STM32L4 technical training

Digital Filter for Sigma Delta Modulator (DFSDM)

Hands-on session

life.augmented

DFSDM Lab

connection of single microphone and collect PCM data

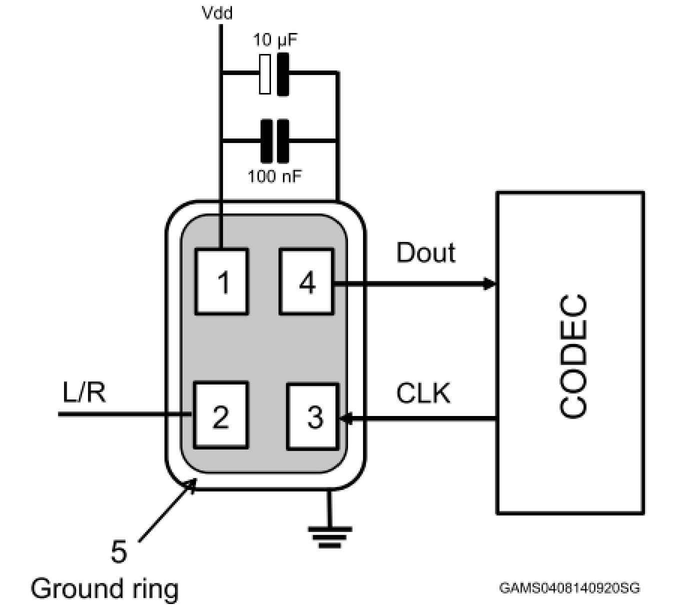
life.augmented

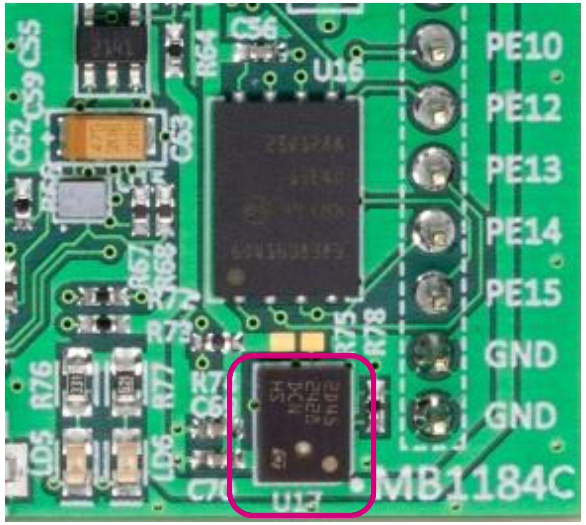
MEMS microphone connection to DFSDM

- Objective
  - Learn how to connect MEMS microphone to DFSDM in STM32CubeMX
  - Learn how to configure DFSDM to convert PDM to PCM signal
  - How to Generate Code in STM32CubeMX and use HAL functions

- Goal
  - Configure DFSDM peripheral in order to collect PDM data and convert them into PCM format.

Figure 4. MP34DT01-M electrical connections (Top view)



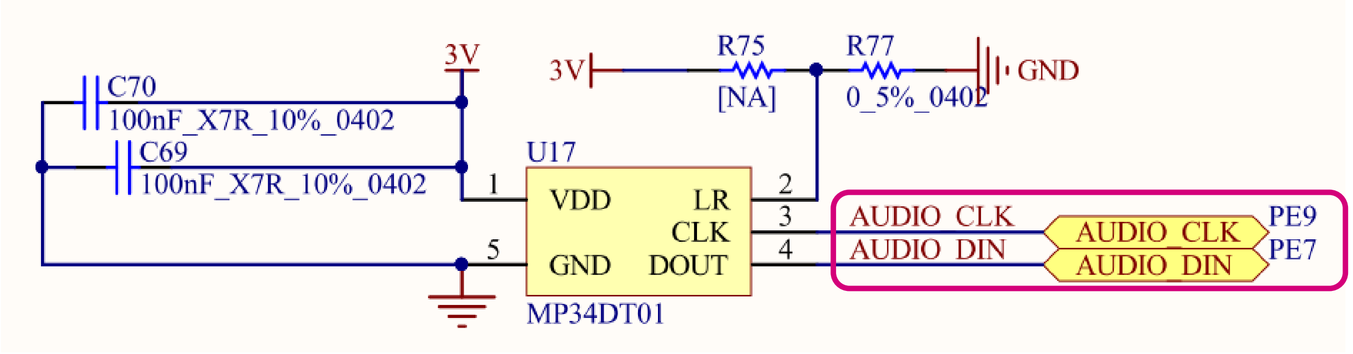


life.augmented

MP34DT01 microphone connection

STM32L476RG-Discovery

- STM32F476RG-Discovery is equipped with one MP34DT01 microphone connected to pins:
  - PE7 (DFSDM channel2)
  - PE9 (DFSDM clock out)



- It requires an external clock in range 1MHz to 3.25MHz delivered by DFSDM for proper operation.

Table 3. Acoustic and electrical characteristics

Symbol	Parameter	Test condition	Min.	Typ. (1)	Max.	Unit
Clock	Input clock frequency (3)		1	2.4	3.25	MHz
Ton	Turn-on time (4)	Guaranteed by design			10	ms
Top	Operating temperature range		-40		+85	°C
V_{IOL}	Low level logic input/output voltage	I_{out} = 1 mA	-0.3		0.35xVdd	V
V_{IOH}	High level logic input/output voltage	I_{out} = 1 mA	0.65xVdd		Vdd+0.3	V

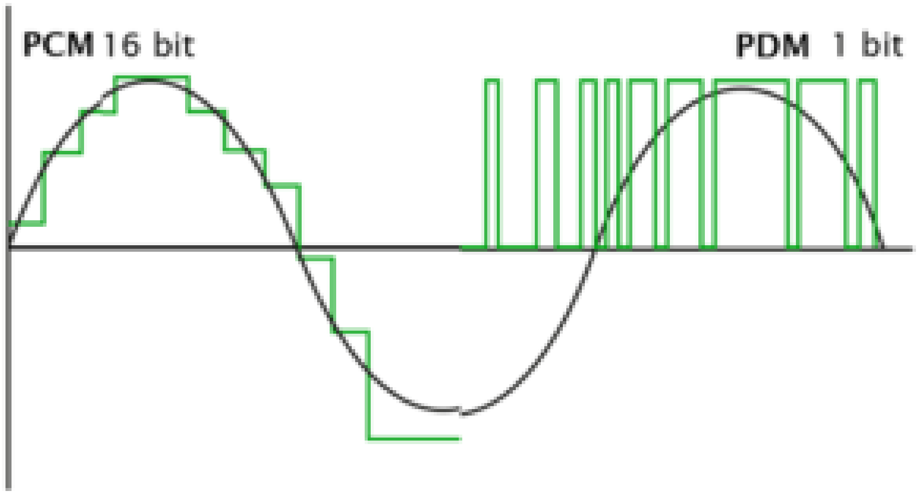
1. Typical specifications are not guaranteed.
2. Input clock in static mode.
3. Duty cycle: min = 40% max = 60%.
4. Time from the first clock edge to valid output data.

life.augmented

Selection of the system parameters

- The digital audio output from the microphone is coded in PDM (Pulse Density Modulation) and is connected to PE7. When CLK = 0 (PE9) , the audio PDM signal is sent on PE7.

- Let's select the following parameters of our system:
  - DFSDM clock: **80MHz** (system clock)
  - Microphone input clock: **2MHz**
  - Output sampling frequency: **8kHz**
  - Resolution of the output signal: **24bits** (signed)

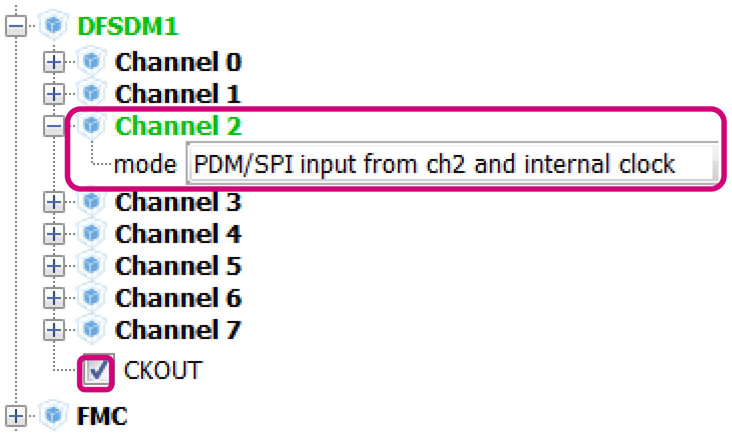


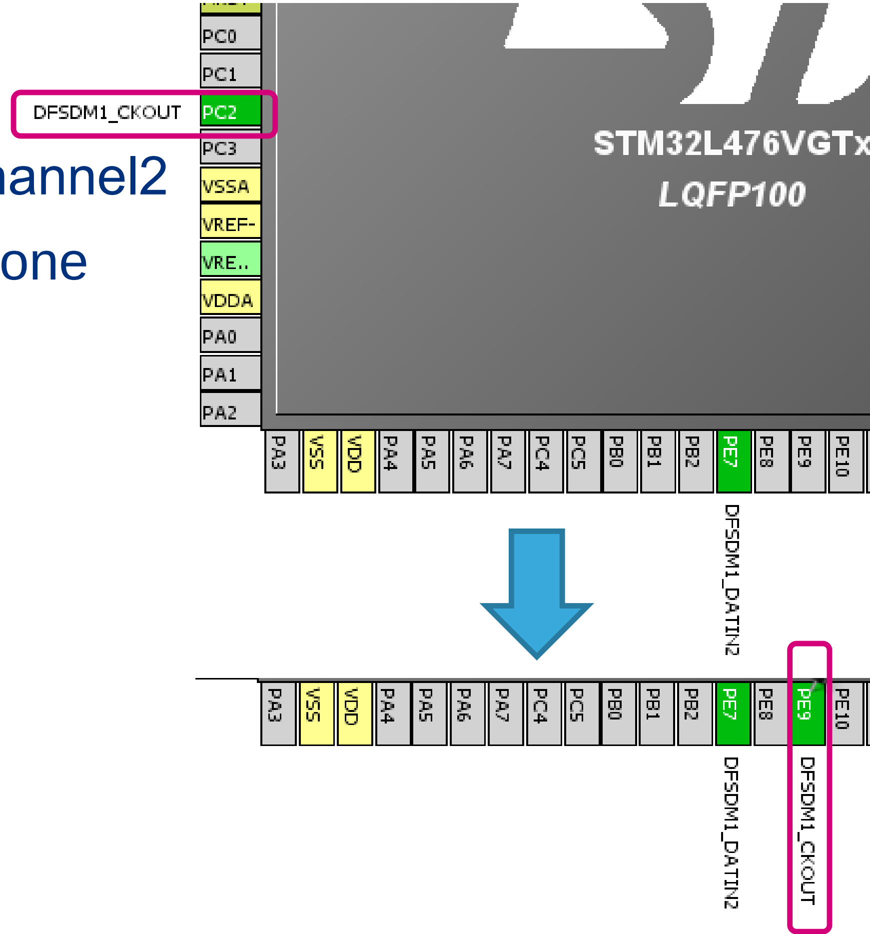
life.augmented

STM32CubeMX

Selecting DFSDM

- Create project in STM32CubeMX
  - Menu > File > New Project
  - Select STM32L4 -> STM32L4x6 -> LQFP100 package -> STM32L476VGTx
- Select DFSDM1:
  - Select "PDM/SPI Input from ch2 and internal clock" option for Channel2
  - Select CKOUT to enable clock connection from MCU to microphone
  - Change default DFSDM1_CKOUT pin (**PC2**) to alternative **PE9**



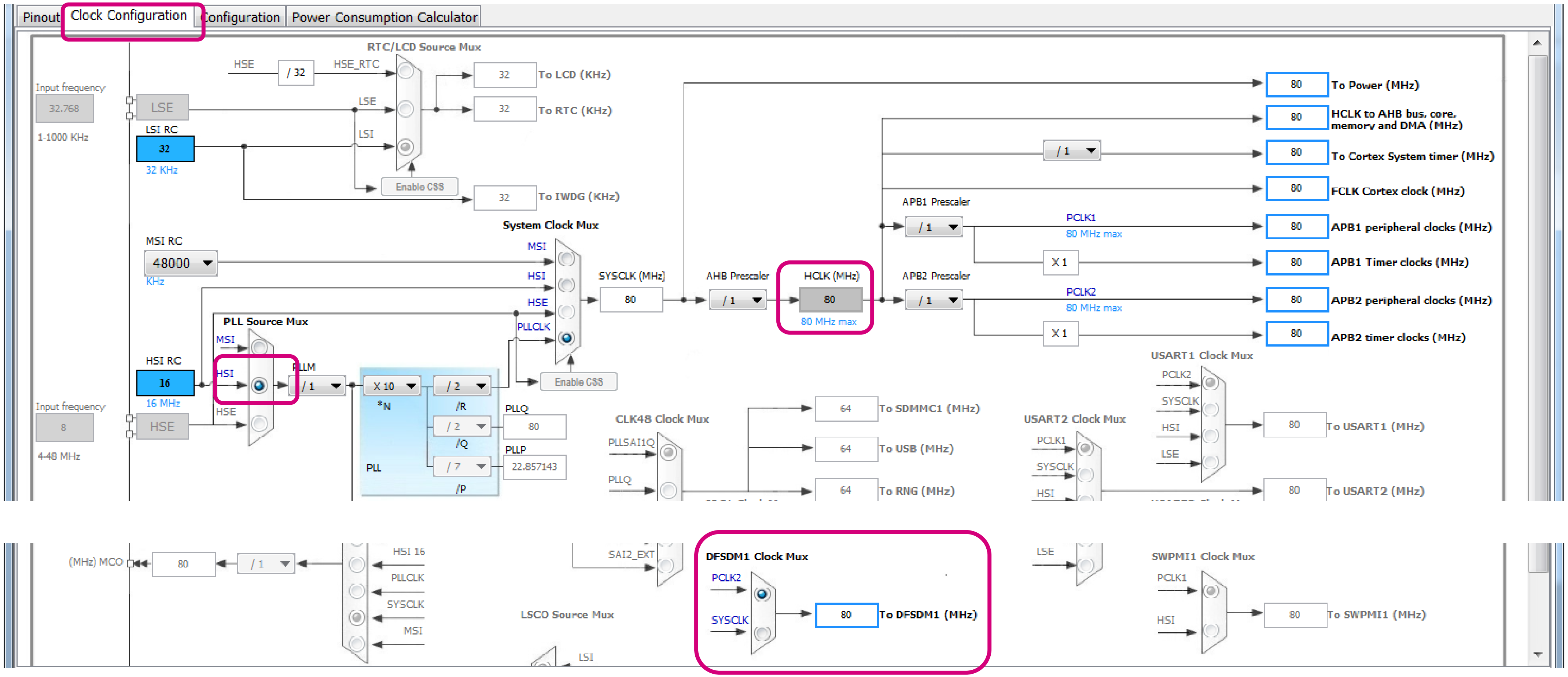


life.augmented

STM32CubeMX

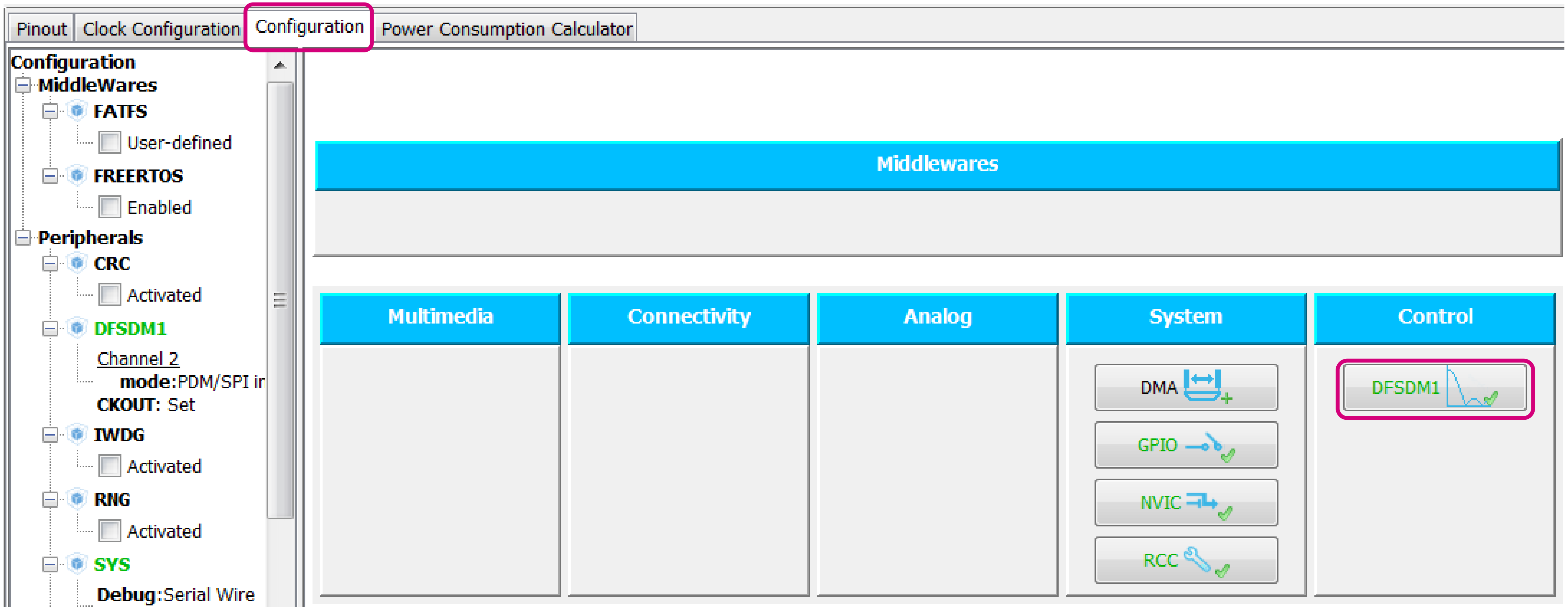
clock configuration

- Go to **Clock Configuration** tab and configure system clock (HCLK) and DFSDM to 80MHz using HSI 16MHz oscillator and PLL



life.augmented

- Go to **Configuration** tab and select DFSDM peripheral

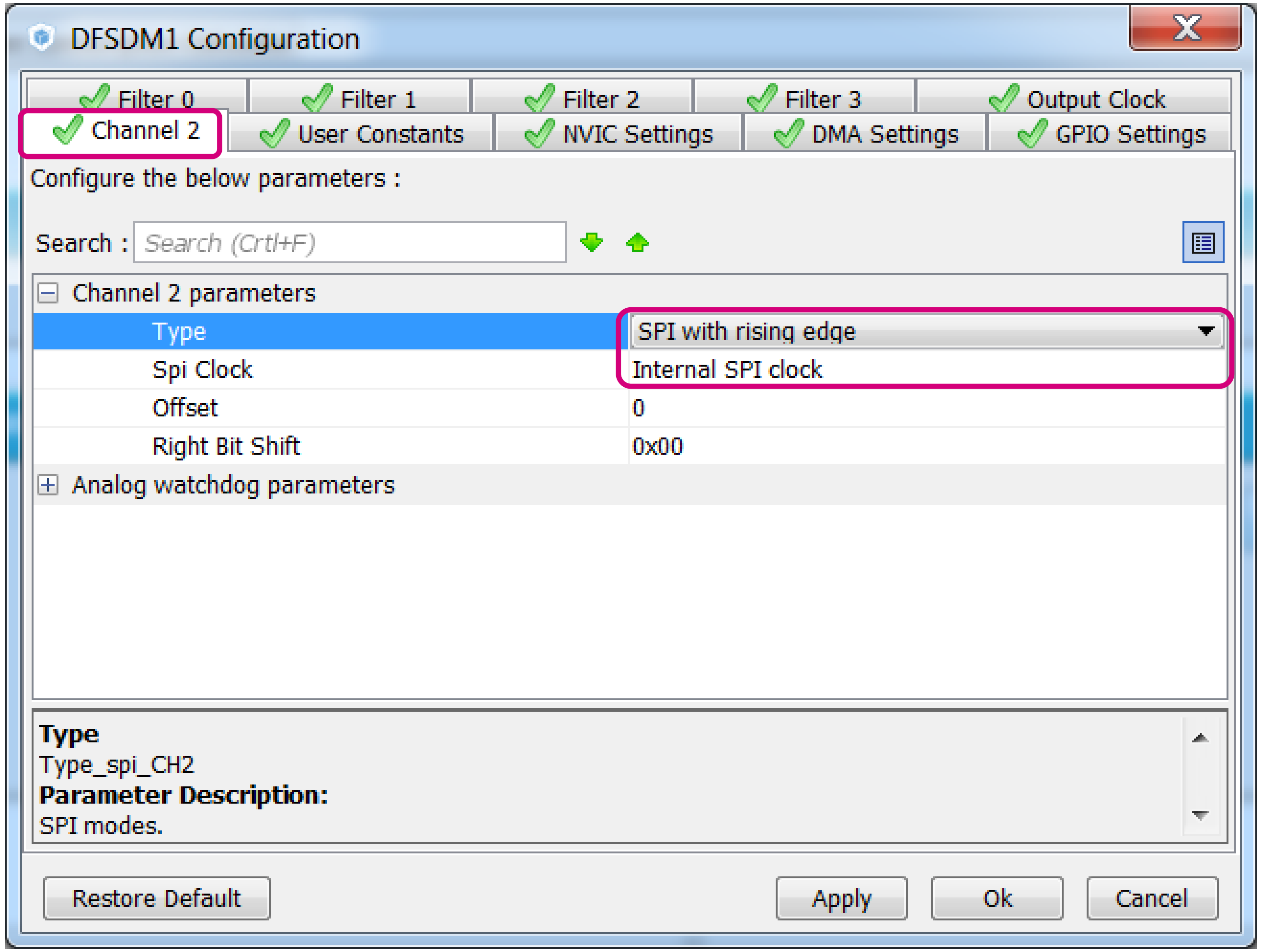


life.augmented

STM32CubeMX

configuration of the DFSDM input channel

- Select **Channel2** tab (as the microphone is connected to this input channel)
  - Set **SPI with rising edge** in Type field
  - Select **Internal SPI clock**
  - Do not configure offset and bit shift, neither analog watchdog
- Press **Apply** to confirm the configuration

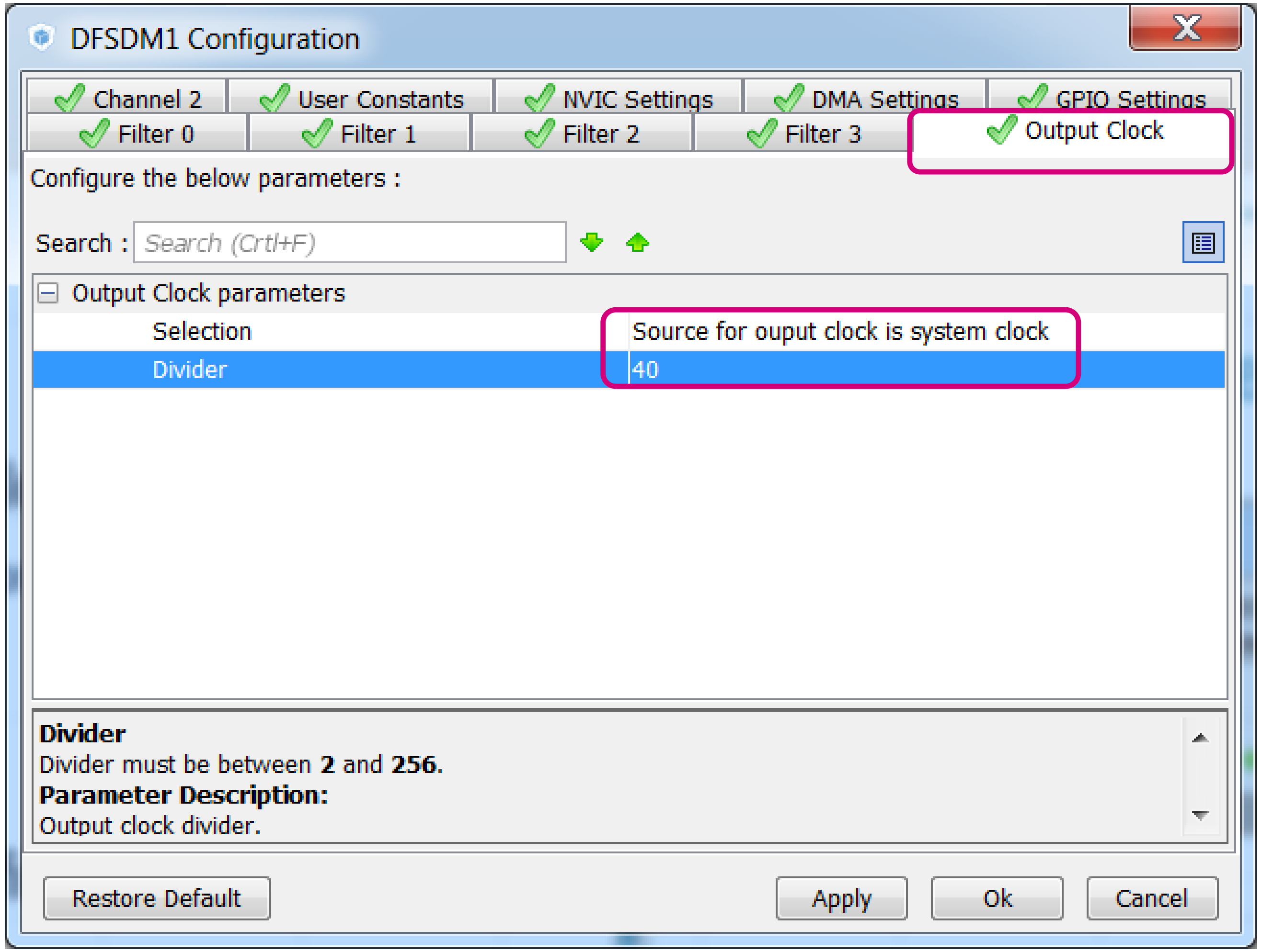


life.augmented

STM32CubeMX

configuration of the DFSDM output clock

- Select **Output Clock** tab
  - Select: **Source for output clock is system clock** (80MHz)
  - To have 2MHz clock signal for the microphone we need to set a divider to **40** (as 80MHz/2MHz=40)
- Press **Apply** to confirm the configuration

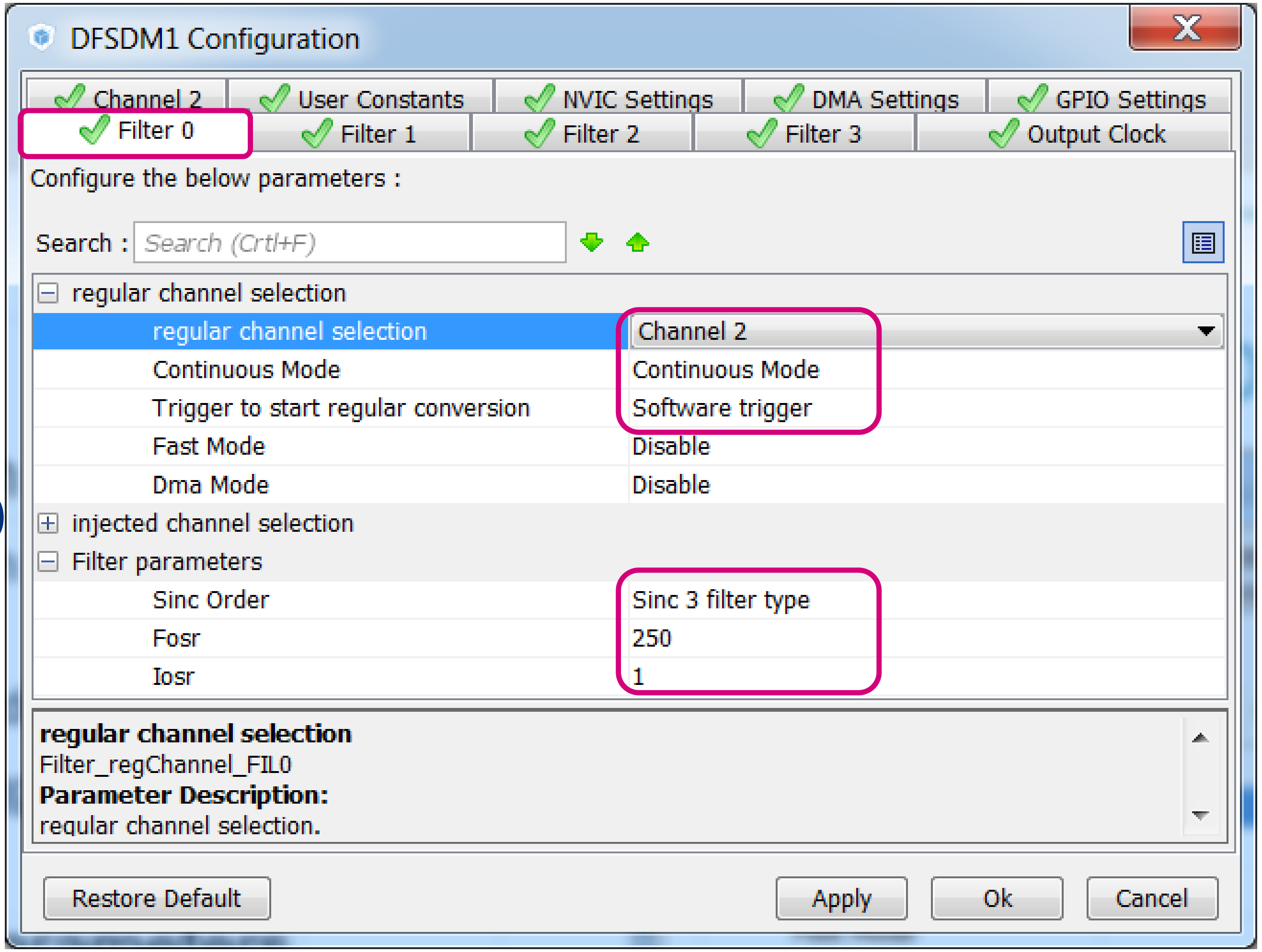


life.augmented

STM32CubeMX

configuration of the DFSDM filter – part 1/2

- Select **Filter0** (can be different one)
  - Select **Channel 2** in regular channel selection field
  - Set **continuous mode**
  - Select **software trigger**
- Configure Filter parameters
  - **Sinc order** set to Sinc 3 filter type
  - Oversampling (**Fosr**) set to **250** (to have output sampling rate 8kHz from input 2MHz)
  - **Iosr** set to **1** (we will not use it)
- Press **Apply** to confirm settings

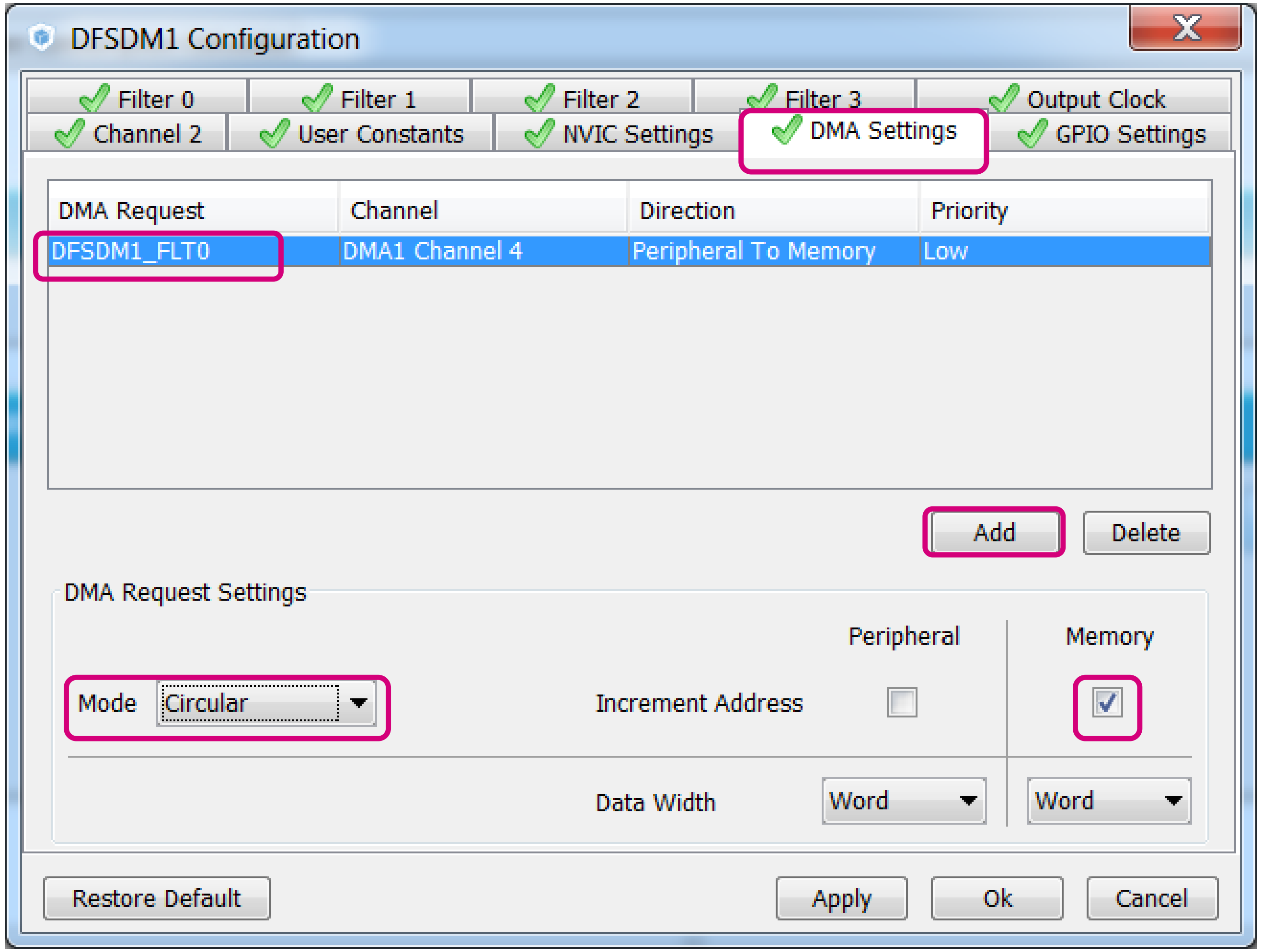


life.augmented

STM32CubeMX

configuration of the DMA for DFSDM

- Select **DMA Settings** tab
- Click **Add** button
  - Select **DFSDM1_FLT0** from DMA request
  - Set **incrementation on Memory side**
  - Select **Word** Data Width for both sides
  - Select **Circular** mode
- Press **Apply** to confirm configuration



life.augmented

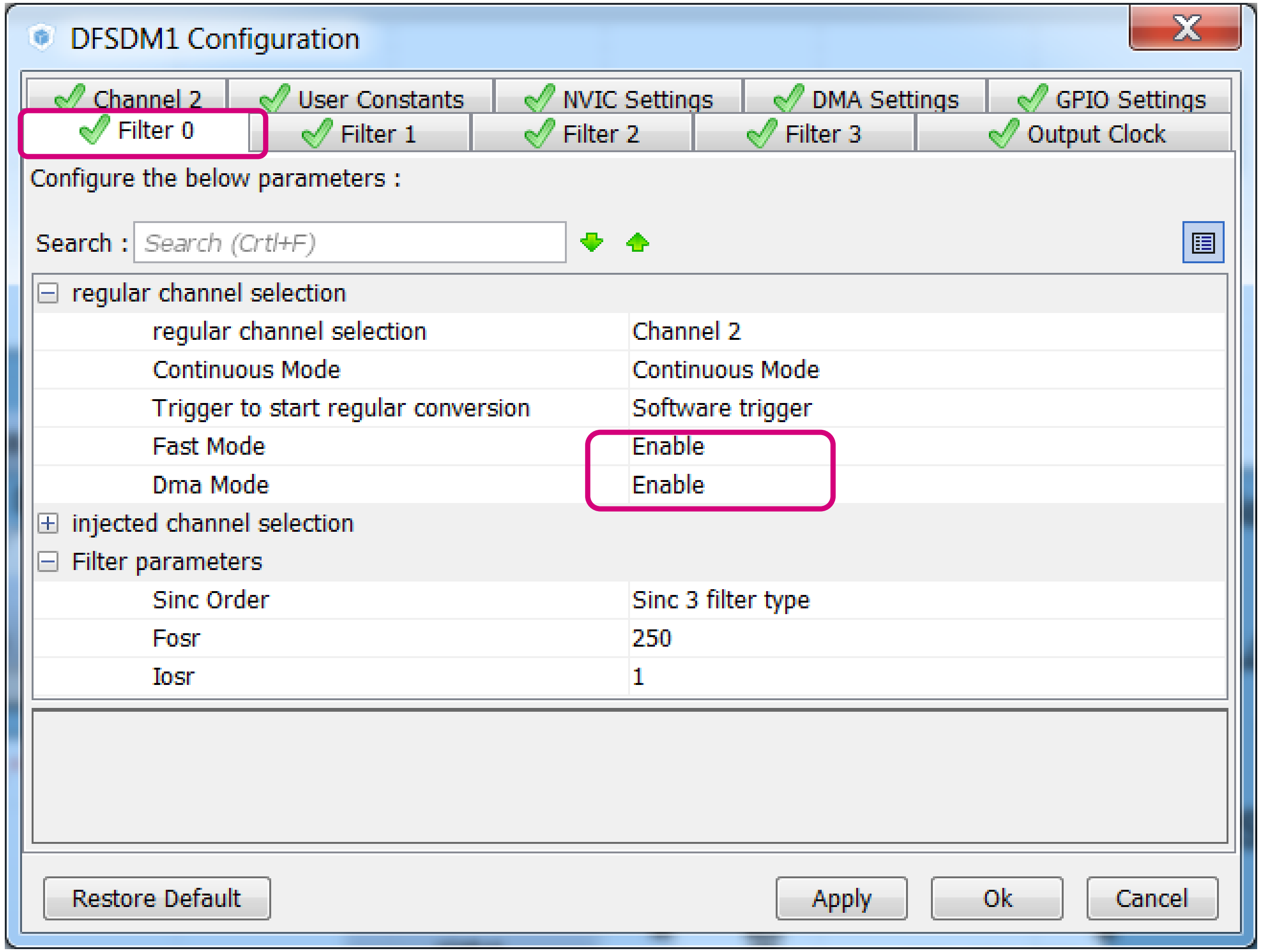
STM32CubeMX

configuration of the DFSDM filter – part 2/2

- Once DMA channel is configured we can come back to **Filter0** settings and perform two more steps:
  - Select Fast Mode: **Enable**
  - Select Dma Mode: **Enable**

- Press **OK** to confirm the changes

- DFSM peripheral is now fully configured

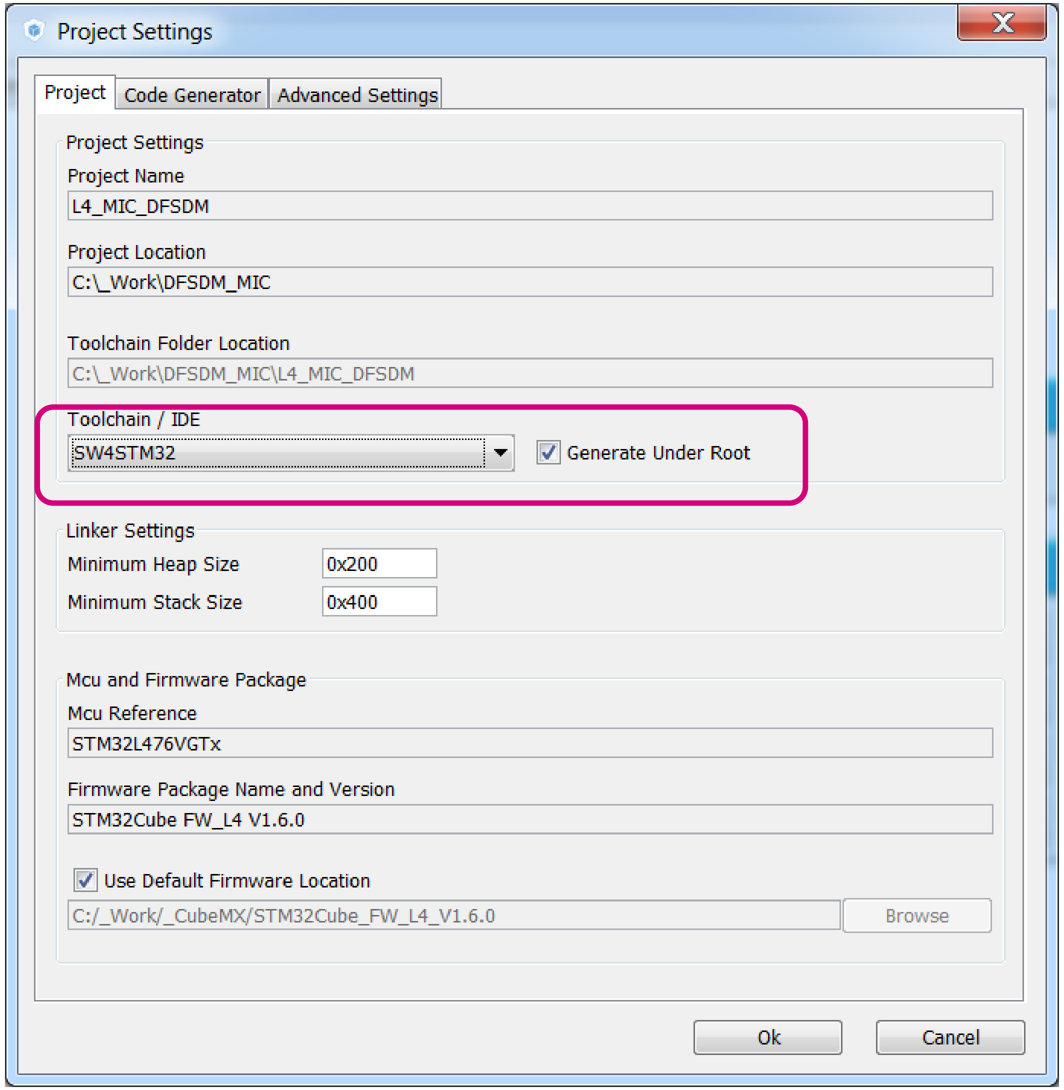


life.augmented

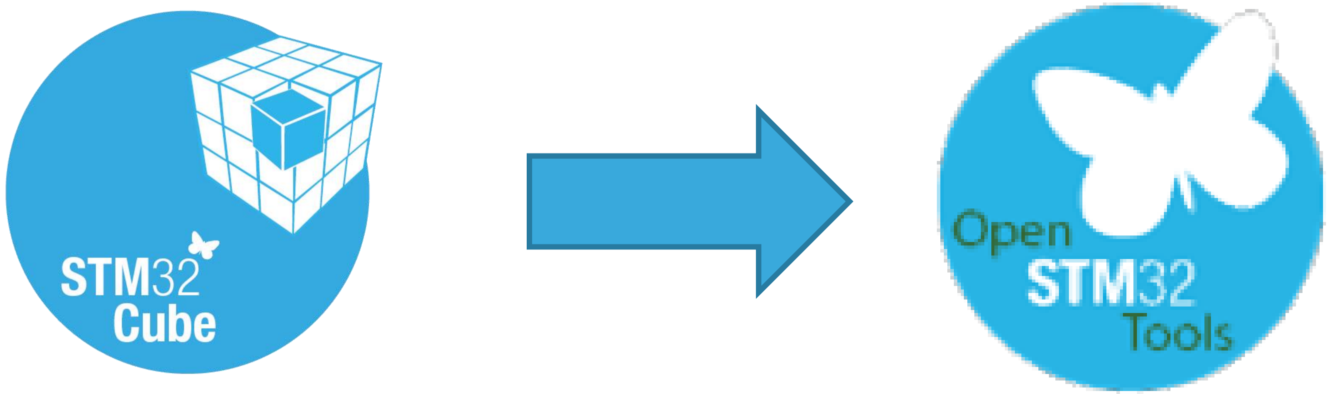
STM32CubeMX

Project generation

- Now we set the project details for generation
  - Menu > Project > Project Settings
  - Set the project name
  - Project location
  - Type of toolchain
- Now we can Generate Code
  - Menu > Project > Generate Code



life.augmented



- After successful code generation by STM32CubeMX this is the right time to import it into SW4STM32 toolchain for further processing

Modifying the code

data declaration and DFSDM start - main.c file

Tasks:

1. Declare the size of the buffer: 1024 words
2. Declare data buffer for DFSDM to store PCM data (32bit signed values)
3. Start DFSDM peripheral in DMA mode for regular conversion for configured channel and its assigned filter to store AUDIO_BUF number of PCM samples into RecBuff buffer

```
/* USER CODE BEGIN PV */
/* Private variables ---------------------------------------------------------*/
#define AUDIO_BUF      1024
int32_t RecBuff[AUDIO_BUF];
/* USER CODE END PV */
```

```
/* USER CODE BEGIN 2 */
HAL_DFSDM_FilterRegularStart_DMA(&hdfsdm1_filter0, (int32_t *)RecBuff, AUDIO_BUF);
/* USER CODE END 2 */
```

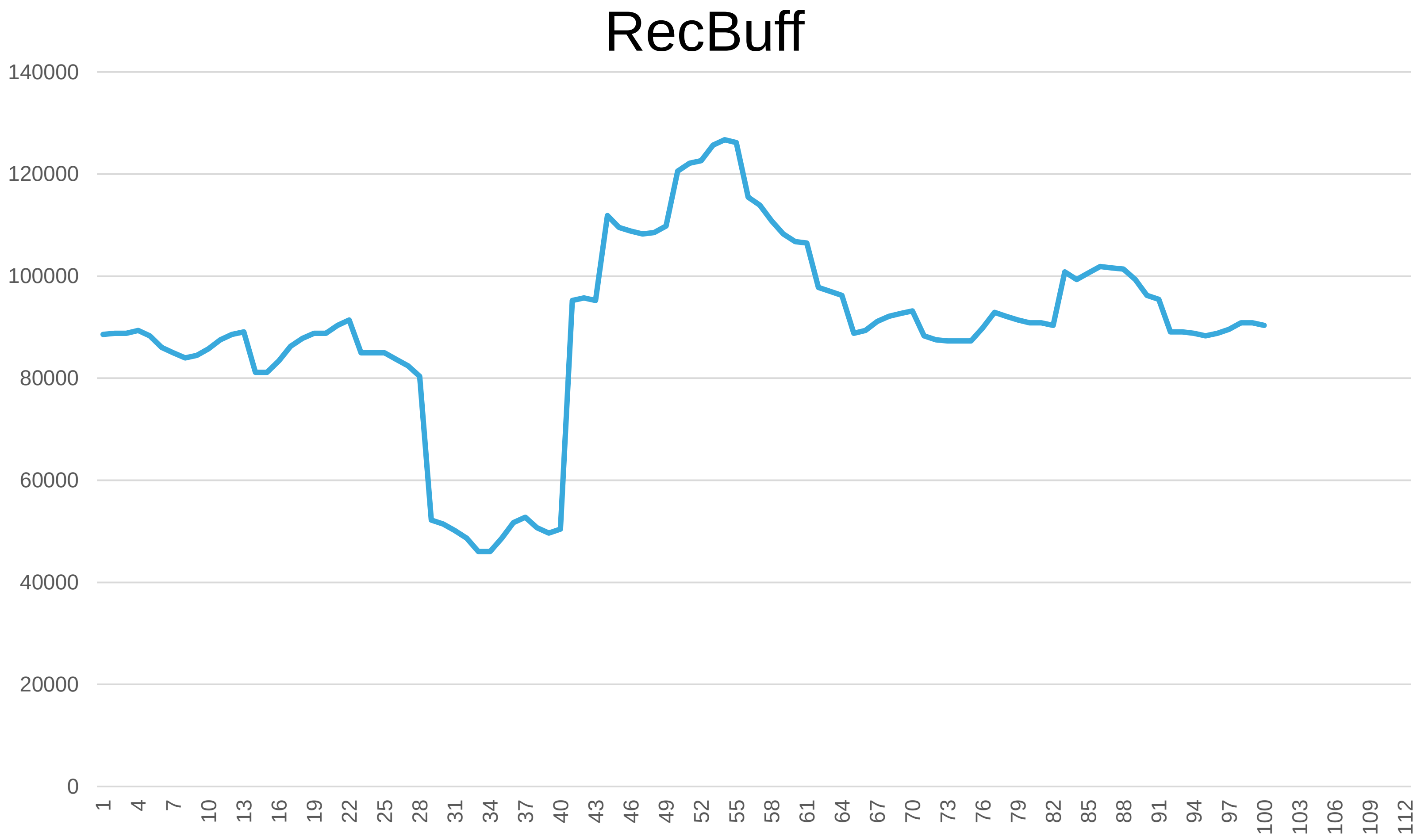
life.augmented

- Now we can compile the code and run the debug session
- As a result we should monitor RecBuff[] table content.
- We can copy the content of this table to a selected PC application for further analysis

Example of the final result

- An example of measurement of input acoustic signal which was a sine wave f=600Hz

How can we improve our signal?



life.augmented

- DMA is taking data from DFSDM_FLTxRDATAR register (there is separate register for each filter)
- The 24bits PCM data are located on bits 8-32.
- We should shift right the buffer data by 8 bits to get valid acoustic samples
- To do it in the proper timing we need to synchronize with DMA status flags (half transfer complete and transfer complete)

DFSDM Lab extension

further processing of the PCM data (DMA transfers management)

life.augmented

- We will continue with our previous lab on DFSDM
- Now we will configure and use interrupt callbacks raised by DMA transferring data from DFSDM
- The goal of this part is to perform some additional postprocessing of the PCM data to have valuable acoustic samples.

Modifying the code

postprocessing of the PCM data - main.c file 1/2

Tasks:

1. Declare buffer for post processed PCM data (same size like RecBuff[])
2. Implement callback functions for DMA Half transfer and DMA transfer complete interrupts

```
/* USER CODE BEGIN PV */
/* Private variables ---------------------------------------------------------*/
int i=0;
int32_t     PlayBuff[AUDIO_BUF];
uint32_t    DmaRecHalfBuffCplt  = 0;
uint32_t    DmaRecBuffCplt      = 0;
```

```
 /* USER CODE BEGIN 4 */
 void HAL_DFSDM_FilterRegConvHalfCpltCallback(DFSDM_Filter_HandleTypeDef *hdfsdm_filter)
{
  DmaRecHalfBuffCplt = 1;
}

void HAL_DFSDM_FilterRegConvCpltCallback(DFSDM_Filter_HandleTypeDef *hdfsdm_filter)
{
  DmaRecBuffCplt = 1;
}
 /* USER CODE END 4 */
```

Modifying the code

postprocessing of the PCM data - main.c file 2/2

Tasks:

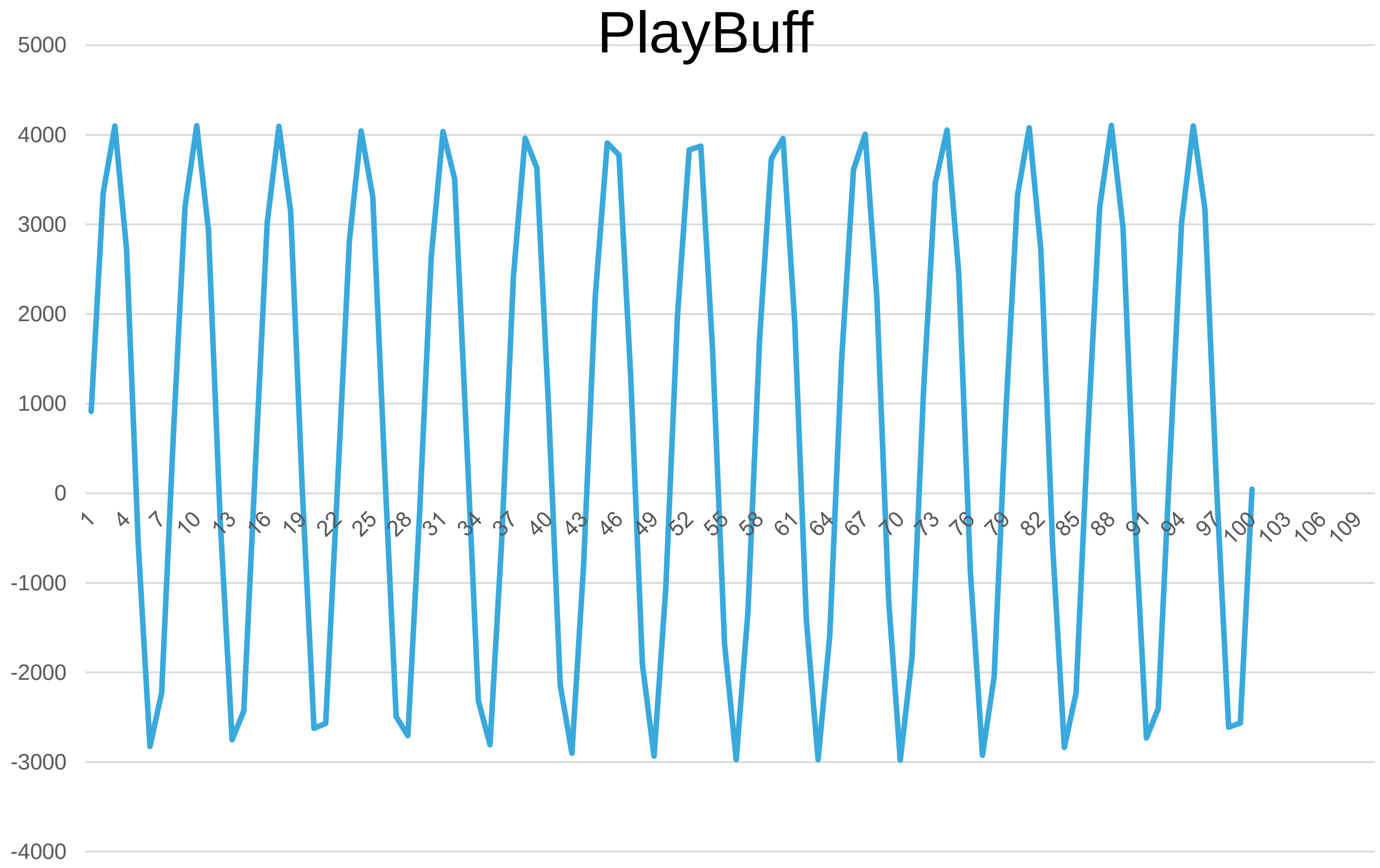
3. Perform postprocessing of the PCM data on completed part of the buffer

```
 /* USER CODE BEGIN 3 */
    if(DmaRecHalfBuffCplt == 1)    //processing of the first half of the buffer
    {
/* Store values on Play buff */
      for(i = 0; i < AUDIO_BUF/2; i++)
        PlayBuff[i]      = RecBuff[i] >> 8;     //example of PCM data postprocessing
      DmaRecHalfBuffCplt  = 0;
    }

    if(DmaRecBuffCplt == 1)        //processing of the second half of the buffer
    {
      /* Store values on Play buff */
      for(i = AUDIO_BUF/2; i < AUDIO_BUF; i++)
       PlayBuff[i]      = RecBuff[i] >> 8;      //example of PCM data postprocessing
      DmaRecBuffCplt  = 0;
    }
```

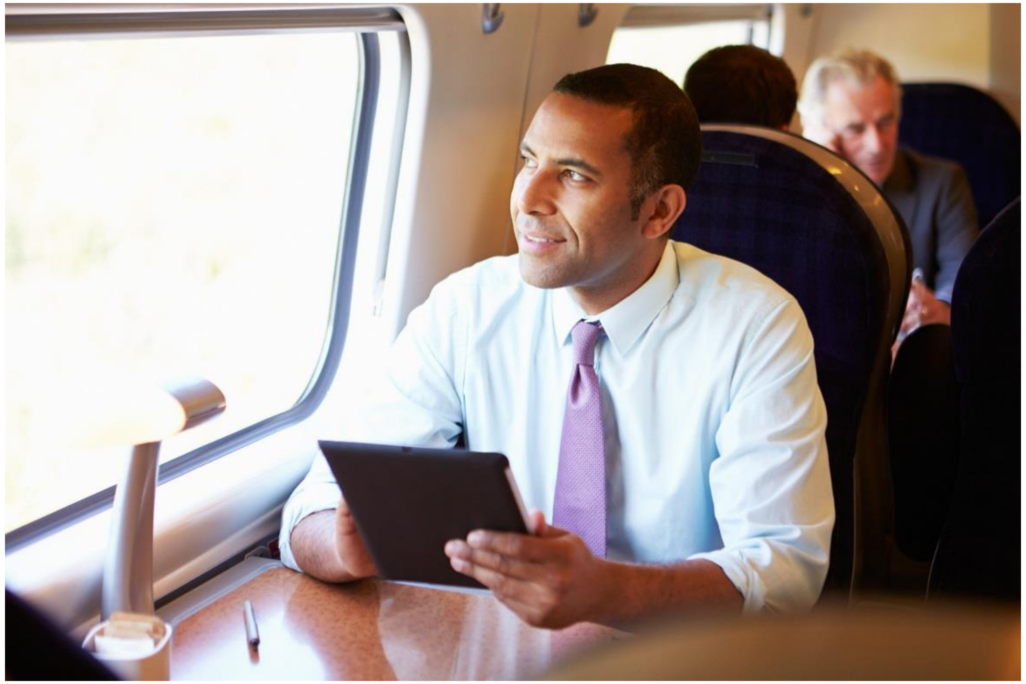
Example of the final result

- An example of measurement of input acoustic signal which was a sine wave f=600Hz



Further reading

- **AN4427** – Gasket design for optimal acoustic performance in MEMS microphones
- **AN4426** – Tutorial for MEMS microphones
- **MP34DT01-M** datasheet
- www.st.com/mems



life.augmented

Enjoy!

www.st.com/mcu

life.augmented