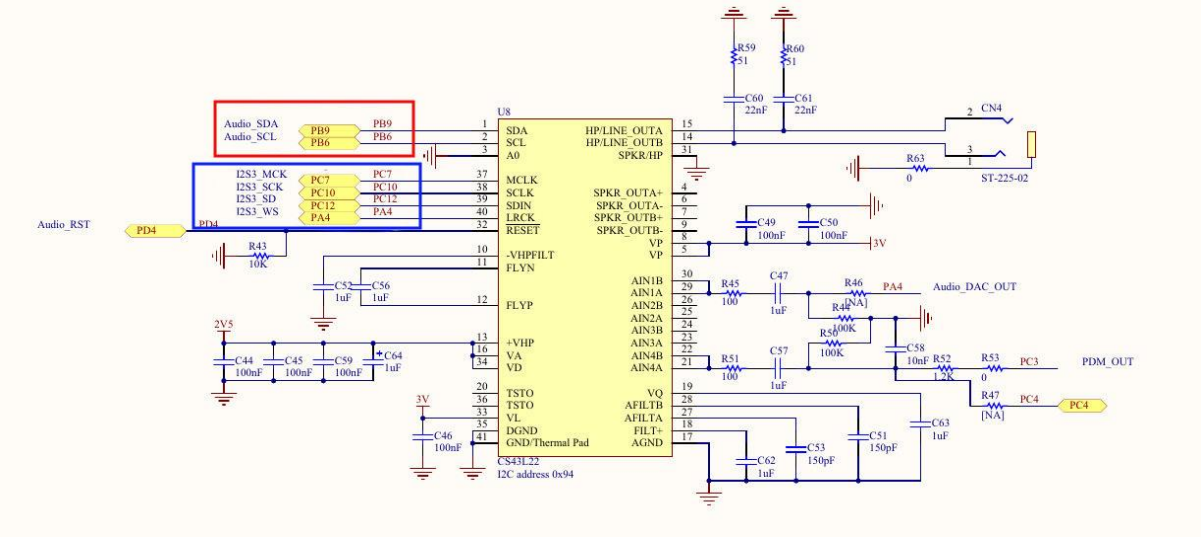
«به نام خدا»

گام به گام ساخت پروژه

Steps Create Projects

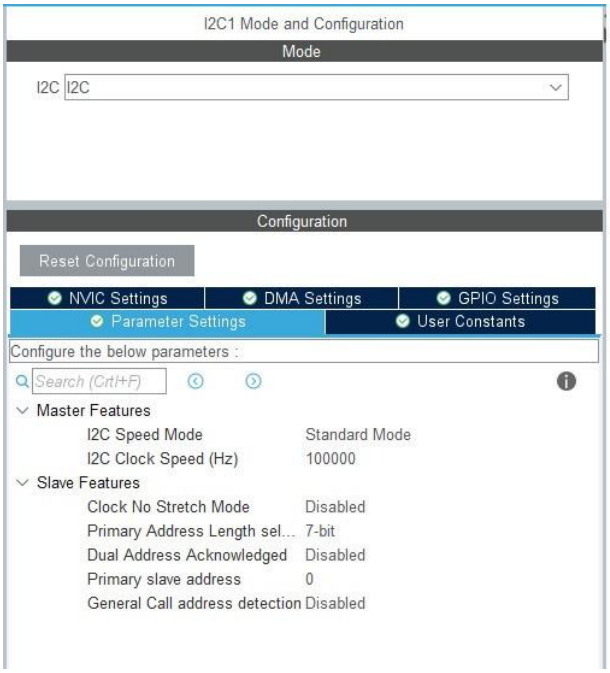
راه اندازی CubeMX :

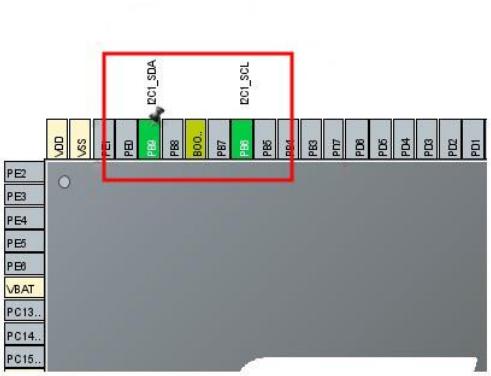
در زیر تصویر شماتیک های بردی که برای این پروژه استفاده می کنیم را مشاهده می کنید. تمام لوازم جانبی را بر اساس ارتباط آنها با آی سی CS۴۳L۲۲ تنظیم خواهم کرد.



I2Cمحیطی

اول از همه دستگاه جانبی I2C را مطابق شکل زیر تنظیم می کنیم.

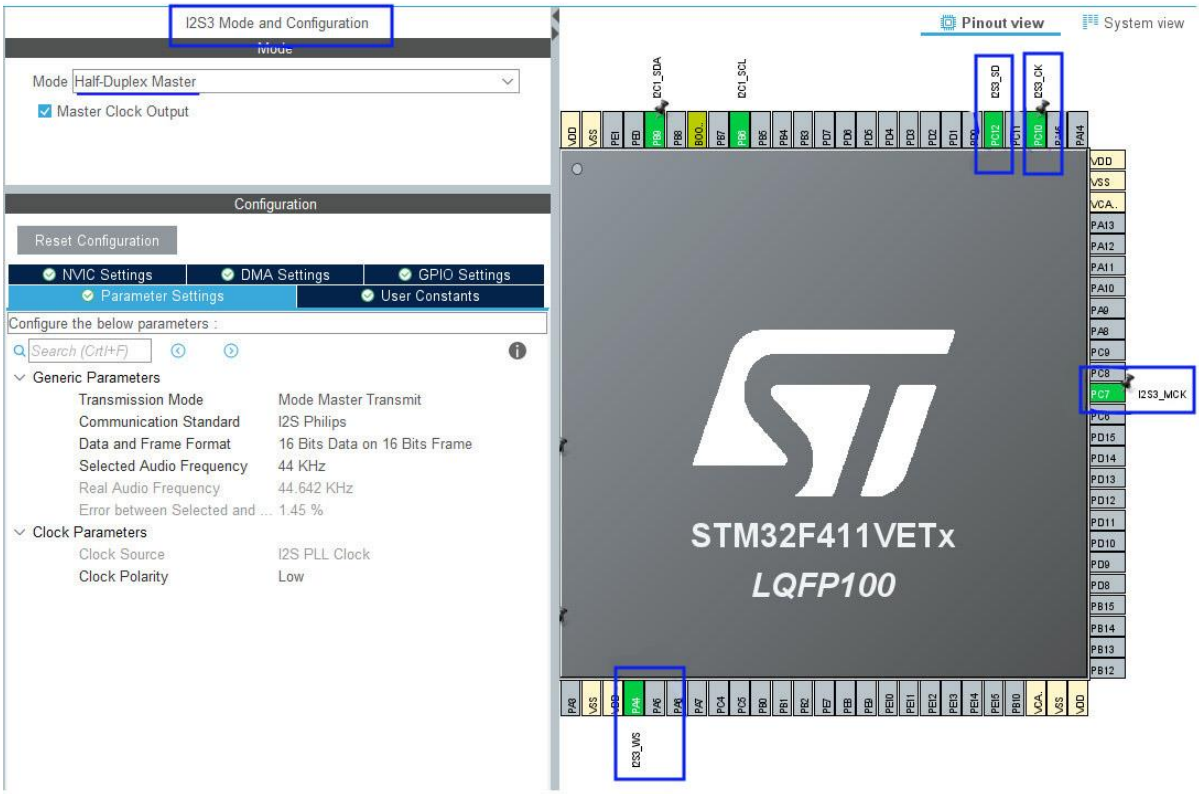




در اینجا باید پین پیش‌فرض SDA (PB7) را به PB9 تغییر دهم، زیرا این پین متصل به آی‌سی است.

I2Cمحیطی:

پس از I2C، زمان راه اندازی دستگاه جانبی I2S است.



اطمینان حاصل می کنیم که تمام پین ها را همانطور که در شماتیک نشان داده شده است انتخاب کرده ایم.

PC12 -> I2S3_SD

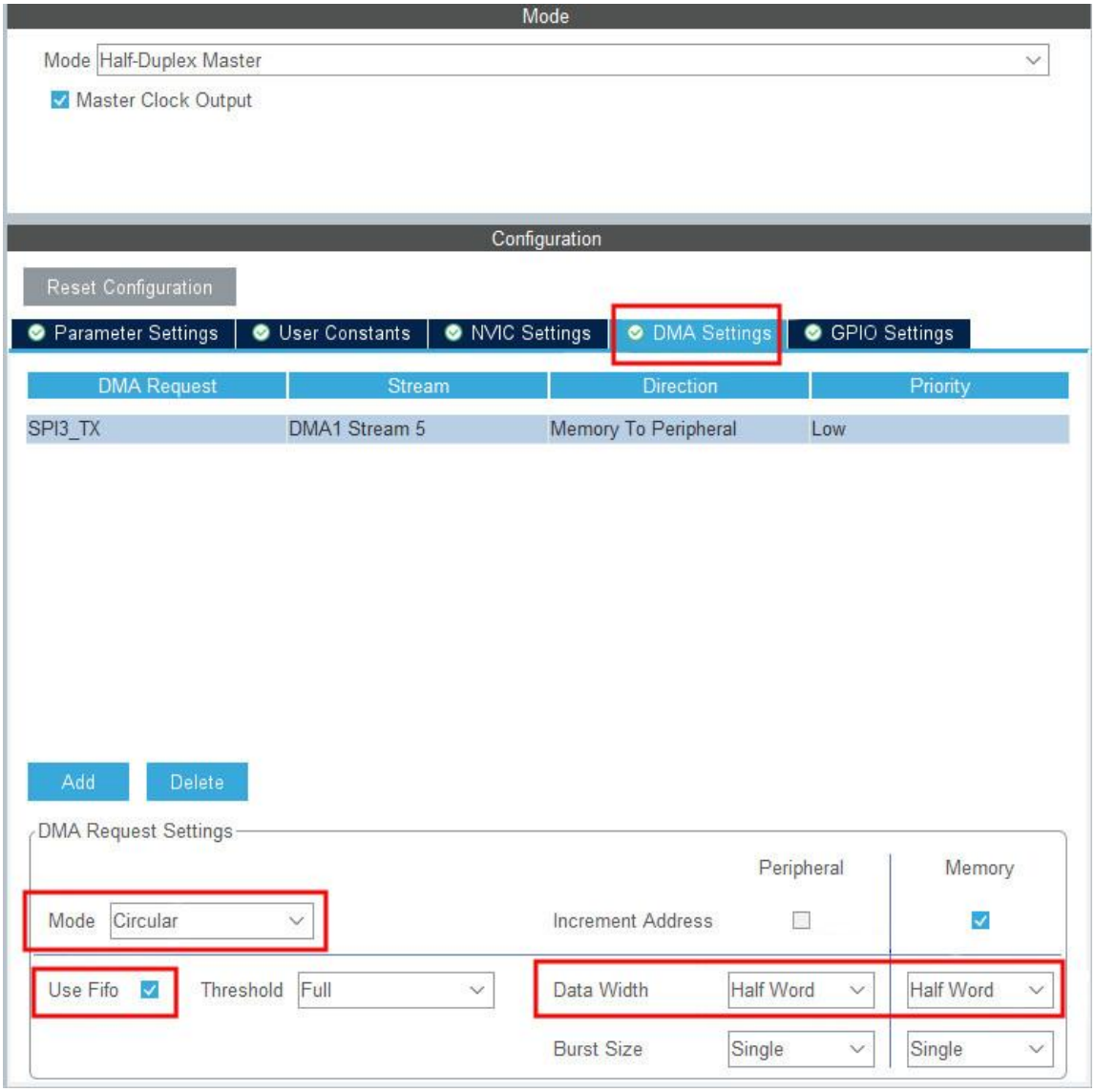
PC10 -> I2S3_CK

PC7 -> I2S3_MCK

PA4 -> I2S3_WS

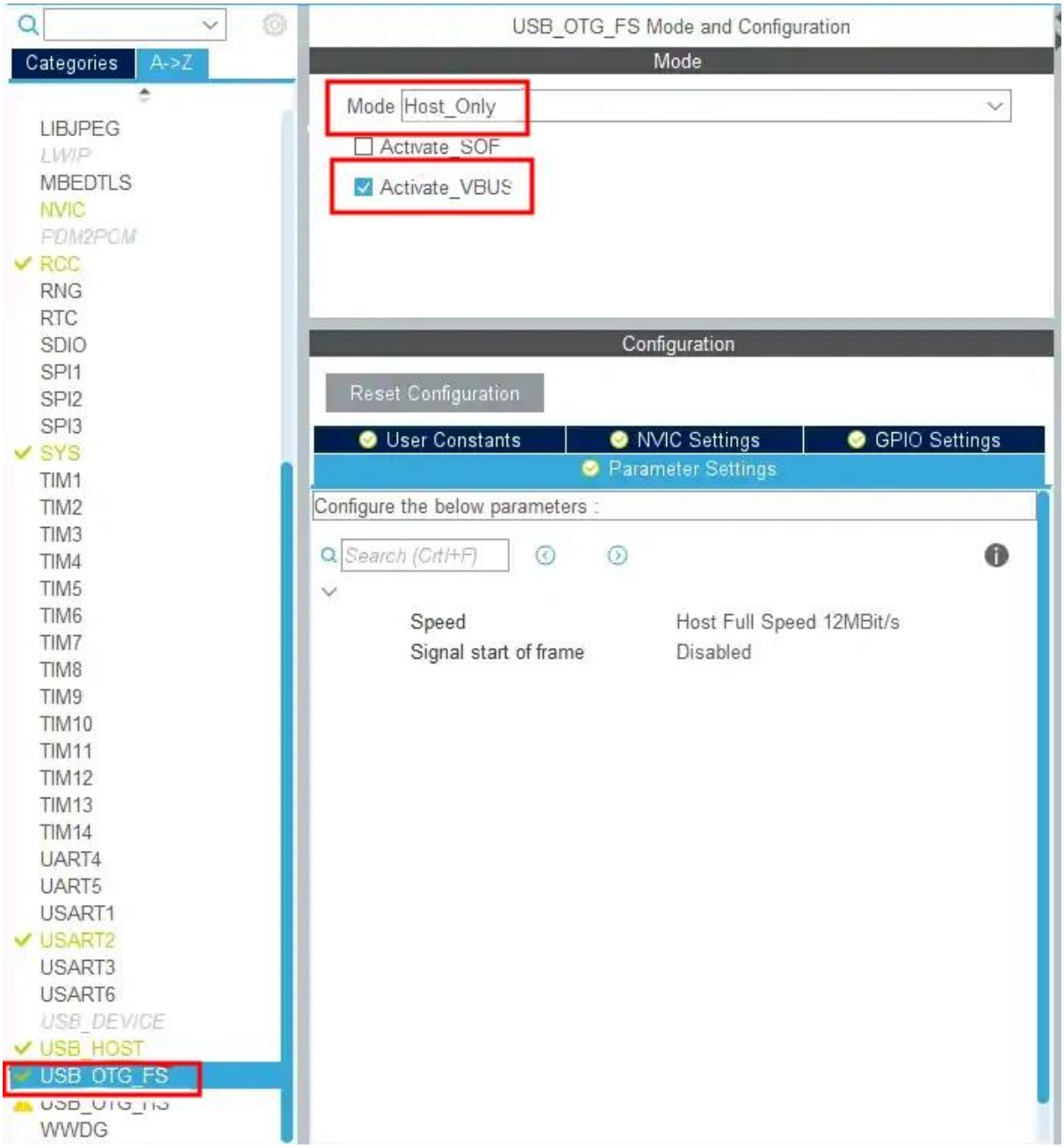
در اینجا نیز باید DMA را مانند شکل زیر فعال کنیم.

DMAدر circular mode انتخاب می کنیم و پهنای داده در حالت Half widthاست زیرا با ۱۶ بیت سروکار خواهیم داشت.


Mode
Mode Half-Duplex Master
Master Clock Output
Configuration
Reset Configuration
Parameter Settings
User Constants
NVIC Settings
DMA Settings
GPIO Settings
DMA Request
Stream
Direction
Priority
SPI3_TX
DMA1 Stream 5
Memory To Peripheral
Low
Add
Delete
DMA Request Settings
Peripheral
Memory
Mode Circular
Increment Address
Use Fifo
Threshold Full
Data Width
Half Word
Half Word
Burst Size
Single
Single

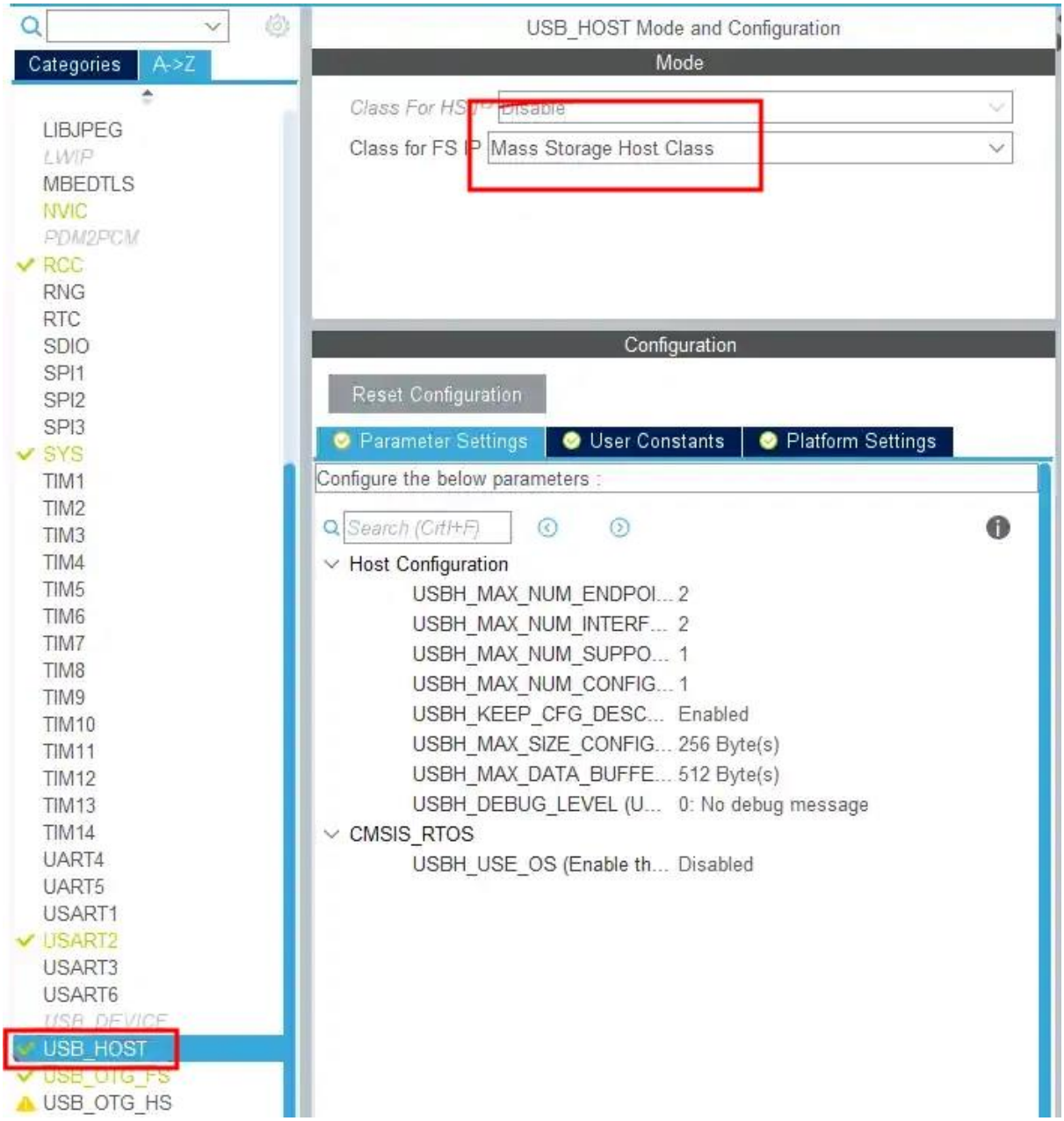
USB HOST:

بعد باید USB HOST را فعال کنیم تا بتوانیم فایل های صوتی را از USB بخوانیم.

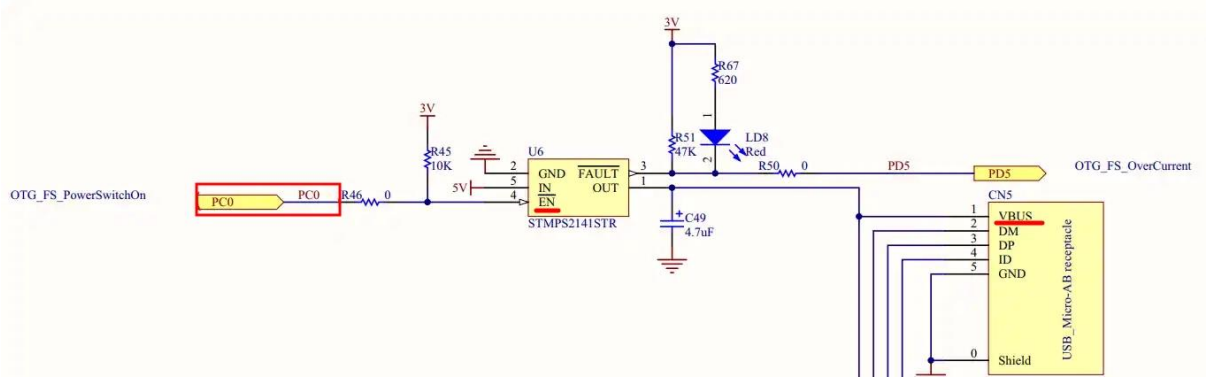


همچنین مطمئن شویم که VBUS را فعال کرده ایم، زیرا دستگاه های USB عمدتا منبع تغذیه ندارند و به همین دلیل HOST مسئول تامین برق مورد نیاز برای چنین دستگاه هایی است.

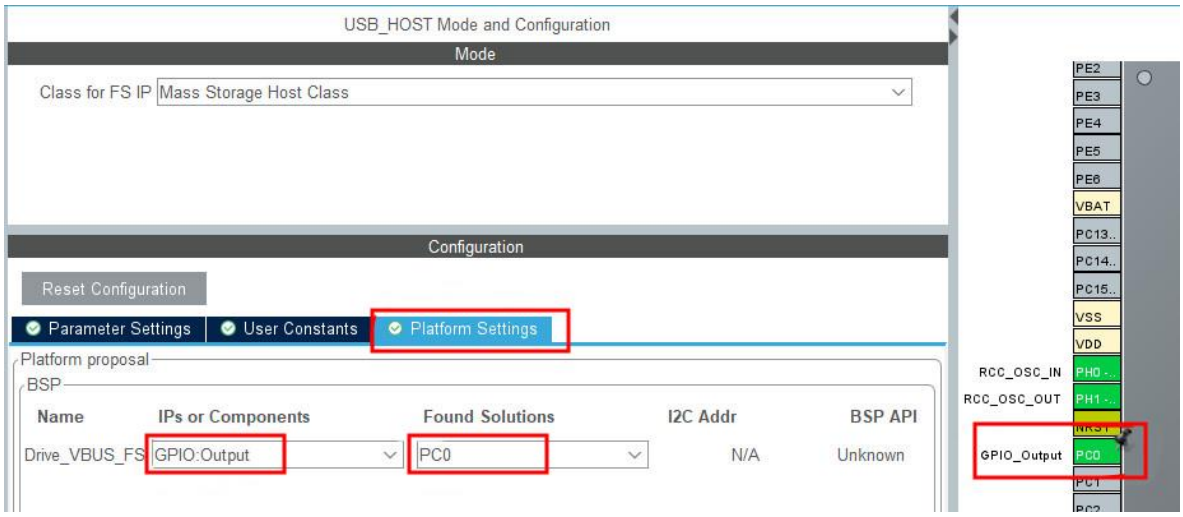
سپس USB_HOST را انتخاب و کلاس را به عنوان کلاس ذخیره‌سازی انبوه انتخاب می‌کنیم. همه چیز را در اینجا به حالت پیش فرض می‌گذاریم.



و اکنون باید منبع ولتاژ پین VBUS را فعال کنیم. برای این کار باید به دفترچه راهنمای برد نگاه کنیم. ما از برد STM32F4 Discovery استفاده می کنیم و نمودار زیر را برای USB داریم.



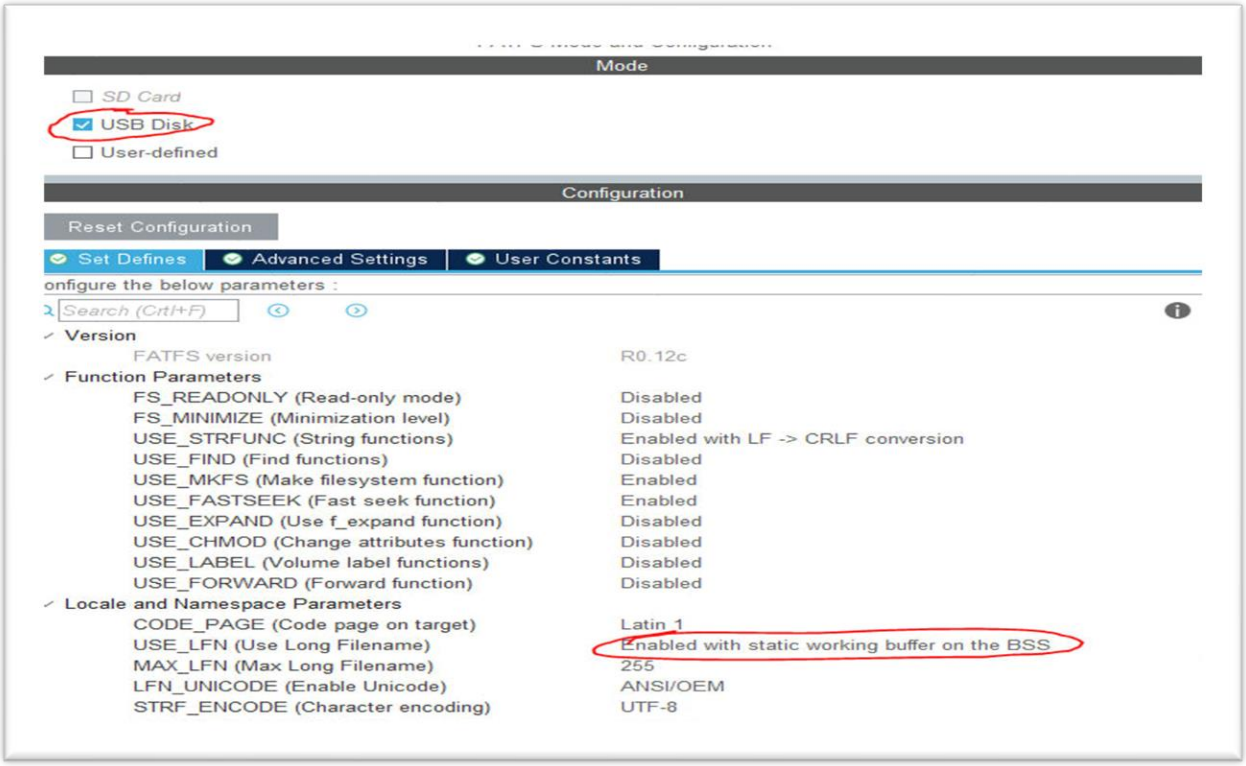
متصل است که یک پایه EN‌ه به پین PC‌ه تغذیه می شود. اما PC از پین VBUS همانطور که در بالا می بینیم، ه را پایین بیاوریم یا اساساً آن PC، باید پین VBUS است. این بدان معناست که برای تامین ولتاژ به active Low تنظیم کنیم.LOWرا



همانطور که در بالا نشان داده شده است، PC‌ه را در تنظیمات پلتفرم فعال می‌کنیم.

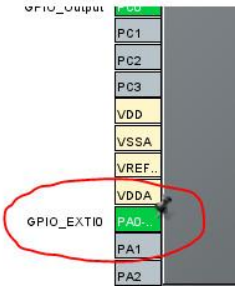
سیستم فایل FAT:

در حال حاضر ما سیستم فایل FAT را راه اندازی خواهیم کرد



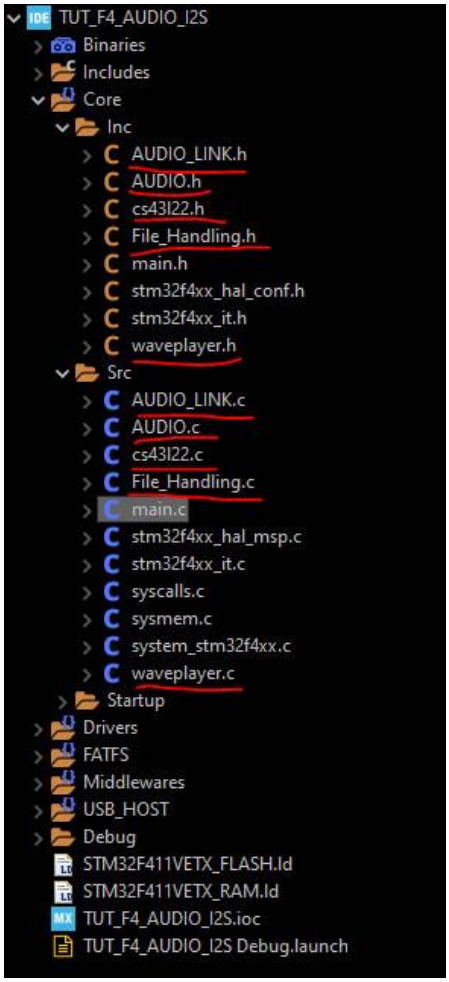
حالت دیسک USB را انتخاب کنید و همه چیز را به صورت پیش فرض در اینجا بگذارید. من استفاده از نام طولانی را فعال کرده ام فقط در صورتی که پرونده هایی با نام های طولانی داشته باشم.

و در نهایت من از PA0 به عنوان یک کلید وقفه خارجی استفاده می کنم. این برای کنترل عملیات در بازیکن مانند NEXT، PREVIOUS، PAUSE، RESUME و غیره است..



برخی از بینش به کد:

باید کپی کنیم فایل های کتابخانه مربوطه به SRC و Inc پوشه. تصویر نهایی این کتابخانه ها در زیر نشان داده شده است.



فایل های زیر کتابخانه هایی هستند که باید در پروژه خود اضافه کنید. شما می توانید انها را در پایان این دانلود کنید کنید.

تعریف:

اول از همه ما نیاز به قرار دادن فایل های هدر در فایل اصلی ما. ما فقط می خواهیم استفاده از توابع از فایل ها موج Player.h و File_Handling ساعت

includes

```
/*---------------------------------------------------------- Private includes */
/* USER CODE BEGIN Includes */

"include "waveplayer.h#
"include "File_Handling.h#

/* USER CODE END Includes */
```

بعد، ما توابع را برای بررسی وضعیت USB و حالت AUDIO تعریف خواهیم کرد

defines

```
;extern ApplicationTypeDef Appli_state
;extern AUDIO_PLAYBACK_StateTypeDef AudioState

;int IsFinished = 0
```

ApplicationTypeDef Appli_state در حال حاضر در فایل **usb_host.c** تعریف شده است و به همین دلیل ما باید ان را به عنوان یک extern اعلام کنیم USB .می تواند حالت های زیر را داشته باشد:

- APPLICATION_IDLE
- APPLICATION_START
- APPLICATION_READY
- APPLICATION_DISCONNECT

AUDIO_PLAYBACK_StateTypeDef AudioState در فایل waveplayer.c تعریف شده است و دارای حالت های زیر است:

- AUDIO_STATE_IDLE
- AUDIO_STATE_WAIT
- AUDIO_STATE_INIT
- AUDIO_STATE_PLAY
- AUDIO_STATE_RECORD
- AUDIO_STATE_NEXT
- AUDIO_STATE_PREVIOUS
- AUDIO_STATE_FORWARD
- AUDIO_STATE_BACKWARD
- AUDIO_STATE_STOP
- AUDIO_STATE_PAUSE
- AUDIO_STATE_RESUME
- AUDIO_STATE_VOLUME_UP
- AUDIO_STATE_VOLUME_DOWN
- AUDIO_STATE_ERROR

دیگر پس از ان این، من نیز تعریف یک متغیر **IsFinished** برای بررسی اگر پخش صوتی متوقف شده است.

قطع تماس

در حال حاضر ما یک تماس وقفه خارجی برای دکمه ارسال خواهیم کرد.

EXTI callback

```
void HAL_GPIO_EXTI_Callback(uint16_t GPIO_Pin)
}
if (GPIO_Pin == GPIO_PIN_0)
}
;AudioState = AUDIO_STATE_NEXT      //
if (AudioState == AUDIO_STATE_PLAY)
}
;AudioState = AUDIO_STATE_PAUSE
{

if (AudioState == AUDIO_STATE_WAIT)
}
;AudioState = AUDIO_STATE_RESUME
{
{
{
```

در اینجا ما می توانیم کنترل کنیم که دکمه چه کاری انجام می دهد. در کد بالا، دکمه به PAUSE برنامه ریزی شده است و صدا را از سر می گیرد.

تابع اصلی

کد اصلی ما در داخل حلقه while نوشته خواهد شد

```
while (1)
}
/* USER CODE END WHILE */
;()MX_USB_HOST_Process

/* USER CODE BEGIN 3 */

if (Appli_state == APPLICATION_READY)
}
;()Mount_USB
;AUDIO_PLAYER_Start(0)

while (!IsFinished)
}
;AUDIO_PLAYER_Process(TRUE)

if (AudioState == AUDIO_STATE_STOP)
}
;IsFinished = 1
{
{
{


{
```

در کد بالا، ابتدا ما به طور مداوم بررسی می کنیم که ایا USB برای برقراری ارتباط اماده است. وقتی اماده شد

- سوار USB .
- پخش کننده صوتی را شروع کنید. پارامتر شاخص اهنگ (اهنگ) است که می خواهید با ان شروع به پخش کنید.
- **AUDIO_PLAYER_Process**پردازش AUDIO را تا زمانی که متوقف شود، ادامه خواهد داد. این پارامتر "**isLoop**" را می گیرد، به این معنی که ایا حلقه AUDIO را می خواهید؟ اگر **TRUE**تنظیم شود، اهنگ اول پس از اخرین اهنگ پخش می شود یا در صورت **FALSE**، بازیکن پس از اخرین اهنگ متوقف می شود.
- اگر وضعیت AUDIO STOP باشد، IsFinished به ۱ تنظیم می شود و پردازش متوقف می شود.