**Lesson 48**

**Part 1**

**USB DEVICE AUDIO**

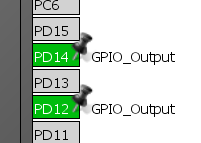
[*Previous lesson*](http://narodstream.ru/stm-urok-47-podklyuchaem-giroskop-lsm6ds3-chast-1/) [*Programming STM32 STM32*](http://narodstream.ru/programmirovanie-mk-stm32/) [*Next part*](http://narodstream.ru/stm-urok-48-usb-device-audio-chast-2/)

Today we will continue to study the new interface - the I2S bus. And we will work as well as in the last lesson on I2S - with the chip CS43L22 installed on the STM32F4-DISCOVERY board. Only in this lesson we will try to take the digital audio stream not from a WAV file, but from a PC via the USB bus, that is, we will try to create a sound card from our board. The sound card will, of course, not be complete, since it will only work on playback (only in DAC mode), but nevertheless, I think we will fix the work with the I2S bus thoroughly, and also we will study another class of USB - DEVICE Audio.

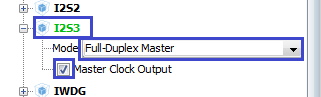
We will create our project from an old project, in which nothing extra is connected - TEST002, and we will name the new project USB\_DEVICE\_AUDIO.

Open our project in Cube MX and make the necessary settings.

First turn off unnecessary LEDs, leaving only red and green.

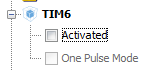


Turn on the bus itself I2S

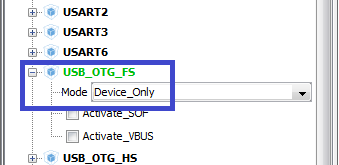


No legs are redefined. Let's leave it as it is. It is on these legs that the Audio DAC chip is connected.

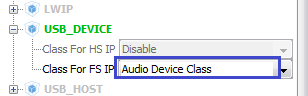
Disable the 6th timer



Turn on USB\_OTG\_FS in Device\_Only mode

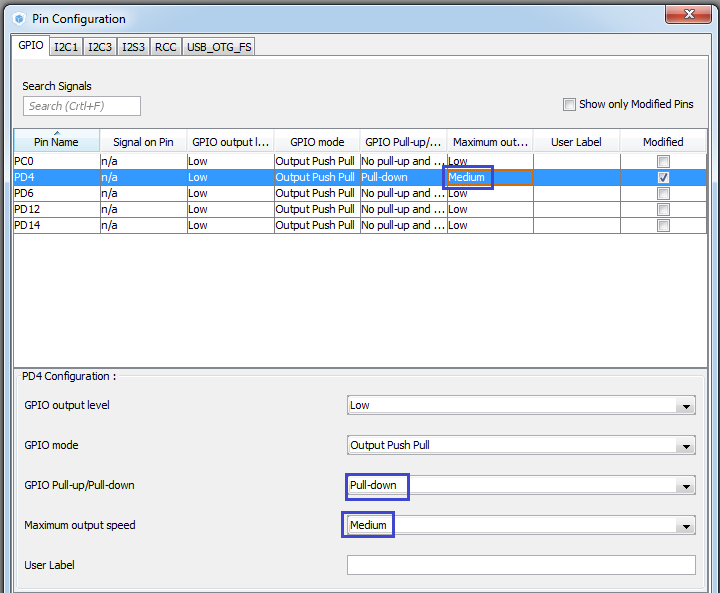


In USB\_DEVICE in the Class For FS IP section, select Audio Device Class

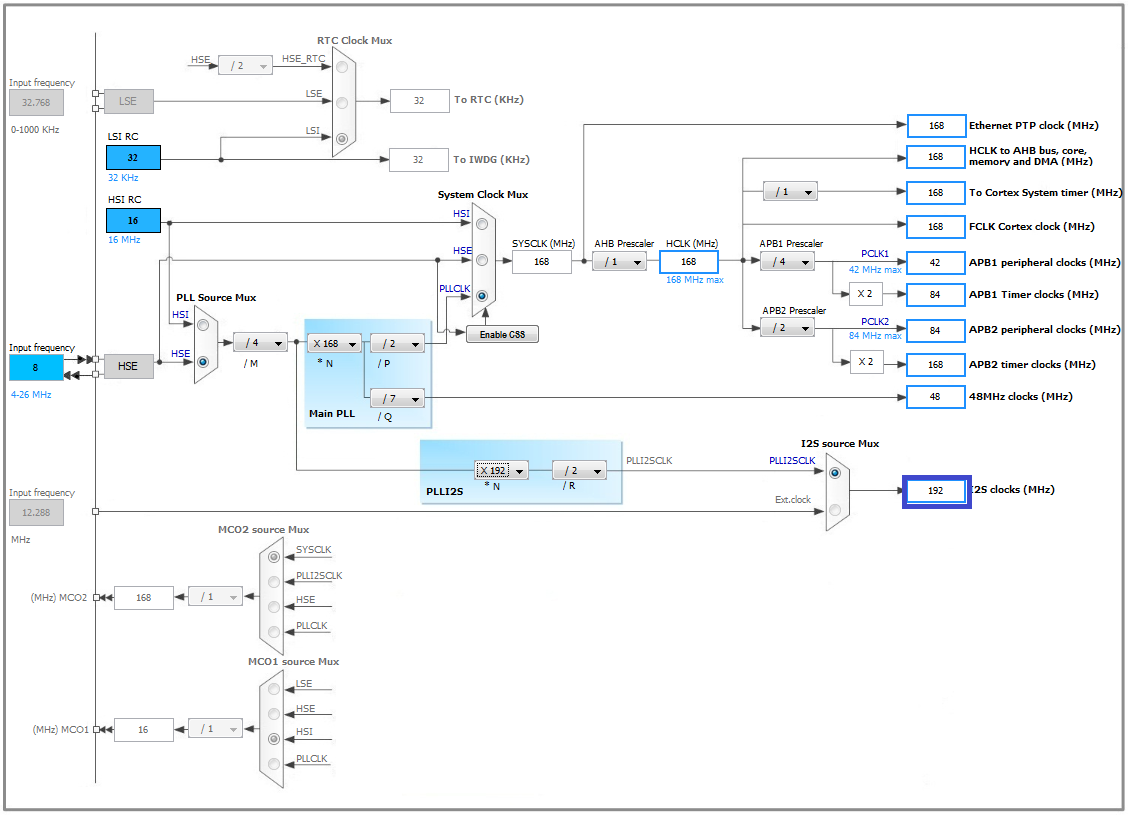


For the work of the RESET foot of the microcircuit, we will still need to turn on the output and adjust this port foot (PD4)

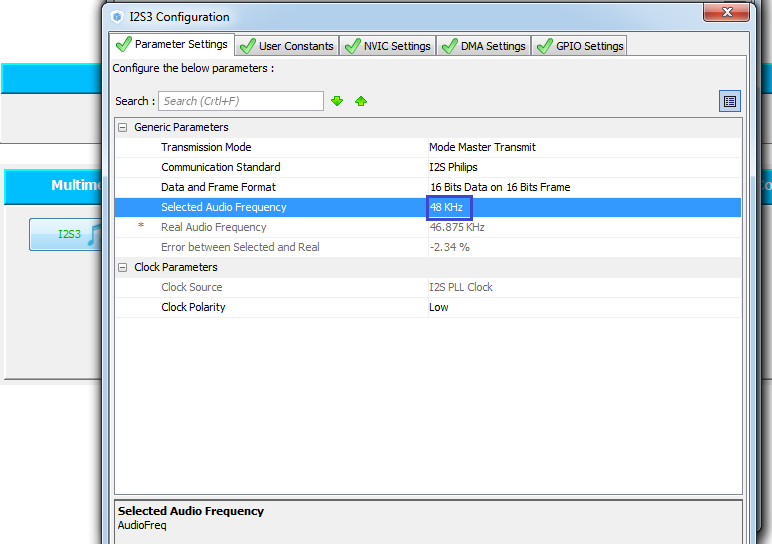




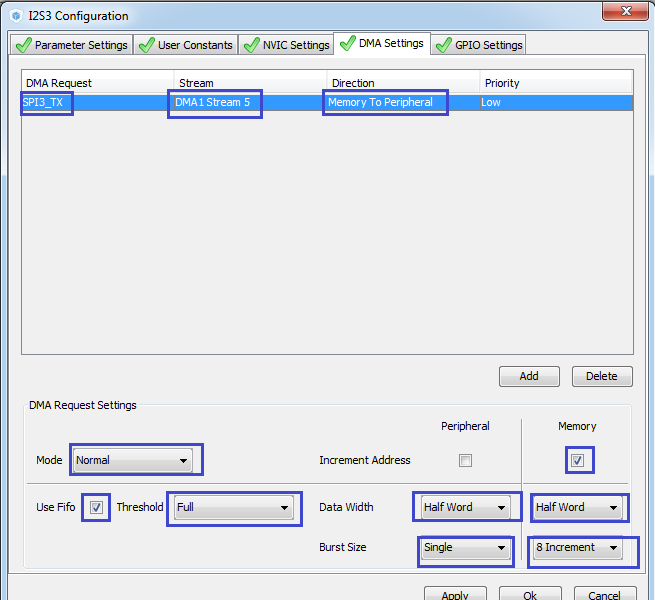
In the "Clock Configuration" section, turn everything on to the maximum



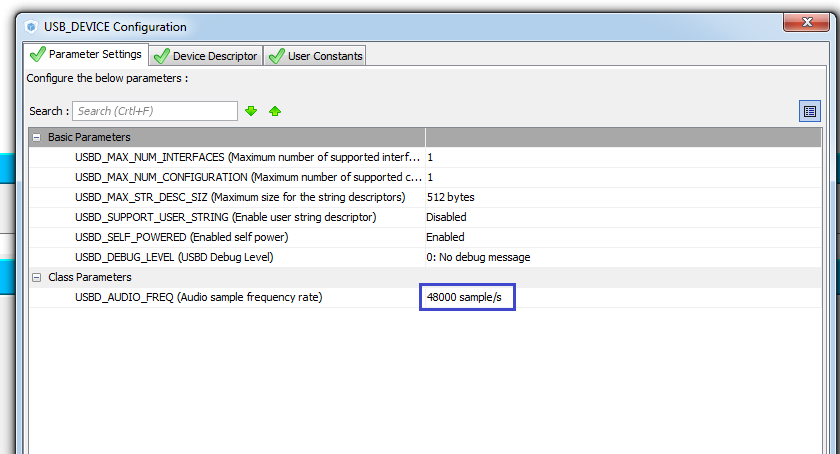
We will also make some adjustments to the I2S settings



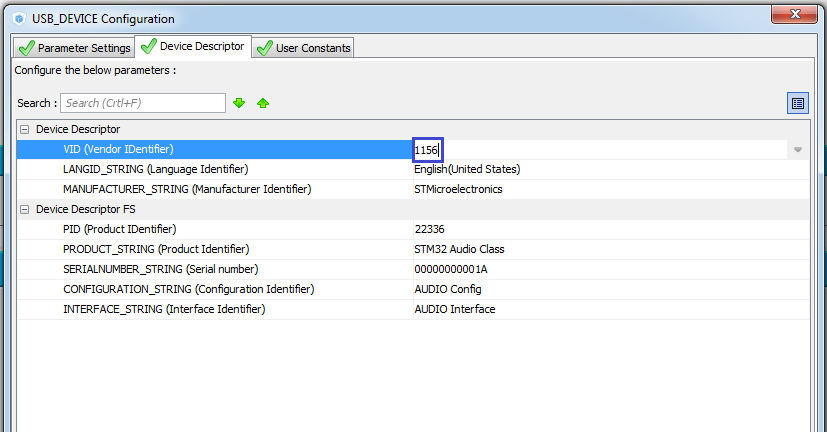
Also add and configure DMA on the I2S bus



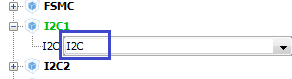
In USB Device we will increase the sampling rate



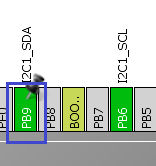
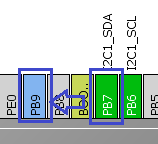
It is also desirable to change the device descriptor, otherwise if you have already used a virtual port, there may be problems



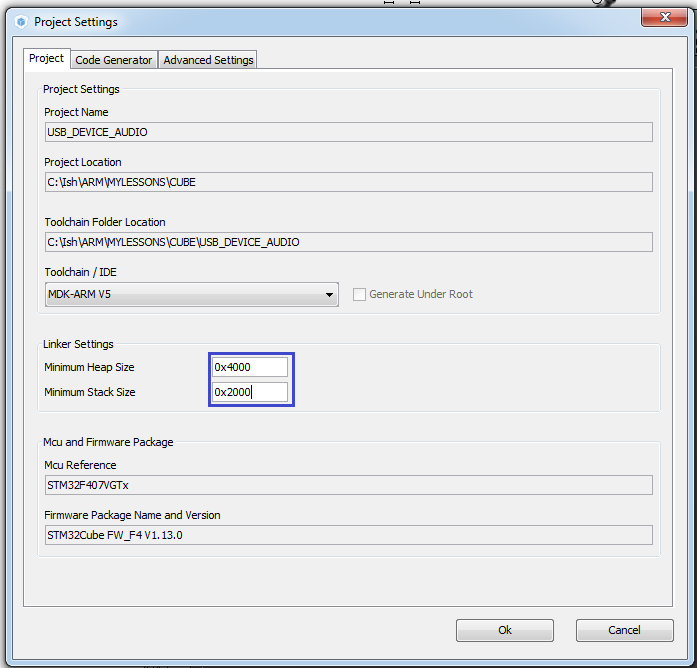
And since the chip management occurs in contrast to the main audio stream already on the I2C bus, it is necessary to include also this bus. Turn on the bus I2C1



The PB7 foot will need to be redefined on PB9. I think everyone already knows how to do this. First, reset it, and then redefine it



In Project -> Setting, set the following stack and heap settings



Generate the project for the Keil environment, configure the programmer for autocutting and compile the project.

Add to the project files designed to work with audio, audioplay.c and audioplay.h from the project I2S\_AUDIO.

Connect the file audioplay.c in the Application / User tree

Also add the connection of this library to the main module main.h

#include "stm32f4xx.h"

**#include "audioplay.h"**

Delete the following line here:

~~uint8\_t tim6\_counter;~~

Remove all the code from an infinite loop

  / \* USER CODE BEGIN 3 \* /

~~if (HAL\_GPIO\_ReadPin (GPIOA, GPIO\_PIN\_0) == GPIO\_PIN\_SET)~~

~~{~~

~~HAL\_TIM\_Base\_Start (& htim6);~~

~~HAL\_TIM\_Base\_Start\_IT (& htim6);~~

~~}~~

~~else~~

~~{~~

~~tim6\_counter = 0;~~

~~HAL\_TIM\_Base\_Stop (& htim6);~~

~~HAL\_TIM\_Base\_Stop\_IT (& htim6);~~

~~HAL\_GPIO\_WritePin (GPIOD, GPIO\_PIN\_12 | GPIO\_PIN\_13 | GPIO\_PIN\_14 | GPIO\_PIN\_15, GPIO\_PIN\_RESET);~~

~~}~~

In the file audioplay.c remove the connection of unnecessary libraries

#include "audioplay.h"

~~#include "fatfs.h"~~

~~#include "usb\_host.h"~~

~~#include "lcd.h"~~

Also delete some more lines related to the use of remote libraries

~~static uint32\_t WaveDataLength = 0;~~

-

~~extern FIL WavFile;~~

-

~~extern ApplicationTypeDef Appli\_state;~~

~~char str2 [20];~~

~~char str3 [20];~~

-

~~UINT bytesread = 0;~~

-

~~extern WAVE\_FormatTypeDef \* waveformat;~~

-

~~AudioTotalSize = waveformat-> FileSize;~~

~~/ \* Get Data from USB Flash Disk \* /~~

~~WaveDataLength = waveformat-> FileSize;~~

-

~~LCD\_SetPos (0,3);~~

~~LCD\_String (str2);~~

~~LCD\_String (str3);~~

Remove some unnecessary ads from the file audioplay.c

~~/ \* Position in the audio play buffer \* /~~

~~\_\_IO BUFFER\_StateTypeDef buffer\_offset = BUFFER\_OFFSET\_NONE;~~

-

~~/ \* Defines for the Audio playing process \* /~~

~~#define PAUSE\_STATUS ((uint32\_t) 0x00) / \* Audio Player in Pause Status \* /~~

~~#define RESUME\_STATUS ((uint32\_t) 0x01) / \* Audio Player in Resume Status \* /~~

~~#define IDLE\_STATUS ((uint32\_t) 0x02) / \* Audio Player in Idle Status \* /~~

-

~~#define AUDIO\_RESET\_GPIO\_CLK\_ENABLE () \_\_GPIOD\_CLK\_ENABLE ()~~

-

~~/ \* Variables used in normal mode to manage audio file during DMA transfer \* /~~

~~uint32\_t AudioTotalSize = 0xFFFF; / \* This variable holds the total size of the audio file \* /~~

~~int32\_t AudioRemSize = 0xFFFF; / \* This variable holds the remaining data in the audio file \* /~~

~~uint16\_t \* CurrentPos; / \* This variable holds the current position of the audio pointer \* /~~

-

~~\_\_IO uint32\_t PauseResumeStatus = IDLE\_STATUS;~~

-

~~static uint32\_t WaveDataLength = 0;~~

-

~~#define AUDIO\_BUFFER\_SIZE 0X8000~~

-

~~uint8\_t Audio\_Buffer [AUDIO\_BUFFER\_SIZE];~~

~~AUDIO\_StateMachine Audio;~~

~~uint32\_t samplerate;~~

~~uint32\_t offsetpos;~~

~~uint32\_t cnt;~~

~~uint32\_t dur; // variable for the remaining time of playing the file~~

AudioPlay\_Init and AudioPlay\_Start functions will be deleted with all content

~~void AudioPlay\_Init (uint32\_t AudioFreq)~~

~~{~~

~~samplerate = AudioFreq;~~

~~\_\_IO uint8\_t volume = 70;~~

~~if (AudioOut\_Init (OUTPUT\_DEVICE\_AUTO, volume, samplerate)! = 0)~~

~~{~~

~~Error ();~~

~~}~~

~~}~~

// ------------------

~~void AudioPlay\_Start (uint32\_t AudioFreq)~~

~~{~~

-

~~}~~

Also delete the contents of the event handler functions and slightly rename them

// ------------------

void AudioPlay\_HalfTransfer\_Call **B**ack (void)

{

}

// ------------------

void AudioPlay\_Transfer **Complete**\_Call **B**ack (void)

{

}

Rename the function also in the file audioplay.h in the declaration of its prototype

void AudioPlay\_HalfTransfer\_Call **B**ack (void);

void AudioPlay\_Transfer **Complete**\_Call **B**ack (void);

Also, connect the file usbd\_audio\_if.h in this file

#include "stm32f4xx\_hal.h"

**#include "usbd\_audio\_if.h"**

Remove the function declaration from this file

~~void AudioPlay\_Start (uint32\_t AudioFreq);~~

And the announcement of the function AudioPlay\_Init will be corrected in the declaration of another function

uint8\_t AudioOut\_Init (uint16\_t OutputDevice, uint8\_t Volume, uint32\_t AudioFreq);

Also remove all declarations of variables and structures except SPI3, and instead of them we add others, taking them from the si-file

#include "usbd\_audio\_if.h"

// ----------------

#define I2S3 SPI3

**/ \* Codec output DEVICE \* /**

**#define OUTPUT\_DEVICE\_SPEAKER 1**

**#define OUTPUT\_DEVICE\_HEADPHONE 2**

**#define OUTPUT\_DEVICE\_BOTH 3**

**#define OUTPUT\_DEVICE\_AUTO 4**

**/ \* Audio status definition \* /**

**#define AUDIO\_OK ((uint8\_t) 0)**

**#define AUDIO\_ERROR ((uint8\_t) 1)**

**#define AUDIO\_TIMEOUT ((uint8\_t) 2)**

// ----------------

uint8\_t AudioOut\_Init (uint16\_t OutputDevice, uint8\_t Volume, uint32\_t AudioFreq);

Add a couple more macros

**/ \* Codec POWER DOWN modes \* /**

**#define CODEC\_PDWN\_HW 1**

**#define CODEC\_PDWN\_SW 2**

Paste the interrupt handlers in main.c

/ \* USER CODE BEGIN 4 \* /

**void HAL\_I2S\_TxCpltCallback (I2S\_HandleTypeDef \* hi2s3)**

**{**

**if (hi2s3-> Instance == I2S3)**

**{**

**AudioPlay\_TransferComplete\_CallBack ();**

**}**

**}**

**// -------------------------**

**void HAL\_I2S\_TxHalfCpltCallback (I2S\_HandleTypeDef \* hi2s)**

**{**

**if (hi2s-> Instance == I2S3)**

**{**

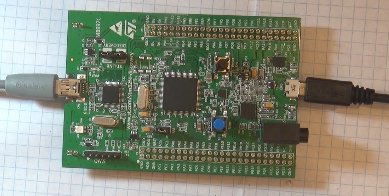
**AudioPlay\_HalfTransfer\_CallBack ();**

**}**

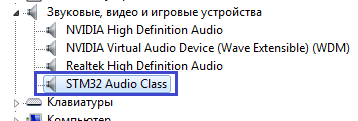
**}**

/ \* USER CODE END 4 \* /

Connect our board to the PC, and ST-Link, and USB OTG



We will collect the project, we will impose the controller and make sure that the driver of our future sound card is normally installed



In the file usbd\_audio\_if.c we connect our library

/ \* USER CODE BEGIN INCLUDE \* /

**#include "audioplay.h"**

/ \* USER CODE END INCLUDE \* /

In the same file in the AUDIO\_Init\_FS function, we call the initialization of the microcircuit

static int8\_t AUDIO\_Init\_FS (uint32\_t AudioFreq, uint32\_t Volume, uint32\_t options)

{

  / \* USER CODE BEGIN 0 \* /

**AudioOut\_Init (OUTPUT\_DEVICE\_AUTO, Volume, AudioFreq);**

  return (USBD\_OK);

  / \* USER CODE END 0 \* /

}

In the [**next part of the**](http://narodstream.ru/stm-urok-48-usb-device-audio-chast-2/) lesson, we will write a few more functions and connect them to the drivers of the sound card driver.

**Lesson 48**

**Part 2**

# USB DEVICE AUDIO

In the [**last part of**](http://narodstream.ru/stm-urok-48-usb-device-audio-chast-1/) this lesson, we created and configured a project in Cube MX, created the application framework, added all the macros and global variables that we will need in the process of writing the code. Also we started to write some functions of the audio library.

In the file audioplay.c write delay functions in microseconds. We use this function often

// ---------------------

**\_\_STATIC\_INLINE void DelayMicro (\_\_ IO uint32\_t micros)**

**{**

**micros \* = (SystemCoreClock / 1000000) / 5;**

**while (micros-);**

**}**

// ---------------------

void Error (void)

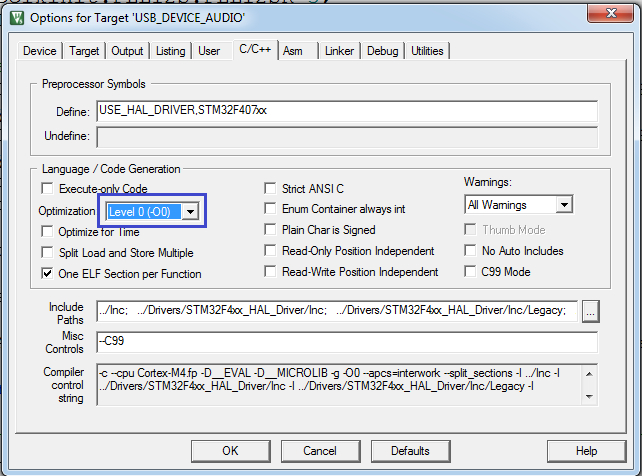
In the functions **AudioOut\_Init** and **cs43l22\_Init,** we replace the delay functions HAL\_Delay with the DelayMicro functions, since with HAL latency functions I had hangs and debugging did not work

**DelayMicro (5000);**

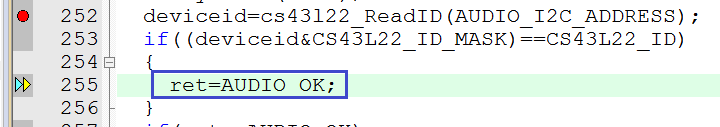
  HAL\_GPIO\_WritePin (AUDIO\_RESET\_GPIO, AUDIO\_RESET\_PIN, GPIO\_PIN\_SET);

**DelayMicro (5000);**

Disable optimization for better debugging



We compile the code and check to see if we have an audio DAC ID and whether it's actually been read.We will check this with debugging. Set the breakpoint, start debugging, get to the breakpoint, and then go in steps. If everything is properly read, we will get to the place marked in the figure



Next, insert another line in the initialization

  counter + = CODEC\_IO\_Write (DeviceAddr, 0x04, OutputDev);

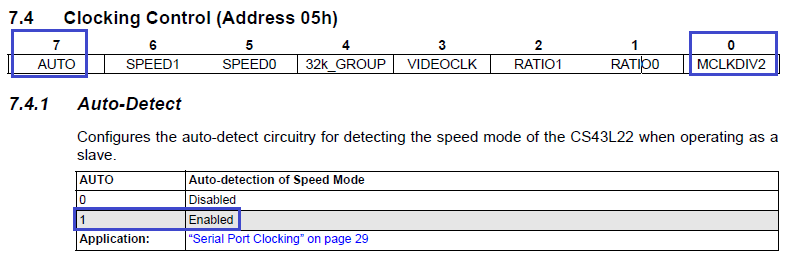
**/ \* Clock configuration: Auto detection \* /**

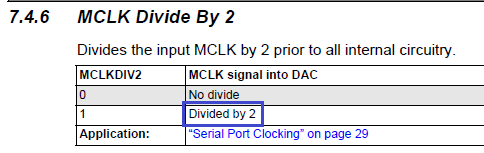
**counter + = CODEC\_IO\_Write (DeviceAddr, 0x05, 0x81);**

  / \* Set the Slave Mode and the audio Standard \* /

  counter + = CODEC\_IO\_Write (DeviceAddr, 0x06, CODEC\_STANDARD);

Thus we in the DAC include auto-detection of the speed, also a divisor by 2





Add one more input argument to the function cs43l22\_Stop in the file audioplay.c

// ------------------

uint32\_t cs43l22\_Stop (uint16\_t DeviceAddr **, uint32\_t CodecPdwnMode**)

{

Let's fix the AudioPlay\_Stop function

**uint8\_t** AudioPlay\_Stop ( **uint32\_t** Option)

{

**uint8\_t ret = AUDIO\_OK;**

**/ \* Call Audio Codec Stop function \* /**

**if (**cs43l22\_Stop (AUDIO\_I2C\_ADDRESS **, Option)! = 0)**

**{**

**ret = AUDIO\_ERROR;**

**}**

**if (ret == AUDIO\_OK)**

**{**

**if (Option == CODEC\_PDWN\_HW)**

**{**

**/ \* Wait at least 100us \* /**

**DelayMicro (2000);**

**}**

**/ \* Stop DMA transfer of PCM samples to the serial audio interface \* /**

**if (HAL\_I2S\_DMAStop (& hi2s3)! = HAL\_OK)**

**{**

**ret = AUDIO\_ERROR;**

**}**

**}**

**/ \* Return AUDIO\_OK when all operations are correctly done \* /**

**return ret;**

}

// ------------------

In the file audioplay.h add a prototype of this function

uint8\_t AudioOut\_Init (uint16\_t OutputDevice, uint8\_t Volume, uint32\_t AudioFreq);

**uint8\_t AudioPlay\_Stop (uint32\_t Option);**

Call this function in the file usbd\_audio\_if.c in a specific handler

static int8\_t AUDIO\_DeInit\_FS (uint32\_t options)

{

  / \* USER CODE BEGIN 1 \* /

**AudioPlay\_Stop (CODEC\_PDWN\_SW);**

  return (USBD\_OK);

  / \* USER CODE END 1 \* /

}

Now in the file audioplay.c change the call to the function cs43l22\_Play

uint32\_t cs43l22\_Play (uint16\_t DeviceAddr **, uint16\_t \* pBuffer, uint16\_t Size**)

{

We insert two functions of the following content

// ------------------

**uint8\_t AudioPlay\_Play (uint16\_t \* pBuffer, uint32\_t Size)**

**{**

**uint8\_t ret = AUDIO\_OK;**

**if (cs43l22\_Play (AUDIO\_I2C\_ADDRESS, pBuffer, Size)! = 0)**

**{**

**Error ();**

**ret = AUDIO\_ERROR;**

**}**

**if (ret == AUDIO\_OK)**

**{**

**if (HAL\_I2S\_Transmit\_DMA (& hi2s3, (uint16\_t \*) pBuffer, DMA\_MAX (Size / AUDIODATA\_SIZE))! = HAL\_OK)**

**{**

**Error ();**

**ret = AUDIO\_ERROR;**

**}**

**}**

**return ret;**

**}**

**// ------------------**

**void AudioPlay\_ChangeBuffer (uint16\_t \* pData, uint16\_t Size)**

**{**

**HAL\_I2S\_Transmit\_DMA (& hi2s3, (uint16\_t \*) pData, Size);**

**}**

// ------------------

Well, everything is standard here, I think there's nothing to explain.

We paste the prototypes into the file audioplay.h

uint8\_t AudioPlay\_Stop (uint32\_t Option);

**uint8\_t AudioPlay\_Play (uint16\_t \* pBuffer, uint32\_t Size);**

**void AudioPlay\_ChangeBuffer (uint16\_t \* pData, uint16\_t Size);**

In the file usbd\_audio\_if.c in the handler we insert function calls

static int8\_t AUDIO\_AudioCmd\_FS (uint8\_t \* pbuf, uint32\_t size, uint8\_t cmd)

{

  / \* USER CODE BEGIN 2 \* /

  switch (cmd)

  {

    case AUDIO\_CMD\_START:

**AudioPlay\_Play ((uint16\_t \*) pbuf, size);**

    break;

    case AUDIO\_CMD\_PLAY:

**AudioPlay\_ChangeBuffer ((uint16\_t \*) pbuf, size);**

    break;

  }

  return (USBD\_OK);

  / \* USER CODE END 2 \* /

}

In the [**next part of**](http://narodstream.ru/stm-urok-48-usb-device-audio-chast-3/) this lesson we will finish the study of this topic and hear the work of the sound card written by our hands.

**Lesson 48**

**Part 3**

# USB DEVICE AUDIO

[*Previous part*](http://narodstream.ru/stm-urok-48-usb-device-audio-chast-2/) [*Programming STM32 MK*](http://narodstream.ru/programmirovanie-mk-stm32/) [*Next lesson*](http://narodstream.ru/stm-urok-49-hal-magnitometr-lsm303dlhc-chast-1/)

In the [**last part of**](http://narodstream.ru/stm-urok-48-usb-device-audio-chast-2/) this lesson, we wrote a few more functions and connected them to the drivers of the sound card driver.

Insert another function into the file audioplay.c

**// ------------------**

**uint8\_t AudioPlay\_SetVolume (uint8\_t Volume)**

**{**

**uint8\_t ret = AUDIO\_OK;**

**/ \* Call the codec volume control function with converted volume value \* /**

**if (cs43l22\_SetVolume (AUDIO\_I2C\_ADDRESS, Volume)! = 0)**

**{**

**ret = AUDIO\_ERROR;**

**}**

**/ \* Return AUDIO\_OK when all operations are correctly done \* /**

**return ret;**

**}**

**// ------------------**

We paste the prototype of this function into the file audioplay.h

void AudioPlay\_ChangeBuffer (uint16\_t \* pData, uint16\_t Size);

**uint8\_t AudioPlay\_SetVolume (uint8\_t Volume);**

In the file usbd\_audio\_if.c in the AUDIO\_VolumeCtl\_FS handler, we call this function

static int8\_t AUDIO\_VolumeCtl\_FS (uint8\_t vol)

{

  / \* USER CODE BEGIN 3 \* /

**AudioPlay\_SetVolume (vol);**

  return (USBD\_OK);

  / \* USER CODE END 3 \* /

}

Insert another function into the file audioplay.c

// ------------------

**uint8\_t AudioPlay\_SetMute (uint32\_t Cmd)**

**{**

**uint8\_t ret = AUDIO\_OK;**

**/ \* Call the Codec Mute function \* /**

**if (cs43l22\_SetMute (AUDIO\_I2C\_ADDRESS, Cmd)! = 0)**

**{**

**ret = AUDIO\_ERROR;**

**}**

**/ \* Return AUDIO\_OK when all operations are correctly done \* /**

**return ret;**

**}**

// ------------------

Let's write a prototype for it

uint8\_t AudioPlay\_SetVolume (uint8\_t Volume);

**uint8\_t AudioPlay\_SetMute (uint32\_t Cmd);**

Insert the function call in the file usbd\_audio\_if.c in the same name handler

static int8\_t AUDIO\_MuteCtl\_FS (uint8\_t cmd)

{

  / \* USER CODE BEGIN 4 \* /

**AudioPlay\_SetMute (cmd);**

  return (USBD\_OK);

  / \* USER CODE END 4 \* /

}

In the file audioplay.c in the handlers we will add calls to these handlers in the USB library

// ------------------

void AudioPlay\_HalfTransfer\_Callback (void)

{

**HalfTransfer\_CallBack\_FS ();**

**}**

// ------------------

void AudioPlay\_TransferComplete\_CallBack (void)

{

**TransferComplete\_CallBack\_FS ();**

}

We will collect the project and we will sew the controller.

Also after the firmware it is desirable to enter the parameters of the playback devices and set this device as the default

