Application Note

32-bit Cortex™-M0 MCU NuMicro® Family

How to use I2S with CODEC in NUC1xx?

Table of Contents-

1 INTRODUCTION 2
1.1 Scope 2
1.2 Features 2
1.3 Structure 2
1.3.1 I2S Block Diagram 3
1.3.2 I2S Operation 3
1.3.3 FIFO operation 4
2 CODE SECTION 5
2.1 Main Function (in Smpl_DrvI2S.c) 5
2.2 I2C CINFIGURATION 7
2.3 CODEC WAU8822 CINFIGURATION 7
3 CALLING SEQUENCE 10
3.1 I2S and CODEC Integration 10
3.2 NUC1xx Settings 10
3.3 Handle Rx Request 11
3.4 Handle Rx Request 11
3.5 API Usage Reference 11
4 EXECUTION ENVIRONMENT SETUP AND RESULT 12
4.1 Test Smpl_DrvI2S 12
4.2 Result 12
5 REVISION HISTORY 13

1 INTRODUCTION

This document explains the sample code, “Smpl_DrvI2S” and demonstrates how to use I2S with audio CODEC through the related IP drivers.

1.1 Scope

This article is provided for programmers applying the I2S IP to audio application. Here, the CODEC used in this application is WAU8822. It is assumed that the reader is familiar with the relative feature and settings of the WAU8822.

1.2 Features

- I2S can operate as either master or slave.
- Capable of handling 8, 16, 24 and 32 bit word size.
- Mono and stereo audio data supported.
- Standard I2S and MSB justified data format supported.
- Two 8 word deep FIFO are provided, one for transmit and the other for receive.
- Generate interrupt request when FIFO level cross a programmable boundary.
- Two DMA requests, one for transmit and the other for receive.

1.3 Structure

I2S is equipped with MCLK, LRCLK and BCLK. The MCLK is to generate clock to audio CODEC. The BCLK and LRCLK are to generate corresponding sampling rate when I2S operates as master mode.

1.3.1 I2S Block Diagram

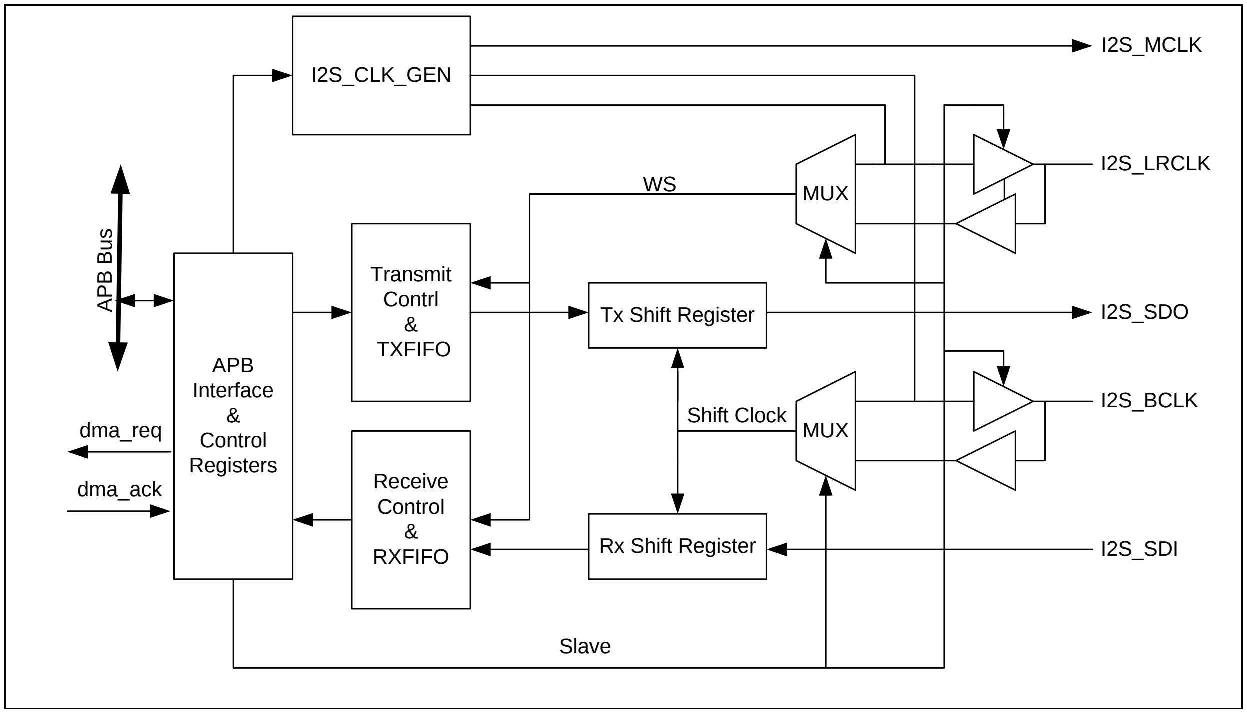

Figure 1 I2S Controller Block Diagram

1.3.2 I2S Operation

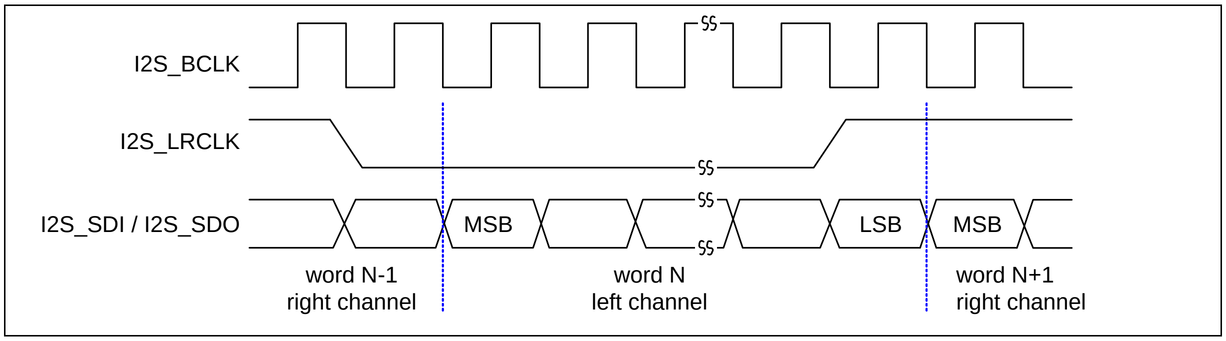

Figure 2 I2S Bus Timing Diagram (Format =0)

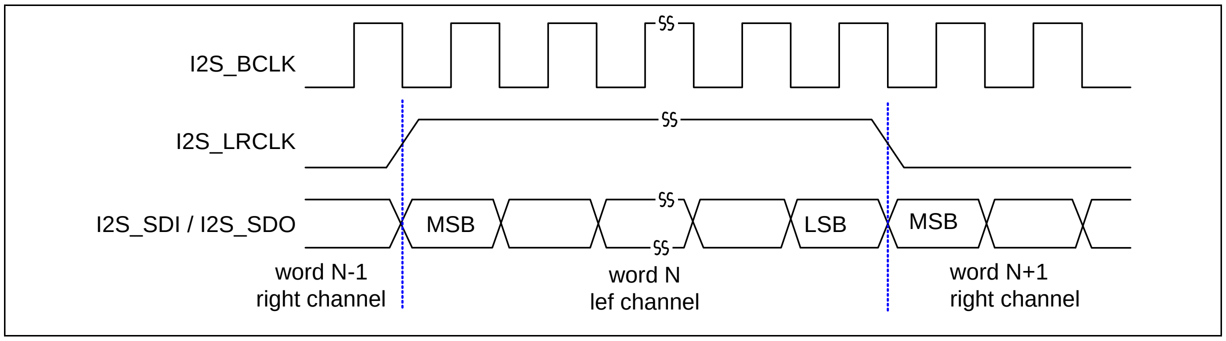

Figure 3 MSB Justified Timing Diagram (Format=1)

1.3.3 FIFO operation

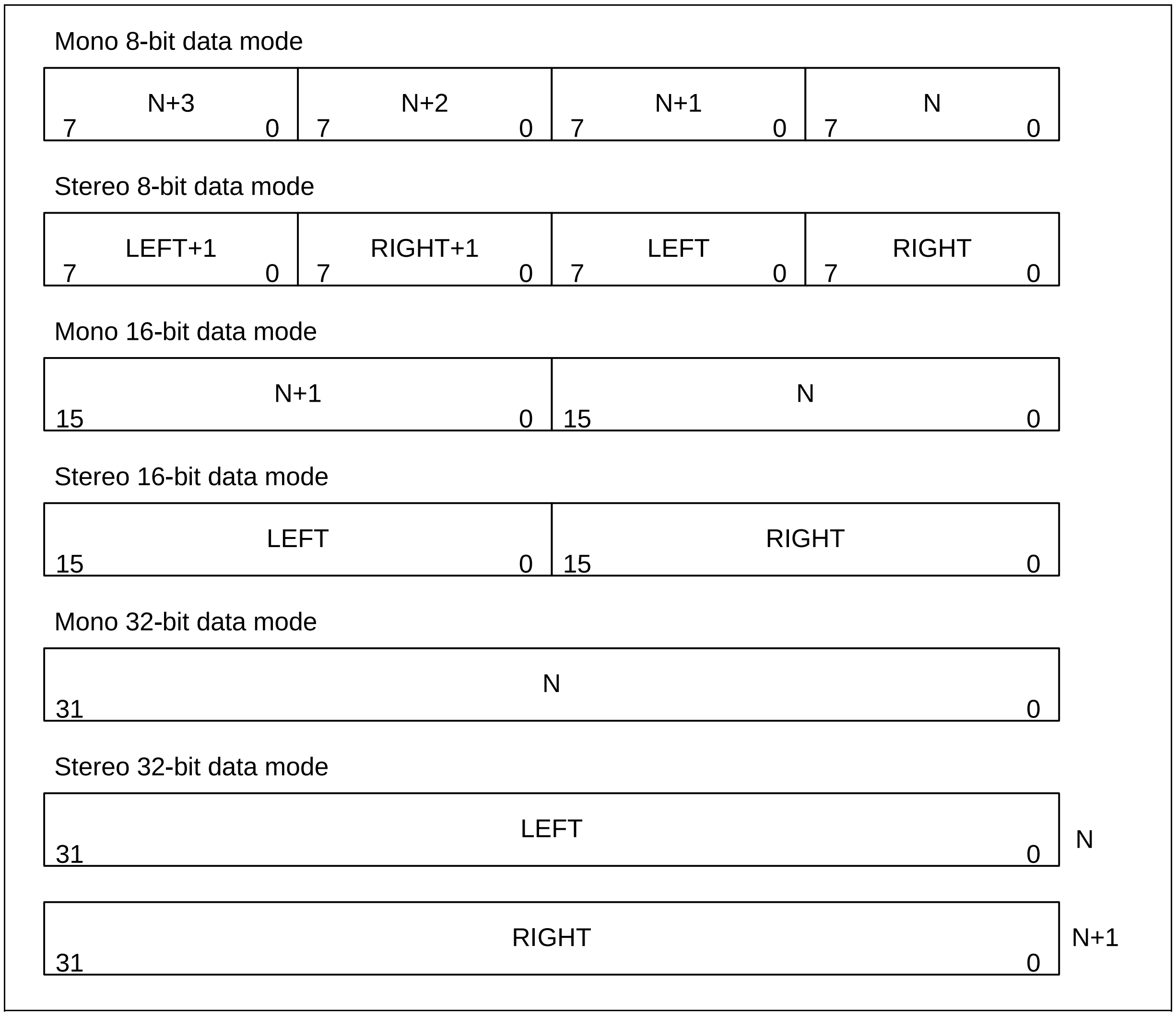

Figure 4 FIFO contents for various I2S modes

2 CODE SECTION

2.1 Main Function (in Smpl_DrvI2S.c)

Refer chapter of Calling Sequence for its calling sequence.

This sample code will show audio demo with I2S and CODEC. I2S receives 16-bit audio data from ADC of WAU8822, and transfer data to Rx FIFO. When word data in Rx FIFO reaches to Rx FIFO threshold level, the I2S interrupt will request MCU to transfer data into audio buffer. And, The I2S interrupt will request MCU to transfer data from audio buffer to Tx FIFO when word data in Tx FIFO is less than Tx FIFO threshold level. Then, I2S sends 16-bit audio data to DAC of WAU8822.

In the main function, the WAU8822 CODEC will be configured with I2C0 interface. After hardware initialization, the DrvI2S_Open() is used to configure I2S mode, data format, data width, FIFO threshold level and etc. Because WAU8822 can drive LRCLK and BCLK for precise sampling rate, I2S will operate in slave mode. When the settings of the CODEC and I2S are finish, the MCLK will be enabled to generated master clock for WAU8822.

```
int32_t main (void)
{
        uint32_t u32clock, u32HCLK;
        __IO uint32_t u32startFlag = 1;

        STR_UART_T sParam;
        S_DRVI2S_DATA_T st;

        /* Set I2S I/O */
    DrvGPIO_InitFunction(FUNC_I2S);

        /* Set I2C I/O */
    DrvGPIO_InitFunction(FUNC_I2C0);

        /* Set UART I/O */
        DrvGPIO_InitFunction(FUNC_UART0);

        /* UART Setting */
    sParam.u32BaudRate              = 115200;
    sParam.u8cDataBits              = DRVUART_DATABITS_8;
    sParam.u8cStopBits        = DRVUART_STOPBITS_1;
    sParam.u8cParity          = DRVUART_PARITY_NONE;
sParam.u8cRxTriggerLevel = DRVUART_FIFO_1BYTES;

        /* Set UART Configuration */
        DrvUART_Open(UART_PORT0, &sParam);

        printf("+------------------------------------------------------------------------+\n");
printf("|                  I2S Driver Sample Code with WAU8822                  |\n");
printf("+------------------------------------------------------------------------+\n");

        //u32HCLK = DrvSYS_GetHCLK() * 1000;
        u32HCLK = 12000000;

        /* Open I2C0 and I2C1, and set clock = 100Kbps */
        DrvI2C_Open(I2C_PORT0, u32HCLK, 100000);

        /* Get I2C0 clock */
        u32clock = DrvI2C_GetClock(I2C_PORT0, u32HCLK);
        printf("I2C0 clock %d Hz\n", u32clock);

        /* Enable I2C0 interrupt and set corresponding NVIC bit */
        DrvI2C_EnableInt(I2C_PORT0);

        st.u32SampleRate          = 16000;
st.u8WordWidth                    = DRVI2S_DATABIT_16;
st.u8AudioFormat       = DRVI2S_STEREO;
        st.u8DataFormat           = DRVI2S_FORMAT_I2S;
st.u8Mode                         = DRVI2S_MODE_SLAVE;
st.u8TxFIFOThreshold = DRVI2S_FIFO_LEVEL_WORD_0;
st.u8RxFIFOThreshold = DRVI2S_FIFO_LEVEL_WORD_8;

        DrvI2S_Open(&st);

        WAU8822_Setup();

        /* Set MCLK and enable MCLK */
        DrvI2S_SetMCLK(12000000);

        u32clock = DrvI2S_GetMCLK();
        printf("MCLK %d\n", u32clock);
        DrvI2S_EnableMCLK(TRUE);

        /* Enable Rx threshold level interrupt and install its callback function */
        DrvI2S_EnableInt(I2S_RX_FIFO_THRESHOLD, Rx_thresholdCallbackfn);

        /* Enable I2S Rx function to receive data */
        DrvI2S_EnableRx(TRUE);

        while(1)
        {
                if (u32startFlag)
                {
                        /* Enable I2S Tx function to send data when data in the buffer
is more than half of buffer size */
                        if (u32BuffPos >= BUFF_LEN/2)
                        {
                                DrvI2S_EnableInt(I2S_TX_FIFO_THRESHOLD,
Tx_thresholdCallbackfn);
                                DrvI2S_EnableTx(TRUE);
                                u32startFlag = 0;
                        }
                }
        }
}
```

2.2 I2C CINFIGURATION

Refer to I2C sample code and relative document.

2.3 CODEC WAU8822 CINFIGURATION

Refer to WAU8822 Design Guide for detail settings of WAU8822.

```
void WAU8822_Setup()
{
        printf("WAU8822 Setup\n");

        I2C_WriteWAU8822(0,  0x000);   /* Reset all registers */
        Delay(0x200);

        I2C_WriteWAU8822(1,  0x02F);
        I2C_WriteWAU8822(2,  0x1B3);   /* Enable L/R Headphone, ADC Mix/Boost,
ADC */

        I2C_WriteWAU8822(3,  0x00F);   /* Enable L/R main mixer, DAC */

        I2C_WriteWAU8822(4,  0x010);   /* 16-bit word length, I2S format, Stereo */

        I2C_WriteWAU8822(5,  0x000);   /* Companding control and loop back mode
(all disable) */

        I2C_WriteWAU8822(6,  0x1AD);   /* Divide by 6, 16K */

        I2C_WriteWAU8822(7,  0x006);   /* 16K for internal filter cofficients */

        I2C_WriteWAU8822(10, 0x008);   /* DAC softmute is disabled, DAC
oversampling rate is 128x */

        I2C_WriteWAU8822(14, 0x108);   /* ADC HP filter is disabled, ADC
oversampling rate is 128x */

        I2C_WriteWAU8822(15, 0x1EF);   /* ADC left digital volume control */
        I2C_WriteWAU8822(16, 0x1EF);   /* ADC right digital volume control */

        I2C_WriteWAU8822(44, 0x000);   /* LLIN/RLIN is not connected to PGA */

        I2C_WriteWAU8822(47, 0x050);   /* LLIN connected, and its Gain value */
        I2C_WriteWAU8822(48, 0x050);   /* RLIN connected, and its Gain value */
        I2C_WriteWAU8822(50, 0x001);   /* Left DAC connected to LMIX */
        I2C_WriteWAU8822(51, 0x001);   /* Right DAC connected to RMIX */
}
```

3 CALLING SEQUENCE

3.1 I2S and CODEC Integration

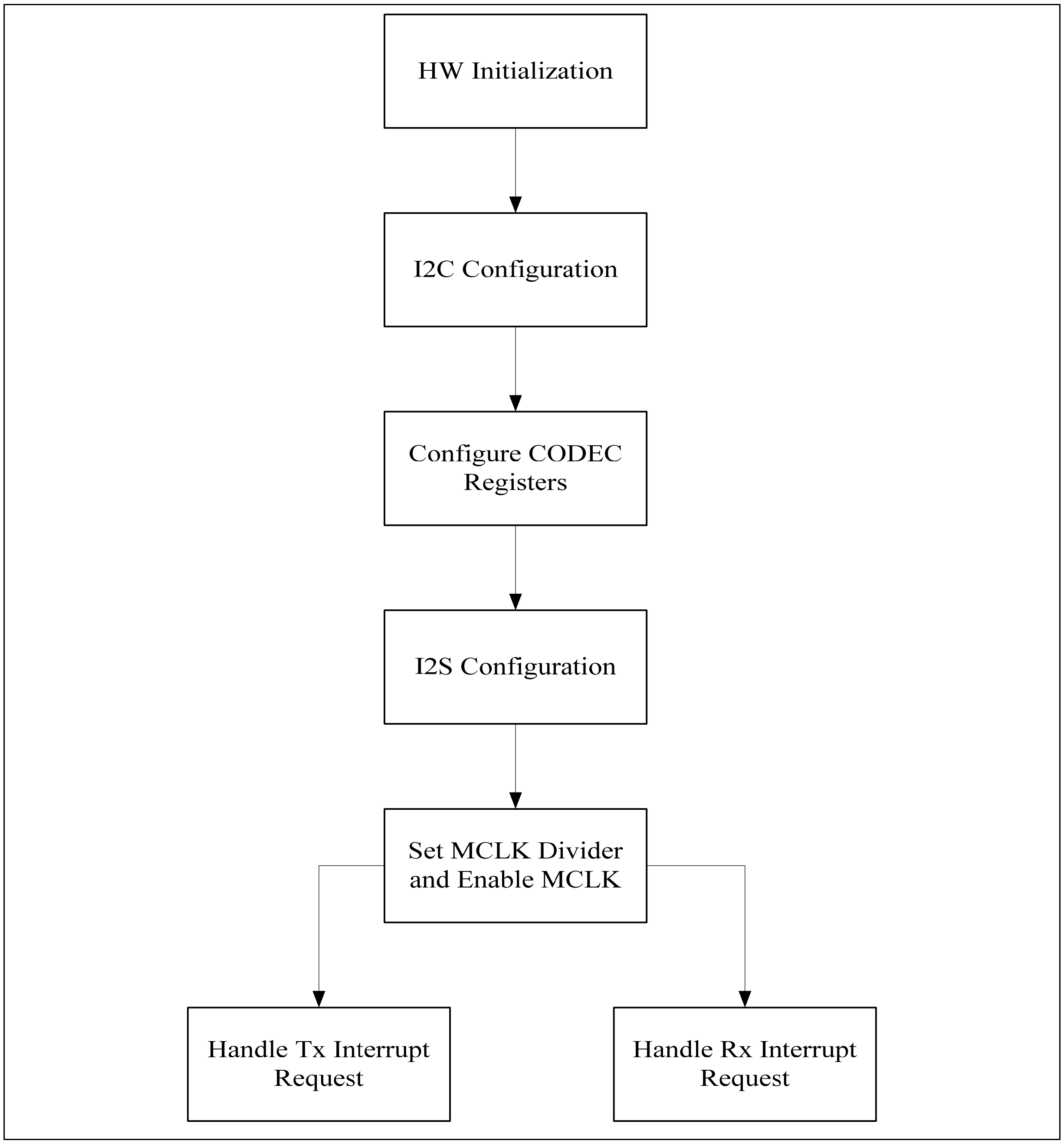
HW Initialization
I2C Configuration
Configure CODEC Registers
I2S Configuration
Set MCLK Divider and Enable MCLK
Handle Tx Interrupt Request
Handle Rx Interrupt Request

Figure 5 Control Flow of I2S and CODEC integration

3.2 NUC1xx Settings

1. Hardware initialization. Set PLL control, HCLK source, and Pin function of I2C and I2S.
2. Open I2C driver to initialize I2C interface, set I2C bus clock, and install I2C callback function.

Refer to I2C Driver Reference Guide.

3. Use I2C interface to configure CODEC relative settings. The master clock source of CODEC is from external pin. Enable CODEC to drive LRCLK and BCLK outputs, and set output clock frequency and audio operation.
4. Open I2S driver to initialize I2C interface. Set I2S as slave mode, audio/data format, and Tx/Rx FIFO threshold level.
5. Set MCLK divider for CODEC clock source, and enable MCLK to CODEC.

3.3 Handle Rx Request

1. I2S receives data from ADC of CODEC and transfer data into Rx FIFO. When word data in Rx FIFO is equal or more than Rx FIFO threshold level, the I2S receive interrupt will be asserted.
2. Read out data from Rx FIFO and transfer to audio buffer.

3.4 Handle Rx Request

1. When word data in Tx FIFO is equal or less than Tx FIFO threshold level, the I2S transmit interrupt will be asserted.
2. Get data from audio buffer and transfer to Tx FIFO.
3. I2S sends data to DAC of CODEC.

3.5 API Usage Reference

- I2S Driver Reference Guide.doc
- I2C Driver Reference Guide.doc

4 EXECUTION ENVIRONMENT SETUP AND RESULT

4.1 Test Smpl_DrvI2S

The I2S sample code, Smpl_DrvI2S, could be built by Keil MDK tool and download to NUC1xx series DEV Board through ICE. Then user is able to execute the code in ICE environment or reset the DEV board to execute the code which had been programmed in on-chip Program Flash.

Then, Input audio source to ADC of WAU8822 and connect DAC of WAU8822 to speaker.

4.2 Result

Hear correct audio from speaker.

5 REVISION HISTORY

REV.	DATE	DESCRIPTION
1.00	March, 2010	Created.

Important Notice