start address + 0x400), and plot the contents of that file using the MATLAB function fm4\_logfft(). Sketch the magnitude frequency response on the axes of figure 11.



Figure 11: *Magnitude* frequency response observed using program fm4\_sysid\_CMSIS\_intr.c

**(6 marks)** for showing a curve that is flat between 100 Hz and 3.5 kHz at a level of -6.5dB but rolls off steeply at low frequencies (due to ac-coupling) and at high frequencies due to antialiasing and reconstruction filters in the ADC and DAC respectively.

The frequency range over which the magnitude frequency response has been identified is equal to half the sampling rate of the codec. In order to observe the frequency response of the codec beyond half its sampling frequency we will identify the characteristics of an audio card with a sampling rate of 8 kHz using a second system with a sampling rate of 16 kHz.

### Estimating WM8731 codec bandwidth using two FM4 BOards

Connect two FM4 boards together as shown in Figure 12. Make sure that program fm4\_loop\_intr.c (sampling rate 8 kHz) is running on one system before running program fm4\_sysid\_CMSIS\_intr.c (sampling rate 16 kHz) for a short time on the other. The sampling rate used by program fm4\_sysid\_CMSIS\_intr.c is determined by parameter FS\_16000\_HZ passed to function fm4\_wm8731\_init().

Change the statement

fm4\_wm8731\_init(FS\_8000\_HZ, WM8731\_LINE\_IN,IO\_METHOD\_INTR,

WM8731\_HP\_OUT\_GAIN\_0\_DB, WM8731\_LINE\_IN\_GAIN\_0\_DB);

in program fm4\_sysid\_CMSIS\_intr.c to read

fm4\_wm8731\_init(FS\_16000\_HZ, WM8731\_LINE\_IN,IO\_METHOD\_INTR,

WM8731\_HP\_OUT\_GAIN\_0\_DB, WM8731\_LINE\_IN\_GAIN\_0\_DB);

Change the statement

#define NUM\_TAPS 256

to read

#define NUM\_TAPS 512

**and** change the sampling frequency in program fm4\_loop\_intr.c from 48 kHz to 8 kHz. Also change the input used by program fm4\_loop\_intr.c from WM8731\_MIC\_IN to WM8731\_LINE\_IN . After running and halting program fm4\_sysid\_CMSIS\_intr.c, save the values of the 512 adaptive filter coefficients firCoeffs32 to a file using the command

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

Where end address is equal to (start address + 0x800). Plot the saved values using MATLAB function fm4\_logfft(). Sketch the magnitude frequency response on the axes of figure 13.

FM4 Starter Kit 1

FM4 Starter Kit 2

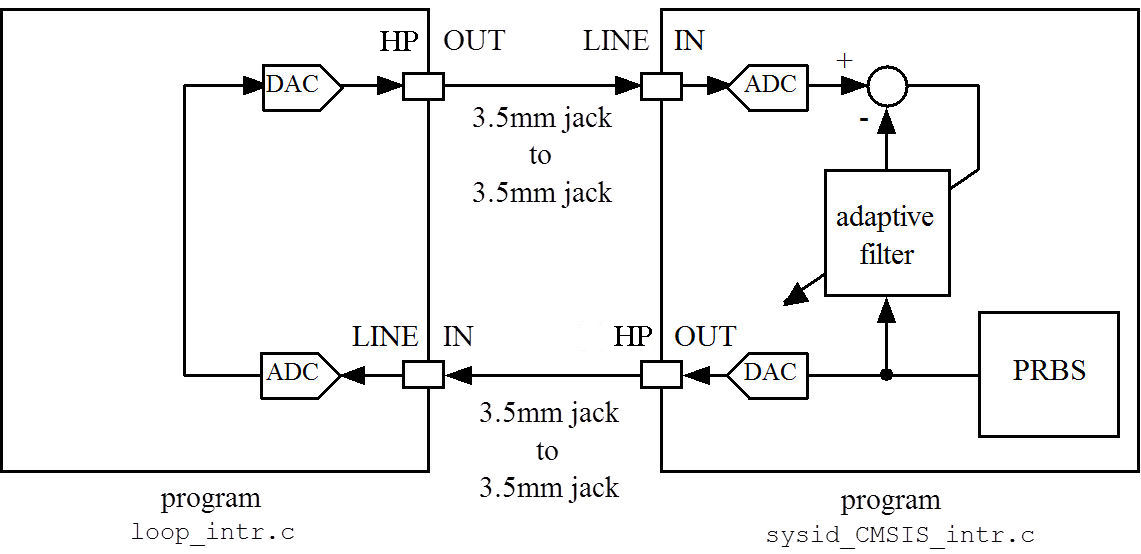


Figure 12: Connection diagram for WM8731 codec bandwidth identification using program fm4\_sysid\_CMSIS\_intr.c and two boards.

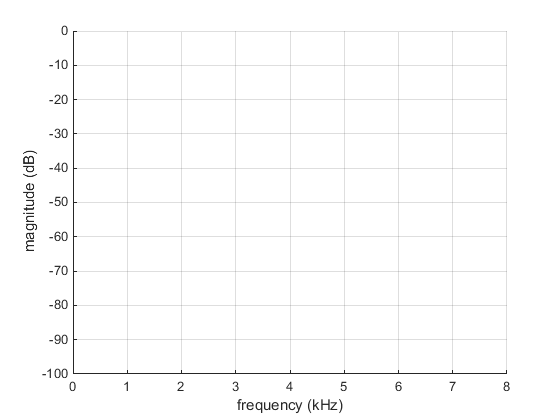


Figure 13: *Magnitude* frequency response observed using program sysid\_CMSIS\_intr.c and two audio cards.

**(5 marks)** for showing a curve that is flat between 100 Hz and 3.5 kHz at a level of -13dB but rolls off steeply at low frequencies (due to ac-coupling) and at frequencies above 4 kHz to a level of approximately -75dB.

# conclusions

At the end of this exercise, you should have gained an awareness of the limited bandwidth of digital signal processing systems and of the importance and characteristics of antialiasing and reconstruction filters.