GigaDevice Semiconductor Inc.

GD32G5x3 FFT Module User Guide

Application Note
AN248

Revision 1.0

(Dec. 2024)

Table of Contents

List of Figures

List of Tables

1. Preface

This document is specifically designed for engineers developing with GD32 MCUs. It mainly introduces the functional configuration and usage precautions of the FFT module in the GD32G5x3 series devices. The aim is to help GD32 MCU developers use the FFT module, thereby shortening the development cycle.

2. Introduction to the FFT Module

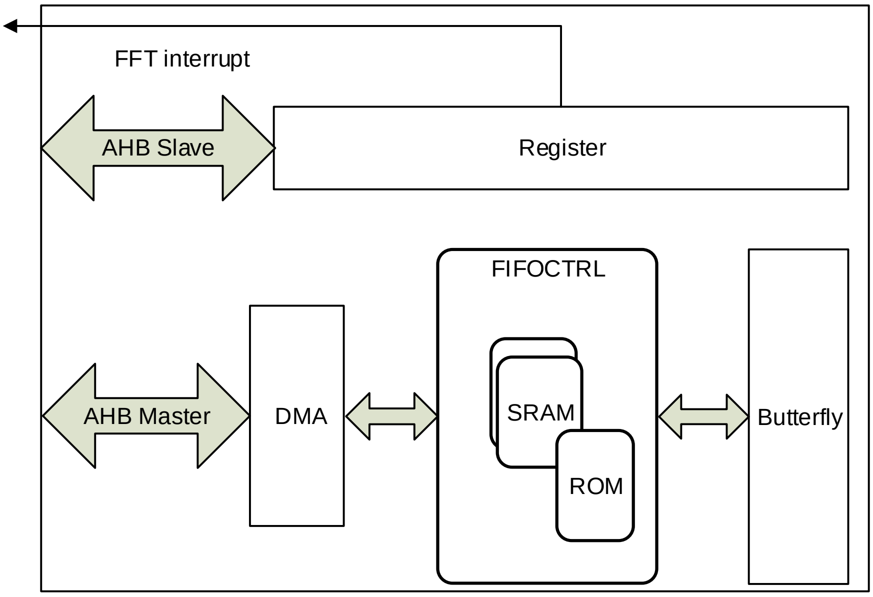
The Fast Fourier Transform (FFT) is an efficient computation of the Discrete Fourier Transform (DFT). The module supports CPU to offload FFT operations. Compared to a software implementation, it can accelerate calculations and time critical tasks. The module supports 6 configuration FFT point number up to 1024, and input and output data should be IEEE-754 single precision float point complex number.

The main features of the FFT module are as follows:

- Support 1024/512/256/128/64/32 points FFT
- Support IFFT mode
- IEEE-754 single precision float point complex number input and output data
- DMA master to load and store data
- Support window function configured in memory
- Support input down sample

2.1. Operation Process

Figure 2-1. FFT module block diagram



The FFT module consists of DMA Master, FIFOCTRL submodule, Butterfly submodule and register, as shown in ***Figure 2-1. FFT module block diagram***. The module is based on redix-2 FFT algorithm. The input data including real, image and window data in the format of IEEE754 single precision float point is loaded from memory, and after window function operation and input bits reverse operation they are written into internal SRAM. The rotation factor already be ready in ROM.

The butterfly is reused in FFT calculation, and calculate interatly (N / 2) * $\log_2 N$ times. The calculation is same address calculation, that is, the output data of each calculation are write back to same address where the input data stored before.

The DMA master transfer the output data from internal SRAM to memory after all butterfly calculation interation finished.

2.2. Operation guide

This section describes the advised operation guide for FFT.

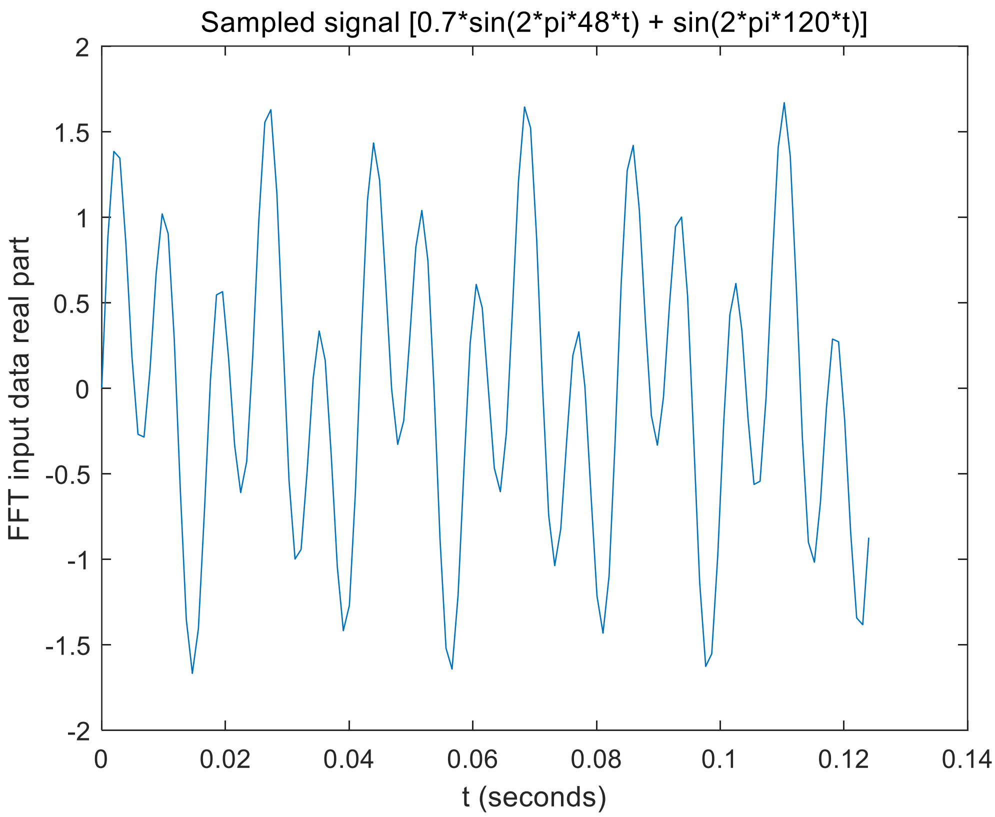
1. Configure the FFT_IMSADDR to set the FFT image start address, if necessary, that is, the image data is not zero.
2. Configure the FFT_RESADDR to set the FFT real start address.
3. Configure the WINEN and FFT_WSADDR to set the FFT window start address, if necessary.
4. Configure the FFT_OSADDR to set the FFT output start address.
5. Configure the FFT_LOOPLEN, if necessary.
6. Configure the IFFTMODE bit, if necessary.
7. Configure the NUMPT[2:0] bits in the FFT_CSR register to set number of FFT points.
8. Configure FFTEN.
9. Wait FFTEN clear, or CCF set.

3. Example of FFT Configuration Code

The example mainly demonstrates using the FFT module to perform an FFT transform on 128 input data (without windowing). The input data, window function, and output data of the FFT module are all complex, containing real and imaginary parts.

The 128 input data are collected from the signal $[0.7*\sin(2\pi*48*t)+\sin(2\pi*120*t)]$, with a sampling frequency of 1024KHz and a sampling duration of 0.125s. It is evident that this signal is a superposition of two sine signals, one with a frequency of 48Hz and an amplitude of 0.7, and the other with a frequency of 120Hz and an amplitude of 1.0, as shown in the ***Figure 3-1. FFT Input data***. Its imaginary part defaults to 0.

Figure 3-1. FFT Input data



The real part of the input data is defined in the array *fft_real_buf[FFT_DATA_LENGTH]*, with the imaginary part defaulting to 0. The code is as follows:

Table 3-1. FFT input data code

```
/* Specify the parameters of a signal with a sampling frequency of 1024 Hz and a signal duration of 0.125 seconds.Form a signal containing a 48 Hz sinusoid of amplitude 0.7 and a 120 Hz sinusoid of amplitude 1 input data = 0.7*sin(2*pi*48*t) + sin(2*pi*120*t) */
#define FERQ_SAMPLE            1024          //uint:Hz
#define PI                     3.1415926f
#define FFT_DATA_LENGTH        128U
```

```
float fft_real_buf[FFT_DATA_LENGTH] = {0};

for(index=0; index<FFT_DATA_LENGTH; index++){
fft_real_buf[index] = 0.7*sin(2*PI*48*index/FERQ_SAMPLE) + sin(2*PI*120*index/FERQ_SAMPLE);
}
```

The FFT module configuration code is as follows:

Table 3-2. FFT module configuration code

```
/* fft parameter variable */
fft_parameter_struct fftconfig;
/* enable FFT clock */
rcu_periph_clock_enable(RCU_FFT);

/* reset the FFT */
fft_deinit();
/* initialize the FFT filter parameter struct */
fft_struct_para_init(&fftconfig);
/* setting FFT parameter */
fftconfig.mode_sel         = FFT_MODE;
fftconfig.point_num        = FFT_POINT_128;
fftconfig.window_enable = FFT_WINDOW_DISABLE;
fftconfig.downsamp_sel    = FFT_DOWNSAMPLE_1;
fftconfig.image_source    = FFT_IM_ZERO;
fftconfig.loopbuf_len      = 0U;
fftconfig.loopbuf_index = 0U;
fftconfig.real_addr        = (uint32_t)fft_real_buf;
fftconfig.image_addr       = NULL;
fftconfig.window_addr      = NULL;
fftconfig.output_addr      = (uint32_t)fft_output_buf;
/* config FFT parameter */
fft_init(&fftconfig);
/* start FFT calculation */
fft_calculation_start();
```

The result after the FFT transform is 128 output data, stored in the array *fft_output_buf[FFT_DATA_LENGTH*2]*. Each output data is represented by two elements, with even-numbered elements being the real part and odd-numbered elements being the imaginary part. For example, the real part of the first output data is *fft_output_buf[0]*, and the imaginary part is *fft_output_buf[1]*, and so on for the other data. The code for the output data is as follows:

Table 3-3. FFT output data code

```
/* output data */
float fft_output_buf[FFT_DATA_LENGTH*2] = {0};
```

The results of the FFT transform can be output through the serial port as follows:

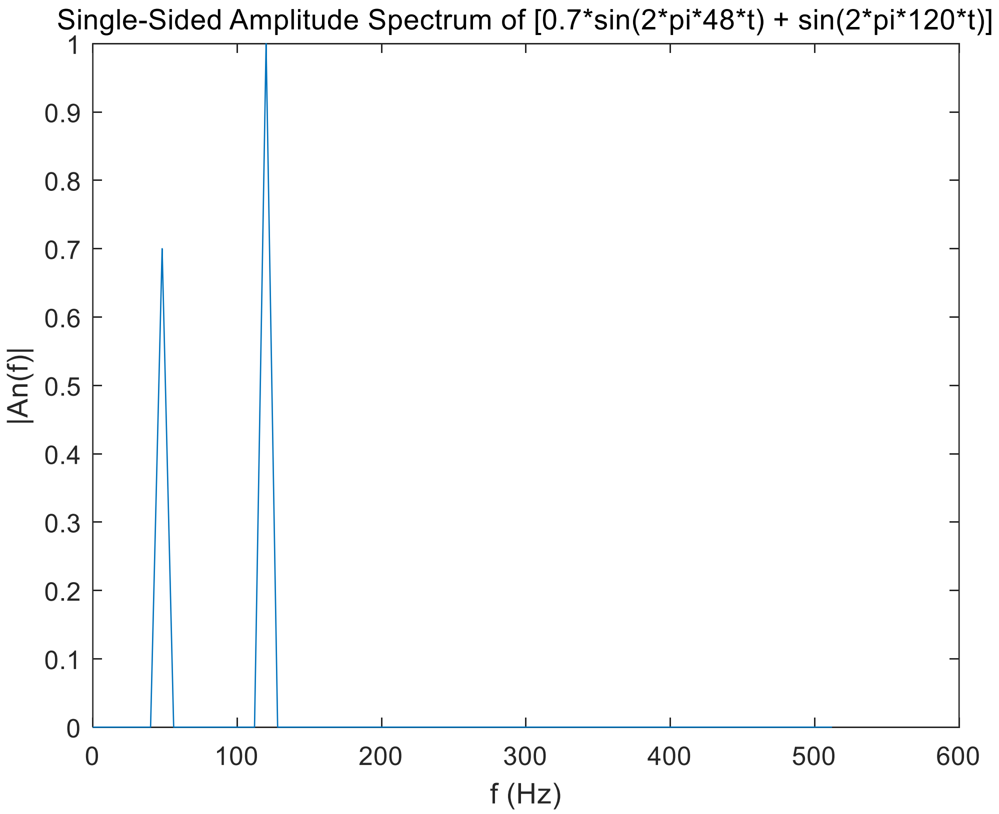
Table 3-4. FFT output results

```
FFT input data:

0.000000+0*i, 0.874758+0*i, 1.384084+0*i, 1.344315+0*i, 0.841806+0*i, 0.182527+0*i, -0.270391+0*i, -
0.286645+0*i, 0.112291+0*i, 0.666867+0*i, 1.018484+0*i, 0.901419+0*i, 0.287692+0*i, -0.590805+0*i, -
1.355038+0*i, -1.668654+0*i, -1.407107+0*i, -0.718927+0*i, 0.052364+0*i, 0.545101+0*i, 0.563592+0*i,
0.174369+0*i, -0.334832+0*i, -0.611566+0*i, -0.428905+0*i, 0.189789+0*i, 0.976833+0*i, 1.554358+0*i,
1.627501+0*i, 1.136808+0*i, 0.290884+0*i, -0.537751+0*i, -1.000000+0*i, -0.944151+0*i, -0.486918+0*i,
0.054591+0*i, 0.334070+0*i, 0.161101+0*i, -0.396262+0*i, -1.044898+0*i, -1.418853+0*i, -1.271523+0*i, -
0.607962+0*i, 0.311590+0*i, 1.099346+0*i, 1.433252+0*i, 1.216425+0*i, 0.620792+0*i, -0.007105+0*i, -
0.328937+0*i, -0.190983+0*i, 0.297341+0*i, 0.823447+0*i, 1.038642+0*i, 0.745359+0*i, 0.006915+0*i, -
0.877654+0*i, -1.521333+0*i, -1.643492+0*i, -1.210734+0*i, -0.451630+0*i, 0.262099+0*i, 0.606286+0*i,
0.468361+0*i, 0.000004+0*i, -0.468355+0*i, -0.606285+0*i, -0.262103+0*i, 0.451622+0*i, 1.210728+0*i,
1.643489+0*i, 1.521336+0*i, 0.877661+0*i, -0.006908+0*i, -0.745356+0*i, -1.038644+0*i, -0.823453+0*i, -
0.297348+0*i, 0.190978+0*i, 0.328938+0*i, 0.007110+0*i, -0.620784+0*i, -1.216419+0*i, -1.433251+0*i, -
1.099350+0*i, -0.311597+0*i, 0.607956+0*i, 1.271521+0*i, 1.418857+0*i, 1.044904+0*i, 0.396268+0*i, -
0.161095+0*i, -0.334071+0*i, -0.054595+0*i, 0.486914+0*i, 0.944146+0*i, 1.000000+0*i, 0.537754+0*i, -
0.290873+0*i, -1.136801+0*i, -1.627500+0*i, -1.554359+0*i, -0.976842+0*i, -0.189795+0*i, 0.428903+0*i,
0.611570+0*i, 0.334841+0*i, -0.174363+0*i, -0.563589+0*i, -0.545103+0*i, -0.052370+0*i, 0.718921+0*i,
1.407099+0*i, 1.668653+0*i, 1.355043+0*i, 0.590809+0*i, -0.287685+0*i, -0.901416+0*i, -1.018487+0*i, -
0.666871+0*i, -0.112300+0*i, 0.286641+0*i, 0.270392+0*i, -0.182518+0*i, -0.841798+0*i, -1.344311+0*i, -
1.384085+0*i, -0.874766+0*i,
--------------------------------
FFT calculation completion!
------------------------------------

FFT calculation result:

0.000018+(0.000000*i), 0.000004+(0.000004*i), -0.000007+(-0.000003*i), 0.000009+(0.000004*i),
0.000010+(0.000009*i), -0.000001+(0.000007*i), 0.000001+(-44.799988*i), -0.000004+(0.000008*i), -
0.000003+(0.000004*i), -0.000002+(0.000012*i), -0.000000+(0.000003*i), 0.000004+(0.000013*i),
0.000006+(0.000018*i), 0.000004+(0.000042*i), 0.000001+(0.000074*i), -0.000234+(-63.999992*i), -
0.000004+(-0.000078*i), -0.000004+(-0.000035*i), 0.000003+(-0.000031*i), -0.000009+(-0.000018*i), -
0.000018+(-0.000024*i), -0.000003+(-0.000044*i), 0.000021+(-0.000026*i), 0.000012+(-0.000005*i), -
0.000001+(-0.000018*i), 0.000008+(-0.000012*i), 0.000009+(-0.000014*i), 0.000010+(-0.000004*i),
0.000012+(-0.000012*i), 0.000011+(0.000004*i), 0.000010+(-0.000008*i), 0.000001+(0.000005*i), -
0.000006+(-0.000009*i), -0.000002+(-0.000003*i), 0.000011+(-0.000010*i), 0.000007+(-0.000003*i), -
0.000004+(0.000004*i), 0.000002+(-0.000014*i), 0.000007+(-0.000005*i), 0.000000+(0.000011*i), -
0.000007+(-0.000022*i), 0.000002+(-0.000002*i), 0.000012+(-0.000021*i), 0.000010+(0.000004*i),
0.000010+(-0.000006*i), 0.000011+(0.000003*i), 0.000007+(-0.000000*i), 0.000000+(0.000003*i),
0.000001+(-0.000003*i), -0.000009+(0.000008*i), -0.000015+(-0.000014*i), 0.000009+(-0.000026*i),
0.000026+(-0.000009*i), 0.000014+(0.000010*i), 0.000000+(-0.000004*i), 0.000011+(-0.000010*i),
```

```
0.000017+(0.000011*i), 0.000006+(-0.000000*i), 0.000002+(0.000008*i), 0.000006+(-0.000009*i),
0.000011+(0.000001*i), 0.000003+(0.000016*i), -0.000000+(-0.000003*i), 0.000001+(0.000008*i), -
0.000003+(0.000000*i), 0.000001+(-0.000008*i), -0.000000+(0.000003*i), 0.000003+(-0.000016*i),
0.000011+(-0.000001*i), 0.000006+(0.000009*i), 0.000002+(-0.000008*i), 0.000006+(0.000000*i),
0.000017+(-0.000011*i), 0.000011+(0.000010*i), 0.000000+(0.000004*i), 0.000014+(-0.000010*i),
0.000026+(0.000009*i), 0.000009+(0.000026*i), -0.000015+(0.000014*i), -0.000009+(-0.000008*i),
0.000001+(0.000003*i), 0.000000+(-0.000003*i), 0.000007+(0.000000*i), 0.000011+(-0.000003*i),
0.000010+(0.000006*i), 0.000010+(-0.000004*i), 0.000012+(0.000021*i), 0.000002+(0.000002*i), -
0.000007+(0.000022*i), 0.000000+(-0.000011*i), 0.000007+(0.000005*i), 0.000002+(0.000014*i), -
0.000004+(-0.000004*i), 0.000007+(0.000003*i), 0.000011+(0.000010*i), -0.000002+(0.000003*i), -
0.000006+(0.000009*i), 0.000001+(-0.000005*i), 0.000010+(0.000008*i), 0.000011+(-0.000004*i),
0.000012+(0.000012*i), 0.000010+(0.000004*i), 0.000009+(0.000014*i), 0.000008+(0.000012*i), -
0.000001+(0.000018*i), 0.000012+(0.000005*i), 0.000021+(0.000026*i), -0.000003+(0.000044*i), -
0.000018+(0.000024*i), -0.000009+(0.000018*i), 0.000003+(0.000031*i), -0.000004+(0.000035*i), -
0.000004+(0.000078*i), -0.000234+(63.999992*i), 0.000001+(-0.000074*i), 0.000004+(-0.000042*i),
0.000006+(-0.000018*i), 0.000004+(-0.000013*i), -0.000000+(-0.000003*i), -0.000002+(-0.000012*i), -
0.000003+(-0.000004*i), -0.000004+(-0.000008*i), 0.000001+(44.799988*i), -0.000001+(-0.000007*i),
0.000010+(-0.000009*i), 0.000009+(-0.000004*i), -0.000007+(0.000003*i), 0.000004+(-0.000004*i)
```

The amplitude-frequency plot corresponding to the above output data fft_output_buf is as shown

Figure 3-2. Frequency amplitude characteristics of FFT output data



4. Revision history

Table 4-1. Revision history

Revision No.	Description	Date
1.0	Initial Release	Dec.31 2024

Important Notice