快速傅立葉轉換使用示範

NuMicro® 32 位系列微控制器範例代碼介紹

文件資訊

代碼簡述	使用快速傅立葉轉換將 ADC 取樣訊號轉換至頻率域，示範實作方法與結果。
BSP 版本	NUC121_125_Series_BSP_CMSIS_V3.01.003
開發平台	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 功能介紹

1.1 簡介

快速傅立葉轉換，用於將時間域的離散訊號轉換至頻率域表示，使用者得以觀察該訊號的頻率組成成分與分布情形，是常見的訊號分析手段。此範例將示範如何透過 ADC 與 TIMER 的搭配使用，取樣出一段離散訊號，並進行快速傅立葉轉換，最後觀察頻譜的數值分布是否符合原始輸入訊號的特性。支援三種訊號長度範例，分別為 128 取樣點，256 取樣點，以及 512 取樣點。

1.2 原理

此範例採用 TIMER trigger ADC 的功能進行實作。首先利用波形產生器輸出頻率固定為 60 Hz 的正弦波，模擬一般應用情境下欲分析的訊號來源，給予微控制器使用 ADC 進行取樣。使用者可根據應用上的需求自行調整取樣頻率，注意其數值會直接影響頻譜的頻寬與解析度。每次的 TIMER 逾時事件，硬體便會自動觸發 ADC 進行一次取樣動作，軟體則須於 ADC 轉換完畢後的中斷服務內，將資料以位元反轉的排序寫入緩衝區當中。

本範例程式提供必要參數的查表數值，可簡化微處理機的運算量，包含傅立葉轉換過程所需要的旋轉因子常數，以及 Decimate-in-time 作法所需的位元反轉排序，開發者已經事前計算完畢並定義在標頭檔當中。由於尋找表的內容會根據訊號長度的不同而有異，不同長度設定的尋找表無法共用，使用者必須於標頭檔中選擇範例提供的指定長度參數編譯，流程主要可分為五個部分:

1. 指定訊號長度以及取樣頻率
2. 系統與周邊硬體初始化
3. ADC 取樣至指定數量為止
4. 執行傅立葉轉換
5. 後置處理，顯示頻譜資訊

1.3 執行結果

1.3.1 起始畫面

等待使用者確認 ADC 輸入腳位 PD.2 已經接上待量測的訊號源，按下任意建開始取樣。

```
+------------------------------------------------------------+
|            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....
```

圖 1-1 Terminal 起始畫面

1.3.2 時間域資料顯示

取樣完畢後顯示時間域訊號資訊，如圖 1-2 所示。因資料列數過多，僅擷取部分資料數值顯示。顯示內容分為三欄，左側欄位為資料的原取樣順序，中間欄位為資料根據位元反轉表寫入緩衝區的位置，右側欄位則為 ADC 數值經處理後實際寫入緩衝區的資料。

```
+------------------------------------------------------------+
| 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
```

圖 1-2 Terminal 局部時間域訊號顯示

1.3.3 頻率域資料顯示

圖1-3為執行傅立葉轉換後得出的局部頻譜，顯示內容分為四欄，由左至右。第一欄為資料索引，第二欄為該點所表示的頻率，第三欄為原訊號含有該頻率訊號的成份大小，由頻率域資料實數部分與虛數部分的平方和求得，單位可先忽略不計。最後一欄則表示該頻率成份於此頻譜佔有的強度百分比。藉由觀察第四欄可看出該訊號的組成成分幾乎正好落在 60 Hz 的頻率點上，符合波形產生器給予的 60 Hz 正弦波頻率。

```
+------------------------------------------------------------+
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%
```

圖 1-3 Terminal 局部頻率域資訊顯示

1.3.4 時間域資料作圖

擷取整份時間域資料進行作圖，可得到下圖1-4，橫坐標為取樣點索引，縱座標為 ADC 轉換數值，呈現出標準的正弦波。

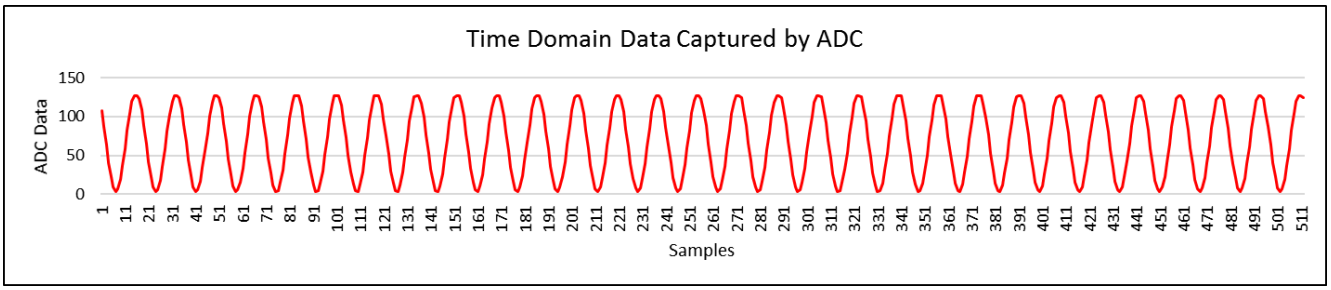


圖1-4 NUC121 ADC 取樣點作圖

1.3.5 頻率域資料作圖

擷取整份頻率域資料進行作圖，可得到下圖1-5。經過傅立葉轉換後得到的頻譜，其左半部以及右半部會呈現出共軛對稱分布。假設使用者所設定的取樣頻率為F_s，則可以得到該訊號組成頻率為 $F_s/2 \sim -F_s/2$ 區間內的強度分布，其中負頻率的部分可以忽略。橫坐標為頻率，縱座標為強度百分比，由圖中可清楚地觀察到該訊號的能量集中在頻率為 60 Hz 的座標點上。

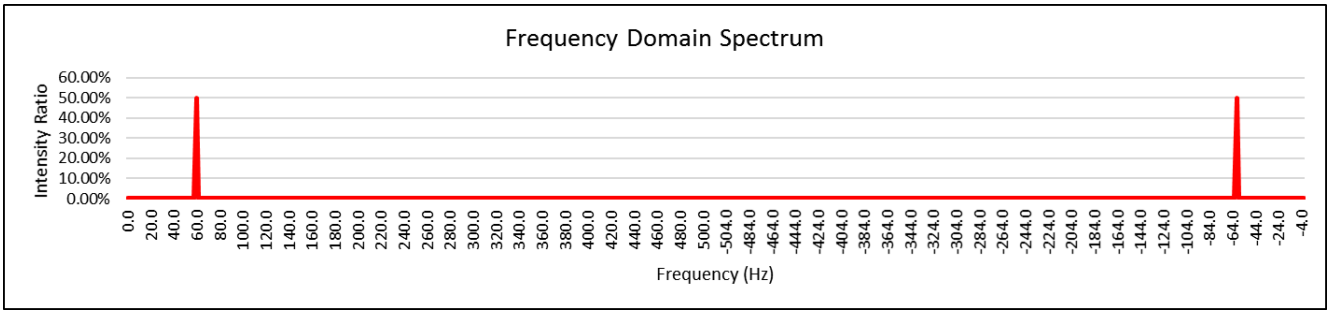


圖1-5 傅立葉轉換頻率域作圖

2 代碼介紹

2.1 指定訊號長度

定義訊號的長度，此範例提供 128 取樣點，256 取樣點，與 512 取樣點。長度越長，頻譜的解析度越高，但所需的資源與運算時間也會相對提升。

```
#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 依據長度設定配給尋找表

提供旋轉因子的正弦 SIN_LUT 與餘弦數值表 COS_LUT，以及位元反轉數值表 REV_LUT。旋轉因子 $exp(-j2\,\pi k/N)$ 為傅立葉轉換當中，取樣點所必須乘上的複數常數，可拆解為實部以及虛部兩項 $\cos(2\pi k/N) - j\,\sin(2\pi k/N)$ 。並根據先前定義的訊號長度，給予的第 k 角度正弦與餘弦的常數值。位元反轉表用於重新排列原本的時間域訊號順序，根據 Decimate-in-time 的作法，時間域訊號必須先依照此規則排序再進行後續計算，因此資料會依據此表格內容填入緩衝區的指定位址。

```
/* 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 參數設定

取樣頻率 SAMPLE_RATE 由使用者根據應用需求進行配置。SAMPLE_DATA_RSHIFT 參數用以防止傅立葉轉換時發生溢位，將 ADC 的原始資料向右位移指定數量個位元，在不發生溢位的前提下，此參數數值建議盡可能地減少，可以保留更多原始資料的特徵，可由使用者自行配置。

```
/* 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 計數器初始化設定

設定 TIMER0 工作頻率為取樣頻率，開啟 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 初始化設定

```
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 中斷服務處理

轉換完畢的 ADC 原始資料會先經過數值縮減處理，再以位元反轉的順序寫入緩衝區。

```
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 快速傅立葉轉換

當指定長度的訊號已經全部取樣完畢後，首先移除訊號的直流成份(平均值)，並執行快速傅立葉演算法。

```
    /* 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 快速傅立葉演算法

```
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 後置處理

計算並顯示頻率域資訊，程式執行結束。

```
    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);
```

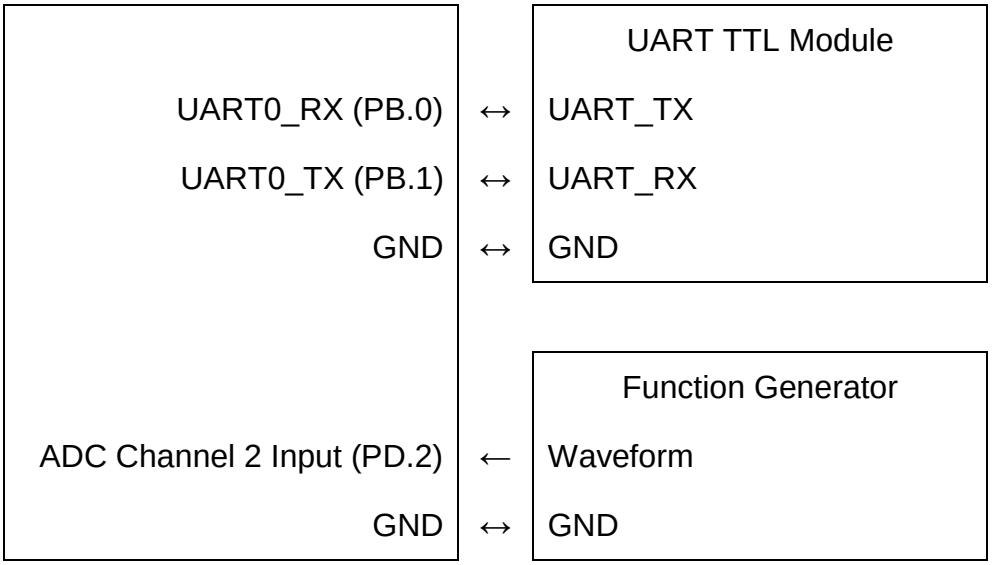
3 軟體與硬體環境

● 軟體環境

- ■ BSP 版本
  - ◆ NUC121_125_Series_BSP_CMSIS_V3.01.003
- ■ IDE 版本
  - ◆ Keil uVersion 5.26

● 硬體環境

- ■ 電路元件
  - ◆ NuTiny-SDK-NUC121_V1.0 開發版
  - ◆ USB-UART-TTL 模組 (Optional)
  - ◆ 波形產生器 (Optional)
- ■ 示意圖



4 目錄資訊

- 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 如何執行範例程式

1. 根據目錄資訊章節進入 ExampleCode 路徑中的 KEIL 資料夾，雙擊 NUC121_FFT_Demo.uvprojx。
2. 進入編譯模式介面
   a. 編譯
   b. 下載代碼至記憶體
   c. 進入 / 離開除錯模式
3. 進入除錯模式介面
   a. 執行代碼

6 修訂紀錄

Date	Revision	Description
Sep. 30, 2019	1.00	1. 初始發布.

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.