M480 Pausing and Resuming MP3

Example Code Introduction for 32-bit NuMicro® Family

Document Information

Application	This sample code implements play, stop, pause, and resume MP3 by external audio codec NAU88L25.
BSP Version	M480 Series BSP CMSIS V3.05.001
Hardware	NuMaker-PFM-M487 v3.0

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. Overview

This sample code uses external codec NAU88L25 to act as a MP3 player to play MP3 files stored in SD card. This MP3 player provides play, stop, pause, and resume functions.

First, user needs to store the MP3 file in the SD card and modify the file name to *test.mp3* or modify *#define MP3_FILE* in *mp3.c* to the correct name. After storing the MP3 file to SD card, this demo application will read the MP3 file, use NAU88L25 codec to decode, and play the music automatically through I^{2}S interface in the M480 series microcontroller (MCU).

1.1 Demo Result

The first demo pauses and resumes music as shown in Figure 1-1 User can push SW3 button to pause the music and push SW3 button again to resume the music where it paused.

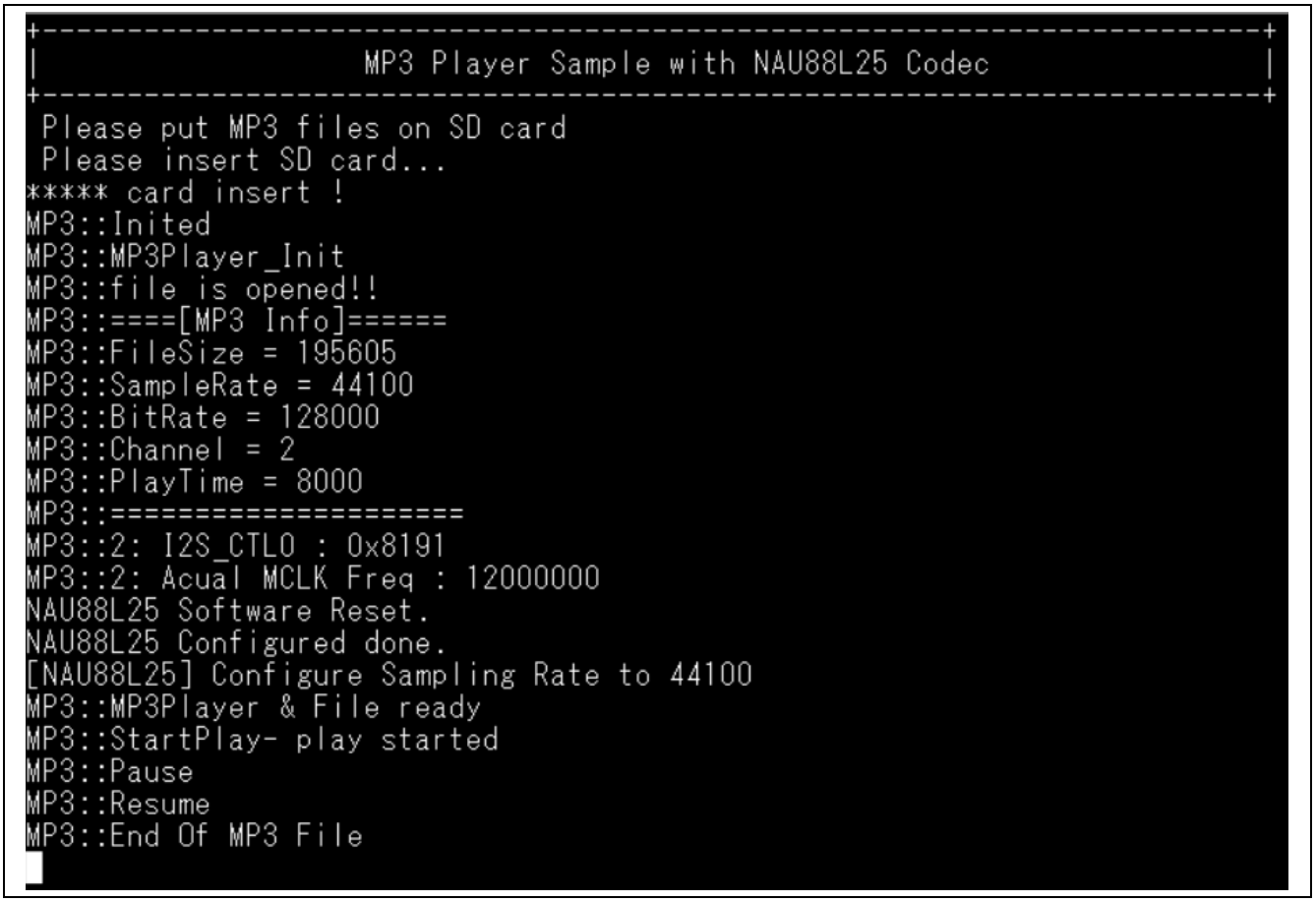

Figure 1-1 Pause and Resume Music Demo Result

The second demo stops and replays the music as shown in Figure 1-2 User can push SW2 button to stop the music and push SW2 button again to replay the music.

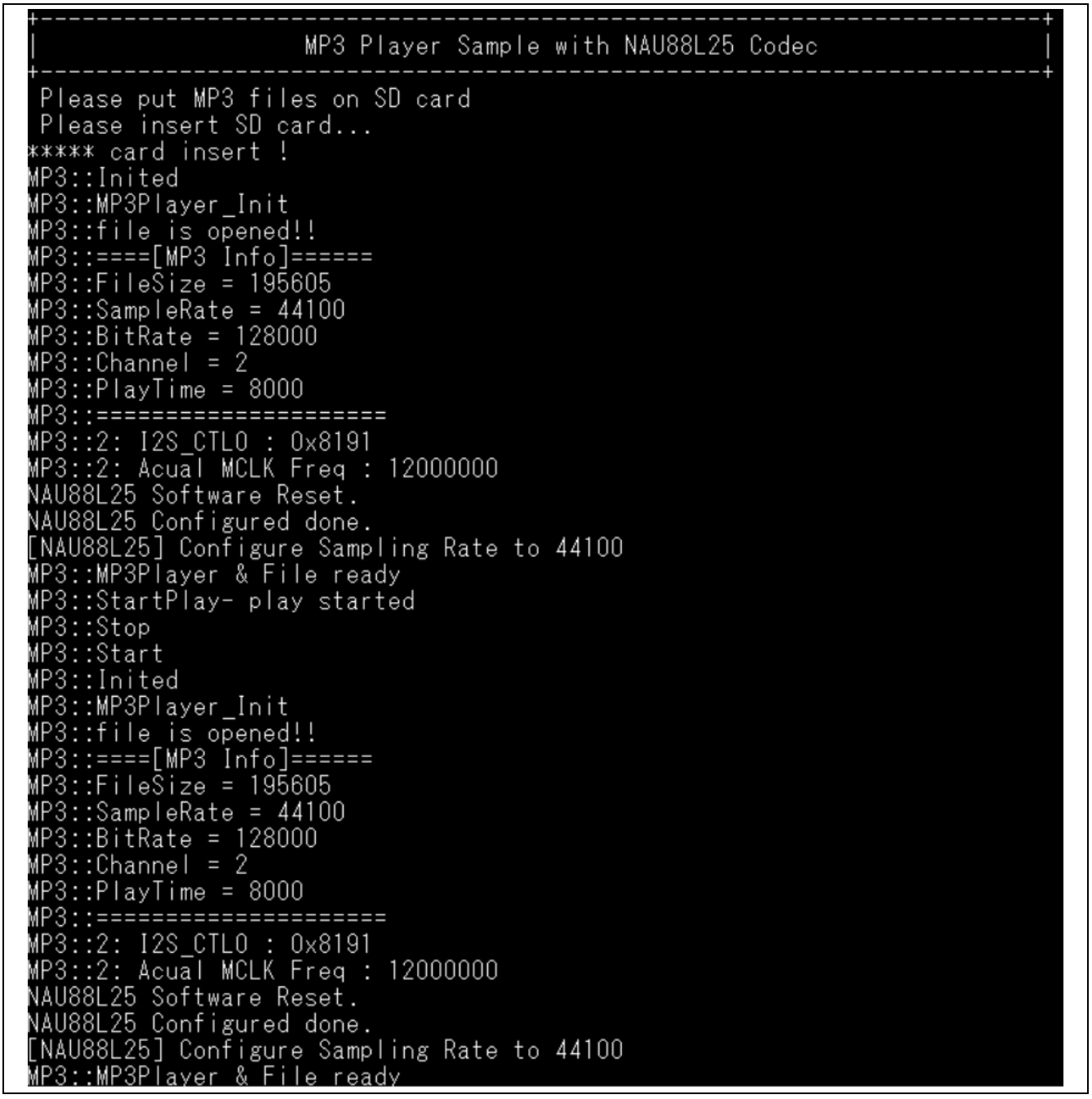

Figure 1-2 Stop and Replay Music Demo Result

2. Code Description

The *main.c* will initialize I²C, SD, buttons on the evaluation board, and NAU88L25. Then call *mp3.c* to play the music until the music ends.

```
int32_t main (void)
{
    TCHAR    sd_path[] = { '0', ':', 0 };    /* SD drive started from 0 */

    MP3RESULT u8MainMP3Result = MP3_OK;

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

    /* Configure PG.15 as Quasi-bidirection mode and enable interrupt by falling edge
trigger */
    GPIO_SetMode(PG, BIT15, GPIO_MODE_QUASI);
    GPIO_EnableInt(PG, 15, GPIO_INT_FALLING);
    NVIC_EnableIRQ(GPG_IRQn);

    /* Enable interrupt de-bounce function and select de-bounce sampling cycle time is
1024 clocks of LIRC clock */
    GPIO_SET_DEBOUNCE_TIME(GPIO_DBCTL_DBCLKSRC_LIRC, GPIO_DBCTL_DBCLKSEL_1024);
    GPIO_ENABLE_DEBOUNCE(PG, BIT15);

    /* Configure PF.11 as Quasi-bidirection mode and enable interrupt by falling edge
trigger */
    GPIO_SetMode(PF, BIT11, GPIO_MODE_QUASI);
    GPIO_EnableInt(PF, 11, GPIO_INT_FALLING);
    NVIC_EnableIRQ(GPF_IRQn);

    /* Enable interrupt de-bounce function and select de-bounce sampling cycle time is
1024 clocks of LIRC clock */
    GPIO_SET_DEBOUNCE_TIME(GPIO_DBCTL_DBCLKSRC_LIRC, GPIO_DBCTL_DBCLKSEL_1024);
    GPIO_ENABLE_DEBOUNCE(PF, BIT11);

    /* BOD Level Set */
    /* Unlock protected registers before setting Brown-out detector function and Power-
down mode */
    SYS_UnlockReg();

    /* Enable Brown-out detector function */
    SYS_ENABLE_BOD();

    /* Set Brown-out detector voltage level as 2.4V */
    SYS_SET_BOD_LEVEL(SYS_BODCTL_BODVL_2_4V);

    /* Enable Brown-out detector interrupt function */
    SYS_DISABLE_BOD_RST();

    /* Enable Brown-out detector and Power-down wake-up interrupt */
    NVIC_EnableIRQ(BOD_IRQn);

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

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

    printf("+-----------------------------------------------------------------------
+\n");
    printf("|                   MP3 Player Sample with NAU88L25 Codec
|\n");
    printf("+-----------------------------------------------------------------------
+\n");
    printf(" Please put MP3 files on SD card \n");

    printf(" Please insert SD card... \n");
    while(1)
    {
        if (SDH_Open_Disk(SDH0, CardDetect_From_GPIO) == Successful)
            break;
    }
/* set default path */
f_chdrive(sd_path);

    /* Init I2C2 to access WAU88L25 */
    I2C2_Init();

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

    SetMP3PlayerStatus(eMP3PLAYER_STATUS_START);

    while(1)
    {
        /* play mp3 */
        if (SD0.IsCardInsert == TRUE)
        {
            u8MainMP3Result = TASK_MP3Player();
            if(u8MainMP3Result > MP3_MAD_BUFLEN_ERR)
            {
                printf("MP3 Play Error code : %d\n\r", u8MainMP3Result);
                printf("Exit MP3 Player\n\r");
                break;
            }
        }
    }
}
```

The *mp3.c* will initialize I²S and show the MP3 file information. *TASK_MP3Player* function will read and set the MP3 player status flag, and execute the corresponding function.

```
MP3RESULT TASK_MP3Player()
{
    MP3RESULT mp3Result = MP3_OK;
    /* Set MP3 Player Status by Button and Reset Flag */
    switch (Status_Flag)
    {
    case eMP3PLAYER_STATUS_RESUME:
        SetMP3PlayerStatus(eMP3PLAYER_STATUS_RESUME);
```

```
        Status_Flag = eMP3PLAYER_STATUS_NONE;
        break;
    case eMP3PLAYER_STATUS_PAUSE:
        SetMP3PlayerStatus(eMP3PLAYER_STATUS_PAUSE);
        Status_Flag = eMP3PLAYER_STATUS_NONE;
        break;
    case eMP3PLAYER_STATUS_STOP:
        SetMP3PlayerStatus(eMP3PLAYER_STATUS_STOP);
        Status_Flag = eMP3PLAYER_STATUS_NONE;
        break;
    case eMP3PLAYER_STATUS_START:
        SetMP3PlayerStatus(eMP3PLAYER_STATUS_START);
        Status_Flag = eMP3PLAYER_STATUS_NONE;
        break;
    default:
        break;
    }
    newplayerstatus = u8MP3PlayerStatus;
    if(currentplayerstatus != newplayerstatus)
    {
        currentplayerstatus = newplayerstatus;
        switch(newplayerstatus)
        {
        case eMP3PLAYER_STATUS_START :
            NAU88L25_Unmute();
            MP3Player_DeviceInit();
            mp3Result = MP3Player_DataInit();
            printf("MP3::MP3Player & File ready\n\r");
            break;
        case eMP3PLAYER_STATUS_STOP:
            NAU88L25_Mute();
            MP3Player_PlayStop();
            break;

        default:
            break;
        }
    }
    if(eMP3PLAYER_STATUS_STOP != currentplayerstatus)
        mp3Result = MP3Player_PlayStart(newplayerstatus);
}
```

3. Software and Hardware Requirements

3.1 Software Requirements

- BSP version
  - M480 Series BSP CMSIS V3.05.001
- IDE version
  - Keil uVersion 5.26

3.2 Hardware Requirements

- Circuit components
  - NuMaker-PFM-M487 v3.0
  - SD card
- Pin Connect

	V_{DD}	↔	V_{DD}	
	GND	↔	GND	
	PE.7	↔	SD_CMD	
	PE.6	↔	SD_CLK	
	PE.5	↔	SD_DATA3	
	PE.4	↔	SD_DATA2	
	PE.3	↔	SD_DATA1	
	PE.2	↔	SD_DATA0	
	PD.13	↔	SD_nCD	
	NuMaker-PFM-M487		SD CARD	
	V_{DD}	↔	V_{DD}	
	GND	↔	GND	
	PD.0	↔	I2C_SCL	
	PD.1	↔	I2C_SSA	
	PF.10	↔	I2S_BCLK	

	PF.9	↔	I2S_MCLK	
	PF8	↔	I2S_DI	
	PF7	↔	I2S_DO	
	PF6	↔	I2S_LRCK	
	NuMaker-PFM-M487		NAU88L25	

Figure 3-1 Pin Connect

4. Directory Information

EC_M480_ Pause_and_Resume_MP3_Player_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 which 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

Figure 4-1 Directory Structure

5. Example Code Execution

1. Browse the sample code folder as described in the Directory Information section and double-click *Pause_and_Resume_MP3_Player.uvproj*.
2. Enter Keil compile mode.
   - Build
   - Download
   - Start/Stop debug session
3. Enter debug mode.
   - Run

6. Revision History

Date	Revision	Description
2022.06.21	1.00	Initial version.

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.