M460 MP3 錄音機

NuMicro® 32 位系列微控制器範例代碼介紹

文件資訊

應用簡述	本範例將音頻編解碼器錄製到的聲音編碼為 MP3 格式並將其儲存至微型 SD 卡中，而此 MP3 檔也可以通過音頻編解碼器來撥放
BSP 版本	M460 BSP CMSIS V3.00.001
開發平台	NuMaker-M467HJ 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 and microprocessor 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 概述

本範例使用音頻編解碼器來錄製聲音，將此聲音數據編碼為 MP3 格式後，會將編碼後的數據寫入微型 SD 卡中。

為了使用此範例代碼，使用者需要準備一個微型 SD 卡並將其插入 T-Flash 卡槽。接著，使用者可以按下按鈕 BTN0 來開始錄製聲音，當按下按鈕 BTN1 時即會停止錄製。MP3 檔會在錄製完成後自動撥放，也可以透過按下按鈕 BTN1 來進行撥放。使用者可以藉由定義 REC_IN_RT 與否來決定是否進行實時錄音。如果有定義 REC_IN_RT，當接收到聲音數據的同時，會將其編碼為 MP3 格式並儲存至微型 SD 卡。相反地，如果沒有定義 REC_IN_RT，則錄製到的聲音數據會先儲存在 HyperRAM，直到停止錄製後才開始將這些數據編碼為 MP3 格式並進一步儲存至微型 SD 卡中。

此 MP3 錄音機適用於人聲錄製。支持單聲道模式，數據長度為 16 位，取樣率預設為 16 千赫茲，比特率預設為 64 千位每秒。此外，該錄音機提供了不同取樣率及比特率的組合，如表 1-1 所列，使用者可以藉由定義 REC_SAMPLE_RATE 及 REC_BIT_RATE 來做配置。

取樣率（赫茲）	比特率（千位每秒）
32,000	<= 128
16,000	<= 64
8,000	<= 32

表 1-1 支持的取樣率及比特率

1.1 執行結果

執行結果以及打印資料顯示如下。

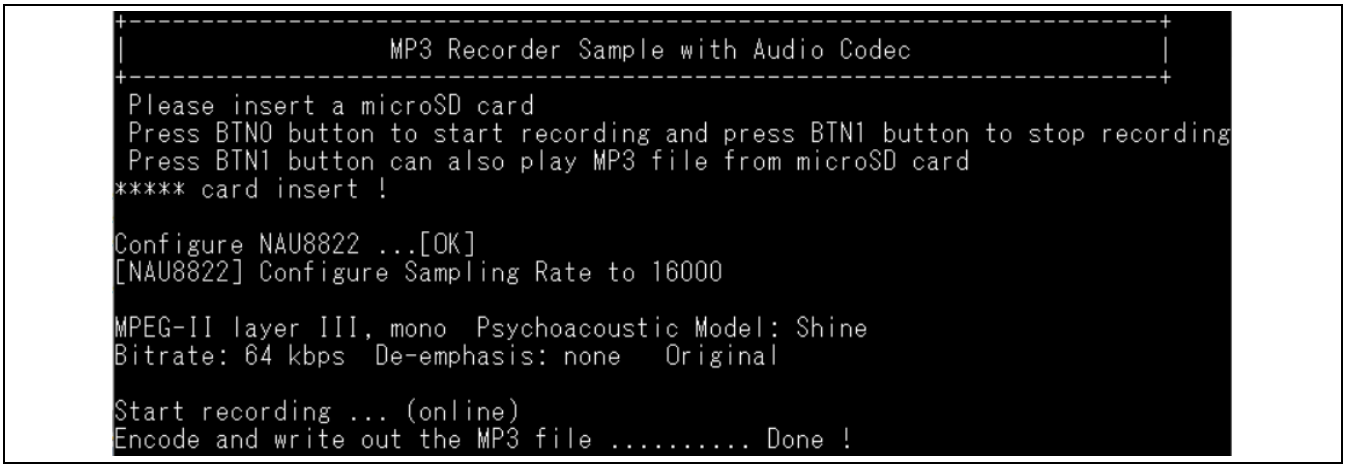

圖 1-1 在線錄音執行結果

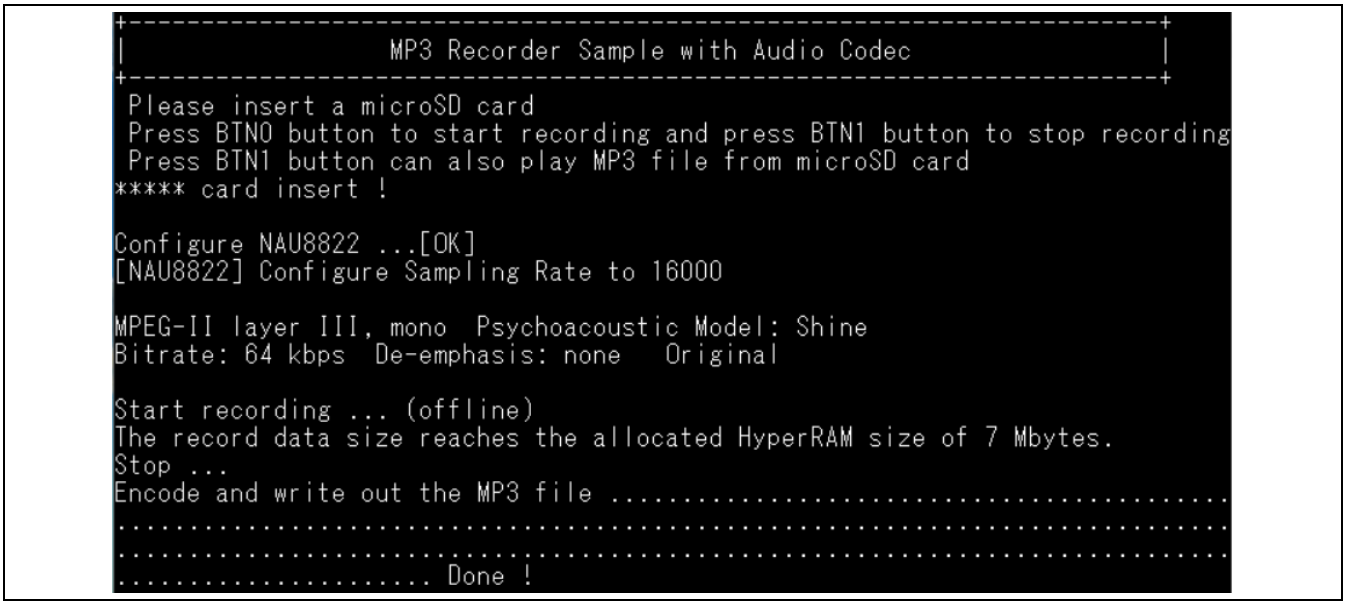

圖 1-2 離線錄音執行結果

圖 1-3 播放聲音執行結果

2 代碼介紹

本範例為使用者提供在線錄音及離線錄音，使用者可以根據 config.h 中是否定義了 REC_IN_RT 來選擇其中一個使用。

在 main.c 的 main() 中，呼叫 SYS_Init() 及 SD_Init() 來初始化周邊時鐘及相關的多功能引腳，包含 UART、I²S、I²C、PDMA、HBI 以及 SD。在 while(1) 迴圈中，它會一直檢查 PH.0 與 PH.1 引腳的狀態以及 g_u32RecordStart 與 g_u32RecordDone 的數值來開始或停止錄製。

當 PH.0 為 1 且 g_u32RecordStart 與 g_u32RecordDone 為 0 時，首先，只在 REC_IN_RT 沒被定義時呼叫 ClearHyperRAM() 來清除 HyperRAM 裡所有數據，接著呼叫 Recorder_Init() 來配置錄音條件、初始化 I²S、音頻編解碼器以及編碼器，然後使能 I²S 接收功能來開始接收聲音數據。若沒有定義 REC_IN_RT，當 g_u32RecordDone 為 1 時，禁用 I²S 接收功能以停止接收聲音數據，接著呼叫 MP3Recorder() 來將聲音數據編碼為 MP3 格式且將編碼後的數據儲存至微型 SD 卡。

若 REC_IN_RT 有被定義且 g_u32RecordStart 為 1，當 I²S 每接收到一個音框點數的數據時，編碼聲音數據以及寫微型 SD 卡的動作就會被執行。值得注意的是，如果編碼聲音數據及寫微型 SD 卡的時間大於 I²S 接收一個音框點數數據的時間時，I²S 在這段時間所接收到的聲音數據將不會被處理。這個情況會在微型 SD 卡性能不佳時發生，此時，錯誤旗標 g_u32ErrorFlag 將不再為零，UART 信息會告知使用者部分聲音已經遺失。如果使用者想要得到高的錄音品質，請更換更高性能或者更高速的微型 SD 卡。

而在停止錄音後或開始錄音前，當 PH.1 為 1 時，呼叫 MP3Player() 可以撥放微型 SD 卡中的 MP3 檔：

```
int32_t main(void)
{
    int32_t i32Written;
    uint8_t *pu8Data;
    uint32_t u32PrintFlag = 1;

    TCHAR sd_path[] = { '0', ':', 0 };    /* SD drive started from 0 */

    /* Unlock protected registers */
    SYS_UnlockReg();

    /* Init System, peripheral clock and multi-function I/O */
    SYS_Init();

    /* Init SD */
    SD_Init();
```

```
    /* Init UART to 115200-8n1 for print message */
    UART_Open(UART0, 115200);

    printf("+-----------------------------------------------------------------------+\n");
    printf("|                  MP3 Recorder Sample with Audio Codec                 |\n");
    printf("+-----------------------------------------------------------------------+\n");
    printf(" Please insert a microSD card\n");
    printf(" Press BTN0 button to start recording and press BTN1 button to stop recording
            \n");
    printf(" Press BTN1 button can also play MP3 file from microSD card\n");

    /* Configure PH.0 and PH.1 as Output mode */
    GPIO_SetMode(PH, BIT0, GPIO_MODE_OUTPUT);
    GPIO_SetMode(PH, BIT1, GPIO_MODE_OUTPUT);

    /* Enable PH.1 interrupt by falling edge trigger */
    GPIO_EnableInt(PH, 1, GPIO_INT_FALLING);
    NVIC_EnableIRQ(GPH_IRQn);
    NVIC_SetPriority(GPH_IRQn, (1 << __NVIC_PRIO_BITS) - 2);

    /* Configure FATFS */
    SDH_Open_Disk(SDH0, CardDetect_From_GPIO);
    f_chdrive(sd_path);          /* Set default path */

    /* Init I2C2 to access audio codec */
    I2C2_Init();

    /* Select source from HXT(12MHz) */
    CLK_SetModuleClock(I2S0_MODULE, CLK_CLKSEL3_I2S0SEL_HXT, 0);

    while (1)
    {
        /* Read pin state of PH.0 and PH.1 */
        u32BTN0 = (PH->PIN & (1 << 0)) ? 0 : 1;
        u32BTN1 = (PH->PIN & (1 << 1)) ? 0 : 1;

        if (SD0.IsCardInsert == TRUE)
        {
#ifdef REC_IN_RT

            /* Inform user about microSD card usage */
            if ((u32PrintFlag == 1) && (g_u32ErrorFlag != 0))
            {
                printf("\n\nSome sounds have been lost due to poor microSD card
                        performance.\nPlease replace !!!\n\n");

                u32PrintFlag = 0;
                g_u32ErrorFlag = 0;
            }

            /* Encode sound data to MP3 format and store in microSD card */
            if (g_u32RecordStart == 1)
            {
                if ((g_u32WriteSDToggle == 0) && (g_u32BuffPos1 > 0) &&
                    (g_u32BuffPos1 == ((samples_per_pass * config.wave.channels) >> 1)))
                {
                    pu8Data = shine_encode_buffer_interleaved(s,
                              (int16_t *)(&g_au32PcmBuff1), (int *)&i32Written);

                    if (Write_MP3(i32Written, pu8Data, &config) != i32Written)
                    {
                        printf("shineenc: write error\n");
                    }

                    g_u32BuffPos1 = 0;
                    g_u32WriteSDToggle = 1;
```

```
                }
                else if ((g_u32WriteSDToggle == 1) && (g_u32BuffPos2 > 0) &&
                         (g_u32BuffPos2 == ((samples_per_pass * config.wave.channels)
                         >> 1)))
                {
                    pu8Data = shine_encode_buffer_interleaved(s,
                              (int16_t *)(&g_au32PcmBuff2), (int *)&i32Written);

                    if (Write_MP3(i32Written, pu8Data, &config) != i32Written)
                    {
                        printf("shineenc: write error\n");
                    }

                    g_u32BuffPos2 = 0;
                    g_u32WriteSDToggle = 0;
                }
            }

#endif

            if ((u32BTN0 == 1) && (g_u32RecordStart == 0) && (g_u32RecordDone == 0))
            {
#ifndef REC_IN_RT

                /* Clear HyperRAM */
                if (ClearHyperRAM(HYPER_RAM_MEM_MAP, HYPER_RAM_MEM_MAP + 0x800000) < 0)
                    return -1;

#endif

                /* Configure recording condition, init I2S, audio codec and encoder */
                Recorder_Init();

#ifdef REC_IN_RT

                printf("Start recording ... (online)\n");

#else

                printf("Start recording ... (offline)\n");

#endif

                /* Enable I2S RX function to receive sound data */
                I2S_ENABLE_RX(I2S0);

                g_u32RecordStart = 1;

#ifdef REC_IN_RT

                u32PrintFlag = 1;

                printf("Encode and write out the MP3 file ");

#endif
            }

            /* Play MP3 */
            if ((u32BTN1 == 1) && (g_u32RecordStart == 0))
            {
                MP3Player();
            }

            if (g_u32RecordDone == 1)
            {
                /* Disable I2S RX function */
```

```
                I2S_DISABLE_RX(I2S0);

#ifndef REC_IN_RT

                /* Encode sound data to MP3 format and store in microSD card */
                MP3Recorder();

#endif

                /* Close encoder */
                shine_close(s);

                f_close(&mp3FileObject);

                printf(" Done !\n\n");

                g_u32RecordStart = 0;
                g_u32RecordDone = 0;

                /* Play MP3 */
                MP3Player();
            }
        }
    }
}
```

在 mp3recorder.c 的 Recorder_Init() 中，它會初始化 I²S 介面、音頻編解碼器以及編碼器。呼叫 f_open() 以在微型 SD 卡中建立一個由 MP3_FILE 定義檔名的可寫檔案。使用者可以藉由定義 config.h 中 REC_SAMPLE_RATE 及 REC_BIT_RATE 來配置錄音取樣率以及錄音比特率：

```
void Recorder_Init(void)
{
    FRESULT res;

    /* Open I2S0 interface and set to slave mode, audio format, I2S format */
    I2S_Open(I2S0, I2S_MODE_SLAVE, REC_SAMPLE_RATE, I2S_DATABIT_16, REC_FORMAT,
             I2S_FORMAT_I2S);

    NVIC_EnableIRQ(I2S0_IRQn);
    NVIC_SetPriority(I2S0_IRQn, 4);

    /* Set MCLK and enable MCLK */
    I2S_EnableMCLK(I2S0, 12000000);

    /* NAU8822 will store data in left channel */
    I2S_SET_MONO_RX_CHANNEL(I2S0, I2S_MONO_LEFT);

    /* Initialize NAU8822 codec */
    NAU8822_Setup();

    /* Configure NAU8822 to specific sample rate */
    NAU8822_ConfigSampleRate(REC_SAMPLE_RATE);

    /* Enable RX threshold level interrupt */
    I2S_EnableInt(I2S0, I2S_IEN_RXTHIEN_Msk);

    res = f_open(&mp3FileObject, MP3_FILE, FA_CREATE_ALWAYS | FA_WRITE);

    if (res != FR_OK)
    {
        printf("Open file error!\n");
```

```
        return;
    }

    set_defaults(&config);

    if (REC_FORMAT == I2S_MONO)
        config.wave.channels = 1;
    else
        config.wave.channels = 2;

    config.wave.samplerate = REC_SAMPLE_RATE;
    config.mpeg.bitr = REC_BIT_RATE;

    /* See if samplerate and bitrate are valid */
    if (shine_check_config(config.wave.samplerate, config.mpeg.bitr) < 0)
        printf("Unsupported samplerate/bitrate configuration.");

    /* Set to stereo mode if wave data is stereo, mono otherwise. */
    if (config.wave.channels > 1)
        config.mpeg.mode = STEREO;
    else
        config.mpeg.mode = MONO;

    /* Initiate encoder */
    s = shine_initialise(&config);

    check_config(&config);

    samples_per_pass = shine_samples_per_pass(s);
}
```

在 isr.c 的 I2S0_IRQHandler() 中，如果有定義 REC_IN_RT，接收到的每個音框點數的聲音數據會被輪流儲存在兩個 SRAM 緩衝器。當接收到的數據點數達到一個音框時，它會一直檢查 g_u32WriteSDToggle 來確認上一筆被編碼的音框數據是否已經被寫入微型 SD 卡了。如果 g_u32WriteSDToggle 不符合預期，g_u32ErrorFlag 會往上計數。相反地，如果 REC_IN_RT 沒被定義，則接收到的聲音數據會被儲存在 HyperRAM 中。其間，若錄製到的聲音數據點數到達所配置的 HyperRAM 大小 7 兆字節時，它會自動停止 I²S 的接收功能：

```
void I2S0_IRQHandler(void)
{
    uint32_t u32Reg;
    uint32_t u32Len, i;
    int32_t *pi32BuffRx1, *pi32BuffRx2;

    u32Reg = I2S_GET_INT_FLAG(I2S0, I2S_STATUS0_RXTHIF_Msk);

    if (u32Reg & I2S_STATUS0_RXTHIF_Msk)
    {
        /* Read RX FIFO Level */
        u32Len = I2S_GET_RX_FIFO_LEVEL(I2S0);

#ifdef REC_IN_RT

        if ((g_u32BuffPos1 > 0) &&
            (g_u32BuffPos1 == ((samples_per_pass * config.wave.channels) >> 1)))
        {
            if (g_u32WriteSDToggle != 0)
            {
                g_u32ErrorFlag++;
                g_u32BuffPos1 = g_u32BuffPos2 = 0;
```

```
            }

            g_u32BuffToggle = 1;
        }

        if ((g_u32BuffPos2 > 0) &&
            (g_u32BuffPos2 == ((samples_per_pass * config.wave.channels) >> 1)))
        {
            if (g_u32WriteSDToggle != 1)
            {
                g_u32ErrorFlag++;
                g_u32BuffPos1 = g_u32BuffPos2 = 0;
            }

            g_u32BuffToggle = 0;
        }

        if (g_u32BuffToggle == 0)
        {
            pi32BuffRx1 = (int32_t *)&g_au32PcmBuff1[g_u32BuffPos1];

            for (i = 0; i < u32Len; i++)
            {
                pi32BuffRx1[i] = I2S_READ_RX_FIFO(I2S0);
            }

            g_u32BuffPos1 += u32Len;
        }
        else if (g_u32BuffToggle == 1)
        {
            pi32BuffRx2 = (int32_t *)&g_au32PcmBuff2[g_u32BuffPos2];

            for (i = 0; i < u32Len; i++)
            {
                pi32BuffRx2[i] = I2S_READ_RX_FIFO(I2S0);
            }

            g_u32BuffPos2 += u32Len;
        }

#else

        for (i = 0; i < u32Len; i++)
        {
            outp32(HYPER_RAM_MEM_MAP + g_u32BuffPos + 4 * i, I2S_READ_RX_FIFO(I2S0));
        }

        g_u32BuffPos += 4 * u32Len;

        if (g_u32BuffPos >= 0x700000)
        {
            /* Disable I2S RX function */
            I2S_DISABLE_RX(I2S0);
            g_u32RecordDone = 1;

            printf("The record data size reaches the allocated HyperRAM size of 7
                    Mbytes.\n");
        }

#endif
    }
}
```

在 mp3recorder.c 的 MP3Recorder() 中，它會呼叫 shine_encode_buffer_interleaved() 來將儲存在 HyperRAM 的數據做編碼，以及呼叫 Write_MP3() 將編碼為 MP3 格式的數據寫入 MP3_FILE 檔中：

```
void MP3Recorder(void)
{
    int32_t i32Written;
    uint8_t *pu8Data;
    uint32_t u32BuffPos = 0;
    uint32_t u32Len;

    u32Len = samples_per_pass * config.wave.channels;

    printf("Stop ...\n");
    printf("Encode and write out the MP3 file ");

    for (u32BuffPos = 0; u32BuffPos < g_u32BuffPos; u32BuffPos += 2 * u32Len)
    {
        pu8Data = shine_encode_buffer_interleaved(s,
                  (int16_t *)(HYPER_RAM_MEM_MAP + u32BuffPos), (int *)&i32Written);

        if (Write_MP3(i32Written, pu8Data, &config) != i32Written)
        {
            printf("shineenc: write error\n");
        }
    }

    g_u32BuffPos = 0;
}

int32_t Write_MP3(long bytes, void *buffer, void *config)
{
    f_write(&mp3FileObject, buffer, bytes, &ReturnSize);

    if (i32Cnt++ >= (bytes / 10))
    {
        i32Cnt = 0;
        printf(".");
    }

    return bytes;
}
```

3 軟體與硬體需求

3.1 軟體需求

- BSP 版本
  - M460 Series BSP CMSIS V3.00.001
- IDE 版本
  - Keil uVersion 5.28

3.2 硬體需求

- 電路元件
  - M467HJHAE
  - T-Flash 卡 (B8502A-13SB-HPA)
  - 音頻編解碼器 (NAU88C22 帶有 9.7D2.54mm 麥克風及 2SJ3080-000111F 耳機)
  - HyperRAM (W956A8MBYA5I)
  - 按鈕 (SW-2P-SMD)
- 示意圖

		V_{DD}	↔	V_{DD}		
		GND	↔	GND		
		SD0_DAT0 (PE2)	↔	DAT0		
		SD0_DAT1 (PE3)	↔	DAT1		
	M467	SD0_DAT2 (PE4)	↔	DAT2	T-Flash 卡	
		SD0_DAT3 (PE5)	↔	DAT3		
		SD0_CLK (PE6)	↔	CLK		

		SD0_CMD (PE7)	↔	CMD	
		SD0_nCS (PD13)	↔	CD	
		V_{DD}	↔	V_{DD}	
		GND	↔	GND	
		I2C2_SDA (PD0)	↔	SDIO	
		I2C2_SCL (PD1)	↔	SCLK	
	M467	I2S0_BCLK (PI6)	↔	BCLK	音頻編解碼器
		I2S0_MCLK (PI7)	↔	MCLK	
		I2S0_DI (PI8)	↔	ADCOUT	
		I2S0_DO (PI9)	↔	DACIN	
		I2S0_LRCK (PI10)	↔	FS	
		V_{DD}	↔	V_{DD}	
		GND	↔	GND	
		HBI_nRESET (PJ2)	↔	RESET#	
		HBI_D3 (PJ3)	↔	DQ3	
		HBI_D2 (PJ4)	↔	DQ2	
		HBI_D1 (PJ5)	↔	DQ1	
		HBI_D0 (PJ6)	↔	DQ0	
	M467	HBI_nCS (PJ7)	↔	CS#	HyperRAM
		HBI_nCK (PH12)	↔	CK#	
		HBI_CK (PH13)	↔	CK	
		HBI_RWDS (PH14)	↔	RSDS	
		HBI_D4 (PH15)	↔	DQ4	
		HBI_D5 (PD7)	↔	DQ5	
		HBI_D6 (PD6)	↔	DQ6	

	HBI_D7 (PD5)	↔	DQ7	
	V_{DD}	↔	V_{DD}	
M467	GND	↔	GND	按鈕
	BTN0 (PH0)	↔	BTN_0	
	BTN1 (PH1)	↔	BTN_1	

圖 3-1 線路連接

4 目錄資訊

EC_M460_MP3_Recorder_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
ThirdParty	
FatFs	A generic FAT file system module for small embedded systems. Its official website is: http://elm-chan.org/fsw/ff/00index_e.html.
LibMAD	A MPEG audio decoder library that currently supports MPEG-1 and the MPEG-2 extension to lower sampling frequencies, as well as the de facto MPEG 2.5 format. All three audio layers — Layer I, Layer II, and Layer III (i.e. MP3) are fully implemented. This library is distributed under GPL license. Please contact Underbit Technologies (http://www.underbit.com/) for the commercial license.
LibShine	A blazing fast MP3 encoding library implemented in fixed-point arithmetic. Its official website is: https://github.com/toots/shine.

圖 4-1 目錄資訊

5 範例程式執行

1. 根據目錄資訊章節進入 ExampleCode 路徑中的 KEIL 資料夾，雙擊 M460_MP3_Recorder.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
2022.05.24	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.