Cooley-Tukey FFT for 16-bit Integer Numbers

Overview

This library allows calculating FFT for the signals entered as an array of 16-bit signed integer numbers. The DSP CPU instructions are not used and the code will work for any 16-bit PIC device from PIC24F to dsPIC33E. The result of the calculation is written to the same array used for the input data. The calculation algorithm is based on Cooley-Tukey FFT algorithm with decimation in frequency. The permutation is performed on the final result to order harmonics. The twiddle coefficients for the transform and result permutation tables are stored in flash (PSV .const section). The valid FFT sizes are 8, 16, 32, 64, 128, 256, 512, 1024 and 2048. The size of the variables in the array is limited by 16-bits for real and imaginary parts. To avoid the overflows and fit into 16-bits, before each iteration (radix 2 butterflies), the results are shifted one bit right (rounded). Though the rounding decreases the result precision and adds a noise to the FFT result the use of just 16-bit numbers makes the FFT calculations fast. For PIC24F @ 16MIPS (32 MHz oscillator frequency), the time required for the calculation is

Signal Points Number	FFT Calculation Time, Instructions	FFT Calculation Time, mS
8	960	0.060
16	2560	0.160
32	6400	0.400
64	15200	0.950
128	32000	2
256	73600	4.6
512	160000	10
1024	352000	22

The quality of the conversion can be good for many applications such as audio signal spectrum indicators, DTFM signals decoding, signal frequency estimation, signal quality estimation and so on. The harmonics are presented as complex numbers with real and imaginary parts. To calculate the harmonics amplitudes and phases the special functions are provided.

Files

To use the FFT library the following files must be included in the project:

fft.h – contains configurations (such as transform size) and FFT functions' prototypes,

fft.c – contains FFT functions' implementation,

fft.s – contains radix 2 butterfly code used in fft.c file.

Memory requirements

Configuration

The library is configured in fft.h file. The following parameter should be set in this header:

```
#define FFT_POINTS_NUMBER                    32
```

- This parameter defines number of points/samples for the calculation. The valid values are 8, 16, 32, 64, 128, 256, 512 and 1024. If required transform size doesn't match exactly the valid sizes then the size should be rounded to largest valid size and unused points should be set to zero.

Functions Description

void FFT(int16_t* re, int16_t* im)

Description: This function calculates FFT for the points provided in real and imaginary parts arrays. In most cases the input signal is real (not complex), the data points must be stored in the real part array and the imaginary array elements must be set to zero. The sizes of the real and imaginary arrays must be equal to the transform size defined by FFT_POINTS_NUMBER in fft.h file. After calculation the result (harmonics) are written to the arrays provided for the input data.

Parameters:

int16_t * re – pointer to array with the input signal vector (real part).

int16_t * im – pointer to array with the input signal vector (imaginary part), for real signals this vector values must be set to zero.

Returned data:

int16_t * re – pointer to array where the harmonics are stored (real part).

int16_t * im – pointer to array where the harmonics are stored (imaginary part).

uint16_t FFTAmplitude(int16_t re, int16_t im)

Description: This function calculates absolute value (harmonic amplitude) of the complex number (A = SQRT(Re^2 + Im^2)).

Parameters:

int16_t re – real part of the complex number.

int16_t im –imaginary part of the complex number.

Returned data:

The function returns the amplitude for the complex number entered.

uint16_t FFTPhase(int16_t re, int16_t im)

Description: This function calculates angle (harmonic phase) of the complex number (D = (180/PI)*ARCTAN(Re/Im)) in degrees with maximum error +-0.17 degree.

Parameters:

int16_t re – real part of the complex number.

int16_t im –imaginary part of the complex number.

Returned data:

The function returns the angle in degrees for the complex number entered.

Getting Started

Let's assume that we need to analyze signal with highest frequency of 2048 Hz and 2048/16 = 128 Hz resolution. In this case by Nyquist-Shannon theorem the input signal must be digitized with 2x2048 = 4096 Hz frequency. To achieve 128 Hz resolution 4096 Hz/ 128 Hz = 32 samples are required. The transform size should be set in fft.h file:

```
#define FFT_POINTS_NUMBER                    32
```

32 samples/points must be stored into the real parts array and zeros must be written into the imaginary parts array. To calculate the FFT, the pointers to these arrays should be passed to void FFT(…) function. If PIC24F is used at the maximum speed (16MIPS) then the result of the calculation will be available in 400 uS in the input data arrays.

Next step is to calculate the amplitude and phase of each harmonic. Only harmonics from 0 (DC) to 15 (1920 Hz) are valid. As effect of digitizing, the spectrum is repeated and harmonics from 16 to 31 will be just a mirrored order of harmonics from 0 to 15 (the values should be approximately same). So amplitude and phase must be calculated only for the first 16 harmonics. It can be done with uint16_t FFTAmplitude(…) and int16_t FFTPhase(…) functions. The DC level in the frequency domain will match the DC level in the time domain. But FFT harmonics amplitudes will be just 0.5 of the signal harmonics in the time domain.

Demo Project

The fft_demo_main.c file demonstrates the FFT library functions usage. To execute the example a MPLABX simulator or Explorer 16 Development board withPIC24FJ128GA010 PIM can be used.