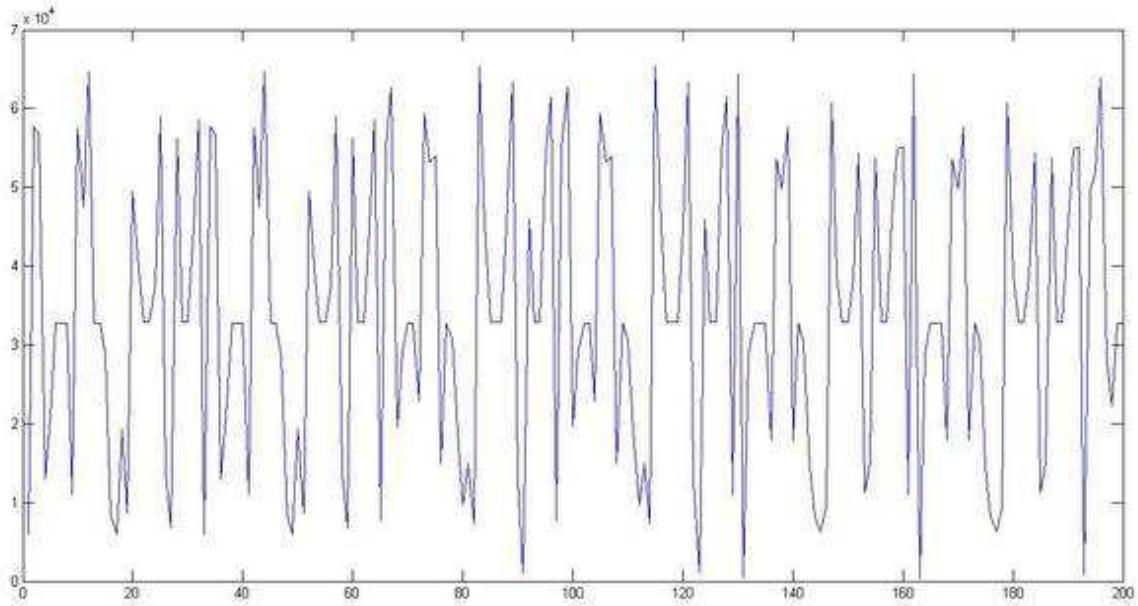


Hallo,

I am working on extracting the data from the MP45DT02. I used PDM filter function. I gave a beep sound to the microphone in the certain frequency, but I noticed that the captured raw data is not given as a sine wave.



Is this kind of data is correct ? because I am expecting some sine wave. The sound frequency that is transmitted to the MIC is 1000Hz. This is my setting for the MP45DT02.

```
#define SPI_Nr      SPI2
#define DMA_SPI     DMA1_Stream3
#define PDM_LENGTH 320
#define PCM_LENGTH 32
/*Private function-----*/
/* Temporary data sample in the DMA memory */
uint16_t AUDIO_SAMPLE_RX_MEM1 [PDM_LENGTH]; //< Array of DMA MEM0 for double buffer
uint16_t AUDIO_SAMPLE_RX_MEM0 [PDM_LENGTH]; //< Array of DMA MEM1 for double buffer

static uint32_t AudioRecInit = 0; //Current state of the audio recorder interface initialization
PDMFilter_InitStruct Filter;

/* Main buffer pointer for the recorded data storing */
uint16_t* pAudioRecBuf;           //< Pointer for Audio record for PDM data collect

uint16_t OutputBuffer0[PCM_LENGTH];    //< Output Buffer 0 after PDM
uint16_t OutputBuffer1[PCM_LENGTH];    //< Output Buffer 1 after PDM
void SignalReceive_Init(void)
{
    /* Configure the SPI with 32KHz Audio frequency */
    WaveRecord_SPI_Init();

    /* Configure the DMA (double buffer) and interrupts */
    WaveRecord_DMA_Init();

    /* Filter LP & HP Init */
    Filter.LP_HZ = 32000;
    Filter.HP_HZ = 10;
    Filter.Fs = 32000;
    Filter.Out_MicChannels = 1;
    Filter.In_MicChannels = 1;

    PDM_Filter_Init((PDMFilter_InitStruct *)&Filter);

    /* Set state of the audio recorder to initialized */
    AudioRecInit = 1;
}
```

```

} // End of SignalReceive_Init()

void WaveRecord_SPI_Init(void)
{
    /*****I2S Initialization *****/
    I2S_InitTypeDef I2S_InitStructure;

    SPI_I2S_DeInit(SPI_Nr);

    I2S_InitStructure.I2S_AudioFreq =64000; //I2S_AudioFreq_48k ; //< Audio is recorded from two
channels, so this value must be twice the value in the PDM filter
    I2S_InitStructure.I2S_Standard = I2S_Standard_LSB;
    I2S_InitStructure.I2S_DataFormat = I2S_DataFormat_16b;
    I2S_InitStructure.I2S_CPOL = I2S_CPOL_High;
    I2S_InitStructure.I2S_Mode = I2S_Mode_MasterRx;
    I2S_InitStructure.I2S_MCLKOutput = I2S_MCLKOutput_Disable;

    /* Initialize the I2S peripheral with the structure above */
    I2S_Init(SPI_Nr, &I2S_InitStructure);

} // End of WaveRecord_SPI_Init()

void WaveRecord_DMA_Init(void)
{
    /***** The DMA Configuration *****/
    DMA_InitTypeDef DMA_InitStructure;

    /***** DMA init (DMA1, Channel0, Stream3)*****/
    DMA_Cmd(DMA_SPI, DISABLE);
    DMA_DeInit(DMA_SPI);
    DMA_InitStructure.DMA_Channel = DMA_Channel_0;
    DMA_InitStructure.DMA_PeripheralBaseAddr = (uint32_t)&(SPI_Nr->DR);
    DMA_InitStructure.DMA_Memory0BaseAddr = (uint32_t)AUDIO_SAMPLE_RX_MEM0;
    DMA_InitStructure.DMA_DIR = DMA_DIR_PeripheralToMemory ;
    DMA_InitStructure.DMA_BufferSize =PDM_LENGTH;
    DMA_InitStructure.DMA_PeripheralInc = DMA_PeripheralInc_Disable;
    DMA_InitStructure.DMA_MemoryInc = DMA_MemoryInc_Enable;
    DMA_InitStructure.DMA_PeripheralDataSize = DMA_PeripheralDataSize_HalfWord;
    DMA_InitStructure.DMA_MemoryDataSize = DMA_MemoryDataSize_HalfWord;
    DMA_InitStructure.DMA_Mode = DMA_Mode_Circular;
    DMA_InitStructure.DMA_Priority = DMA_Priority_Medium ;
    DMA_InitStructure.DMA_FIFOMode = DMA_FIFOMode_Disable;
    DMA_InitStructure.DMA_FIFOThreshold = DMA_FIFOThreshold_HalfFull;
    DMA_InitStructure.DMA_MemoryBurst = DMA_MemoryBurst_Single;
    DMA_InitStructure.DMA_PeripheralBurst = DMA_PeripheralBurst_Single;
    DMA_Init(DMA_SPI, &DMA_InitStructure);

    /*****Setup the Dual Buffer mode for the Rx DMA1 Stream 3 *****/
    DMA_DoubleBufferModeConfig (DMA_SPI, (uint32_t)AUDIO_SAMPLE_RX_MEM1,DMA_Memory_1);
    DMA_DoubleBufferModeCmd(DMA_SPI, ENABLE);

    DMA_Init(DMA_SPI, &DMA_InitStructure);
    /***** Transfer complete interrupt enabling *****/
    DMA_ITConfig(DMA_SPI, DMA_IT_TC, ENABLE);

    /***** DMA enable*****/
    DMA_Cmd(DMA_SPI, ENABLE);

} // End of WaveRecord_DMA_Init()

uint8_t SignalReceive_start(void)
{
    /* Check if the interface has already been initialized */
}

```

```

/*Enable the I2S DMA */
SPI_I2S_DMACmd( SPI_Nr, SPI_I2S_DMAReq_Rx, ENABLE);

/* Enable the SPI peripheral */
I2S_Cmd(SPI_Nr, ENABLE);
/* Return 0 if all operations are OK */
return 0;

I2S_ISR_Indication=0;
}

void DMA1_Stream3_IRQHandler(void)
{
    uint16_t volume;
    volume = 64;
    //*****Check the transfer complete event *****

    if (DMA_GetITStatus(DMA_SPI, DMA_IT_TCIF3)== SET)
    {
        DMA_ClearFlag(DMA_SPI, DMA_FLAG_TCIF3);
        //GPIO_ToggleBits(GPIOD, GPIO_Pin_14);

        if(DMA_GetCurrentMemoryTarget(DMA_SPI)==1) //
        {

            pAudioRecBuf=OutputBuffer0;
            PDM_Filter_64_LSB((uint8_t *)AUDIO_SAMPLE_RX_MEM0,(uint16_t *)pAudioRecBuf, volume ,
(PDMFilter_InitStruct *)&Filter);

        }
        else
        {
            pAudioRecBuf=OutputBuffer1;
            PDM_Filter_64_LSB((uint8_t *)AUDIO_SAMPLE_RX_MEM1,(uint16_t *)pAudioRecBuf, volume ,
(PDMFilter_InitStruct *)&Filter);

        }
    }

    DMA_ClearITPendingBit(DMA_SPI, DMA_IT_HTIF3); ///< clear pending interrupt
} //< End of DMA1_Stream3_IRQHandler

```

Hope someone can help me. Thanks