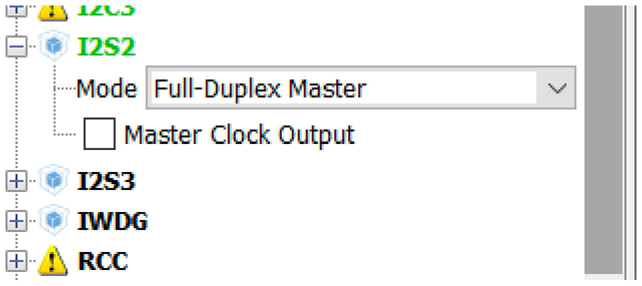
MP45DT02: digital microphone

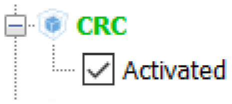
Set up Project configuration in STM32Cube MX

Open I2S Full-Duplex Master Mode



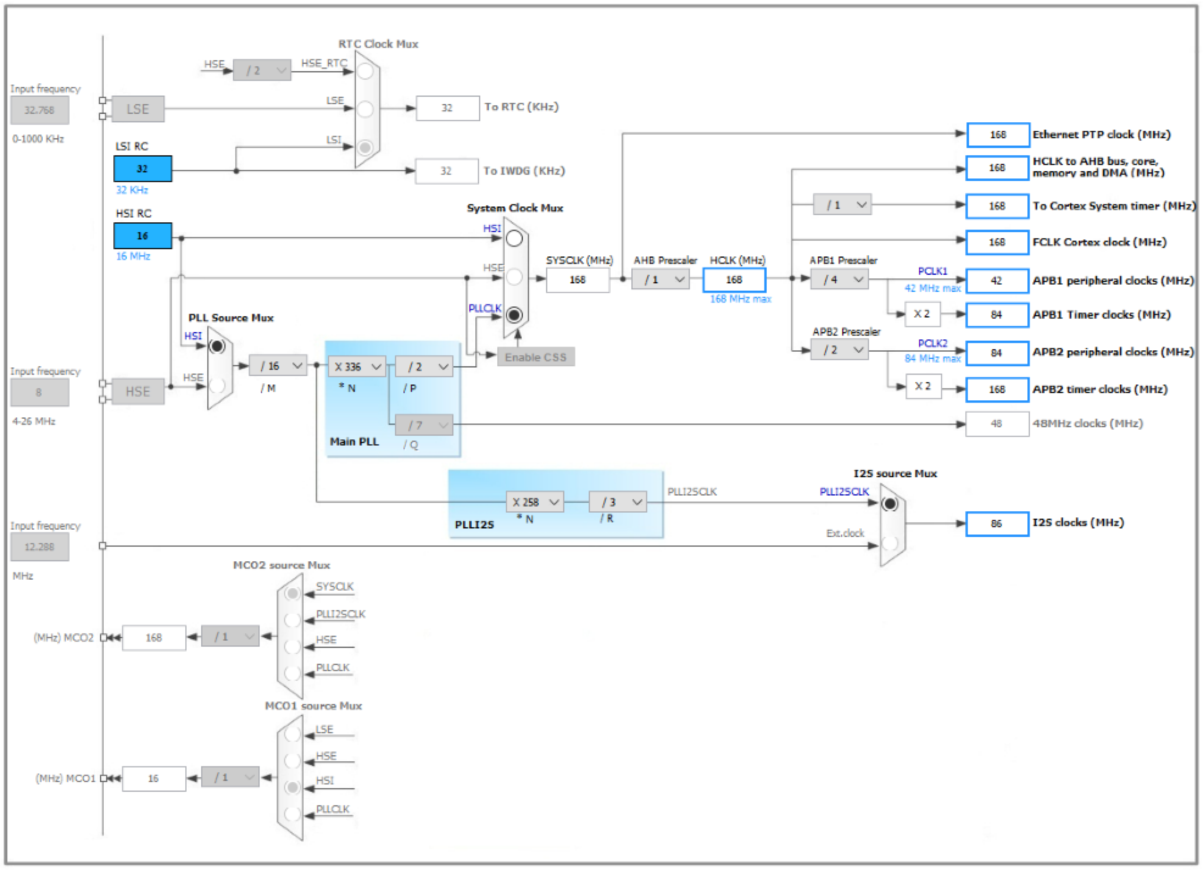
I2S2
Mode: Full-Duplex Master
☐ Master Clock Output
I2S3
IWDG
RCC

Open CRC for using PDM Filter library



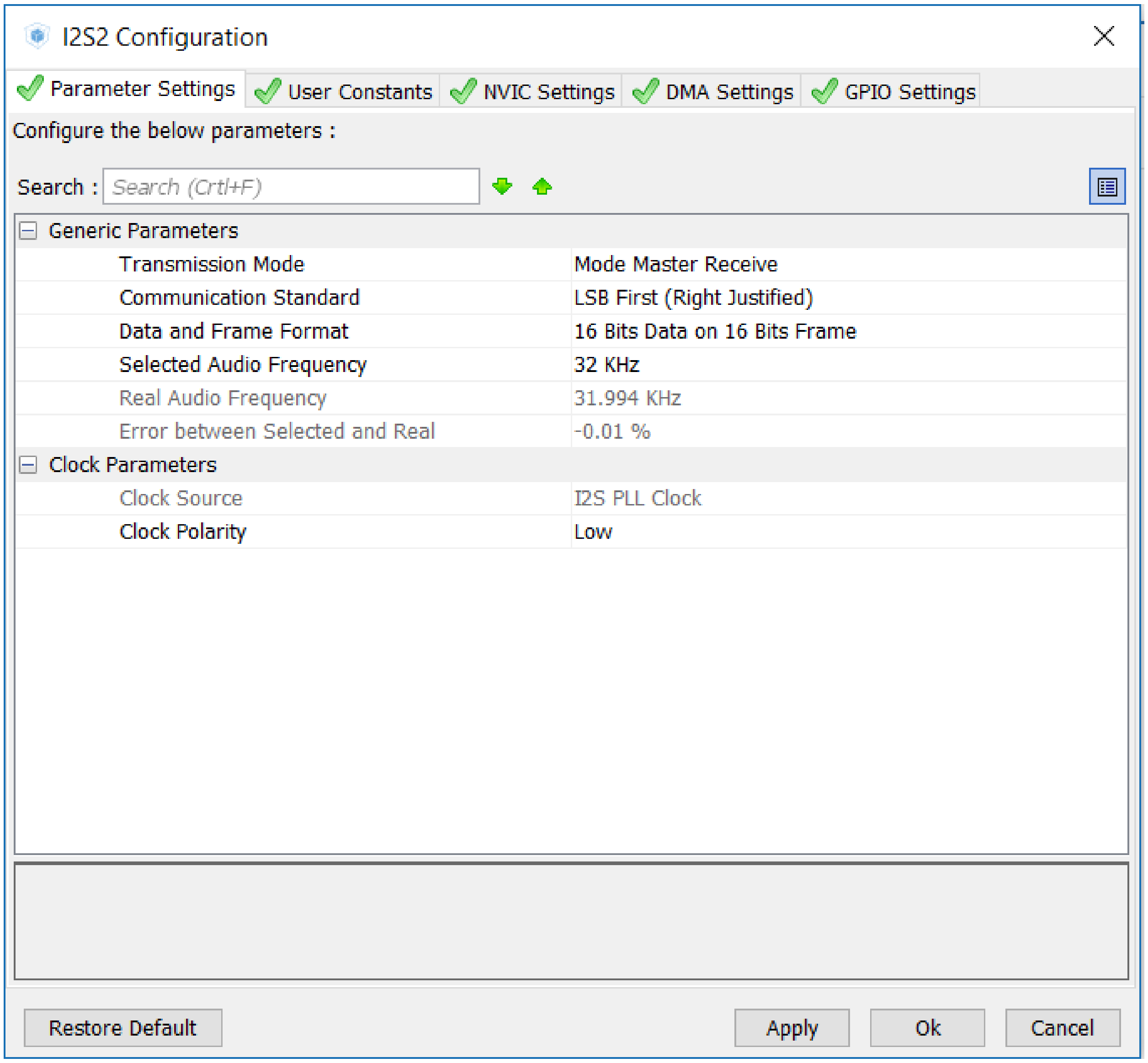
CRC
☑ Activated

Set up clock for I2S



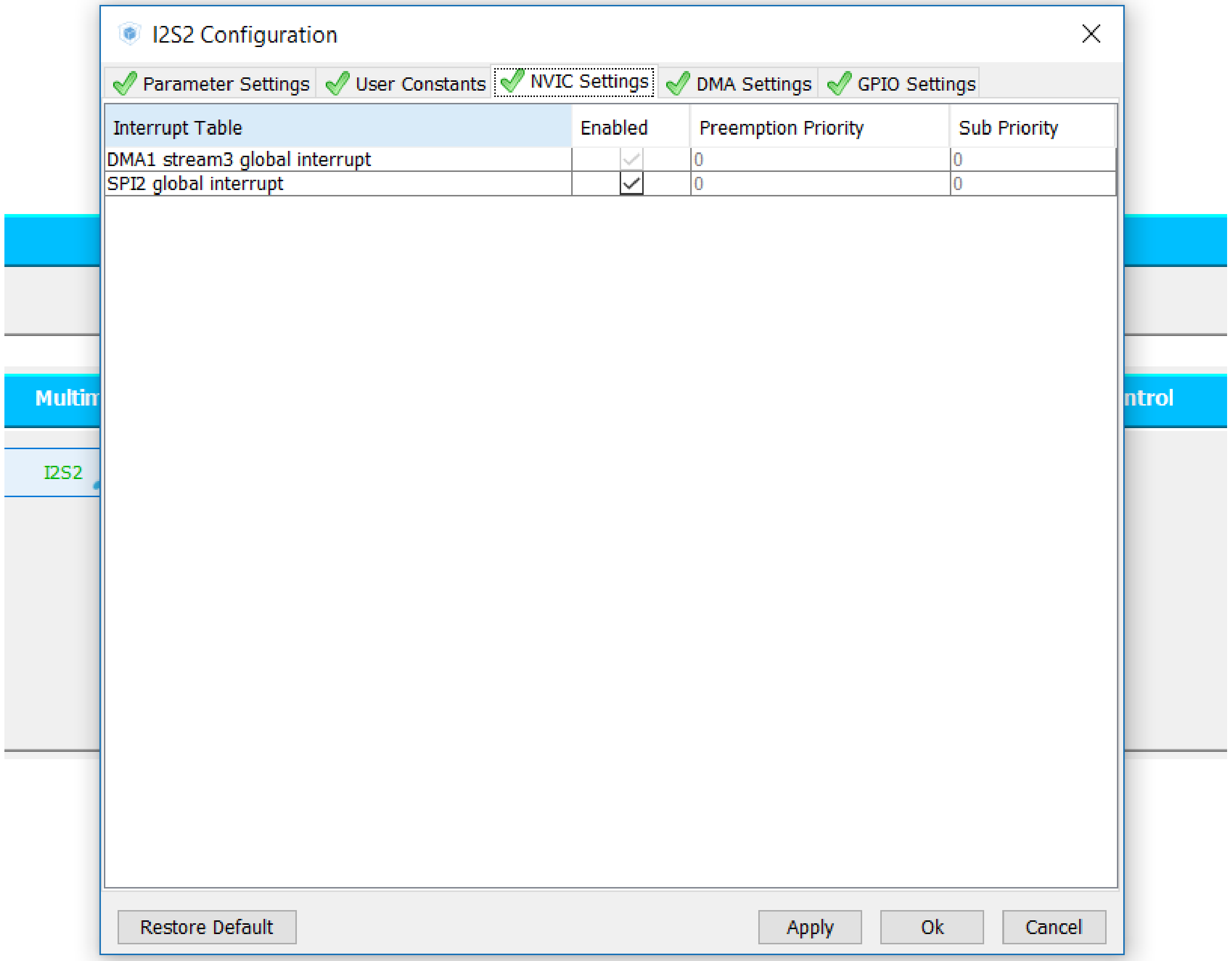
Set up I2S same as configuration below

Nuttapat Kirawittaya 5731035121

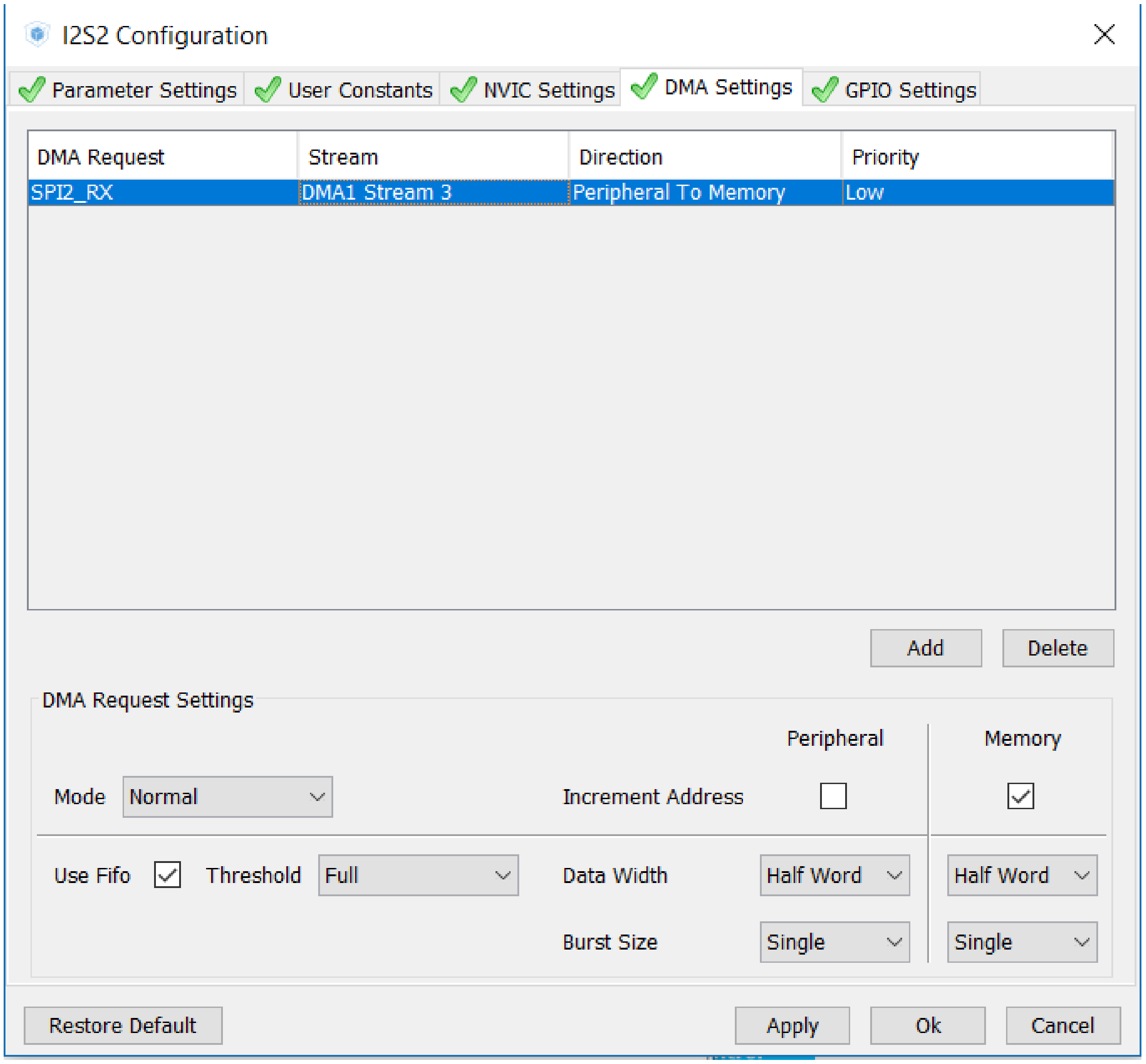


Configure the below parameters :

Generic Parameters	
Transmission Mode	Mode Master Receive
Communication Standard	LSB First (Right Justified)
Data and Frame Format	16 Bits Data on 16 Bits Frame
Selected Audio Frequency	32 KHz
Real Audio Frequency	31.994 KHz
Error between Selected and Real	-0.01 %
Clock Parameters	
Clock Source	I2S PLL Clock
Clock Polarity	Low



Interrupt Table	Enabled	Preemption Priority	Sub Priority
DMA1 stream3 global interrupt	✓	0	0
SPI2 global interrupt	✓	0	0



DMA Request	Stream	Direction	Priority
SPI2_RX	DMA1 Stream 3	Peripheral To Memory	Low

DMA Request Settings

Mode: Normal

Use Fifo ✓ Threshold: Full

	Peripheral	Memory
Increment Address	☐	✓
Data Width	Half Word	Half Word
Burst Size	Single	Single

Code in main.c

```
#include "pdm_filter.h"
```

Add library pdm_filter to our project include to main.c

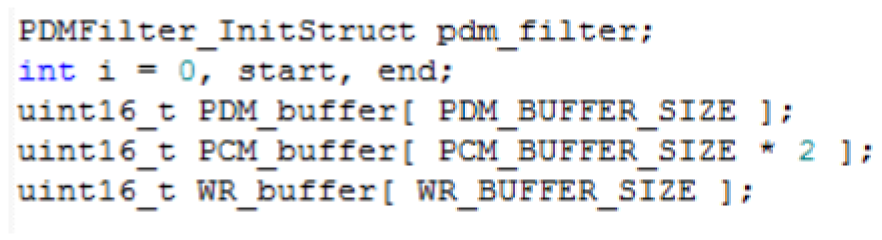
```
#define FS 16000
#define PDM_BUFFER_SIZE 64 // 16000 / 1000 * 1 * 64 / 8 -> uint8_t[128] -> uint8_t[64]
#define PCM_BUFFER_SIZE 16 // 16000 / 1000 * 1
#define WR_BUFFER_SIZE 4096

#define PI 3.14159265f
```

FS is frequecy for PDM Filter

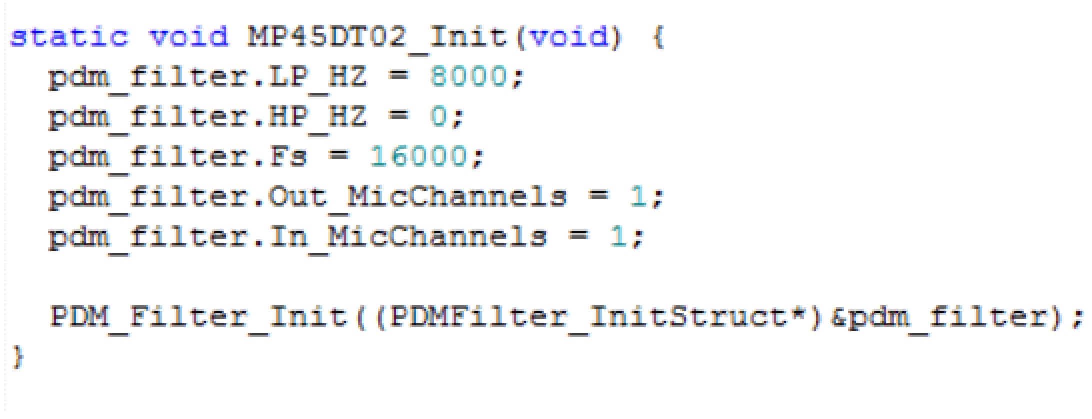
*PDM_BUFFER_SIZE is size of pdm buffer (calculate from Output freq / 1000 * Microphone channel / 8)*

*PCM_BUFFER_SIZE is size of pcm buffer (calculate from Output freq / 1000 * Microphone channel)*



```
PDMFilter_InitStruct pdm_filter;
int i = 0, start, end;
uint16_t PDM_buffer[ PDM_BUFFER_SIZE ];
uint16_t PCM_buffer[ PCM_BUFFER_SIZE * 2 ];
uint16_t WR_buffer[ WR_BUFFER_SIZE ];
```

PDMFilter_InitStruct is initial value for pdm_filter, and declare buffer for PDM, PCM, Write buffer.



```
static void MP45DT02_Init(void) {
  pdm_filter.LP_HZ = 8000;
  pdm_filter.HP_HZ = 0;
  pdm_filter.Fs = 16000;
  pdm_filter.Out_MicChannels = 1;
  pdm_filter.In_MicChannels = 1;

  PDM_Filter_Init((PDMFilter_InitStruct*)&pdm_filter);
}
```

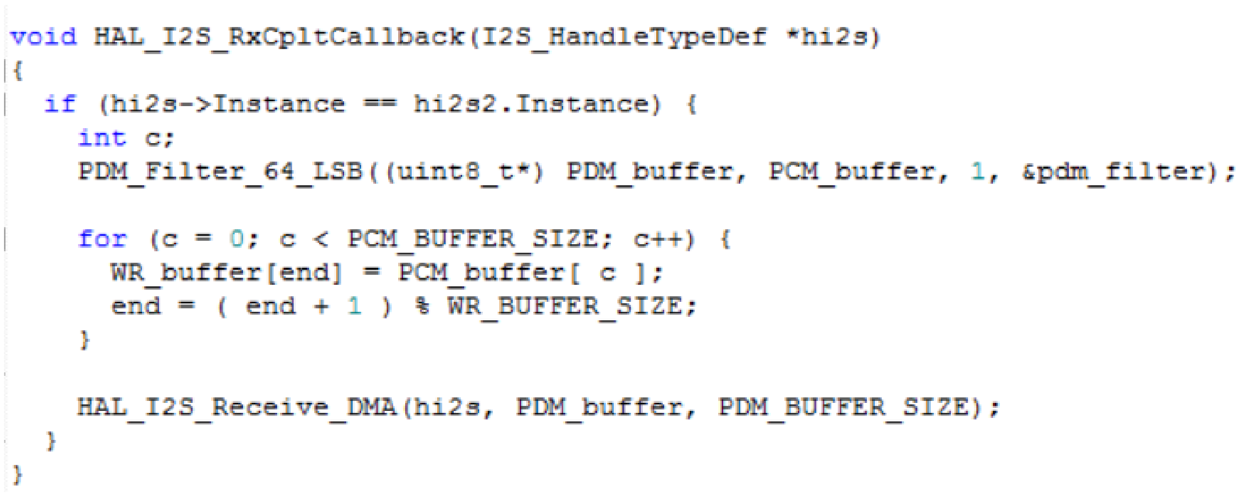
Initial function for MD45T02, setting value to pdm_filter

LP_HZ is low pass filter frequency

HP_HZ is high pass filter frequency

Fs is Frequency of sample

Out and in Microphone channel



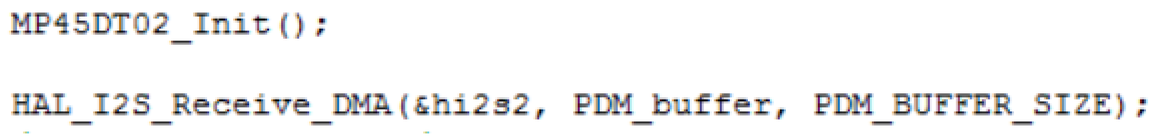
```
void HAL_I2S_RxCpltCallback(I2S_HandleTypeDef *hi2s)
{
  if (hi2s->Instance == hi2s2.Instance) {
    int c;
    PDM_Filter_64_LSB((uint8_t*) PDM_buffer, PCM_buffer, 1, &pdm_filter);

    for (c = 0; c < PCM_BUFFER_SIZE; c++) {
      WR_buffer[end] = PCM_buffer[ c ];
      end = ( end + 1 ) % WR_BUFFER_SIZE;
    }

    HAL_I2S_Receive_DMA(hi2s, PDM_buffer, PDM_BUFFER_SIZE);
  }
}
```

Writing handler callback for I2S, if only hi2s is hi2s2 start calculate by calling PDM_Filter_64_LSB with input buffer(PDM) ,output buffer(PCM) and PDMFilter_InitStruct. After calculate finish put value from PCM to Write buffer(Circular FIFO) then call HAL_I2S_Receive_DMA again to retrieve next data.

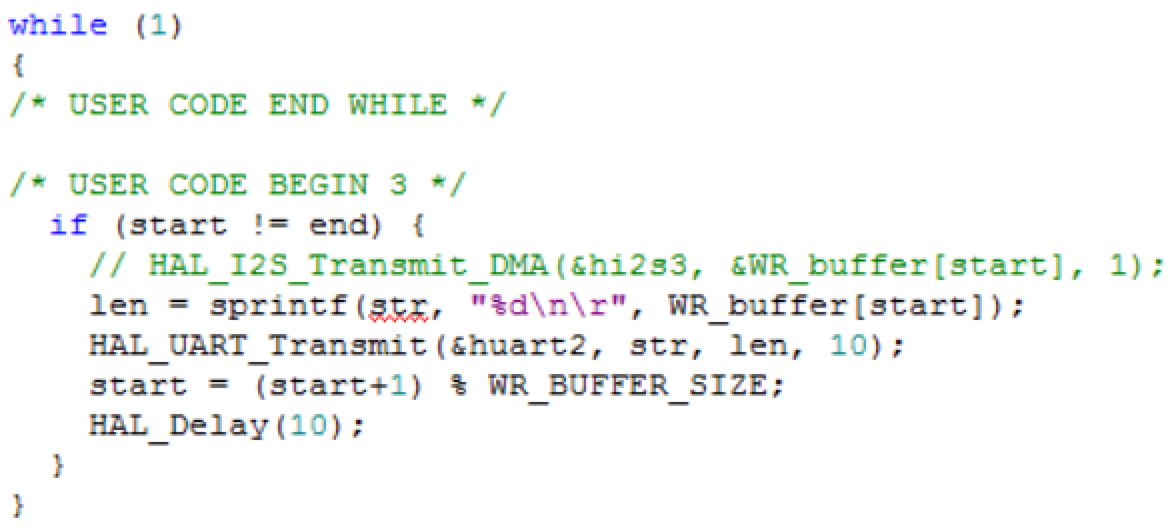
In function main()



```
MP45DT02_Init();

HAL_I2S_Receive_DMA(&hi2s2, PDM_buffer, PDM_BUFFER_SIZE);
```

Call MP4DT02_Init() and start sending receive signal through HAL_I2S_Receive_DMA with hi2s2 and internal buffer with size.



```
while (1)
{
/* USER CODE END WHILE */

/* USER CODE BEGIN 3 */
  if (start != end) {
    // HAL_I2S_Transmit_DMA(&hi2s3, &WR_buffer[start], 1);
    len = sprintf(str, "%d\n\r", WR_buffer[start]);
    HAL_UART_Transmit(&huart2, str, len, 10);
    start = (start+1) % WR_BUFFER_SIZE;
    HAL_Delay(10);
  }
}
```

In while start sending data from WR_buffer through UART and put Delay as 10