# NUC505 WAV Player from internal Flash

Example Code Introduction for 32-bit NuMicro® Family

## Information

| | |
|---|---|
| Application | A WAV player demo using internal codec to play WAV file stored in internal SPI flash. |
| BSP Version | NUC505 Series BSP CMSIS V3.02.000 |
| Hardware | NuTiny-EVB-NUC505 V1.6 |

*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 Function Description

## 1.1 Introduction

This sample code uses NUC505 internal codec to play WAV file stored in internal SPI flash. User needs to store WAV file first and then modify WAVFile_address from config.h to set WAV file storage address. Please note that the WAV file cannot exceed the NUC505 memory location. This sample code will check WAV file header to verify correctness of WAV file.

| WAV file storage address range | WAV file location parameter |
|---|---|
| **0x2000~0x1F_FFFF** | **WAVFile_address** |

This sample code supports mono/stereo mode, data length is 16 bits and sampling rates is 8k, 11025, 12k, 16k, 22050, 24k, 32k, 44100 and 48k. It will automatically set the APLL value according to different sampling rates.

## 1.2 Demo Result

```
Relocate vector table in SRAM (0x20000000) for fast interrupt handling, size = 0xbc.
+------------------------------------------------------------+
|      Play WAV file from internal Flash Sample Code         |
+------------------------------------------------------------+
  Please store WAV file first in the internal Flash.
  WAV file storage location cannot exceed flash size
  NOTE: Need head-phone.

wav.bin pre-stored in SPI Flash address from 0x00002000
wav: sampling rate = 44100
wav: Data size = 864 KB
wav: Channel Num = 2

Configure Internal CODEC ...[OK]
Start Playing ...
Play finish... enter while 1
```

## 2 Code Description

First, initialize the NUC505 in main.c, and put the code into the SRAM to increase its execution speed. Read the WAV file according to the address set by WAVFile_address.

```
int32_t main(void)
{
    /* Init System, IP clock and multi-function I/O */
    SYS_Init();

    /* Init UART0 to 115200-8n1 for print message */
    UART0_Init();

    /* Relocate vector table in SRAM for fast interrupt handling. */
    {
        extern uint32_t __Vectors[];
        extern uint32_t __Vectors_Size[];
        extern uint32_t Image$$ER_VECTOR2$$ZI$$Base[];

        printf("Relocate vector table in SRAM (0x%08X) for fast interrupt handling, size =
0x%x.\n", Image$$ER_VECTOR2$$ZI$$Base, (unsigned int) __Vectors_Size);
        memcpy((void *) Image$$ER_VECTOR2$$ZI$$Base, (void *) __Vectors, (unsigned int)
__Vectors_Size);
        SCB->VTOR = (uint32_t) Image$$ER_VECTOR2$$ZI$$Base;
    }
    /* Init I2S */
    I2S_Init();

    printf("+------------------------------------------------------------+\n");
    printf("|      Play WAV file from internal Flash Sample Code         |\n");
    printf("+------------------------------------------------------------+\n");
    printf("  Please store WAV file first in the internal Flash.\n");
    printf("  WAV file storage location cannot exceed flash size\n");
    printf("  NOTE: Need head-phone. \n\n");

    /* Play wav from internal flash at WAVFile_address */
    ReadFlashPlay(WAVFile_address);

    while (1);
}
```

In the ReadFlashPlay(uint32_t FlashAddr) function, the program will check WAV file header to verify correctness of WAV file, and to parse information about the WAV file. And set the APLL according to the sampling rate.

```
    /* Parsing WAV file header */
    SPIFlash_DMARead(SPIM, FlashAddr, 0, 44, aWavHeader, SPIM_CTL0_CMDCODE_FAST_READ);

    /* Read sampling rate from WAV header */
    u32WavSamplingRate = aWavHeader[6];

    /* Check if the WAV file is correct */
    if (aWavHeader[0] != 0x46464952 | aWavHeader[2] != 0x45564157) {
        printf("WAV File ERROR, please store Wav file at 0x%08x\n", FlashAddr);
        return -1;
    }
    /* Read WAV file total size from WAV header */
    PlayFrameSize = aWavHeader[1] / (PCM_BUFFER_SIZE * 4);
    WavRealSize = aWavHeader[1] / 1024;

    /* Read Channel number from WAV header */
    ChannelNum = (aWavHeader[5] & 0x00030000) >> 16;
    AudioFormat = ChannelNum == 1 ? I2S_MONO : I2S_STEREO;
    /* Check if the MP3 SamplRate frequency is 11025 multiples */
    if (u32WavSamplingRate % 11025) {
        /* APLL is 49152031Hz for 12k, 16k, 24k, 32k, 48k, and 96k sampling rate */
        CLK_SET_APLL(CLK_APLL_49152031);
        if (u32WavSamplingRate == 8000)
            CLK_SetModuleClock(I2S_MODULE, CLK_I2S_SRC_APLL, 1);
    }
```

After setting, use ping-pong buffer to play WAV file. The process is as follows:

```
    while (1) {
        /* Set ping-pong buffer to transmit data */
        if ((aPCMBuffer_Full[0] == 1) && (aPCMBuffer_Full[1] == 1)) {
            /* In beginning, wait all buffers full and start play */
            if (!bAudioPlaying) {
                bAudioPlaying = 1;
                /* I2S DMA settings */
                I2S_SET_TX_TH_LEVEL(I2S, I2S_FIFO_TX_LEVEL_WORD_15);
                I2S_SET_RX_TH_LEVEL(I2S, I2S_FIFO_RX_LEVEL_WORD_16);
                /* Tx Start Address */
```

```
                I2S_SET_TXDMA_STADDR(I2S, (uint32_t) &aPCMBuffer[0][0]);
                /* Tx Threshold Address */
                I2S_SET_TXDMA_THADDR(I2S, (uint32_t) &aPCMBuffer[0][PCM_BUFFER_SIZE - 1]);
                /* Tx End Address */
                I2S_SET_TXDMA_EADDR(I2S, (uint32_t) &aPCMBuffer[1][PCM_BUFFER_SIZE - 1]);
                /* Enable setting -> start to play */
                I2S_ENABLE_TXDMA(I2S);
                I2S_ENABLE_TX(I2S);
                /* Enable interrupt : TX DMA Threshold and TX DMA End Interrupt */
                I2S_EnableInt(I2S, (I2S_IEN_TDMATIEN_Msk | I2S_IEN_TDMAEIEN_Msk));
                NVIC_EnableIRQ(I2S_IRQn);
                printf("Start Playing ...\n");
            }
            /* All buffers are full, wait */
            while ((aPCMBuffer_Full[0] == 1) && (aPCMBuffer_Full[1] == 1));
        }
        /* Read WAV data from internal flash */
        SPIFlash_DMARead(SPIM, FlashAddr + 44, 0, PCM_BUFFER_SIZE * 4,
_UNVMP(&aPCMBuffer[u8PCMBufferTargetIdx][0]), SPIM_CTL0_CMDCODE_FAST_READ);

        /* Move address pointer to next address */
        FlashAddr = FlashAddr + PCM_BUFFER_SIZE * 4;
        count_frame++;
        NVIC_DisableIRQ(I2S_IRQn);
        /* Set buffer full flag */
        aPCMBuffer_Full[u8PCMBufferTargetIdx] = 1;
        NVIC_EnableIRQ(I2S_IRQn);

        if (bAudioPlaying) {
            /* Make sure the current ping-pong buffer is empty */
            while (aPCMBuffer_Full[u8PCMBufferTargetIdx ^ 1]) {
                /* Break if the last audio data played */
                if (count_frame >= PlayFrameSize) {
                    goto stop;
                }
            }
        }
        /* Switch to another buffer */
        u8PCMBufferTargetIdx ^= 1;
    }
```

Clear buffer full flag when buffer transmit completed in nuc505_isr.c

```
void I2S_IRQHandler(void)
{
    uint32_t u32I2SIntFlag;

    /* Check interruput flag */
    u32I2SIntFlag = I2S_GET_INT_FLAG(I2S, I2S_STATUS_TDMATIF_Msk |
I2S_STATUS_TDMAEIF_Msk);

    if (u32I2SIntFlag & I2S_STATUS_TDMATIF_Msk) {
        I2S_CLR_INT_FLAG(I2S, I2S_STATUS_TDMATIF_Msk);

        /* Buffer 0 transfer completed, clear buffer full flag */
        aPCMBuffer_Full[0] = 0;
    } else if (u32I2SIntFlag & I2S_STATUS_TDMAEIF_Msk) {
        I2S_CLR_INT_FLAG(I2S, I2S_STATUS_TDMAEIF_Msk);
        /* Buffer 1 transfer completed, clear buffer full flag */
        aPCMBuffer_Full[1] = 0;
    }
}
```

# 3 Software and Hardware Environment

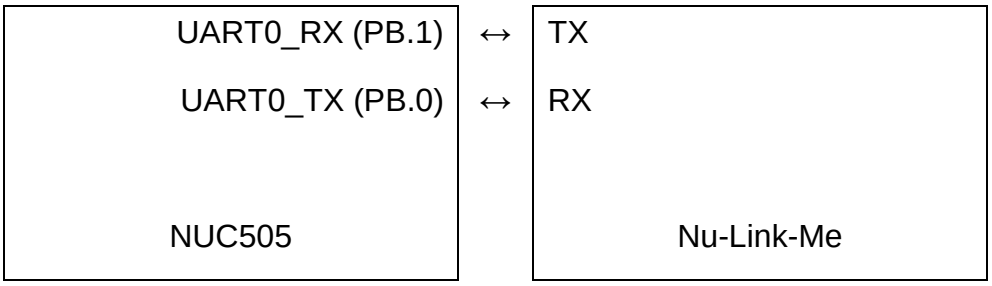
## Software environment

- BSP version
  - NUC505 Series BSP CMSIS V3.02.000
- IDE version
  - Keil uVersion 5.26

## Hardware environment

- Circuit components
  - NuTiny-EVB-NUC505 V1.6
  - 3.5 mm headphones
- Diagram

Use 3.5 mm headphones to connect to onboard socket(HP OUT). NUC505's UART0_RX (PB.1) and UART0_TX (PB.0) are connected to Nu-Link Me to print the message. Set COM port and Baud. The number of the COM Port can be found in the device manager "NuBridge Virtual Com Port (COMX)" and Baud is set to 115200.



# 4 Directory Information

| | |
|---|---|
| EC_NUC505_WavPlayerSPIFlash_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 How to Execute Example Code

1. Browsing into sample code folder by Directory Information (section 4) and double click WavPlayerSPIFlash.uvproj
2. Enter Keil compile mode
   a. Build
   b. Download
   c. Start/Stop debug session
3. Enter debug mode
   a. Run

# 6 Revision History

| Date | Revision | Description |
|---|---|---|
| Jul. 03, 2019 | 1.00 | 1. Initially issued. |

## 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.*