AN14151

MCX Nx4x的MICFIL接口

第1版—2024年1月20日 应用笔记

文档信息

信息	内容
关键词	Nirvana MCX Nx4x、音频、micfil接口
摘要	本应用笔记详细介绍了如何采用不同机制来配置和使用MICFIL接口。

1 介绍

本节提供了相关产品的基本信息。

1.1 芯片概述

MCX Nx4x系列微控制器结合了双核Arm Cortex M33、一个CoolFlux BSP32、一个PowerQuad DSP协处理器、一个NPU以及以150MHz频率运行的多个高速连接选项。为了支持各种应用，MCX N系列集成了先进的串行外设、定时器、高精度模拟和最高水平的安全功能。所有的MCX Nx4x产品均配备了双组闪存，支持对内部闪存的读写操作。MCX Nx4x系列还支持大容量外部串行存储器的配置。

MCX Nx4x系列是双核微控制器系列MCU。CPU0是主Cortex-M33（版本r0p4-00rel0）处理器，支持TrustZone-M、浮点运算单元（FPU）和内存保护单元（MPU）。

此外，MCX Nx4x芯片还包含第二个Cortex-M33实体CPU1，它作为一个副CM33，旨在从主处理器上分担部分工作，以支持特定的专用应用。CPU1的配置不包括MPU、FPU、DSP、ETM、Trustzone-M、安全属性单元（SAU）或协处理器接口。两个内核均支持SYSTICK。

Cortex-M33采用两个32位总线接口：代码总线和系统总线，实现了一种变化的哈佛内存架构。总线接口根据地址范围激活，且在同一总线端口上可同时处理指令获取和操作数数据引用（传统的哈佛架构将指令获取和操作数数据引用严格分离到特定的总线端口上，而与访问地址无关）。代码总线通常用于指令获取和对PC相关的数据访问，而系统总线通常用于对片内外存储器的操作数数据的引用以及外设的访问。此种总线结构完全支持并发指令获取和数据访问，但Cortex-M33的实现可以在每条总线上都生成两种类型的引用。

1.2 Micfil接口

MCX Nx4x具有一个MICFIL模块的实体，支持多达2对麦克风输入，共有4个通道可用。

MICFIL在移动电话等多种应用中可将音频信号从麦克风传送到处理器。由于当前的数字音频系统采用多位音频信号（也称多位PCM）来表示信号，因此该模块实现了所需的数字接口（一系列滤波器），以一个可配置的输出采样率，将一个脉冲密度调制（PDM）麦克风比特流转换为一个24位的PCM音频信号。MICFIL架构旨在节省门控数，并最大限度地降低功耗。MICFIL还支持在多通道模式下工作。所有通道的配置相同，但每个输入通道都可以单独打开或关闭。

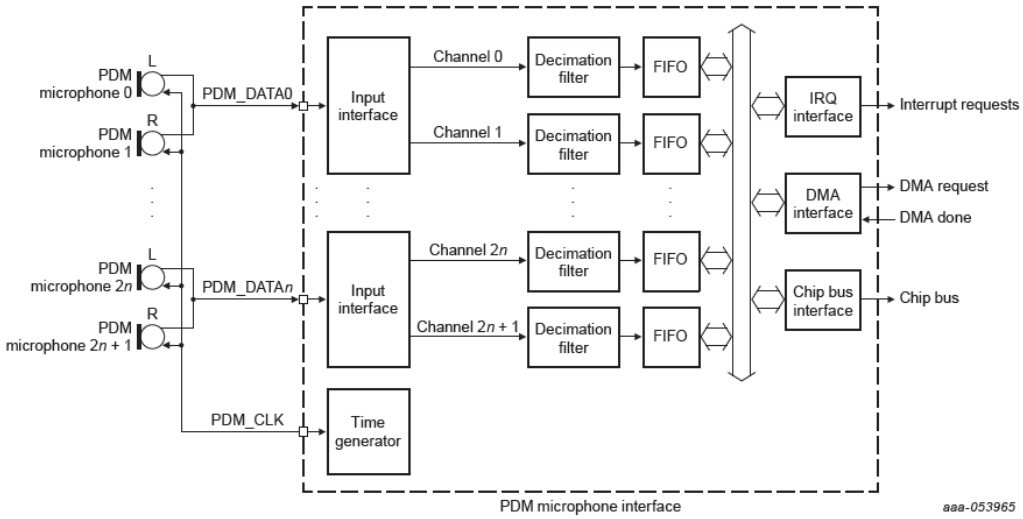

图1. MICFIL的结构框图

主要功能包括：

- 抽取滤波器：
  - 定点过滤。
  - 24位PCM音频输出。
  - 用于生成可编程PDM时钟的内部时钟分频器：支持低频操作的时钟分频器旁路功能。
  - 帧同步。
  - 带有单独启用控制的完整或部分通道操作集。
  - 可编程抽取率。
  - 在输出端可编程的直流消除器。
  - 范围调整功能。
- 带有中断和eDMA功能的FIFO：每个FIFO长度为16个条目。

MICFIL能够发出中断请求或eDMA请求。然后，芯片或eDMA可分别通过内部总线接口访问存储在16个FIFO条目中的滤波结果。

在本应用笔记中，我们将通过示例了解如何利用MICFIL与eDMA、中断功能相结合，将音频流发送到SRAM进行后处理。

这些示例基于MCUXpresso v 11.7和SDK版本2.13.1。

2 基于中断机制的MICFIL

启用后，会触发一个中断，表明过滤结果已存储在FIFO中并准备进行处理。当启用的通道的FIFO超过了水印配置的水平值时，就会触发此中断。

当发生某个异常情况时，如通道输出上溢或下溢，或FIFO缓冲区上溢或下溢，也会产生错误中断请求。

下面的示例演示了如何利用中断机制向CPU发送数据。

在此示例中，MICFIL的时钟频率为24MHz，来自FRO_HF（48MHz）除以2得到的结果。不过，根据实际需求，可以进行多种配置，例如，MICFIL时钟可以连接到FRO_12M、PLL0、CLK_IN、PLL1和SAI0。

不同的时钟源（MICFIL_CLK_ROOT速率）的性能表现不同，并且必须根据目标的MICFIL频率（输出速率）相应地设置MICFIL时钟分频器（CLKDIV）。

$$CLKDIV = \frac{MICFIL\_CLK\_ROOT\ rate}{8 \times OSR \times (output\ rate)}$$

此外，仅启用右通道，FIFO的水印水平为4，采样率为16000。当示例结束时，SRAM数组中存储的数据将显示在终端上。

下面是实现方法：

- 将MICFIL的时钟分频器设置为所需值，并将FRO_HF的时钟连接到MICFIL。

```
/* attach FRO HF to MICD */
CLOCK_SetClkDiv(kCLOCK_DivMicfilFClk, 2u);
CLOCK_AttachClk(kFRO_HF_to_MICFILF);
```

图2. 连接和配置MICFIL的时钟和分频器的示例代码

- PDM的初始化：通过写入CTRL_1寄存器来初始化PDM外设、PDM通道、PDM分频器。

```
int main(void)
{
    uint32_t i = 0U, j = 0U;

    /* attach FRO 12M to FLEXCOMM4 (debug console) */
    CLOCK_SetClkDiv(kCLOCK_DivFlexcom4Clk, 1u);
    CLOCK_AttachClk(BOARD_DEBUG_UART_CLK_ATTACH);

    /* attach FRO HF to MICD */
    CLOCK_SetClkDiv(kCLOCK_DivMicfilFClk, 2u);
    CLOCK_AttachClk(kFRO_HF_to_MICFILF);

    BOARD_InitPins();
    BOARD_PowerMode_OD();
    BOARD_InitBootClocks();
    BOARD_InitDebugConsole();

    PRINTF("PDM interrupt example started! \n\r");

    memset(txBuff, 0U, sizeof(txBuff));

    /* Set up pdm */
    PDM_Init(DEMO_PDM, &pdmConfig);

    PDM_SetChannelConfig(DEMO_PDM, DEMO_PDM_ENABLE_CHANNEL_LEFT, &channelConfig);
    PDM_SetChannelConfig(DEMO_PDM, DEMO_PDM_ENABLE_CHANNEL_RIGHT, &channelConfig);
    if (PDM_SetSampleRateConfig(DEMO_PDM, DEMO_PDM_CLK_FREQ, DEMO_AUDIO_SAMPLE_RATE) != kStatus_Success)
    {
        PRINTF("PDM configure sample rate failed.\r\n");
        return -1;
    }
```

图3. 初始化PDM外设和通道的示例代码

- 启用PDM中断和PDM外设。错误中断和FIFO中断均通过写入CTRL_1寄存器来启用。

```
    PDM_EnableInterrupts(DEMO_PDM, kPDM_ErrorInterruptEnable | kPDM_FIFOInterruptEnable);

    EnableIRQ(PDM_EVENT_IRQn);
#if !(defined FSL_FEATURE_PDM_HAS_NO_INDEPENDENT_ERROR_IRQ && FSL_FEATURE_PDM_HAS_NO_INDEPENDENT_ERROR_IRQ)
    EnableIRQ(PDM_ERROR_IRQn);
#endif
    PDM_Enable(DEMO_PDM, true);

    /* wait data read finish */
    while (!s_dataReadFinishedFlag)
    {
#if (defined(FSL_FEATURE_PDM_HAS_STATUS_LOW_FREQ) && (FSL_FEATURE_PDM_HAS_STATUS_LOW_FREQ == 1U))
        if (s_lowFreqFlag)
        {
            s_lowFreqFlag = false;
            PRINTF("PDM root clock freq too low, please switch to a higher value\r\n");
        }
#endif
    }

    PRINTF("PDM recieve data:\n\r");
    for (i = 0U; i < SAMPLE_COUNT; i++)
    {
        PRINTF("%6x ", txBuff[i]);
        if (++j > 32U)
        {
            j = 0U;
            PRINTF("\r\n");
        }
    }

    PDM_Deinit(DEMO_PDM);
```

图4. 启用PDM中断的示例代码

- 执行PDM的中断服务例程。

当FIFO水位超过配置的水印参数时，将会触发PDM中断服务例程，提取FIFO数据并将其存储到SRAM数组中。为此：

- 通过读取STAT[3-0]寄存器，来检查通道的FIFO是否超过其水印值。如果该值为1，则DATACHn寄存器中的数据可用。
- 如果是这种情况，则使用一个for循环以FIFO的水印值的次数来读取数据，并将数据存储在一个SRAM数组中。这样，DATACHn会被连续读取，给出存储在通道n的FIFO顶部的一个字的数值。

```
void PDM_EVENT_IRQHandler(void)
{
    uint32_t i = 0U, status = PDM_GetStatus(DEMO_PDM);
    /* recieve data */
    if ((1U << DEMO_PDM_ENABLE_CHANNEL_LEFT) & status)
    {
        for (i = 0U; i < DEMO_PDM_FIFO_WATERMARK; i++)
        {
            if (s_readIndex < SAMPLE_COUNT)
            {
                txBuff[s_readIndex] = PDM_ReadData(DEMO_PDM, DEMO_PDM_ENABLE_CHANNEL_LEFT);
                s_readIndex++;
            }
        }
    }

    if ((1U << DEMO_PDM_ENABLE_CHANNEL_RIGHT) & status)
    {
        for (i = 0U; i < DEMO_PDM_FIFO_WATERMARK; i++)
        {
            if (s_readIndex < SAMPLE_COUNT)
            {
                txBuff[s_readIndex] = PDM_ReadData(DEMO_PDM, DEMO_PDM_ENABLE_CHANNEL_RIGHT);
                s_readIndex++;
            }
        }
    }
    PDM_ClearStatus(DEMO_PDM, status);
    if (s_readIndex >= SAMPLE_COUNT)
    {
        s_dataReadFinishedFlag = true;
        PDM_Enable(DEMO_PDM, false);
    }
    __DSB();
```

图5. PDM中断服务例程的示例代码

当异常情况发生时，也可以产生中断。因此，通过读取FIFO_STAT寄存器，可在中断服务例程中处理这一假设。如果寄存器的值与0不同，则表明发生了FIFO上溢或下溢。

在PDM ISR中：

```
    /* handle PDM error status */
#if (defined FSL_FEATURE_PDM_HAS_NO_INDEPENDENT_ERROR_IRQ && FSL_FEATURE_PDM_HAS_NO_INDEPENDENT_ERROR_IRQ)
    pdm_error_irqHandler();
#endif

static void pdm_error_irqHandler(void)
{
    uint32_t status = 0U;

#if (defined(FSL_FEATURE_PDM_HAS_STATUS_LOW_FREQ) && (FSL_FEATURE_PDM_HAS_STATUS_LOW_FREQ == 1U))
    if (PDM_GetStatus(DEMO_PDM) & PDM_STAT_LOWFREQF_MASK)
    {
        PDM_ClearStatus(DEMO_PDM, PDM_STAT_LOWFREQF_MASK);
        s_lowFreqFlag = true;
    }
#endif

    status = PDM_GetFifoStatus(DEMO_PDM);
    if (status != 0U)
    {
        PDM_ClearFIFOStatus(DEMO_PDM, status);
        s_fifoErrorFlag = true;
    }
#if defined(FSL_FEATURE_PDM_HAS_RANGE_CTRL) && FSL_FEATURE_PDM_HAS_RANGE_CTRL
    status = PDM_GetRangeStatus(DEMO_PDM);
    if (status != 0U)
    {
        PDM_ClearRangeStatus(DEMO_PDM, status);
    }
#else
    status = PDM_GetOutputStatus(DEMO_PDM);
    if (status != 0U)
    {
        PDM_ClearOutputStatus(DEMO_PDM, status);
    }

#endif
}

#if !(defined FSL_FEATURE_PDM_HAS_NO_INDEPENDENT_ERROR_IRQ && FSL_FEATURE_PDM_HAS_NO_INDEPENDENT_ERROR_IRQ)
void PDM_ERROR_IRQHandler(void)
{
    pdm_error_irqHandler();
    __DSB();
}
#endif
```

图6. 检测到错误时的PDM中断服务例程的示例代码

`mcxn5xxevk_pdm_interrupt`示例可以在MCX Nx4x的MCUXpresso SDK中找到，位置为`<SDK_LOCATION>\boards\<board_name>\driver_examples\pdm\`。

图7所示为终端中显示的工程运行的结果。

```
PDM interrupt example started!
PDM recieve data:
     0      0  1d800  7de00      0      0  1d800  7de00 30fc600 136fc000 10d99800 f5527400 30fc600 136fc000 10d99800 f5527400 390dc00 a74800 fd610a00 33aea00
 390dc00 a74800 fd610a00 33aea00 fcea1400 2856400 fe286800 1235200 fcea1400 2856400 fe286800 1235200 ff754200
 a7200 4b1600 ff7be000 ff754200  a7200 4b1600 ff7be000 969e00 ff690400 7ed400 ff9c4400 969e00 ff690400 7ed400 ff9c4400 3e6400 ffde1a00  3be00  c5800 3e6400 f
fde1a00  3be00  c5800 ffe40a00 1dc800 ffdee200 199c00 ffe40a00 1dc800 ffdee200 199c00 ffe92600  c4e00
fff6f400 ffffa000 ffe92600  c4e00 fff6f400 ffffe000   de00 fff8fa00  4c600 fff7ae00   de00 fff8fa00  4c600 fff7ae00  41000 fff9ba00  15400 fffc9c00  41000 ff
f9ba00  15400 fffc9c00 fffeb200 fffebc00 ffffd3200 ffff9c00 fffeb200 fffebc00 ffffd3200 ffff9c00 ffffce200 ffff8000 ffffd4a00
fffeee00 ffffce200 ffff8000 ffffd4a00 fffeee00 ffffde400 fffe6000 fffe5800 fffe0e00 ffffde400 fffe6000 fffe5800 fffe0e00 fffe8a00 ffffdfa00 fffe8400 fffe1000 fffe
8a00 ffffdfa00 fffe8400 fffe1000 fffe6800 fffe3000 fffe4a00 fffe4800 fffe6800 fffe3000 fffe4a00 fffe4800 fffe3a00 fffe5200 fffe3600 fffe5400
fffe3a00 fffe5200 fffe3600 fffe5400 fffe3a00 fffe4c00 fffe4000 fffe4600 fffe3a00 fffe4c00 fffe4000 fffe4600 fffe4600 fffe4400 fffe4800 fffe4200 fffe4600 fffe
4400 fffe4800 fffe4200 fffe4800 fffe4400 fffe4600 fffe4600 fffe4800 fffe4400 fffe4600 fffe4600 fffe4600 fffe4600 fffe4600 fffe4600 fffe4600 fffe4600 fffe4600
fffe4600 fffe4600 fffe4600 fffe4600 fffe4600 fffe4600 fffe4600 fffe4600 fffe4600 fffe4600 fffe4600 fffe4600 fffe4600 fffe4600 fffe4600 fffe4600 fffe4600 fffe
4600 fffe4600 fffe4800 fffe4800 fffe4800 fffe4800 fffe4800 fffe4800 fffe4800 fffe4800 fffe4800 fffe4800 fffe4800 fffe4800 fffe4800 fffe4800 fffe4800 fffe4800
fffe4800 fffe4800 fffe4800 fffe4800 fffe4800 fffe4800 fffe4800 fffe4800 fffe4800 fffe4800 fffe4800 fffe4800 fffe4800 fffe4800 fffe4800 fffe4800 fffe4800 fffe
4800 fffe4800 fffe4800 fffe4800 fffe4800 fffe4800 fffe4800 fffe4800 fffe4800 fffe4800 fffe4a00 fffe4a00 fffe4a00 fffe4800 fffe4a00 fffe4a00
fffe4a00 fffe4a00 fffe4a00 fffe4a00 fffe4a00 fffe4a00 fffe4a00 fffe4a00 fffe4a00 fffe4a00 fffe4a00 fffe4a00 fffe4a00 fffe4a00 fffe4a00 fffe4a00 fffe4a00 fffe
4a00 fffe4a00 fffe4a00 fffe4a00 fffe4a00 fffe4a00 fffe4a00
PDM interrupt example finished!
```

图7. 串行终端上的PDM中断示例输出

3 基于eDMA的MICFIL

启用后，当PDM的FIFO超过配置的水印参数时，将触发eDMA传输。通常情况下，如果至少有一个已启用的通道，且所有已启用通道的FIFO都达到了其水印水位，则输出接口就会发出一个eDMA事务的请求。这一过程无需CPU介入，即使系统处于低功耗状态也能完成。

以下示例演示了如何利用eDMA机制向SRAM传输数据。

在此示例中，MICFIL的时钟频率为24MHz，来自FRO_HF（48MHz）除以2得到的结果，与上例相同。

此外，仅启用右通道，FIFO水印水平为4，采样率为16000。当示例结束时，SRAM数组中存储的数据将显示在终端上。

由于中断机制被eDMA取代，因此不执行PDM ISR，仅启用FIFO出错时的中断。处理可能出现的FIFO下溢/上溢错误的PDM ISR与上例相同。

```
#define DEMO_PDM                PDM
#define DEMO_PDM_ERROR_IRQn     PDM_EVENT_IRQn

PDM_EnableInterrupts(DEMO_PDM, kPDM_ErrorInterruptEnable);
EnableIRQ(DEMO_PDM_ERROR_IRQn);
```

图8. 启用PDM错误中断的示例代码

MCX Nx4x系列提供了两个16通道的eDMA控制器，使用控制器0的通道0。通过向DMA0->CH0_MUX寄存器写入MICFIL源编号的值，该通道被配置为与MICFIL连接。

表1. DMAMUX0请求分配

DMAMUX编号	别名	来源说明
9	CTIMER1	DMAREQ_M0
10	CTIMER1	DMAREQ_M1
11	CTIMER2	DMAREQ_M0
12	CTIMER2	DMAREQ_M1
13	CTIMER3	DMAREQ_M0
14	CTIMER3	DMAREQ_M1
15	CTIMER4	DMAREQ_M0
16	CTIMER4	DMAREQ_M1
17	wuuo	Wakeupevent
18	MICFIL0	FIFO_request

必须初始化eDMA外设，并为DMA0通道0创建一个eDMA句柄，以存储回调函数和参数。此处使用默认配置。

eDMA的初始化是通过写入MP_CSR寄存器来完成的。管理页控制（MP_CSR）寄存器定义了DMA的基本操作配置。

```
#define DEMO_EDMA               DMA0
#define DEMO_PDM_EDMA_CHANNEL   kDmaRequestMuxMicfil0FifoRequest
#define DEMO_EDMA_CHANNEL       0
#define DEMO_AUDIO_SAMPLE_RATE 16000
#define BUFFER_SIZE (256)

AT_NONCACHEABLE_SECTION_ALIGN(edma_handle_t dmaHandle, 4);

    /* Create EDMA handle */
    /*
     * dmaConfig.enableRoundRobinArbitration = false;
     * dmaConfig.enableHaltOnError = true;
     * dmaConfig.enableContinuousLinkMode = false;
     * dmaConfig.enableDebugMode = false;
     */
    EDMA_GetDefaultConfig(&dmaConfig);
    EDMA_Init(DEMO_EDMA, &dmaConfig);
    EDMA_CreateHandle(&dmaHandle, DEMO_EDMA, DEMO_EDMA_CHANNEL);
#if defined(FSL_FEATURE_EDMA_HAS_CHANNEL_MUX) && FSL_FEATURE_EDMA_HAS_CHANNEL_MUX
    EDMA_SetChannelMux(DEMO_EDMA, DEMO_EDMA_CHANNEL, DEMO_PDM_EDMA_CHANNEL);
#endif
```

图9. eDMA初始化的示例代码

必须通过以下方式对PDM进行相应的初始化：

- 初始化PDM外设。

```
#define DEMO_PDM                    PDM
#define DEMO_PDM_FIFO_WATERMARK     (4)
#define DEMO_PDM_QUALITY_MODE       kPDM_QualityModeHigh
#define DEMO_PDM_CIC_OVERSAMPLE_RATE  (0U)
static const pdm_config_t pdmConfig = {
    .enableDoze         = false,
    .fifoWatermark      = DEMO_PDM_FIFO_WATERMARK,
    .qualityMode        = DEMO_PDM_QUALITY_MODE,
    .cicOverSampleRate = DEMO_PDM_CIC_OVERSAMPLE_RATE,
};
    /* Set up pdm */
    PDM_Init(DEMO_PDM, &pdmConfig);
```

图10. PDM的初始化和配置的示例代码

- 初始化PDM Rx eDMA句柄，并将其链接到eDMA句柄以便传输PDM数据。此外，定义了一个回调函数。一旦DMA传输完成，就会调用该回调函数。

```
static void pdmEdmallback(PDM_Type *base, pdm_edma_handle_t *handle, status_t status, void *userData);

AT_NONCACHEABLE_SECTION_ALIGN(pdm_edma_handle_t pdmRxHandle, 4);

PDM_TransferCreateHandleEDMA(DEMO_PDM, &pdmRxHandle, pdmEdmallback, NULL, &dmaHandle);
```

图11. PDM的eDMA中断服务例程的示例代码

- 安装PDM Rx EDMA句柄的EDMA描述符内存（TCD，即传输控制描述符）。TCD是每个eDMA通道的结构，包含eDMA的传输信息及其相关属性。

```
AT_QUICKACCESS_SECTION_DATA_ALIGN(edma_tcd_t s_edmaTcd[1], 32U);
PDM_TransferInstallEDMATCDMemory(&pdmRxHandle, s_edmaTcd, 1);
```

图12. 创建一个PDM eDMA内存描述符的示例代码

- 配置PDM通道，此处仅使用左通道（通道0）。该配置由用户自定义，可通过写入PDM CTRL_1寄存器来配置抽取滤波器输出增益、直流去除器截止频率和PDM输出直流去除器截止频率。

```
#define DEMO_PDM_ENABLE_CHANNEL_LEFT  (0U)
static const pdm_channel_config_t channelConfig = {
#if (defined(FSL_FEATURE_PDM_HAS_DC_OUT_CTRL) && (FSL_FEATURE_PDM_HAS_DC_OUT_CTRL))
    .outputCutOffFreq = kPDM_DcRemoverCutOff40Hz,
#else
    .cutOffFreq = kPDM_DcRemoverCutOff152Hz,
#endif
#ifdef DEMO_PDM_CHANNEL_GAIN
    .gain = DEMO_PDM_CHANNEL_GAIN,
#else
    .gain       = kPDM_DfOutputGain4,
#endif
};
PDM_TransferSetChannelConfigEDMA(DEMO_PDM, &pdmRxHandle, DEMO_PDM_ENABLE_CHANNEL_LEFT, &channelConfig);
```

图13. 配置PDM通道的示例代码

- 通过写入PDM CTRL_2寄存器来配置PDM采样率。

```
#define DEMO_PDM_CLK_FREQ            CLOCK_GetMicfilClkFreq()
#define DEMO_AUDIO_SAMPLE_RATE 16000
if (PDM_SetSampleRateConfig(DEMO_PDM, DEMO_PDM_CLK_FREQ, DEMO_AUDIO_SAMPLE_RATE) != kStatus_Success)
{
    PRINTF("PDM configure sample rate failed.\r\n");
    return -1;
}
```

图14. 配置PDM采样率的示例代码

- 初始化并执行eDMA传输。这种机制非常灵活，支持多种数据传输方式。例如，可以发送多PDM通道数据（按信道交错发送或按块发送），并使用链路传输功能支持静态/动态的散点收集。本例中配置了一个简单的传输任务，其目标是将2字节数据存储到一个256字节大小的数组中。

```
AT_NONCACHEABLE_SECTION_ALIGN(static int16_t rxBuff[BUFFER_SIZE], 4);
xfer.data          = (uint8_t *)rxBuff;
xfer.dataSize      = BUFFER_SIZE * 2U;
xfer.linkTransfer = NULL;

PDM_TransferReceiveEDMA(DEMO_PDM, &pdmRxHandle, &xfer);
```

图15. 初始化和配置eDMA传输的示例代码

仅当PDM的FIFO超过水印的参数时，才会触发DMA传输。当DMA传输完成后，将调用之前定义的PDM eDMA回调函数。

`mcxn5xxevk_pdm_edma_transfer`示例可在MCX Nx4x的MCUXpresso SDK中找到，位置为：`<SDK_LOCATION>\boards\<board_name>\driver_examples\pdm\`。

图16所示为终端中显示的工程运行的结果。

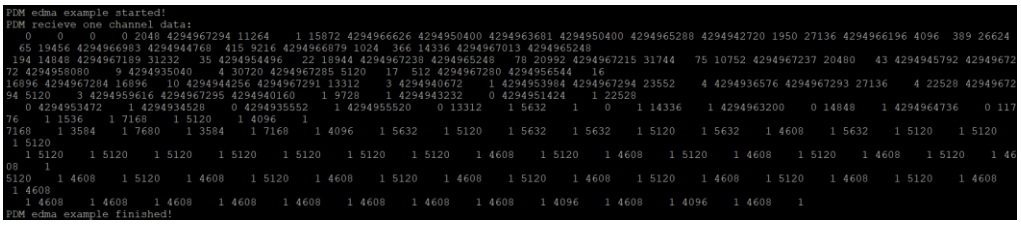

图16. 带有eDMA传输功能的PDM的示例在串行终端上的输出结果

4 结论

本应用笔记简要介绍了MCX Nx4x系列芯片中的MICFIL模块。基于MCUX Nx4x 的SDK示例，阐述了如何利用MICFIL中断或eDMA来处理内存的数据管理。这些机制适用于许多复杂或简单的应用场景，例如使用带I2S的SAI接口将数据直接发送到编解码器，从而无需CPU的介入。

5 关于本文中源代码的说明

本文中所示的示例代码具有以下版权和BSD-3-Clause许可：

2024年恩智浦版权所有；在满足以下条件的情况下，可以源代码和二进制文件的形式重新分发和使用本源代码（无论是否经过修改）：

1. 重新分发源代码必须保留上述版权声明、这些条件和以下免责声明。
2. 以二进制文件形式重新分发时，必须在文档和/或随分发提供的其他材料中复制上述版权声明、这些条件和以下免责声明。
3. 未经事先书面许可，不得使用版权所有者的姓名或参与者的姓名为本软件的衍生产品进行背书或推广。

本软件由版权所有者和参与者“按原样”提供，不承担任何明示或暗示的担保责任，包括但不限于对适销性和特定用途适用性的暗示保证。在任何情况下，无论因何种原因或根据何种法律条例，版权所有者或参与者均不对因使用本软件而导致的任何直接、间接、偶然、特殊、惩戒性或后果性损害（包括但不限于采购替代商品或服务；使用损失、数据损失或利润损失或业务中断）承担责任，无论是因合同、严格责任还是侵权行为（包括疏忽或其他原因）造成的，即使事先被告知有此类损害的可能性也不例外。

6 修订历史

表2. 修订历史

文档ID	发布日期	描述
AN14151 v.1.0	2024年1月20日	初始版本

Legal information

Definitions

Draft — A draft status on a document indicates that the content is still under internal review and subject to formal approval, which may result in modifications or additions. NXP Semiconductors does not give any representations or warranties as to the accuracy or completeness of information included in a draft version of a document and shall have no liability for the consequences of use of such information.

Disclaimers

Limited warranty and liability — Information in this document is believed to be accurate and reliable. However, NXP Semiconductors does not give any representations or warranties, expressed or implied, as to the accuracy or completeness of such information and shall have no liability for the consequences of use of such information. NXP Semiconductors takes no responsibility for the content in this document if provided by an information source outside of NXP Semiconductors.

In no event shall NXP Semiconductors be liable for any indirect, incidental, punitive, special or consequential damages (including - without limitation - lost profits, lost savings, business interruption, costs related to the removal or replacement of any products or rework charges) whether or not such damages are based on tort (including negligence), warranty, breach of contract or any other legal theory.

Notwithstanding any damages that customer might incur for any reason whatsoever, NXP Semiconductors' aggregate and cumulative liability towards customer for the products described herein shall be limited in accordance with the Terms and conditions of commercial sale of NXP Semiconductors.

Right to make changes — NXP Semiconductors reserves the right to make changes to information published in this document, including without limitation specifications and product descriptions, at any time and without notice. This document supersedes and replaces all information supplied prior to the publication hereof.

Suitability for use — NXP Semiconductors products are not designed, authorized or warranted to be suitable for use in life support, life-critical or safety-critical systems or equipment, nor in applications where failure or malfunction of an NXP Semiconductors product can reasonably be expected to result in personal injury, death or severe property or environmental damage. NXP Semiconductors and its suppliers accept no liability for inclusion and/or use of NXP Semiconductors products in such equipment or applications and therefore such inclusion and/or use is at the customer's own risk.

Applications — Applications that are described herein for any of these products are for illustrative purposes only. NXP Semiconductors makes no representation or warranty that such applications will be suitable for the specified use without further testing or modification.

Customers are responsible for the design and operation of their applications and products using NXP Semiconductors products, and NXP Semiconductors accepts no liability for any assistance with applications or customer product design. It is customer's sole responsibility to determine whether the NXP Semiconductors product is suitable and fit for the customer's applications and products planned, as well as for the planned application and use of customer's third party customer(s). Customers should provide appropriate design and operating safeguards to minimize the risks associated with their applications and products.

NXP Semiconductors does not accept any liability related to any default, damage, costs or problem which is based on any weakness or default in the customer's applications or products, or the application or use by customer's third party customer(s). Customer is responsible for doing all necessary testing for the customer's applications and products using NXP Semiconductors products in order to avoid a default of the applications and the products or of the application or use by customer's third party customer(s). NXP does not accept any liability in this respect.

Terms and conditions of commercial sale — NXP Semiconductors products are sold subject to the general terms and conditions of commercial sale, as published at https://www.nxp.com.cn/profile/terms, unless otherwise agreed in a valid written individual agreement. In case an individual agreement is concluded only the terms and conditions of the respective agreement shall apply. NXP Semiconductors hereby expressly objects to applying the customer's general terms and conditions with regard to the purchase of NXP Semiconductors products by customer.

Export control — This document as well as the item(s) described herein may be subject to export control regulations. Export might require a prior authorization from competent authorities.

Suitability for use in non-automotive qualified products — Unless this document expressly states that this specific NXP Semiconductors product is automotive qualified, the product is not suitable for automotive use. It is neither qualified nor tested in accordance with automotive testing or application requirements. NXP Semiconductors accepts no liability for inclusion and/or use of non-automotive qualified products in automotive equipment or applications.

In the event that customer uses the product for design-in and use in automotive applications to automotive specifications and standards, customer (a) shall use the product without NXP Semiconductors' warranty of the product for such automotive applications, use and specifications, and (b) whenever customer uses the product for automotive applications beyond NXP Semiconductors' specifications such use shall be solely at customer's own risk, and (c) customer fully indemnifies NXP Semiconductors for any liability, damages or failed product claims resulting from customer design and use of the product for automotive applications beyond NXP Semiconductors' standard warranty and NXP Semiconductors' product specifications.

Translations — A non-English (translated) version of a document, including the legal information in that document, is for reference only. The English version shall prevail in case of any discrepancy between the translated and English versions.

Security — Customer understands that all NXP products may be subject to unidentified vulnerabilities or may support established security standards or specifications with known limitations. Customer is responsible for the design and operation of its applications and products throughout their lifecycles to reduce the effect of these vulnerabilities on customer's applications and products. Customer's responsibility also extends to other open and/or proprietary technologies supported by NXP products for use in customer's applications. NXP accepts no liability for any vulnerability. Customer should regularly check security updates from NXP and follow up appropriately.

Customer shall select products with security features that best meet rules, regulations, and standards of the intended application and make the ultimate design decisions regarding its products and is solely responsible for compliance with all legal, regulatory, and security related requirements concerning its products, regardless of any information or support that may be provided by NXP.

NXP has a Product Security Incident Response Team (PSIRT) (reachable at PSIRT@nxp.com) that manages the investigation, reporting, and solution release to security vulnerabilities of NXP products.

NXP B.V. — NXP B.V. is not an operating company and it does not distribute or sell products.

Trademarks

Notice: All referenced brands, product names, service names, and trademarks are the property of their respective owners.

NXP — wordmark and logo are trademarks of NXP B.V.

目录

For more information, please visit: https://www.nxp.com.cn

Date of release: 20 January 2024
Document identifier: AN14151