PRACTICAL SESSION 6

INTRODUCTION TO DSP PROGRAMMING Analogic-Digital/Digital-Analogic CONVERTER INPUT-OUTPUT

The aim in this sessions is to review the ADC/DAC programming included in the DSP board: How to program the interrupt routines, data types and data handling.

Download the codec_bios.zip file from “aula global”, save it in a new folder. Open the project “codec_bios” with Code Composer.

Fix the paths of the libraries; they will vary depending on the new location where you have saved the project.

The project’s main file is codec_b.c, and contains the main routines for the session.

This session has three parts:
1. Programming a simple input/output system: The application receives a new data, and sends it to the DAC output.
2. Programming a sine wave signal generator based on fixed values stored in a table.
3. Programming an FIR filter.

All the three parts are already programmed; you will only have to comment/uncomment the active section for each part.

MAIN ROUTINE DESCRIPTION

The routine main() contains the setup stage for the program. Once finished, the DSP enters in an infinite loop waiting for hardware interrupt signals.

This routine has three sections:

1. ADC/DAC Setup

In this section, the programmer defines the ADC/DAC parameters, using the follow instructions:

```
/* Open Handset Codec */
   hHandset = codec_open(HANDSET_CODEC);              /* Acquire handle to handset codec */

   /* Set codec parameters */
   codec_dac_mode(hHandset, CODEC_DAC_15BIT);         /* DAC in 15-bit mode */
   codec_adc_mode(hHandset, CODEC_ADC_15BIT);         /* ADC in 15-bit mode */
   codec_ain_gain(hHandset, CODEC_AIN_6dB);           /* 6dB gain on analog input to ADC */
   codec_aout_gain(hHandset, CODEC_AOUT_MINUS_6dB);   /* -6dB gain on analog output from DAC */
   codec_sample_rate(hHandset,SR_8000);               /* 8 KHz sampling rate */
```

Where you get the ADC/DAC handler (hHandset), which is used to reference the device.

The first two lines program the ADC/DAC working mode (15 bits signed converter). The two next instructions set the input and output gain. The last instruction sets the sampling frequency.

Search in the code composer installation (c:\ti) the codec.h file, and look up the set of valid values for the codec gain (prefix CODEC_AIN and CODEC_AOUT), and the valid values for the sample frequency.

2. Setup global variables

In this zone the programmer must set the initial values for the global variables, if the program uses them.

3. Interruption Mask setup

The last instruction in the main routine sets the interruption mask, which allows handling the hardware interrupt signal form the ADC. To set the new interrupt mask uncomment the following instruction:

```
C54_enableIMR(McBSP1_RX_MASK);
```

This must be the last instruction, otherwise the global variables can be set in unknown state and the program will have an erratic behaviour.

Exercise 1: Input buffer:

The aim in this first test is to characterize the frequency response of the DAC.

Set the input signal to a sine wave with 100 mV peak-to-peak value and 1kHz frequency. Check the output and you must see the input signal.

Once verified, sweep the input frequency from SR/200 up to SR/2, check the cut-off frequencies of these converters.

Change the sampling frequency to verify the new filter cut-off frequencies

Exercise 2: Sine wave generation

With MATLAB, calculate 16 values for one period of a sine wave (between $0{:}2\pi$). Convert to Q15 representation and stores it in sineTable array variable.

Calculate the expression of the output frequency of the sampled sine wave made of the periodic repetition of these 16 values, in function of the ADC/DAC sampling frequency and the sineTable array length.

Exercise 3: FIR filter

Using MATLAB, design a low-pass FIR filter (using fir1 function), with a sampling frequency of 8 kHz and a cut-off frequency of 2 kHz, order 15. Convert the coefficients to Q15 representation and store the in coeffs variable.

Verify your new filter.

Now, with the same coefficients, and modifying the sampling rate, design a new low-pass filter with cut-off frequency 10.2 Khz (aprox).

Now recalculate the coefficients to have a high-pass filter with cutt-off frequency of 500 Hz.

Observe: the code is the same for low-pass, high-pass and band-pass filters, only you have to change the coefficients.