Several "setup" items related to the AUDIO functions are as follows.

1. Near the top of the startup file ***startup_stm32l476xx.s***, increase the value of **Heap_Size to 0x2400**. Some driver functions use *malloc()* to allocate memory from this space.

2. Put the following in the file with your interrupt handlers (these are from the ST CubeMx demo programs – which provide some good examples that you might want to examine.)

```
/******************************************************************************/
/*            Cortex-M4 Processor Interruption and Exception Handlers        */
/******************************************************************************/
extern SAI_HandleTypeDef         BSP_AUDIO_hSai;
extern DFSDM_Filter_HandleTypeDef BSP_AUDIO_hDfsdmLeftFilter;

/**
  * @brief  This function handles SAI DMA interrupt request.
  * @param  None
  * @retval None
  */
void AUDIO_SAIx_DMAx_IRQHandler(void)
{
  HAL_DMA_IRQHandler(BSP_AUDIO_hSai.hdmatx);
}

/**
  * @brief  This function handles DFSDM Left DMA  interrupt request.
  * @param  None
  * @retval None
  */
void AUDIO_DFSDM_DMAx_LEFT_IRQHandler(void)
{
  HAL_DMA_IRQHandler(BSP_AUDIO_hDfsdmLeftFilter.hdmaReg);
}
```

3. In your interrupt handler header (.h) file, add the following (DMA IRQ handlers renamed by HAL functions):

```
void AUDIO_DFSDM_DMAx_LEFT_IRQHandler(void);    //DMA1-CH4
void AUDIO_SAIx_DMAx_IRQHandler(void);          //DMA2-CH1
```

4. AUDIO_IN and AUDIO_OUT "register callback functions" are functions you define. If defined, these are called by the HAL-level drivers when the memory buffer being written/read is half or entirely full (one callback for each), or when there is an error. HAL drivers use callback functions whenever there are HAL-level interrupts, so that you can provide code to be executed when those occur. For example, I use the "*TransferComplete*" callback function during recording to write a full buffer to flash, or during playback to fill a buffer with a new block of data from flash. You must use tell the HAL driver about these via functions

   BSP_AUDIO_IN_RegisterCallBacks(1,2,3);

   where 1,2,3 are the names of the functions you have written for (1)error, (2)half transfer complete, (3) transfer complete. Specify ***NULL*** as the name for any callback function you do not wish to use. In my test, I executed the following, and wrote the two functions in my main.c file. Likewise for ***BSP_AUDIO_OUT_RegisterCallBacks()***

   BSP_AUDIO_IN_RegisterCallBacks(AudioRecordErrorCallback, NULL, AudioRecordTransferComplete);