M480 USB Audio Codec and HID Composite Device

Example Code Introduction for 32-bit NuMicro® Family

Information

Application	This example code uses M480 MCU and NAU88L25 audio codec device to implement an USB composite device that based on the USB audio class 1.0 (UAC1.0) and HID class.
BSP Version	M480 Series BSP CMSIS V3.03.001
Hardware	NuMaker-PFM-M487 V3.0

For additional information or questions, please contact: Nuvoton Technology Corporation.

www.nuvoton.com

1 Function Description

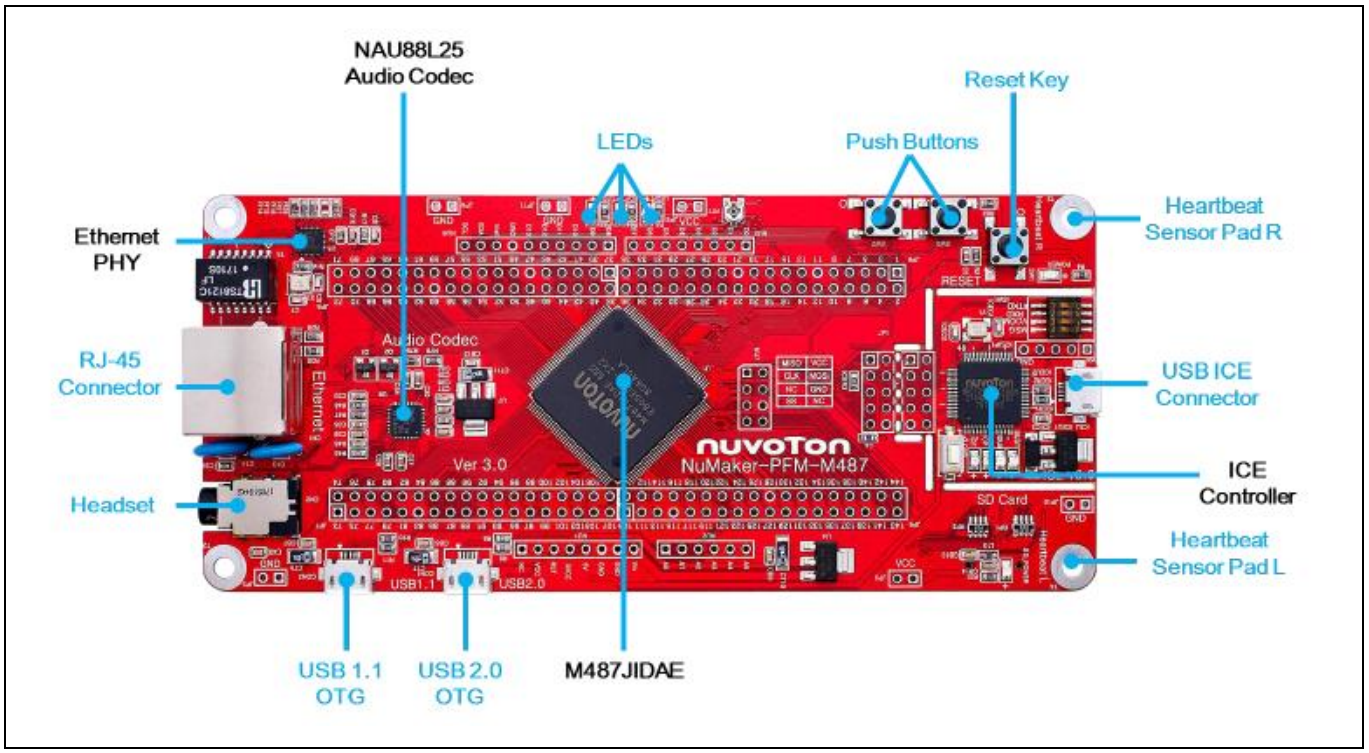
1.1 Introduction

This example code uses M480 MCU and on-board two key buttons and a NAU88L25 audio codec device to implement an USB composite device that based on the UAC1.0 (USB Audio Class 1.0) and HID class specification. The HID class is for reporting the key buttons inputs to control the play/pause state and volume up of multimedia functions thru the USB protocol.

For more information on the NuMaker-PFM-M487 board for the ARM® mbed™ Device Platform, please visit the ARM® mbed™ hardware board website or Nuvoton website to download the User Manual document UM_NuMaker-PFM_M487_User_Manual_EN:

https://developer.mbed.org/platforms/NUMAKER-PFM-M487/

https://www.nuvoton.com/hq/products/microcontrollers/arm-cortex-m4-mcus/User-Manual/?__locale=en&resourcePage=Y&category=&pageIndex=3



NuMaker-PFM-M487 V3.0 Board

1.2 Demo Result

After executed this example code, user can plug-in the USB cable that connect the USB1.1 connector (FS USB OTG) with PC Host and plug-in a headset to the CN2 connector on the NuMaker-PFM-M487 V3.0 board to listen the playing music from PC and press the key buttons (SW3/SW2) to control the play/pause state and volume up of multimedia functions.

2 Code Description

The EP3_Handler() function is used to get the 16-bit playback PCM data of Left and Right channels from PC Host and automatically pass these data to I2S0 port thru the PDMA channel by calling the UAC_GetPlayData() function, and then I2S0 port transmits these PCM data to the external NAU88L25 audio codec device to play out the music.

```
/**
 * @brief       EP3 Handler
 *
 * @param[in]   None
 *
 * @return      None
 *
 * @details     This function is used to process EP3 event
 */
/* Play */
void EP3_Handler(void)
{
    /* ISO OUT transfer ACK */
    UAC_GetPlayData((uint8_t *)((uint32_t)USBD_BUF_BASE + USBD_GET_EP_BUF_ADDR(EP3)),
(uint32_t)USBD_GET_PAYLOAD_LEN(EP3));
    USBD_SET_PAYLOAD_LEN(EP3, EP3_MAX_PKT_SIZE);
}


/**
  * @brief  GetPlayData, To get data from ISO OUT to play buffer.
  * @param  pi16src: The data buffer of ISO OUT.
  *         i16Samples: The sample number in data buffer.
  * @retval None.
  */
void UAC_GetPlayData(uint8_t *psrc, uint32_t u32Samples)
{
    uint32_t i, u32len;
    uint32_t *pBuff;
    uint32_t *pSrc;

    /* if buffer has enough data, play it!! */
    if(!u8AudioPlaying && (u8TxDataCntInBuffer >= (PDMA_TXBUFFER_CNT/2+1))) {
        AudioStartPlay(g_usbd_SampleRate);
        u8AudioPlaying = 1;
```

```
    }

    u32len = u32Samples/4;
    /* Ring buffer check */
    if((u32PlayBufPos + u32len) > u32BuffLen) {
        PcmPlayBuffLen[u32BufPlayIdx] = u32PlayBufPos;
        u32PlayBufPos = 0;
        u32BufPlayIdx ++;

        /* change buffer index */
        if(u32BufPlayIdx >= PDMA_TXBUFFER_CNT)
            u32BufPlayIdx=0;

        /* increase data count in buffer */
        u8TxDataCntInBuffer ++;
    }

    pBuff = &PcmPlayBuff[u32BufPlayIdx][u32PlayBufPos];
    pSrc = (uint32_t *) psrc;
    for(i = 0; i < u32len; i++) {
        pBuff[i] = pSrc[i];
    }

    u32PlayBufPos += u32len;
}
```

These EP2_Handler() and UAC_SendRecData() functions transfer the 16-bit record PCM data of Left and Right channels through the data buffers (PcmRecBuff[]) that receiving from the external NAU88L25 audio codec device and store these data into the internal SRAM of Full Speed USB Device.

```
/**
 * @brief       EP2 Handler
 *
 * @param[in]   None
 *
 * @return      None
 *
 * @details     This function is used to process EP2 event
 */
/* Record */
```

```
void EP2_Handler(void)
{
    /* ISO IN transfer ACK */
    if(g_usbd_UsbAudioState == UAC_START_AUDIO_RECORD)
        UAC_SendRecData();
}

/**
  * @brief  SendRecData, prepare the record data for next ISO transfer.
  * @param  None.
  * @retval None.
  */
void UAC_SendRecData(void)
{
    /* when record buffer full, send data to host */
    if(u8PcmRxBufFull[u32BufRecIdx]) {
        /* Set empty flag */
        u8PcmRxBufFull[u32BufRecIdx] = 0;

        USBD_MemCopy((uint8_t *)((uint32_t)USBD_BUF_BASE + USBD_GET_EP_BUF_ADDR(EP2)),
(void *)&PcmRecBuff[u32BufRecIdx][0], u32RxBuffLen);
        USBD_SET_PAYLOAD_LEN(EP2, u32RxBuffLen);

        /* Change to next PCM buffer */
        u32BufRecIdx ++;
        if(u32BufRecIdx >= PDMA_RXBUFFER_CNT)
            u32BufRecIdx=0;

    } else {
        USBD_SET_PAYLOAD_LEN(EP2, 0);
    }
}
```

The HID_KeyboardReportDescriptor[] table is the HID keyboard report descriptor for the multimedia keys. These EP4_Handler() and HID_UpdateKbData() functions are used to report the HID multimedia key status to Host when the push buttons be pressed or released at a fixed interval 10ms.

```
#define HID_CTRL_MUTE        0x01
#define HID_CTRL_VOLUME_INC  0x02
#define HID_CTRL_VOLUME_DEC  0x04
```

```
#define HID_CTRL_EJECT       0x08
#define HID_CTRL_PLAY        0x01
#define HID_CTRL_STOP        0x02
#define HID_CTRL_PAUSE       0x04
#define HID_CTRL_NEXT        0x08
#define HID_CTRL_PREVIOUS    0x10
#define HID_CTRL_RECORD      0x20
#define HID_CTRL_REWIND      0x40
#define HID_CTRL_FF          0x80

/*!<USB HID Report Descriptor */
uint8_t HID_KeyboardReportDescriptor[] = {
    0x05, 0x0C,         /* Usage Page (Consumer) */
    0x09, 0x01,         /* Usage(Consumer Control) */
    0xA1, 0x01,         /* Collection(Application) */
    0x15, 0x00,         /* Logical Minimum(0x0) */
    0x25, 0x01,         /* Logical Maximum(0x1) */
    0x09, 0xE2,         /* Usage(Mute) */
    0x09, 0xE9,         /* Usage(Volume Increment) */
    0x09, 0xEA,         /* Usage(Volume Decrement) */
    0x75, 0x01,         /* Report Size(0x1) */
    0x95, 0x03,         /* Report Count(0x3) */
    0x81, 0x02,         /* Input(Data, Variable, Absolute, No Wrap, Linear, Preferred
State, No Null Position, Bit Field) */
    0x75, 0x01,         /* Report Size(0x1) */
    0x95, 0x05,         /* Report Count(0x5) */
    0x81, 0x03,         /* Input(Constant, Variable, Absolute) */
    0x09, 0xB0,         /* Usage(Play) */
    0x09, 0xB7,         /* Usage(Stop) */
    0x09, 0xCD,         /* Usage(Play/Pause) */
    0x09, 0xB5,         /* Usage(Scan Next Track) */
    0x09, 0xB6,         /* Usage(Scan Previous Track) */
    0x09, 0xB2,         /* Usage(Record) */
    0x09, 0xB4,         /* Usage(Rewind) */
    0x09, 0xB3,         /* Usage(Fast Forward) */
    0x75, 0x01,         /* Report Size(0x1) */
    0x95, 0x08,         /* Report Count(0x8) */
    0x81, 0x02,         /* Input(Data, Variable, Absolute, No Wrap, Linear, Preferred
State, No Null Position, Bit Field) */
    0x09, 0x00,         /* Usage(Undefined) */
    0x75, 0x08,         /* Report Size(0x8) */
```

```
    0x95, 0x06,         /* Report Count(0x6) */
    0x81, 0x02,         /* Input(Data, Variable, Absolute, No Wrap, Linear, Preferred
State, No Null Position, Bit Field) */
    0x09, 0x00,         /* Usage(Undefined) */
    0x75, 0x08,         /* Report Size(0x8) */
    0x95, 0x08,         /* Report Count(0x8) */
    0x91, 0x00,         /* Output(Data, Array, Absolute, No Wrap, Linear, Preferred State,
No Null Position, Non VolatileBit Field) */
    0xC0
};

/**
 * @brief       EP4 Handler
 *
 * @param[in]   None
 *
 * @return      None
 *
 * @details     This function is used to process EP4 event
 */
/* HID */
void EP4_Handler(void)  /* Interrupt IN handler */
{
    g_u8EP4Ready = 1;
}

/* Update the HID key data to Host */
void HID_UpdateKbData(void)
{
    int32_t i;
    uint8_t *buf;
    uint32_t key = 0xF;
    static uint32_t preKey;
    int32_t n;

    n = 8;
    if(g_u8EP4Ready) {
        buf = (uint8_t *)(USBD_BUF_BASE + USBD_GET_EP_BUF_ADDR(EP4));

        // PF11, play/pause
        // PG15, Vol+
```

```
        key = (!PF11) | (!PG15 << 1);

        if(key == 0) {
            for(i = 0; i < n; i++) {
                buf[i] = 0;
            }

            if(key != preKey) {
                preKey = key;
                g_u8EP4Ready = 0;
                /* Trigger to note key release */
                USBD_SET_PAYLOAD_LEN(EP4, n);
            }
        } else {
            // Don't repeat key when it is media key
            if(preKey != key) {
                preKey = key;
                buf[0] = 0;
                buf[1] = 0;
                if(!PF11)
                    buf[1] |= HID_CTRL_PAUSE;
                else if(!PG15)
                    buf[0] |= HID_CTRL_VOLUME_INC;

                g_u8EP4Ready = 0;
                USBD_SET_PAYLOAD_LEN(EP4, n);
            }
        }
    }
}
```

3 Software and Hardware Environment

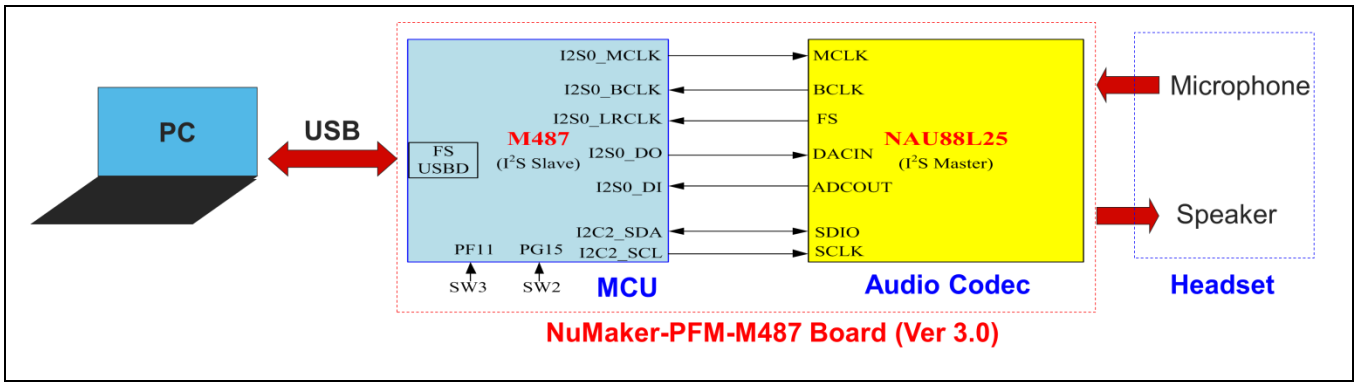
Software Environment

- BSP version
  - M480 Series BSP CMSIS V3.03.001
- IDE version
  - Keil uVersion 4.74

Hardware Environment

- Circuit components
  - NuMaker-PFM-M487 V3.0 board
- Diagram

The M480 MCU uses the I2C2 port to configure the NAU88L25 for I^2S clock settings of I^2S master, I2S0 port to transfer data with NAU88L25 audio codec device and Full Speed USB Device (FS USBD) to transfer PCM streams with PC Host. The SW3 (PF11) and SW2 (PG15) buttons can control the play/pause state and volume up of multimedia functions.



Hardware Block Diagram

4 Directory Information

EC_M480_USBD_Audio_Codec_HID_V1.00	
Library	Sample code header and source code files
CMSIS	Cortex® Microcontroller Software Interface Standard (CMSIS) by Arm® Corp.
Device	CMSIS compliant device header files
StdDriver	All peripheral driver header and source code files
SampleCode	
ExampleCode	Source code files of this example code

5 How to Execute Example Code

1. Browsing into the sample code folder by Directory Information (section 4) and double click this sample code project M480_USBD_Audio_Codec_HID.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
Sept. 26, 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.