Istanbul Technical University
Faculty of Computer and Informatics

BLG438E Digital Signal Processing Lab
Experiment 7

Cem Yusuf Aydoğdu
150120251

Mert Yıldız
150120066

Experiment

In this experiment, Discrete Fourier Transformation was implemented, which allows to convert periodic signals from time domain to frequency domain. This transformation results coefficients of complex sinusoidal functions with different frequencies. Combinations of these sinusoidal functions weighted with calculated coefficients produce the input signal. The term discrete denotes that converted signals consist of discrete-time data sets. Generally, Fast Fourier Transformation algorithms are used to implement DFT in order perform calculations efficiently.

In the code, every sample of the input signal is taken when this interrupt program is called. After taking a sample, all previous samples are shifted. In total, N=16 samples from original signal are collected. Note that the input is also divided with 2^{16} in order to prevent overflow in calculations or output operation.

$$X[k] = \sum_{n=0}^{N-1} x[n]\cos(\frac{jkn2\pi}{N}) - j\sum_{n=0}^{N-1} x[n]\sin(\frac{jkn2\pi}{N})$$

Then, DFT coefficients are calculated according to formula above, first real and then imaginary parts. After that, an output is constructed from these coefficients and written to the output.

```
Int16 input_16, output_16, i,j;
Int16 N=16;
float input,output;
float x[N] = {0};
float Xre, Xim;

for(;;){
    for(j=0; j<N; j++){

       while((Rcv & I2S0_IR) == 0);
       input_16 = I2S0_W0_MSW_R;            //read the input
       while((Xmit & I2S0_IR) == 0);

       input=float(input_16/pow(2,16));

       for(i=N-1; i>0; i--)          //shift samples
            x[i] = x[i-1];
       x[0]=input;

       //calculate coefficients
       Xre=0;
       for(i=0; i<N; i++)
            Xre += x[i] * cos(2*i*j*PI/N);

       Xim=0;
       for(i=0; i<N; i++)
            Xim -= x[i] * sin(2*i*j*PI/N);

       //write output
       output = Xre*Xre + Xim*Xim;
       output_16 = output*pow(2,16);
       I2S0_W0_MSW_W = output_16;
       I2S0_W0_LSW_W = 0;
       I2S0_W1_MSW_W = output_16;
       I2S0_W1_LSW_W = 0;

    }
}
```