# M460 MP3 录音机

NuMicro® 32 位系列微控制器范例代码介绍

## 文件信息

| | |
|---|---|
| 应用简述 | 本范例将音频编解码器录制到的声音编码为 MP3 格式并将其储存至微型 SD 卡中，而此 MP3 档也可以通过音频编解码器来播放 |
| BSP 版本 | M460 BSP CMSIS V3.00.001 |
| 开发平台 | 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

本范例使用音频编解码器来录制声音，将此声音数据编码为 MP3 格式后，会将编码后的数据写入微型 SD 卡中。

为了使用此范例代码，用户需要准备一个微型 SD 卡并将其插入 T-Flash 卡槽。接着，用户可以按下按钮 BTN0 来开始录制声音，当按下按钮 BTN1 时即会停止录制。MP3 档会在录制完成后自动播放，也可以透过按下按钮 BTN1 来进行播放。用户可以藉由定义 REC_IN_RT 与否来决定是否进行实时录音。如果有定义 REC_IN_RT，当接收到声音数据的同时，会将其编码为 MP3 格式并储存至微型 SD 卡。相反地，如果没有定义 REC_IN_RT，则录制到的声音数据会先储存在 HyperRAM，直到停止录制后才开始将这些数据编码为 MP3 格式并进一步储存至微型 SD 卡中。

此 MP3 录音机适用于人声录制。支持单声道模式，数据长度为 16 位，采样率预设为 16 千赫兹，比特率预设为 64 千位每秒。此外，该录音机提供了不同采样率及比特率的组合，如表 1-1 所列，用户可以藉由定义 REC_SAMPLE_RATE 及 REC_BIT_RATE 来做配置。

| 采样率（赫兹） | 比特率（千位每秒） |
|---|---|
| 32,000 | <= 128 |
| 16,000 | <= 64 |
| 8,000 | <= 32 |

表 1-1 支持的采样率及比特率

## 1.1 执行结果

执行结果，以及打印数据。

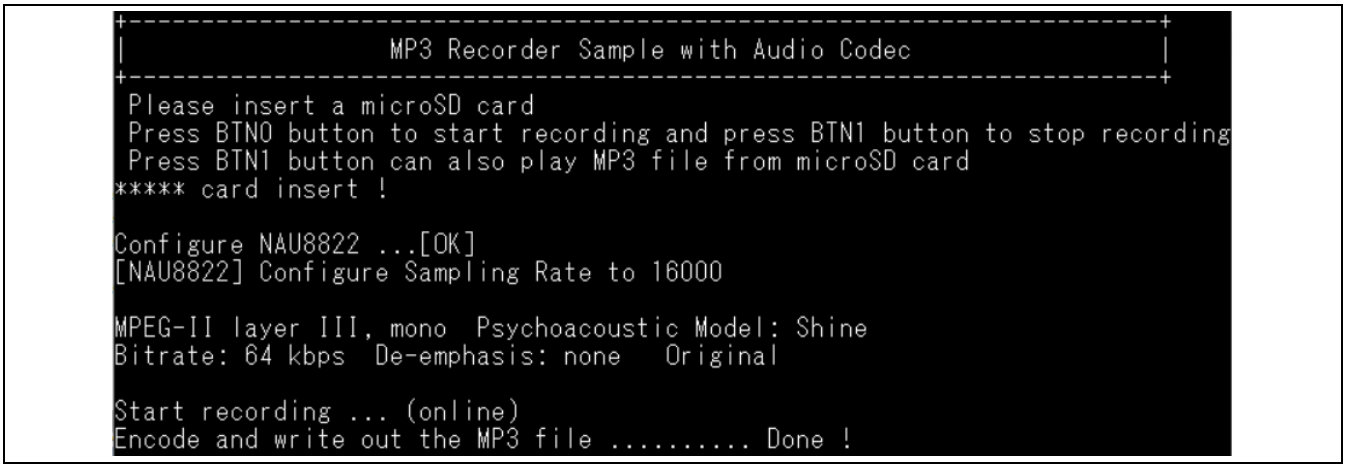


图 1-1 在线录音执行结果

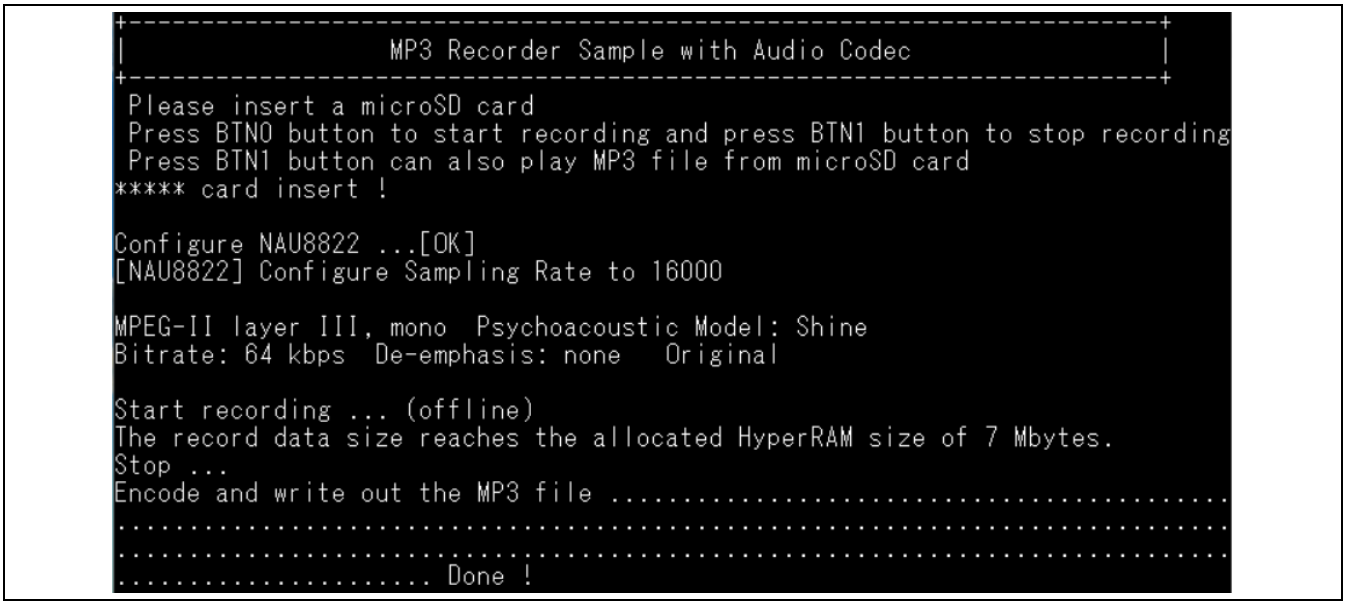


图 1-2 脱机录音执行结果

图 1-3 播放声音执行结果

# 2 代码介绍

本范例为用户提供在线录音及脱机录音，用户可以根据 config.h 中是否定义了 REC_IN_RT 来选择其中一个使用。

在 main.c 的 main() 中，调用 SYS_Init() 及 SD_Init() 来初始化外设时钟及相关的多功能引脚，包含 UART、I²S、I²C、PDMA、HBI 以及 SD。在 while(1) 循环中，它会一直检查 PH.0 与 PH.1 引脚的状态以及 g_u32RecordStart 与 g_u32RecordDone 的数值来开始或停止录制。

当 PH.0 为 1 且 g_u32RecordStart 与 g_u32RecordDone 为 0 时，首先，只在 REC_IN_RT 没被定义时调用 ClearHyperRAM() 来清除 HyperRAM 里所有数据，接着调用 Recorder_Init() 来配置录音条件、初始化 I²S、音频编解码器以及编码器，然后使能 I²S 接收功能来开始接收声音数据。若没有定义 REC_IN_RT，当 g_u32RecordDone 为 1 时，禁用 I²S 接收功能以停止接收声音数据，接着调用 MP3Recorder() 来将声音数据编码为 MP3 格式且将编码后的数据储存至微型 SD 卡。

若 REC_IN_RT 有被定义且 g_u32RecordStart 为 1，当 I²S 每接收到一帧数据时，编码声音数据以及写微型 SD 卡的动作就会被执行。值得注意的是，如果编码声音数据及写微型 SD 卡的时间大于 I²S 接收一帧数据的时间时，I²S 在这段时间所接收到的声音数据将不会被处理。这个情况会在微型 SD 卡性能不佳时发生，此时，错误标志 g_u32ErrorFlag 将不再为零，UART 信息会告知用户部分声音已经遗失。如果用户想要得到高的录音质量，请更换更高性能或者更高速的微型 SD 卡。

而在停止录音后或开始录音前，当 PH.1 为 1 时，调用 MP3Player() 可以播放微型 SD 卡中的 MP3 档：

```
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，它会初始化 I²S 接口、音频编解码器以及编码器。调用 f_open() 以在微型 SD 卡中建立一个由 MP3_FILE 定义档名的可写文件。用户可以藉由定义 config.h 中 REC_SAMPLE_RATE 及 REC_BIT_RATE 来配置录音采样率以及录音比特率：

```
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，如果有定义 REC_IN_RT，接收到的每帧声音数据会被轮流储存在两个 SRAM 缓冲器。当接收到的数据达到一个帧时，它会一直检查 g_u32WriteSDToggle 来确认上一笔被编码的音频数据是否已经被写入微型 SD 卡了。如果 g_u32WriteSDToggle 不符合预期，g_u32ErrorFlag 会往上计数。相反地，如果 REC_IN_RT 没被定义，则接收到的声音数据会被储存在 HyperRAM 中。其间，若录制到的声音数据点数到达所配置的 HyperRAM 大小 7 兆字节时，它会自动停止 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，它会调用 shine_encode_buffer_interleaved() 来将储存在 HyperRAM 的数据做编码，以及调用 Write_MP3() 将编码为 MP3 格式的数据写入 MP3_FILE 档中：

```
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 软件与硬件需求

## 3.1 软件需求

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

## 3.2 硬件需求

- 电路组件
  - M467HJHAE
  - T-Flash 卡 (B8502A-13SB-HPA)
  - 音频编解码器 (NAU88C22 带有 9.7D2.54mm 麦克风及 2SJ3080-000111F 耳机)
  - HyperRAM (W956A8MBYA5I)
  - 按钮 (SW-2P-SMD)
- 示意图

| | | | | | | |
|---|---|---|---|---|---|---|
| | | $V_{DD}$ | ↔ | $V_{DD}$ | | |
| | | 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 | |
| | M467 | $V_{DD}$ | ↔ | $V_{DD}$ | 音频编解码器 |
| | | GND | ↔ | GND | |
| | | I2C2_SDA (PD0) | ↔ | SDIO | |
| | | I2C2_SCL (PD1) | ↔ | SCLK | |
| | | I2S0_BCLK (PI6) | ↔ | BCLK | |
| | | I2S0_MCLK (PI7) | ↔ | MCLK | |
| | | I2S0_DI (PI8) | ↔ | ADCOUT | |
| | | I2S0_DO (PI9) | ↔ | DACIN | |
| | | I2S0_LRCK (PI10) | ↔ | FS | |
| | M467 | $V_{DD}$ | ↔ | $V_{DD}$ | HyperRAM |
| | | GND | ↔ | GND | |
| | | HBI_nRESET (PJ2) | ↔ | RESET# | |
| | | HBI_D3 (PJ3) | ↔ | DQ3 | |
| | | HBI_D2 (PJ4) | ↔ | DQ2 | |
| | | HBI_D1 (PJ5) | ↔ | DQ1 | |
| | | HBI_D0 (PJ6) | ↔ | DQ0 | |
| | | HBI_nCS (PJ7) | ↔ | CS# | |
| | | 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 | 按钮 |
| | | BTN0 (PH0) | ↔ | BTN_0 | |
| | | BTN1 (PH1) | ↔ | BTN_1 | |

图 3-1 线路连接

# 4 目录信息

| | |
|---|---|
| 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. |

图 4-1 目录信息

# 5 范例程序执行

1. 根据目录信息章节进入 ExampleCode 路径中的 KEIL 文件夹，双击 M460_MP3_Recorder.uvprojx。
2. 进入编译模式界面
   a. 编译
   b. 下载代码至内存
   c. 进入 / 离开仿真模式
3. 进入仿真模式界面
   a. 执行代码

# 6 修订纪录

| Date | Revision | Description |
|---|---|---|
| 2022.05.24 | 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.*