快速傅立叶变换使用示范

NuMicro® 32 位系列微控制器范例代码介绍

文件信息

代码简述	使用快速傅立叶变换将 ADC 取样讯号转换至频率域，示范实作方法与结果。
BSP 版本	NUC121_125_Series_BSP_CMSIS_V3.01.003
开发平台	NuTiny-SDK-NUC121_V1.0

The information described in this document is the exclusive intellectual property of Nuvoton Technology Corporation and shall not be reproduced without permission from Nuvoton.

Nuvoton is providing this document only for reference purposes of NuMicro microcontroller based system design. Nuvoton assumes no responsibility for errors or omissions.

All data and specifications are subject to change without notice.

For additional information or questions, please contact: Nuvoton Technology Corporation.

www.nuvoton.com

1 功能介绍

1.1 简介

快速傅立叶变换，用于将时间域的离散讯号转换至频率域表示，用户得以观察该讯号的频率组成成分与分布情形，是常见的讯号分析手段。此范例将示范如何透过 ADC 与 TIMER 的搭配使用，取样出一段离散讯号，并进行快速傅立叶变换，最后观察频谱的数值分布是否符合原始输入讯号的特性。支持三种讯号长度范例，分别为 128 取样点，256取样点，以及512取样点。

1.2 原理

此范例采用 TIMER trigger ADC 的功能进行实作。首先利用波形产生器输出频率固定为 60 Hz 的正弦波，模拟一般应用情境下欲分析的讯号来源，给予微控制器使用 ADC 进行取样。使用者可根据应用上的需求自行调整取样频率，注意其数值会直接影响频谱的带宽与分辨率。每次的 TIMER 逾时事件，硬件便会自动触发 ADC 进行一次取样动作，软件则须于 ADC 转换完毕后的中断服务内，将数据以位反转的排序写入缓冲区当中。

本范例程序提供必要参数的查表数值，可简化微处理机的运算量，包含傅立叶变换过程所需要的旋转因子常数，以及 Decimate-in-time 作法所需的位反转排序，开发者已经事前计算完毕并定义在头文件当中。由于寻找表的内容会根据讯号长度的不同而有异，不同长度设定的寻找表无法共享，使用者必须于头文件中选择范例提供的指定长度参数编译，流程主要可分为五个部分:

1. 指定讯号长度以及取样频率
2. 系统与周边硬件初始化
3. ADC 取样至指定数量为止
4. 执行傅立叶变换
5. 后置处理，显示频谱信息

1.3 执行结果

1.3.1 起始画面

等待使用者确认 ADC 输入脚位 PD.2 已经接上待量测的讯号源，按下任意建开始取样。

```
+------------------------------------------------------------+
|            Fast Fourier Transform Example Code             |
+------------------------------------------------------------+

# Demonstrate the decimate-in-time(DIT) radix-2 FFT
# This example code supports 3 kinds of data length:
    * 128 samples
    * 256 samples
    * 512 samples
# Please confirm the selected length of fft sequence is [512].
    Otherwise, change the fft sequence length settings and
    re-compile again to acquire the correct look-up table.
# Connect PD.2 to the signal source.
# Press any key to start ADC capturing....
```

图 1-1 Terminal 起始画面

1.3.2 时间域数据显示

取样完毕后显示时间域讯号信息，如图 1-2 所示。因资料列数过多，仅撷取部分数据数值显示。显示内容分为三栏，左侧字段为数据的原取样顺序，中间字段为数据根据位反转表写入缓冲区的位置，右侧字段则为 ADC 数值经处理后实际写入缓冲区的数据。

```
+------------------------------------------------------------+
| Display Time-Domain Data                                   +
+------------------------------------------------------------+
#Index before reversal     #Index after reversal     #ADC Data
 0                          0                         108
 1                          256                       87
 2                          128                       63
 3                          384                       41
 4                          64                        22
 5                          320                       9
 6                          192                       4
 7                          448                       7
 8                          32                        19
 9                          288                       37
 10                         160                       59
 11                         416                       82
 12                         96                        104
 13                         352                       120
 14                         224                       127
 15                         480                       127
 16                         16                        123
 17                         272                       109
 18                         144                       88
 19                         400                       65
```

图 1-2 Terminal 局部时间域讯号显示

1.3.3 频率域数据显示

图1-3为执行傅立叶变换后得出的局部频谱，显示内容分为四栏，由左至右。第一栏为数据索引，第二栏为该点所表示的频率，第三栏为原讯号含有该频率讯号的成份大小，由频率域数据实数部分与虚数部分的平方和求得，单位可先忽略不计。最后一栏则表示该频率成份于此频谱占有的强度百分比。藉由观察第四栏可看出该讯号的组成成分几乎正好落在60 Hz 的频率点上，符合波形产生器给予的 60 Hz 正弦波频率。

```
+------------------------------------------------------------+
Spectrum Parameters
# Sample rate: 1024 Hz
# Data length: 512 samples
# Resolution : 2.00 Hz
+------------------------------------------------------------+
```

#Index	#Frequency(Hz)	#Amplitude	#Ratio
0	0.00	16900	0.00%
1	2.00	260	0.00%
2	4.00	3026	0.00%
3	6.00	2929	0.00%
4	8.00	340	0.00%
5	10.00	949	0.00%
6	12.00	2425	0.00%
7	14.00	712	0.00%
8	16.00	117	0.00%
9	18.00	457	0.00%
10	20.00	1060	0.00%
11	22.00	778	0.00%
12	24.00	949	0.00%
13	26.00	205	0.00%
14	28.00	648	0.00%
15	30.00	261	0.00%
16	32.00	17	0.00%
17	34.00	34	0.00%
18	36.00	221	0.00%
19	38.00	458	0.00%
20	40.00	232	0.00%
21	42.00	389	0.00%
22	44.00	1274	0.00%
23	46.00	292	0.00%
24	48.00	629	0.00%
25	50.00	449	0.00%
26	52.00	1481	0.00%
27	54.00	629	0.00%
28	56.00	2525	0.00%
29	58.00	2330	0.00%
30	60.00	255295080	49.95%
31	62.00	1732	0.00%
32	64.00	580	0.00%
33	66.00	45	0.00%
34	68.00	130	0.00%
35	70.00	229	0.00%
36	72.00	20	0.00%
37	74.00	50	0.00%
38	76.00	1	0.00%
39	78.00	106	0.00%
40	80.00	26	0.00%

图 1-3 Terminal 局部频率域信息显示

1.3.4 时间域数据作图

撷取整份时间域数据进行作图，可得到下图1-4，横坐标为取样点索引，纵坐标为 ADC 转换数值，呈现出标准的正弦波。

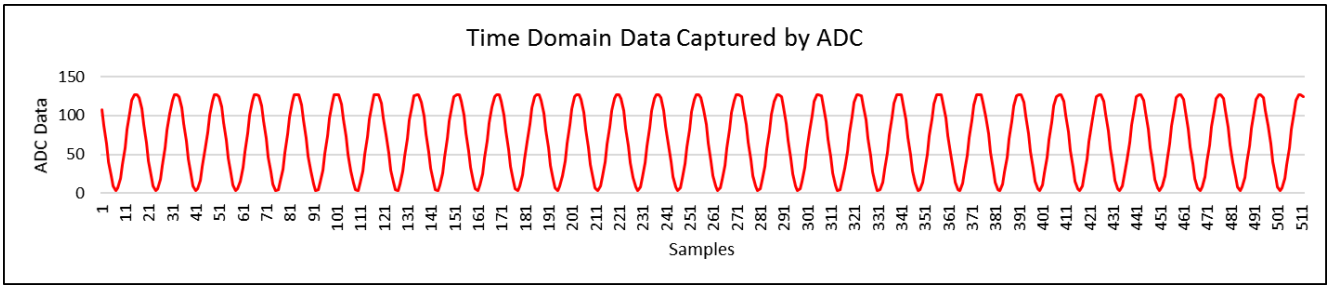


图1-4 NUC121 ADC 取样点作图

1.3.5 频率域数据作图

撷取整份频率域数据进行作图，可得到下图1-5。经过傅立叶变换后得到的频谱，其左半部以及右半部会呈现出共轭对称分布。假设用户所设定的取样频率为F_s，则可以得到该讯号组成频率为$F_s/2 \sim -F_s/2$ 区间内的强度分布，其中负频率的部分可以忽略。横坐标为频率，纵坐标为强度百分比，由图中可清楚地观察到该讯号的能量集中在频率为 60 Hz 的坐标点上。

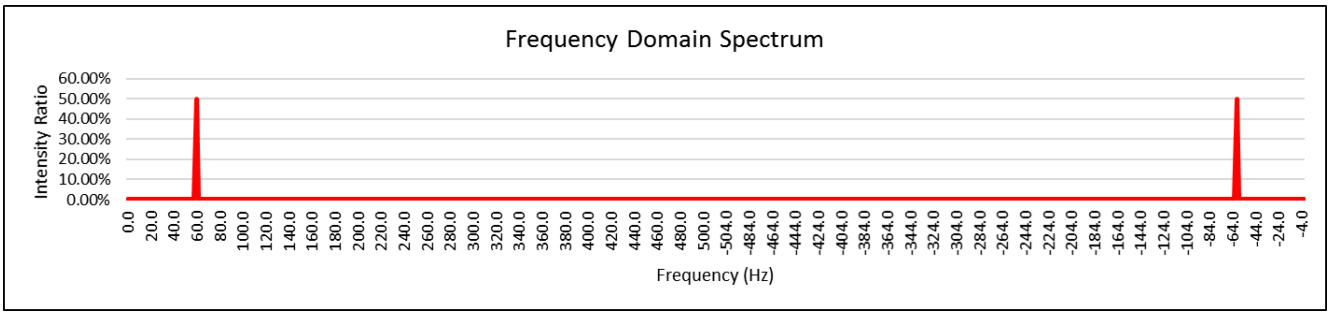


图1-5 傅立叶变换频率域作图

2 代码介绍

2.1 指定讯号长度

定义讯号的长度，此范例提供 128 取样点，256 取样点，与 512 取样点。长度越长，频谱的分辨率越高，但所需的资源与运算时间也会相对提升。

```
#ifndef __FFT_LUT_H__
#define __FFT_LUT_H__

/* This example provides 3 types selectable data length */
#define FFT_128_POINT               (0)
#define FFT_256_POINT               (1)
#define FFT_512_POINT               (2)

/* User can select one of the length setting here. */
#define FFT_N_POINT_SEL             (FFT_512_POINT)
```

2.2 依据长度设定配给寻找表

提供旋转因子的正弦 SIN_LUT 与余弦数值表 COS_LUT，以及位反转数值表 REV_LUT。旋转因子 $exp(-j2\,\pi k/N)$ 为傅立叶变换当中，取样点所必须乘上的复数常数，可拆解为实部以及虚部两项 $\cos(2\pi k/N) - j\,\sin(2\pi k/N)$ 。并根据先前定义的讯号长度，给予的第 k 角度正弦与余弦的常数值。位反转表用于重新排列原本的时间域讯号顺序，根据 Decimate-in-time 的作法，时间域讯号必须先依照此规则排序再进行后续计算，因此数据会依据此表格内容填入缓冲区的指定地址。

```
/* Assign the corresponding look-up table according to the length setting.
   Do not modify the contents here. */
#if (FFT_N_POINT_SEL == FFT_128_POINT)
#define SEQUENCE_LEN                (128)
#define RESCALE(x)                  (x>>7)
#define COS_LUT                     COS_SCALED_LUT_128
#define SIN_LUT                     SIN_SCALED_LUT_128
#define REV_LUT                     REV_LUT_128

#elif (FFT_N_POINT_SEL == FFT_256_POINT)
#define SEQUENCE_LEN                (256)
#define RESCALE(x)                  (x>>8)
```

```
#define COS_LUT                     COS_SCALED_LUT_256
#define SIN_LUT                     SIN_SCALED_LUT_256
#define REV_LUT                     REV_LUT_256

#elif (FFT_N_POINT_SEL == FFT_512_POINT)
#define SEQUENCE_LEN                (512)
#define RESCALE(x)                  (x>>9)
#define COS_LUT                     COS_SCALED_LUT_512
#define SIN_LUT                     SIN_SCALED_LUT_512
#define REV_LUT                     REV_LUT_512

#else
#define "configuration error"
#endif
```

2.3 参数设定

取样频率 SAMPLE_RATE 由用户根据应用需求进行配置。SAMPLE_DATA_RSHIFT 参数用以防止傅立叶变换时发生溢位，将 ADC 的原始数据向右位移指定数量个位，在不发生溢位的前提下，此参数数值建议尽可能地减少，可以保留更多原始数据的特征，可由用户自行配置。

```
/* System clock frequency */
#define PLL_CLOCK                   (48000000)

/* The parameter SAMPLE_RATE is configurable. It is based on your application.
 * The unit of SAMPLE_RATE is hertz.
 */
#define SAMPLE_RATE                 (1024)

/* The parameter SAMPLE_DATA_RSHIFT is configurable, the purpose is to truncate
 * the data size by right shifting the raw ADC data, and it can prevent the buffer
 * from overflow when calculating a large amount N point of FFT. The longer data
 * length is selected, the higher SAMPLE_DATA_RSHIFT is required. If the spectrum
 * buffer is overflow, try to increase the SAMPLE_DATA_RSHIFT parameter.
 */
#define SAMPLE_DATA_RSHIFT          (5)
#define SHRINK(x)                   ((x) >> SAMPLE_DATA_RSHIFT)
```

2.4 计数器初始化设定

设定 TIMER0 工作频率为取样频率，开启 TIMER trigger ADC 功能。

```
void TIMER0_Init(void)
{
    uint32_t u32Freq;

    /* Set TIMER0 working frequency to SAMPLE_RATE */
    u32Freq = TIMER_Open(TIMER0, TIMER_PERIODIC_MODE, SAMPLE_RATE);

    if (u32Freq != SAMPLE_RATE)
    {
        printf("# Warning. Sample rate does not equal to the pre-defined setting.\n\r");
    }

    /* Enable timer timeout interrupt trigger ADC */
    TIMER0->CTL |= TIMER_TRGSRC_TIMEOUT_EVENT | TIMER_TRG_TO_ADC | TIMER_CTL_INTEN_Msk;
}
```

2.5 ADC 初始化设定

```
void ADC_Init(void)
{
    /* Set the ADC operation mode as single, input mode as single-end and enable the
       analog input channel 2 */
    ADC_Open(ADC, ADC_ADCR_DIFFEN_SINGLE_END, ADC_ADCR_ADMD_SINGLE, 0x1<<g_u8AdcChannel);

    /* Enable TIMER Trigger ADC */
    ADC_EnableTimerTrigger(ADC, ADC_ADCR_TRGS_TIMER, 0);

    /* Power on ADC module */
    ADC_POWER_ON(ADC);

    /* Clear the A/D interrupt flag for safe */
    ADC_CLR_INT_FLAG(ADC, ADC_ADF_INT);

    /* Enable the ADC interrupt */
    ADC_EnableInt(ADC, ADC_ADF_INT);
    NVIC_EnableIRQ(ADC_IRQn);
}
```

2.6 ADC 中断服务处理

转换完毕的 ADC 原始数据会先经过数值缩减处理，再以位反转的顺序写入缓冲区。

```
void ADC_IRQHandler(void)
{
    /* clear the A/D conversion flag */
    ADC_CLR_INT_FLAG(ADC, ADC_ADF_INT);

    /* Toggle PB.0 to indicate an ADC data conversion is complete */
    PB0^= 1;

    /* Place the time-domain signal into the real part buffer */
    g_i32RealBuffer [REV(g_u32AdcConvNum)] = \
                          SHRINK(ADC_GET_CONVERSION_DATA(ADC, g_u8AdcChannel));

    /* The image part buffer is initialized to zeros */
    g_i32ImageBuffer[REV(g_u32AdcConvNum)] = 0;

    /* Increase the buffer index by 1 */
    g_u32AdcConvNum++;
}
```

2.7 快速傅立叶变换

当指定长度的讯号已经全部取样完毕后，首先移除讯号的直流成份(平均值)，并执行快速傅立叶算法。

```
    /* Remove the DC component from the spectrum by zero-mean the time-domain signal before
       performing FFT. Otherwise, it will remain a high DC component at the 0 Hz point. */
    Remove_DC_Component(g_i32RealBuffer, SEQUENCE_LEN);

    /* Transform the time domain signal into frequency domain */
    NuFFT(g_i32RealBuffer, g_i32ImageBuffer, SEQUENCE_LEN);
```

2.7.1 快速傅立叶算法

```
void NuFFT(int * pi32RealBuffer, int * pi32ImageBuffer, uint32_t u32Len)
{
    const uint32_t N = u32Len;
    const uint32_t u32TotalStage = log(N) / log(2);

    uint32_t u32Stage, u32Span, u32Node, u32Twiddle, u32Angle;
    int32_t i32X1Real, i32X1Image, i32X2Real, i32X2Image;

    /* Iterations for log2(N) FFT butterfly stages */
    for (u32Stage=0; u32Stage<u32TotalStage; u32Stage++)
    {
        u32Span = pow(2, u32Stage);

        for (u32Twiddle=0; u32Twiddle<u32Span; u32Twiddle++)
        {
            u32Angle = (N>>1)/u32Span * u32Twiddle;

            for (u32Node=u32Twiddle; u32Node<N; u32Node+=2*u32Span)
            {
                /* Get the real and image part of the input variable X1 */
                i32X1Real  = pi32RealBuffer [u32Node];
                i32X1Image = pi32ImageBuffer[u32Node];

                /* Get the real and image part of the input variable X2 */
                i32X2Real  = pi32RealBuffer [u32Node+u32Span];
                i32X2Image = pi32ImageBuffer[u32Node+u32Span];

                /* Y1 = X1 + X2 * twiddle factor
                 * The real part is   real * cos() + image * sin()
                 * The image part is -real * sin() + image * cos()
                 */
                pi32RealBuffer [u32Node] = \
                          i32X1Real + RESCALE(i32X2Real * COS(u32Angle))
                                    + RESCALE(i32X2Image * SIN(u32Angle));
                pi32ImageBuffer[u32Node] = \
                          i32X1Image - RESCALE(i32X2Real * SIN(u32Angle))
                                     + RESCALE(i32X2Image * COS(u32Angle));
```

```
                /* Y2 = X1 - X2 * twiddle factor
                 * The real part is  -real * cos() - image * sin()
                 * The image part is  real * sin() - image * cos()
                 */
                pi32RealBuffer [u32Node+u32Span] = \
                          i32X1Real  - RESCALE(i32X2Real * COS(u32Angle))
                                     - RESCALE(i32X2Image * SIN(u32Angle));
                pi32ImageBuffer[u32Node+u32Span] = \
                          i32X1Image + RESCALE(i32X2Real * SIN(u32Angle))
                                     - RESCALE(i32X2Image * COS(u32Angle));
            }
        }
    }
}
```

2.8 后置处理

计算并显示频率域信息，程序执行结束。

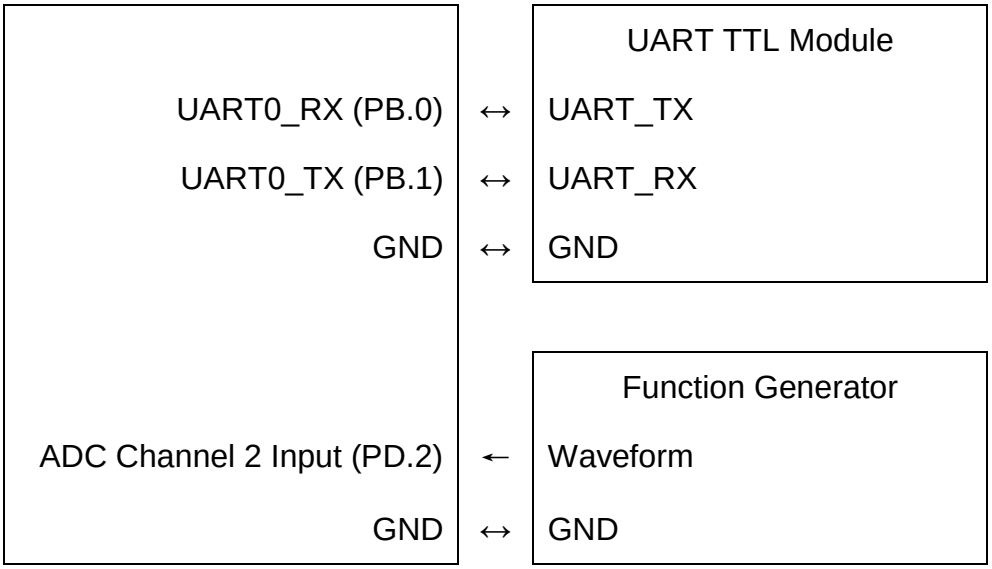
```
    for (i=0; i<SEQUENCE_LEN ;i++)
    {
        /* The amplitude of the spectrum could be presented as the sum of square of the
           real part and the complex part. */
        g_i32Spectrum[i] = (((g_i32RealBuffer[i] * g_i32RealBuffer[i]) +
                            (g_i32ImageBuffer[i] * g_i32ImageBuffer[i])));
    }

    Display_Spectrum(g_i32Spectrum, SEQUENCE_LEN, SAMPLE_RATE);

    while(1);
```

3 软件与硬件环境

- 软件环境
  - BSP 版本
    - NUC121_125_Series_BSP_CMSIS_V3.01.003
  - IDE 版本
    - Keil uVersion 5.26
- 硬件环境
  - 电路组件
    - NuTiny-SDK-NUC121_V1.0 开发版
    - USB-UART-TTL 模块 (Optional)
    - 波形产生器 (Optional)
  - 示意图



4 目录信息

- EC_NUC121_FFT_Demo_V1.00
  - Library — Sample code header and source files
    - CMSIS — Cortex® Microcontroller Software Interface Standard (CMSIS) by Arm® Corp.
    - Device — CMSIS compliant device header file
    - StdDriver — All peripheral driver header and source files
  - SampleCode
    - ExampleCode — Source file of example code

5 如何执行范例程序

1. 根据目录信息章节进入 ExampleCode 路径中的 KEIL 文件夹，双击 NUC121_FFT_Demo.uvprojx。
2. 进入编译模式接口
   a. 编译
   b. 下载代码至内存
   c. 进入 / 离开除错模式
3. 进入除错模式接口
   a. 执行代码

6 修订纪录

Date	Revision	Description
Sep. 30, 2019	1.00	1. 初始发布.

Important Notice

Nuvoton Products are neither intended nor warranted for usage in systems or equipment, any malfunction or failure of which may cause loss of human life, bodily injury or severe property damage. Such applications are deemed, "Insecure Usage".

Insecure usage includes, but is not limited to: equipment for surgical implementation, atomic energy control instruments, airplane or spaceship instruments, the control or operation of dynamic, brake or safety systems designed for vehicular use, traffic signal instruments, all types of safety devices, and other applications intended to support or sustain life.

All Insecure Usage shall be made at customer's risk, and in the event that third parties lay claims to Nuvoton as a result of customer's Insecure Usage, customer shall indemnify the damages and liabilities thus incurred by Nuvoton.

Please note that all data and specifications are subject to change without notice.
All the trademarks of products and companies mentioned in this datasheet belong to their respective owners.