# STM32L4 – Putting All Together

STM32L4 Workshop


life.augmented

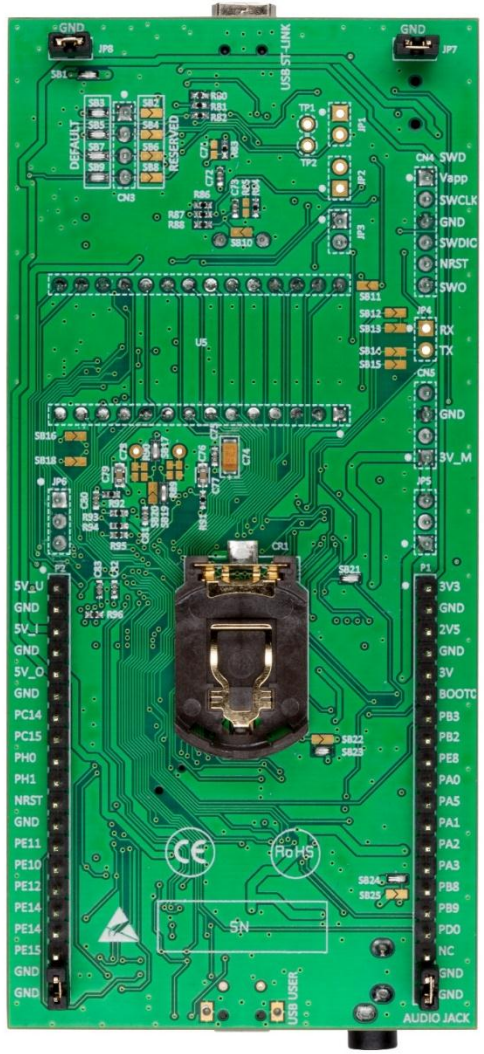

# Goal of this part



- Get some basic knowledge about advanced topics
- Practice a little bit with STM32CubeMX
- Understand the structure of prepared demo application
- Have some fun!

# STM32L476 Discovery

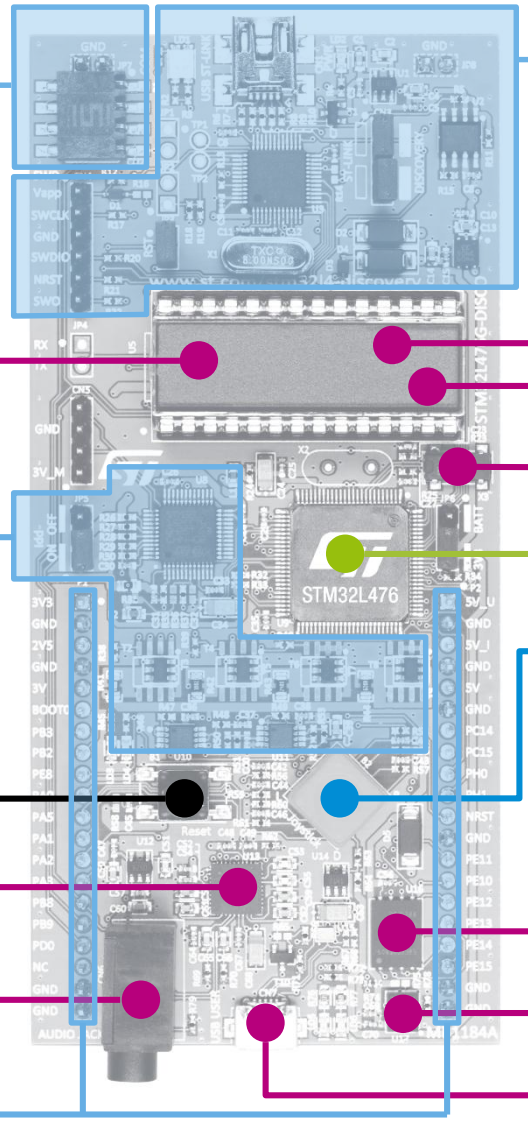



life.augmented

# STM32L476 Discovery






APC connector
(for Apple connector – I²C)

Glass LCD (96 segments)

MFX
(for $I_{DD}$ measurement)

Reset Button

Audio Codec

Audio Out (3.5mm jack)

GPIO access

Integrated ST-Link/V2-1
(for programming and debugging)

Accelerometer + Magnetometer (3-axis)

Gyro (3-axis)

32.768kHz crystal

STM32L476VG
1MB FLASH
LQFP100

Joystick (Push Buttons)

QUADSPI NOR Flash (16MB)

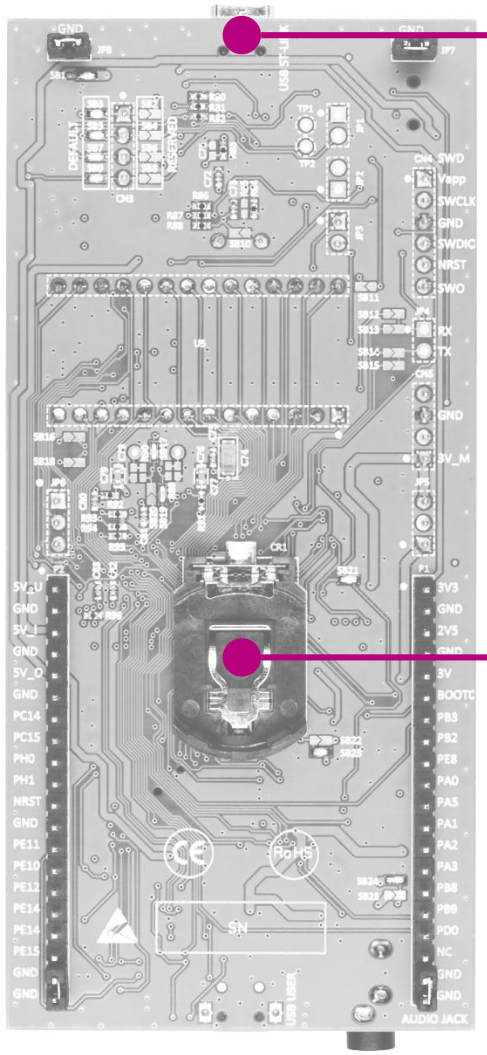
MEMS Microphone

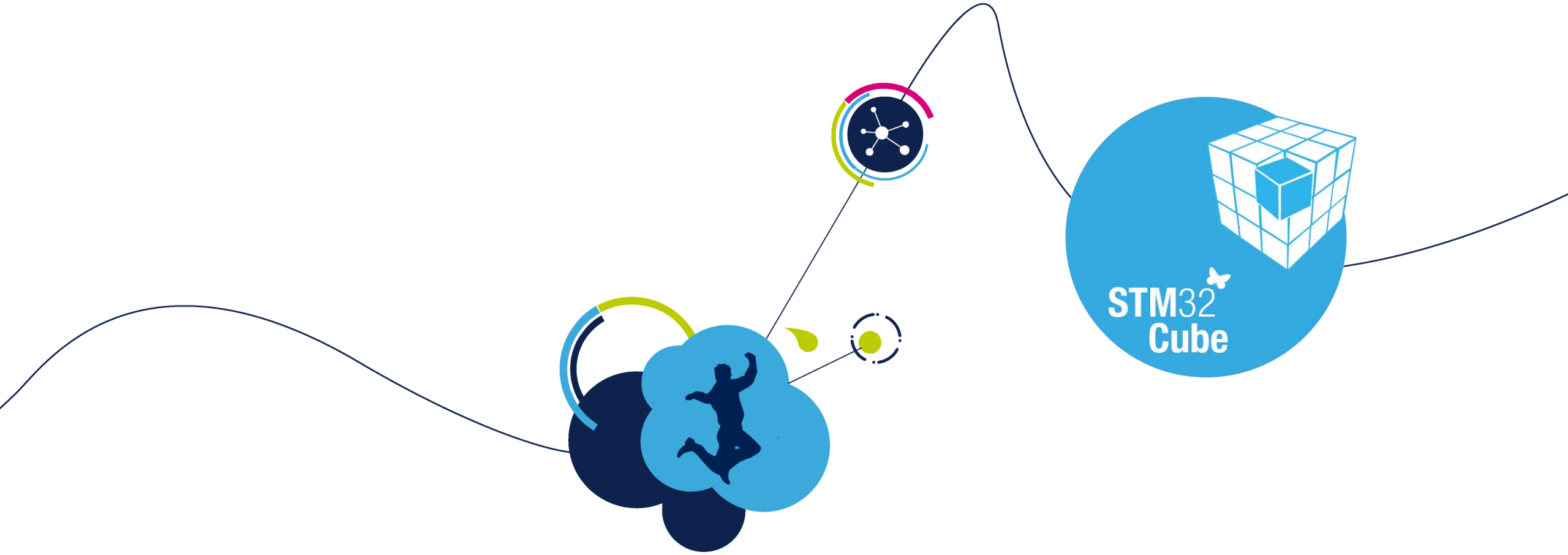
USB OTG connector

# STM32L476 Discovery



Flexible board power supply





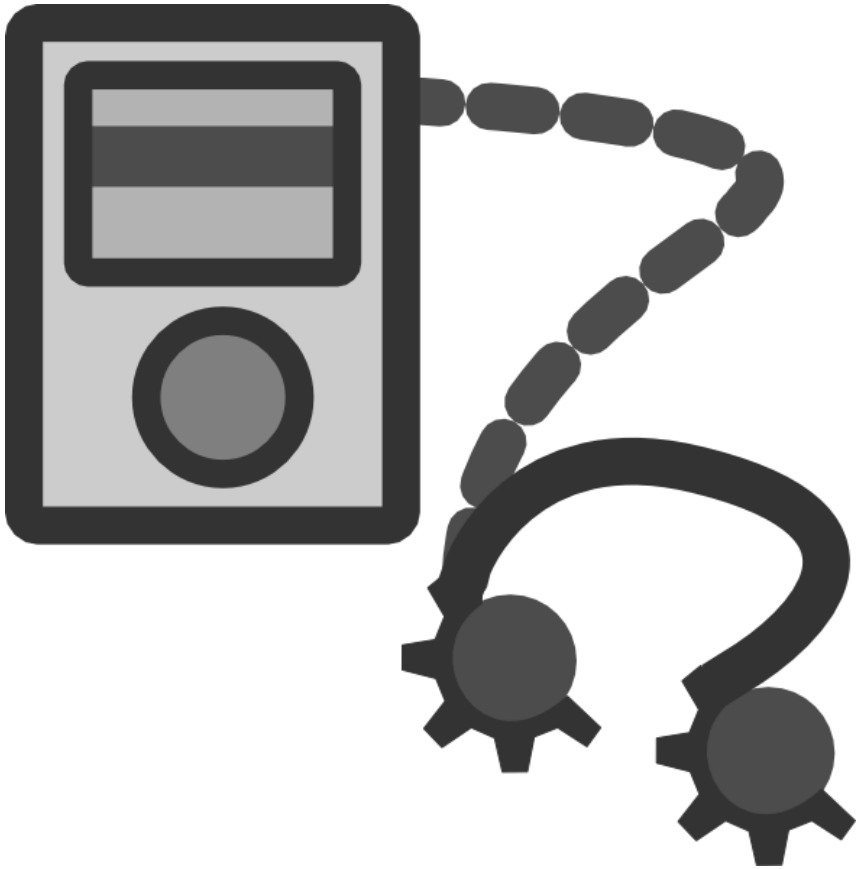

# Let’s start to work!

life.augmented

# Application idea

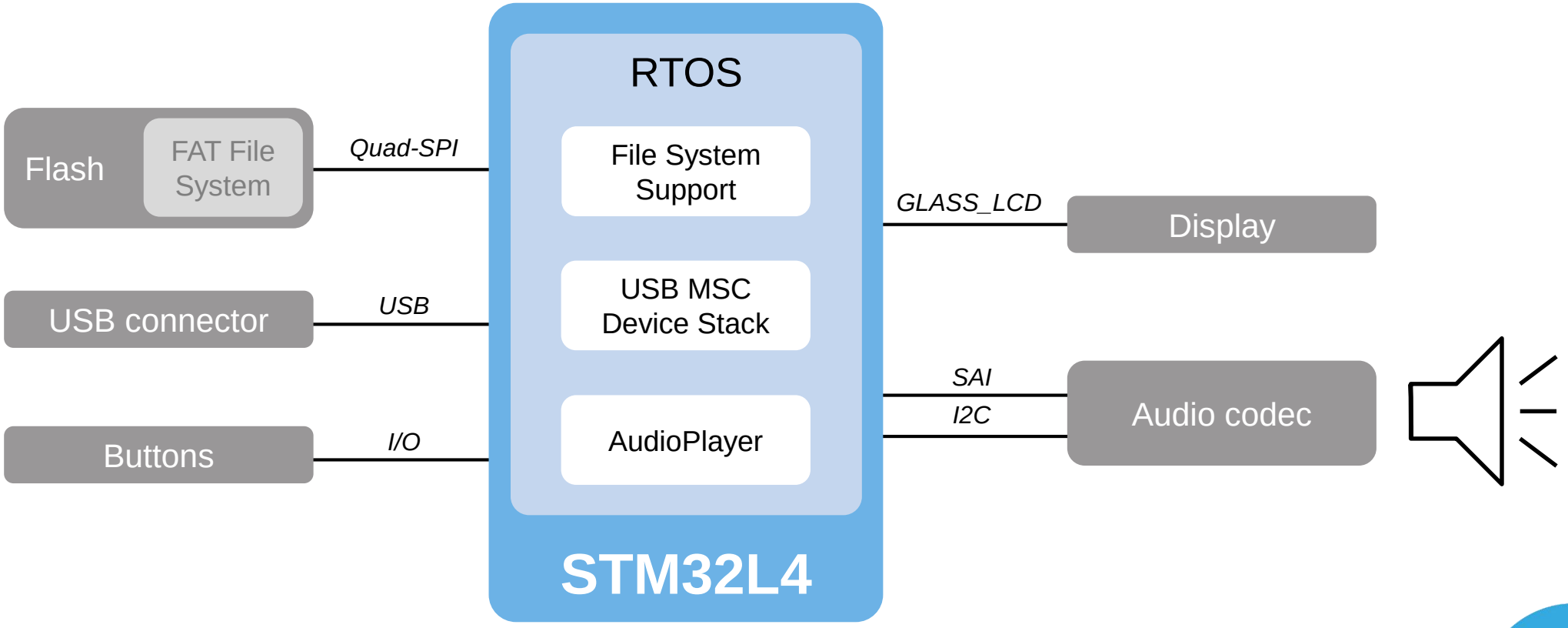


- Create simple Audio Player



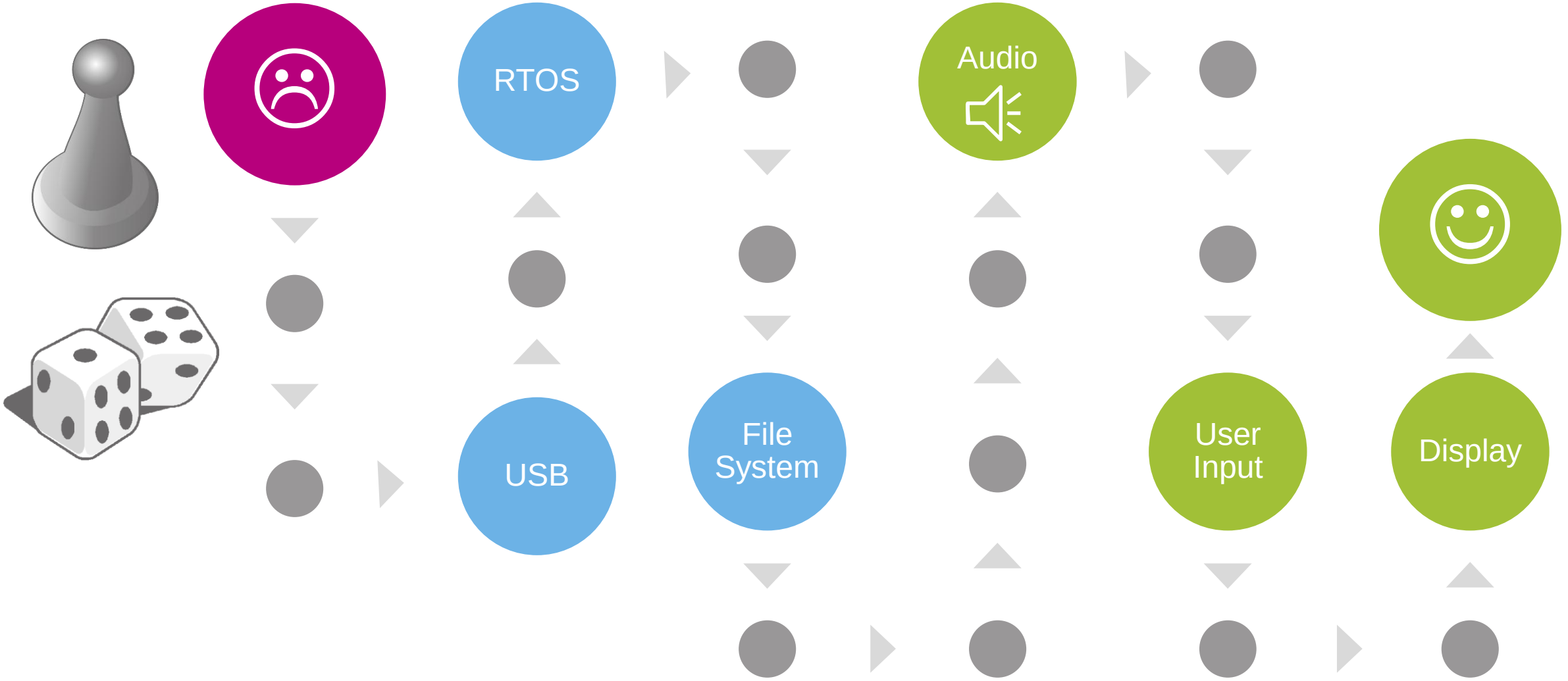
# Simplified target block diagram

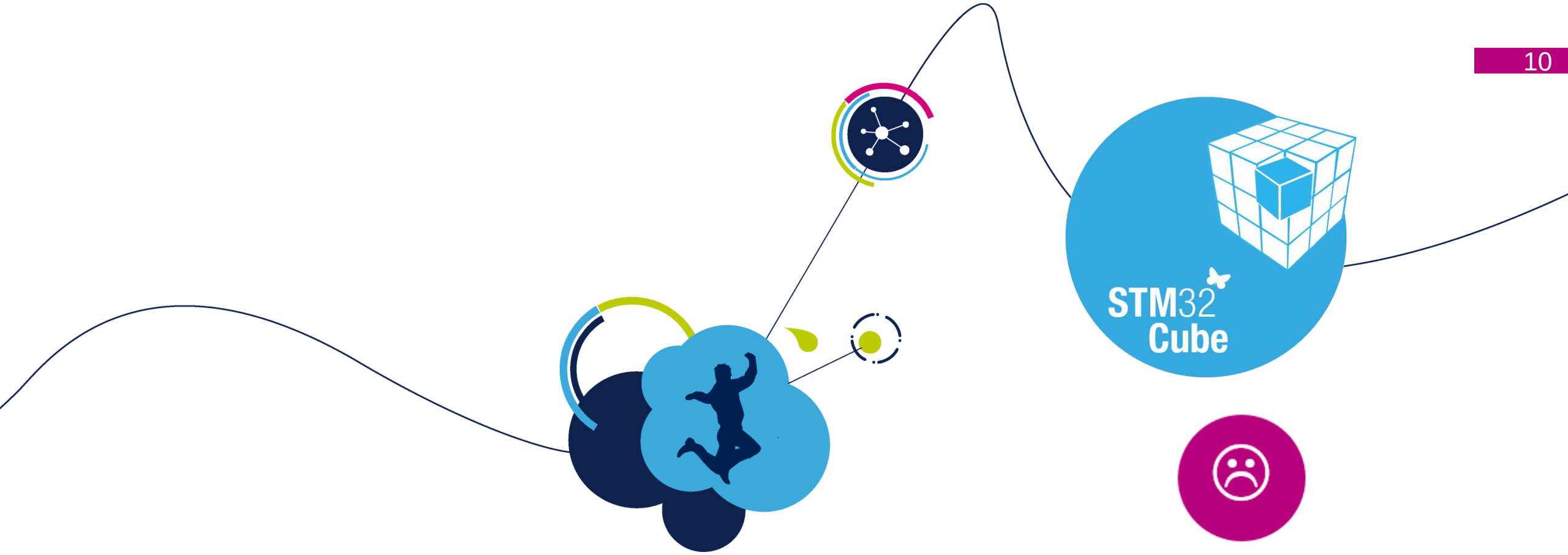




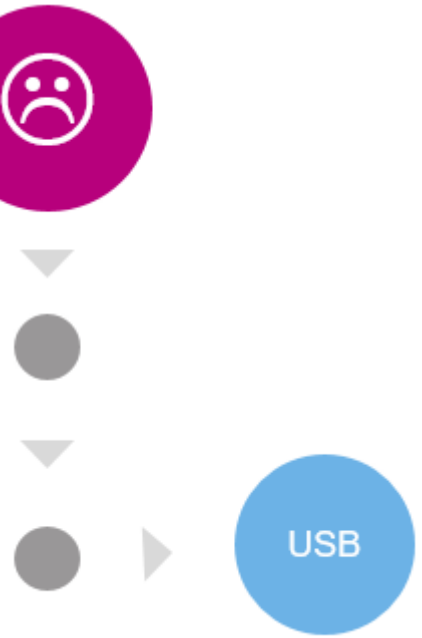
# „Game“ flow




RTOS
Audio
USB
File System
User Input
Display
main target functionality
bonus functionality




# STEP1 – USB MSC Device

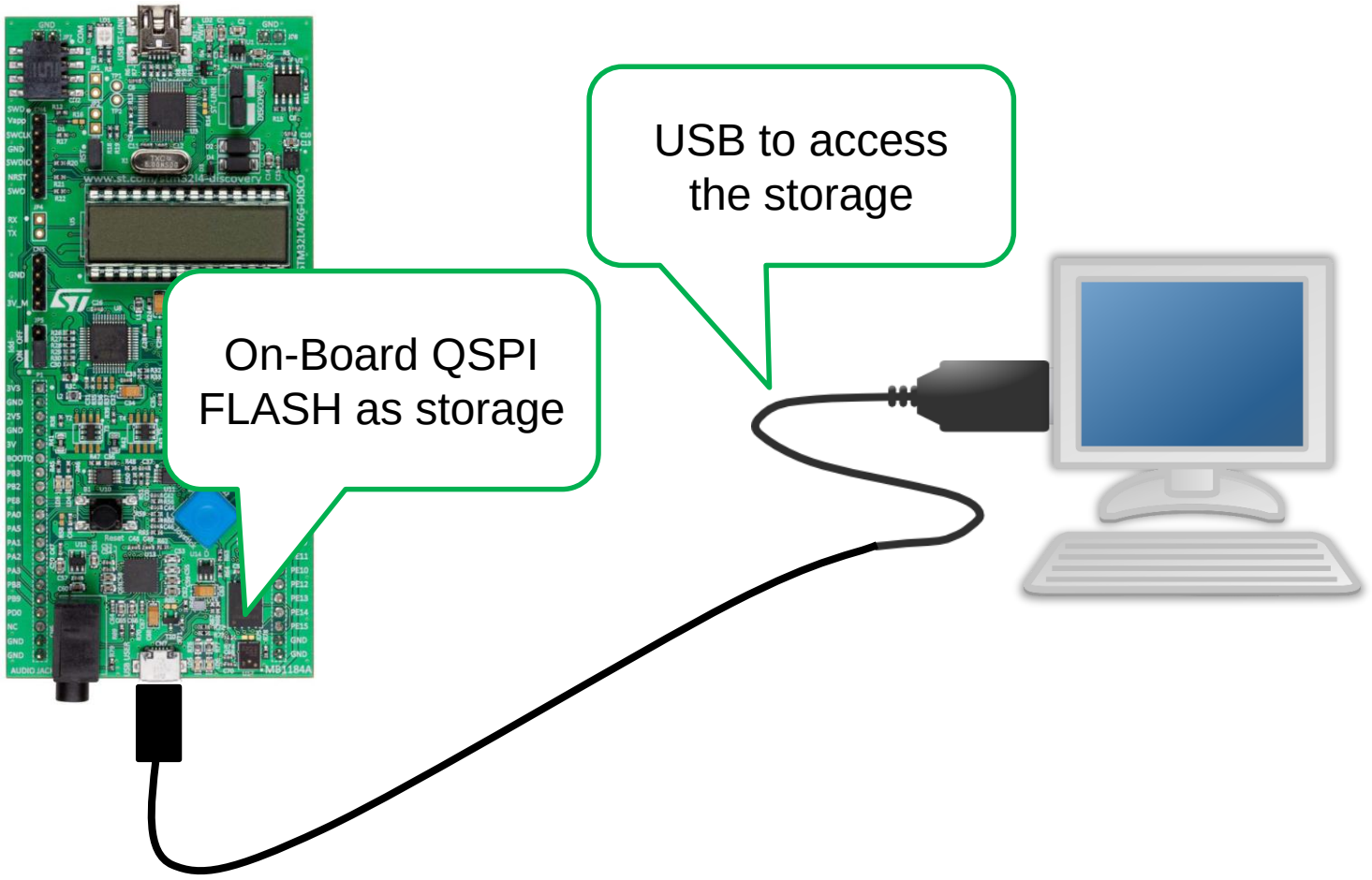



life.augmented

# 1 USB MSC Device with QUADSPI



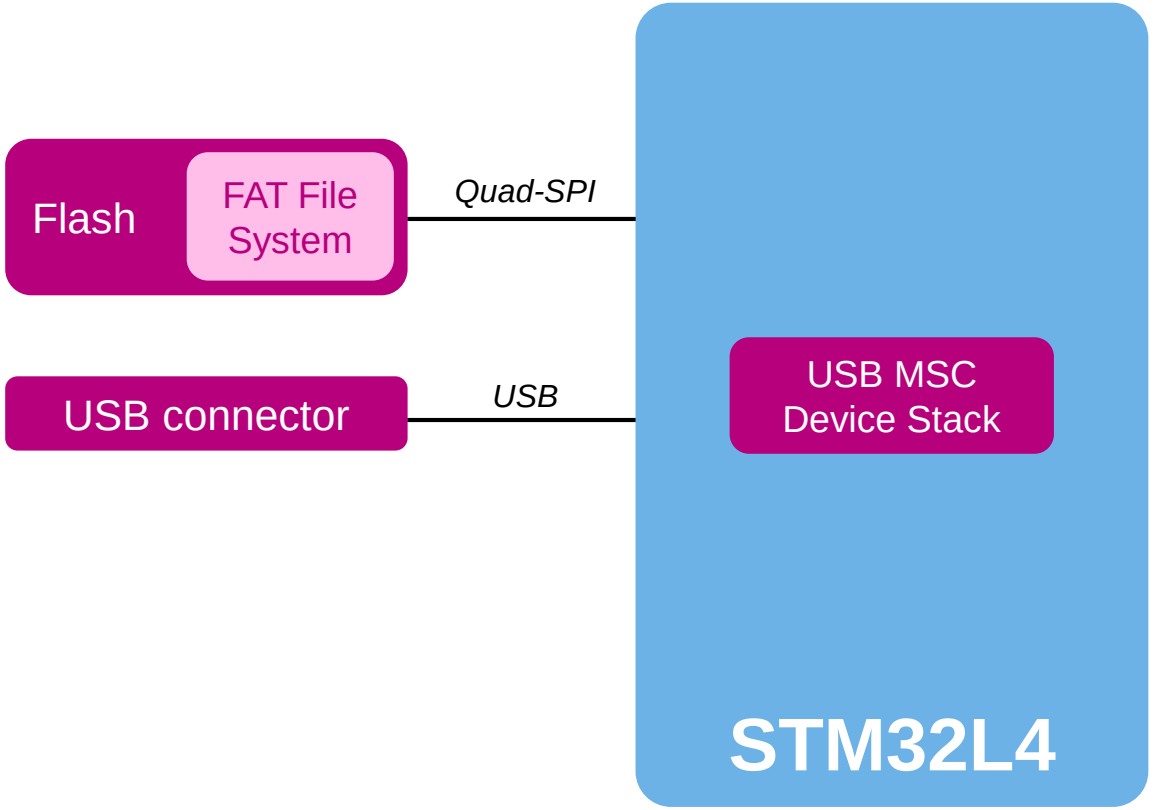
- We need storage for our audio files!



# 1 Adding USB MSC Device functionality



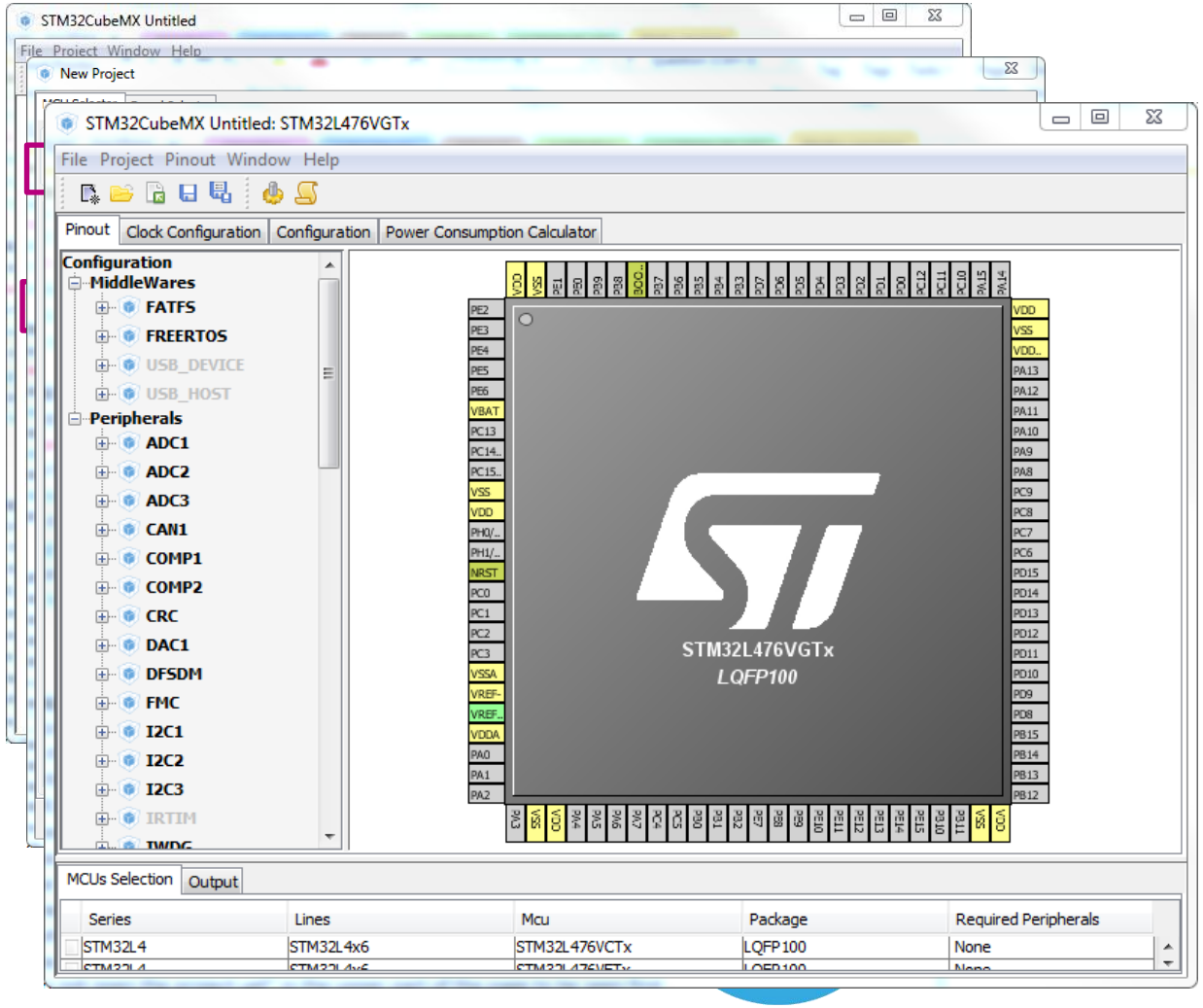
- We need storage for our audio files!



# 1 Create new project in CubeMX



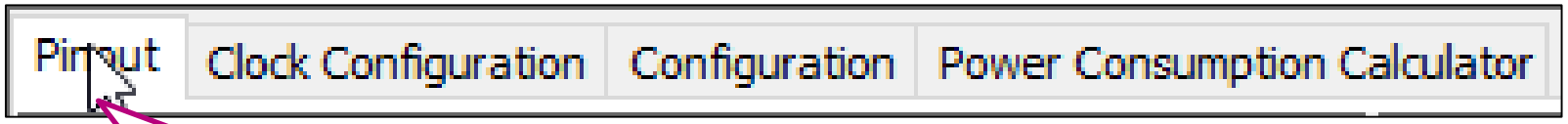
- Run CubeMX tool
- Start new project
  - Click “New Project” desktop shortcut, or
  - Go to “Menu->File->New Project”
- Filter:
  - Series: STM32L4
  - Line: STM32L4x6
  - Package: LQFP100
- Select: STM32L476VGTx



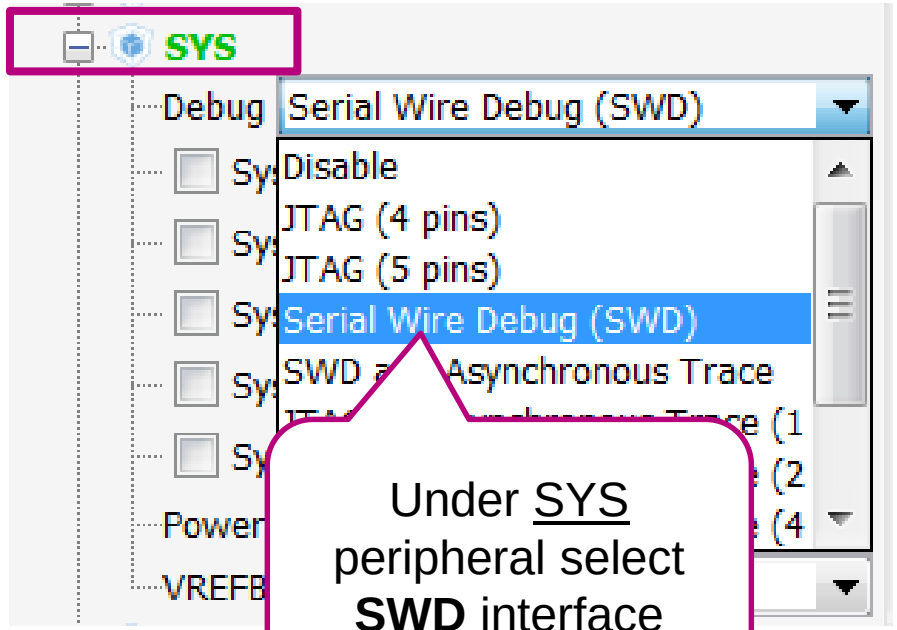

life.augmented

# 1 Configure debug interface





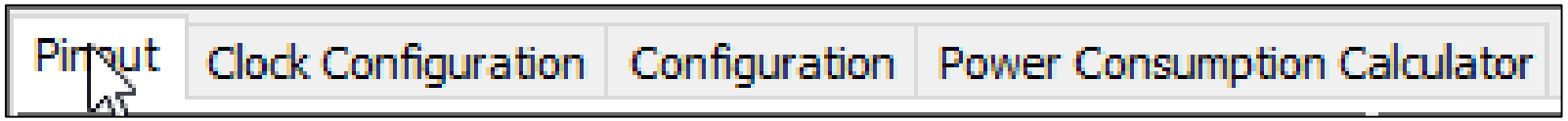

Go to Pinout settings

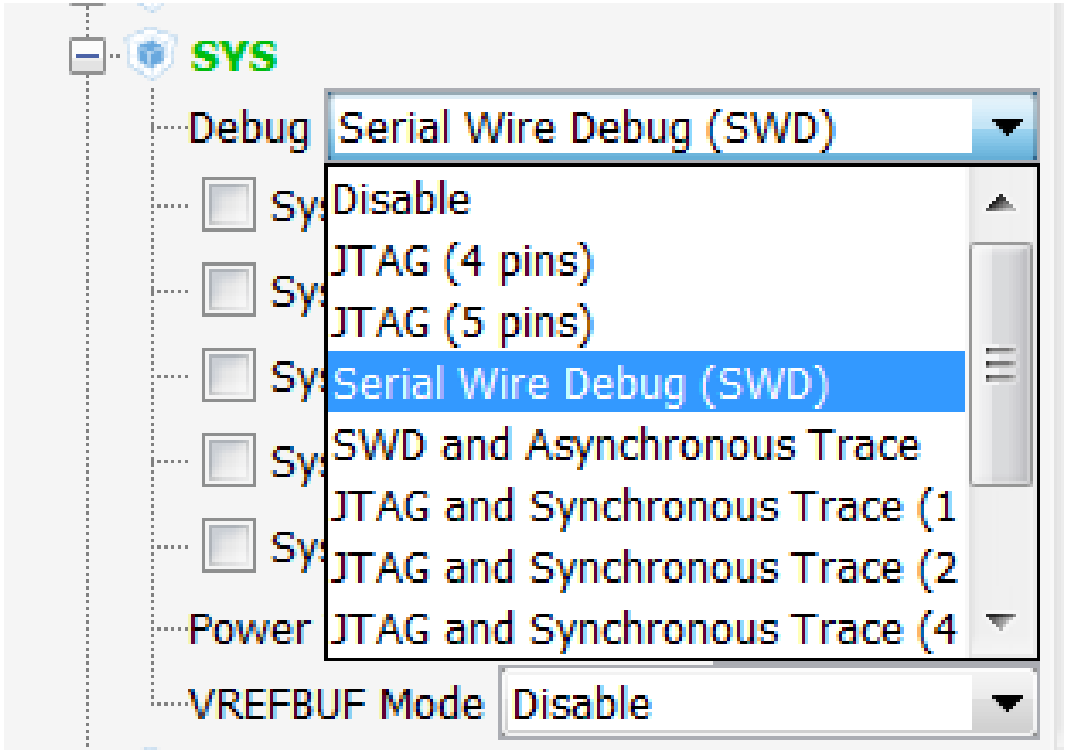


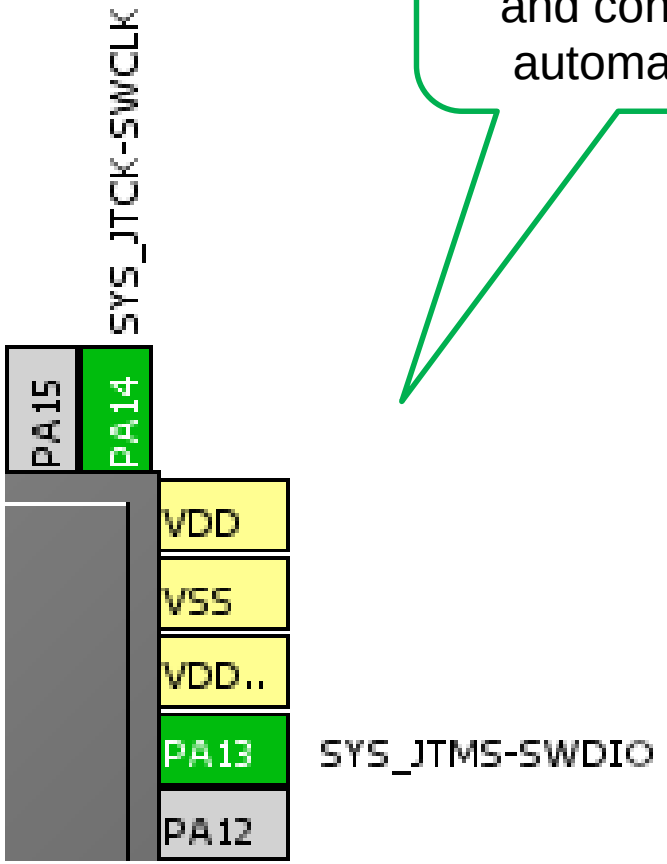

Under SYS peripheral select **SWD** interface

# 1 Configure debug interface





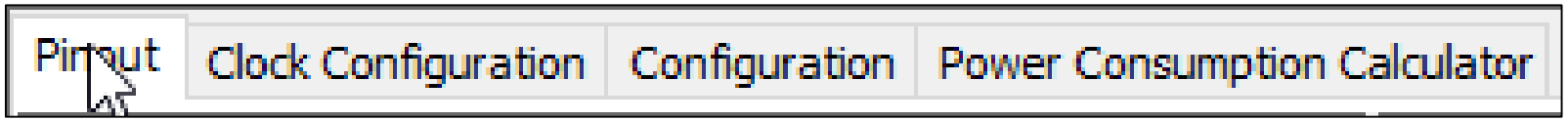

The corresponding pins are assigned and configured automatically!

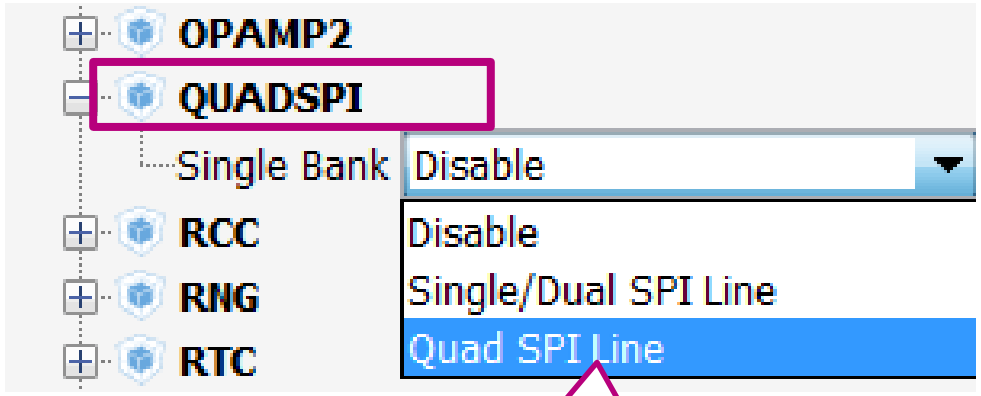
life.augmented

STM32 Cube

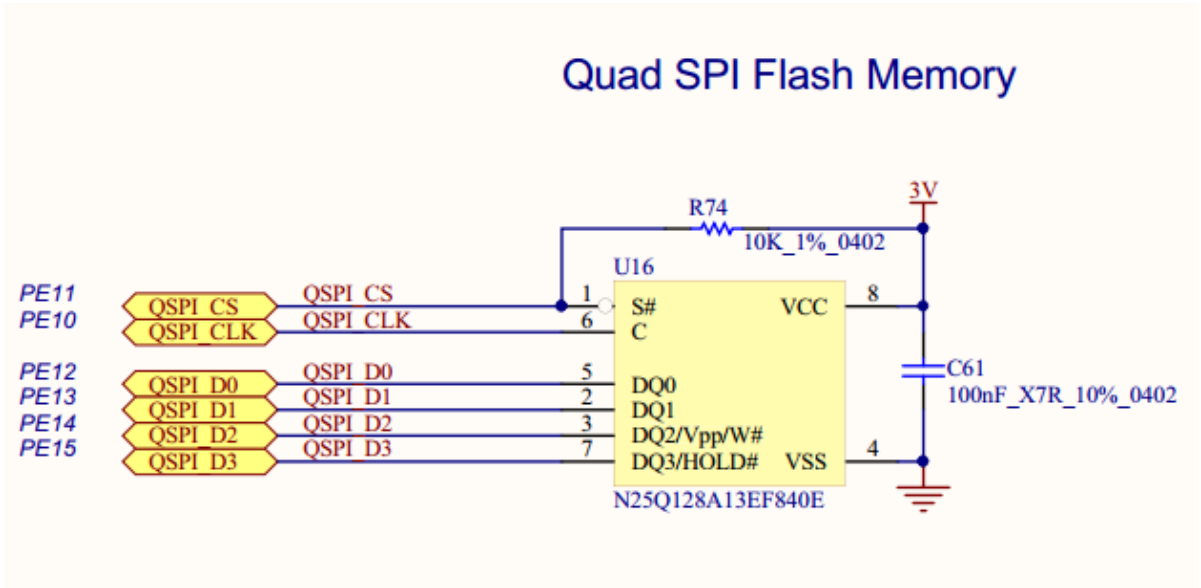
# 1 Configure QUADSPI interface





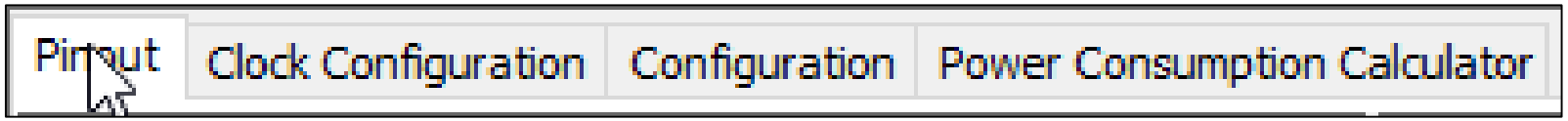




Under QUADSPI peripheral select **Quad SPI Line** configuration

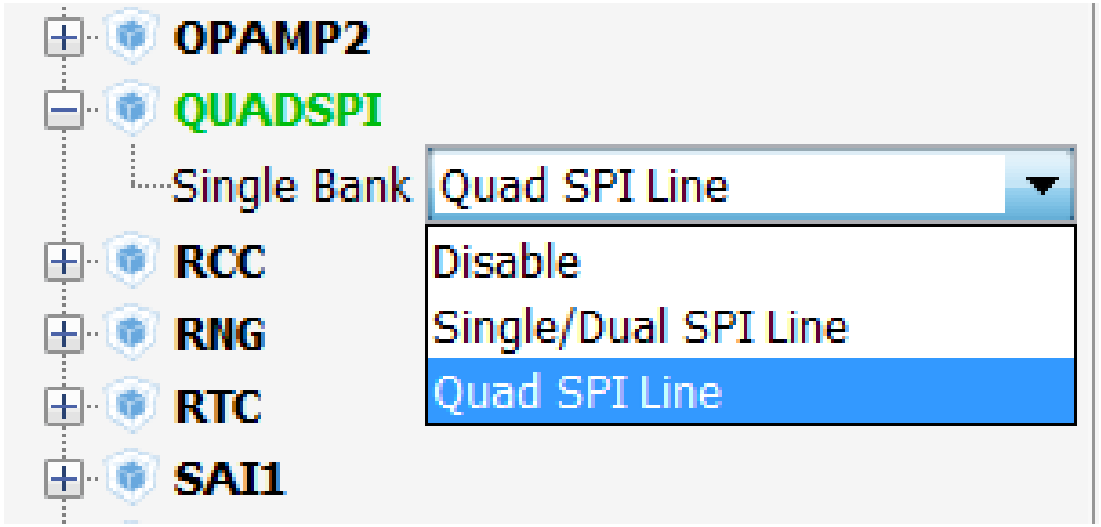


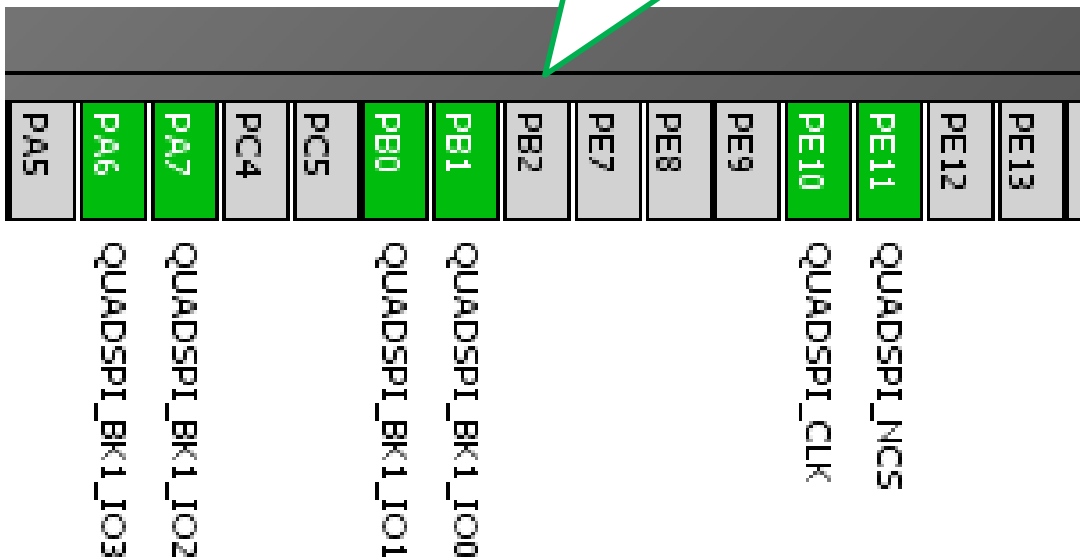
# 1 Configure QUADSPI interface






The corresponding pins are assigned and configured automatically!



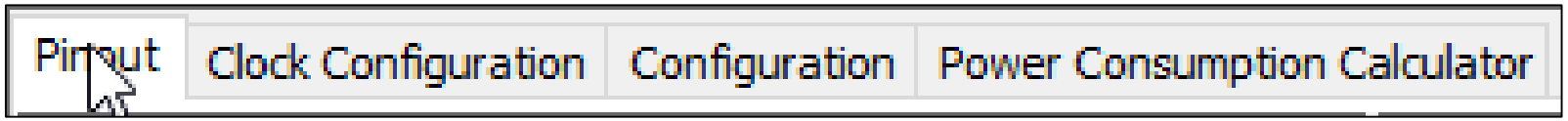



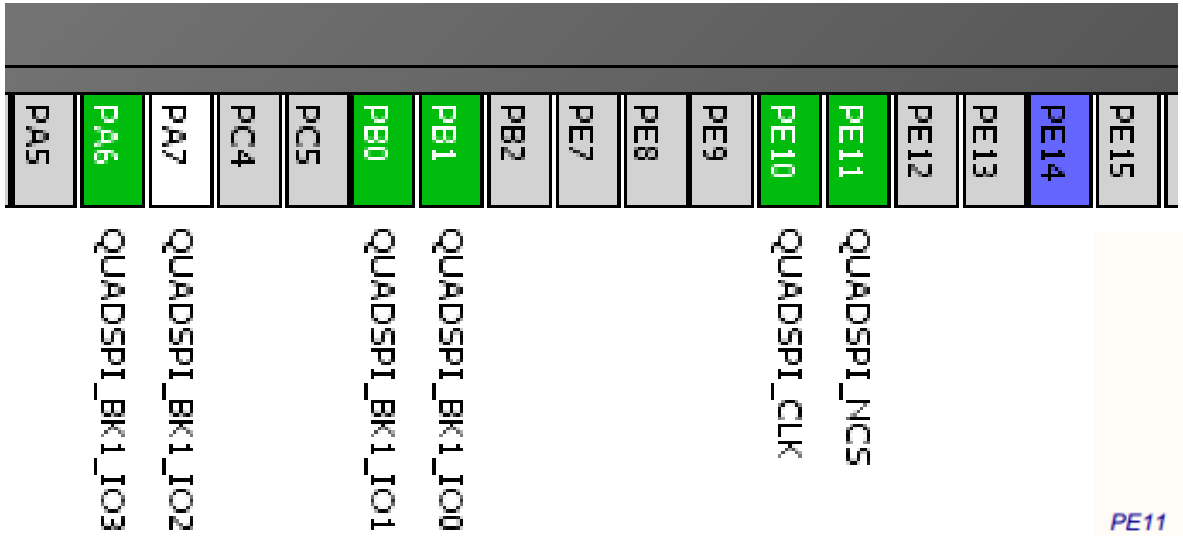

life.augmented

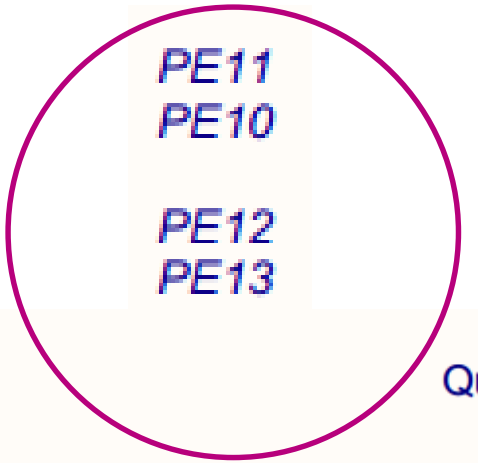
STM32 Cube

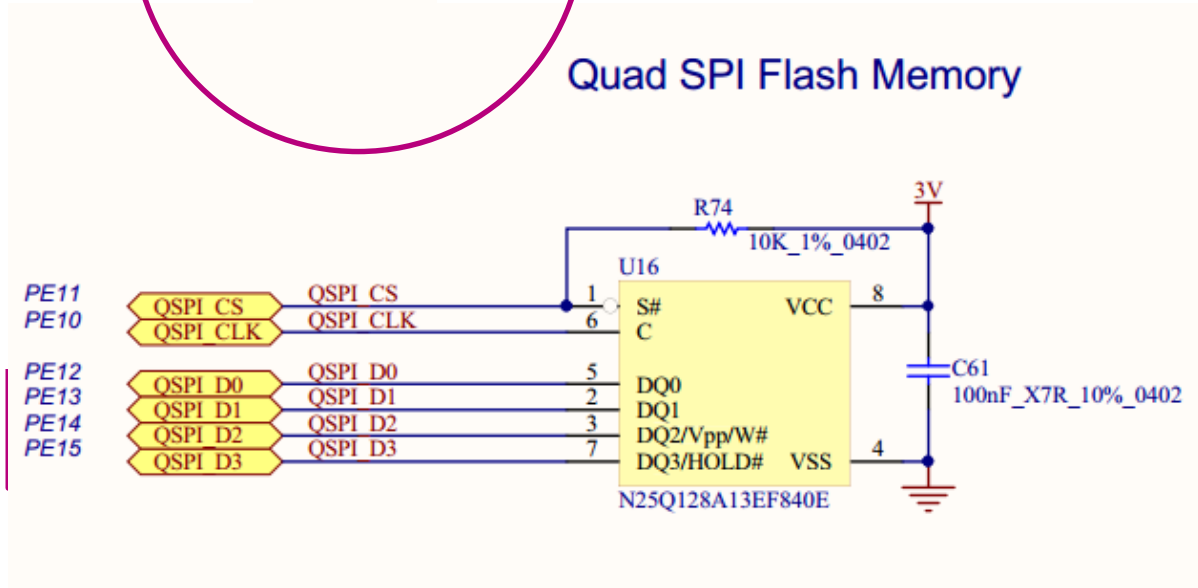
# 1 QUADSPI interface pins remapping





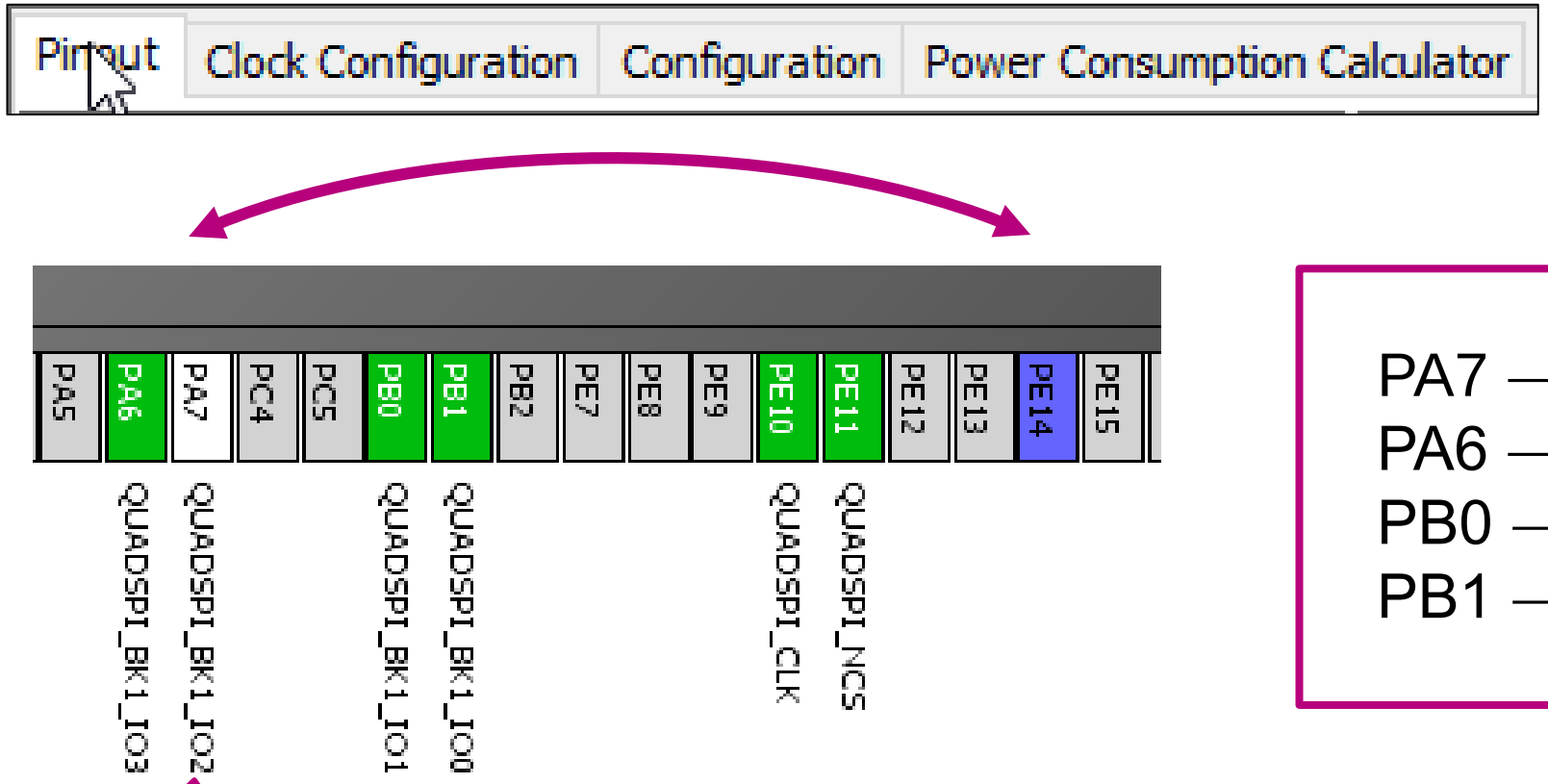




life.augmented

# 1 QUADSPI interface pins remapping






PA7 → PE14
PA6 → PE15
PB0 → PE13
PB1 → PE12

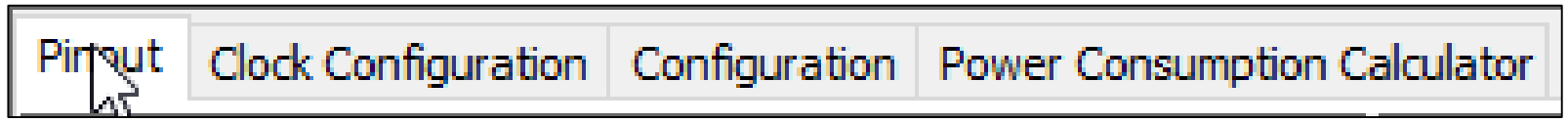
Hold CTRL key → click on the pin → move it to the other possible configuration

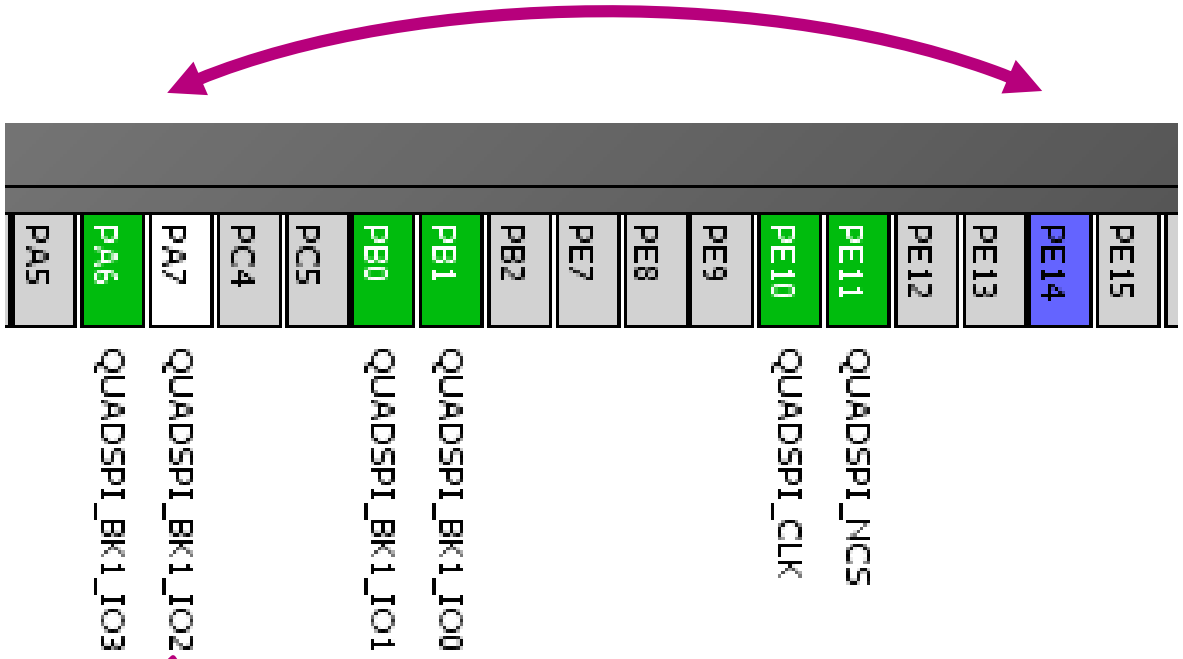
life.augmented

STM32 Cube

# 1 QUADSPI interface pins remapping

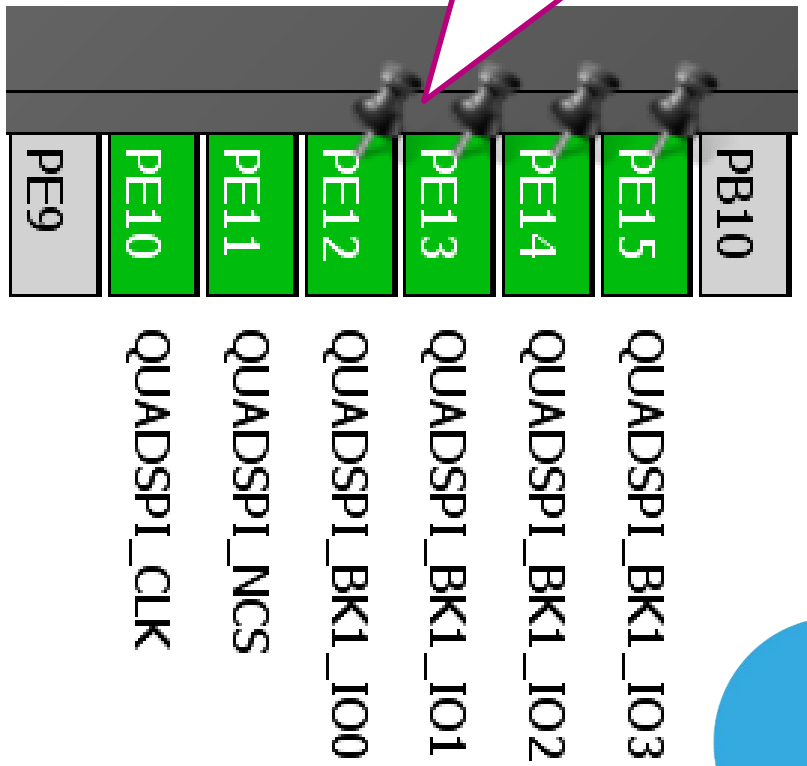





Hold CTRL key → click on the pin → move it to the other possible configuration

Desired configuration

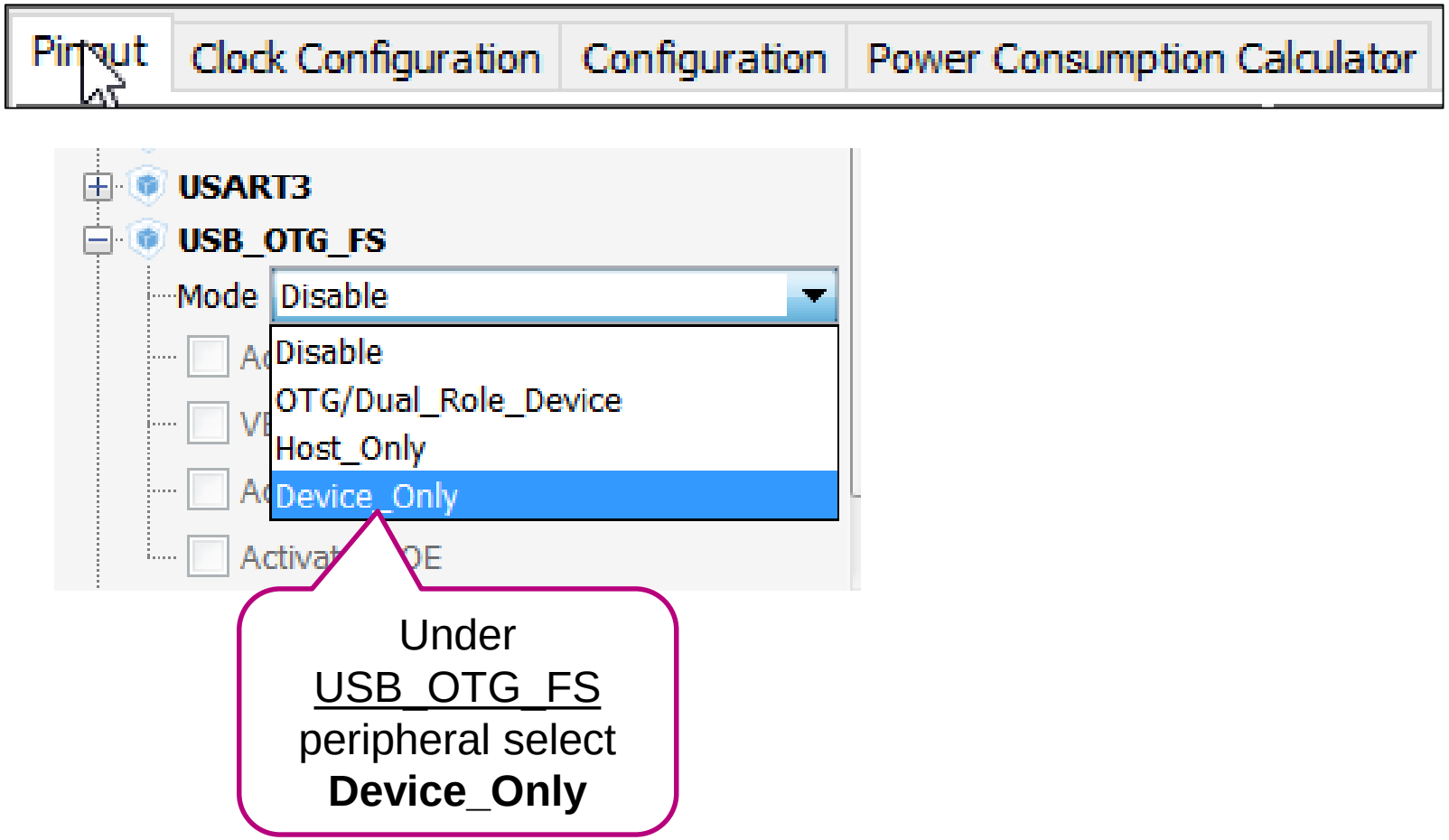



life.augmented

STM32 Cube

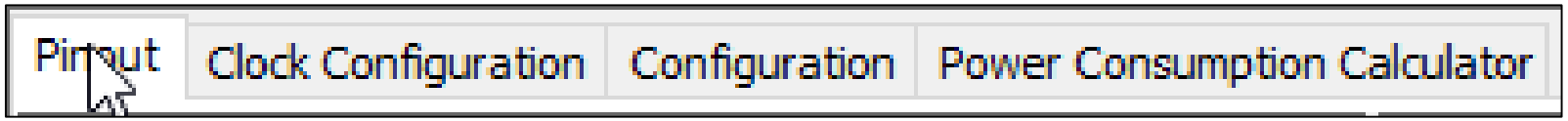
# 1 Configure USB_OTG_FS interface

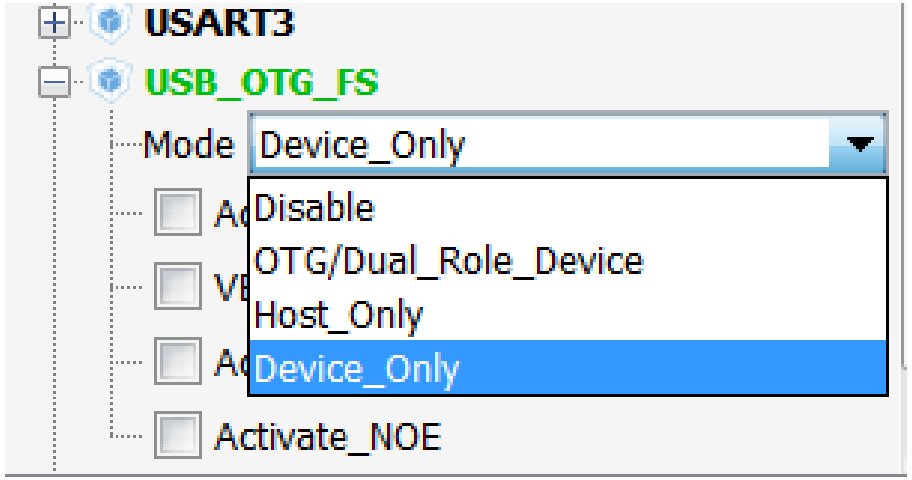




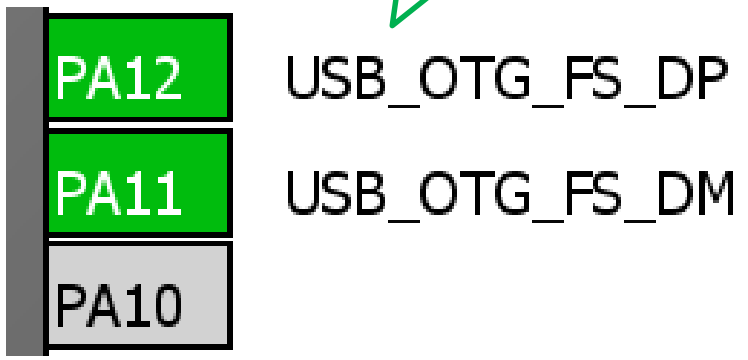
# 1 Configure USB_OTG_FS interface






The corresponding pins are assigned and configured automatically!

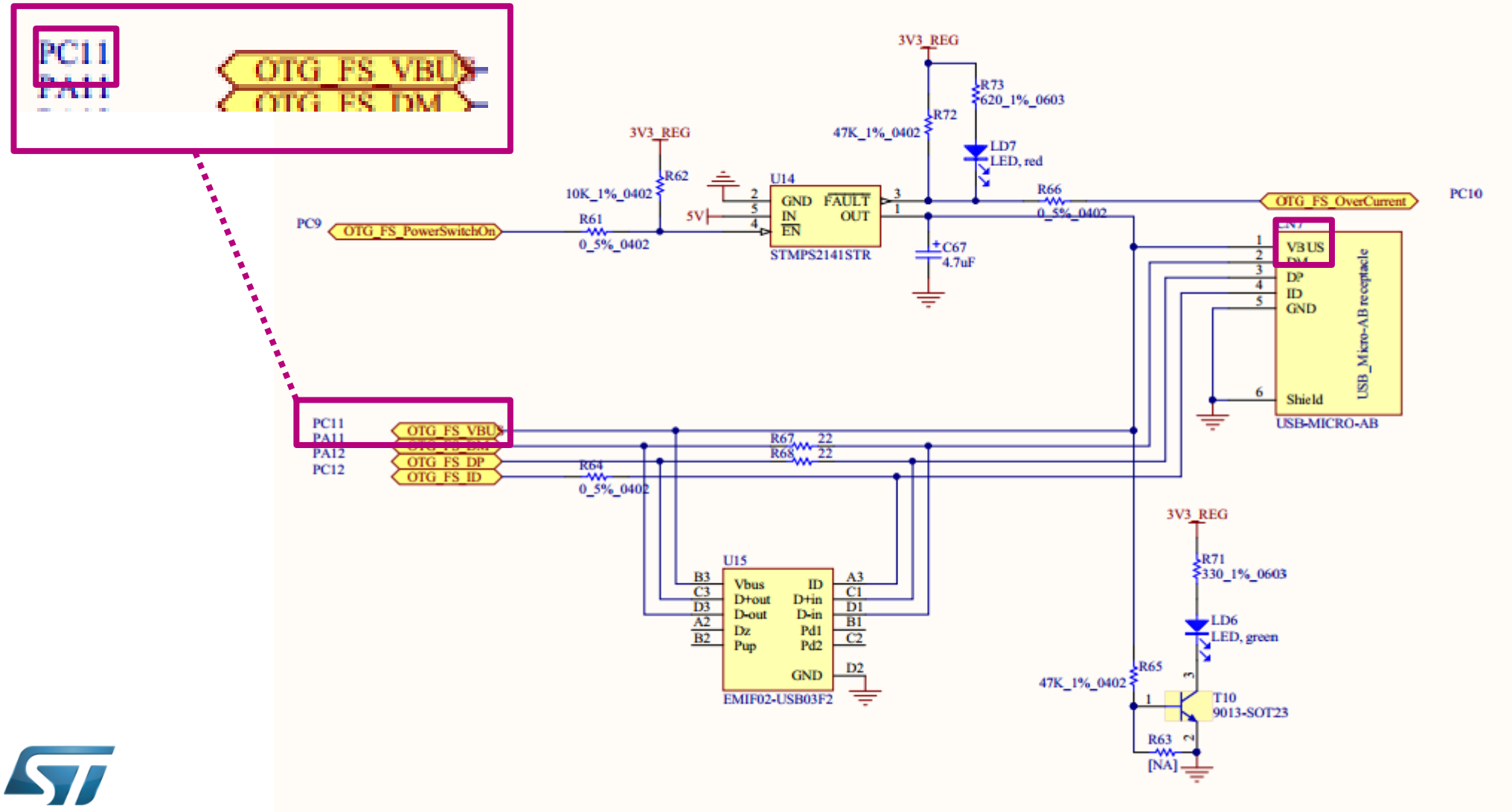



life.augmented

STM32 Cube

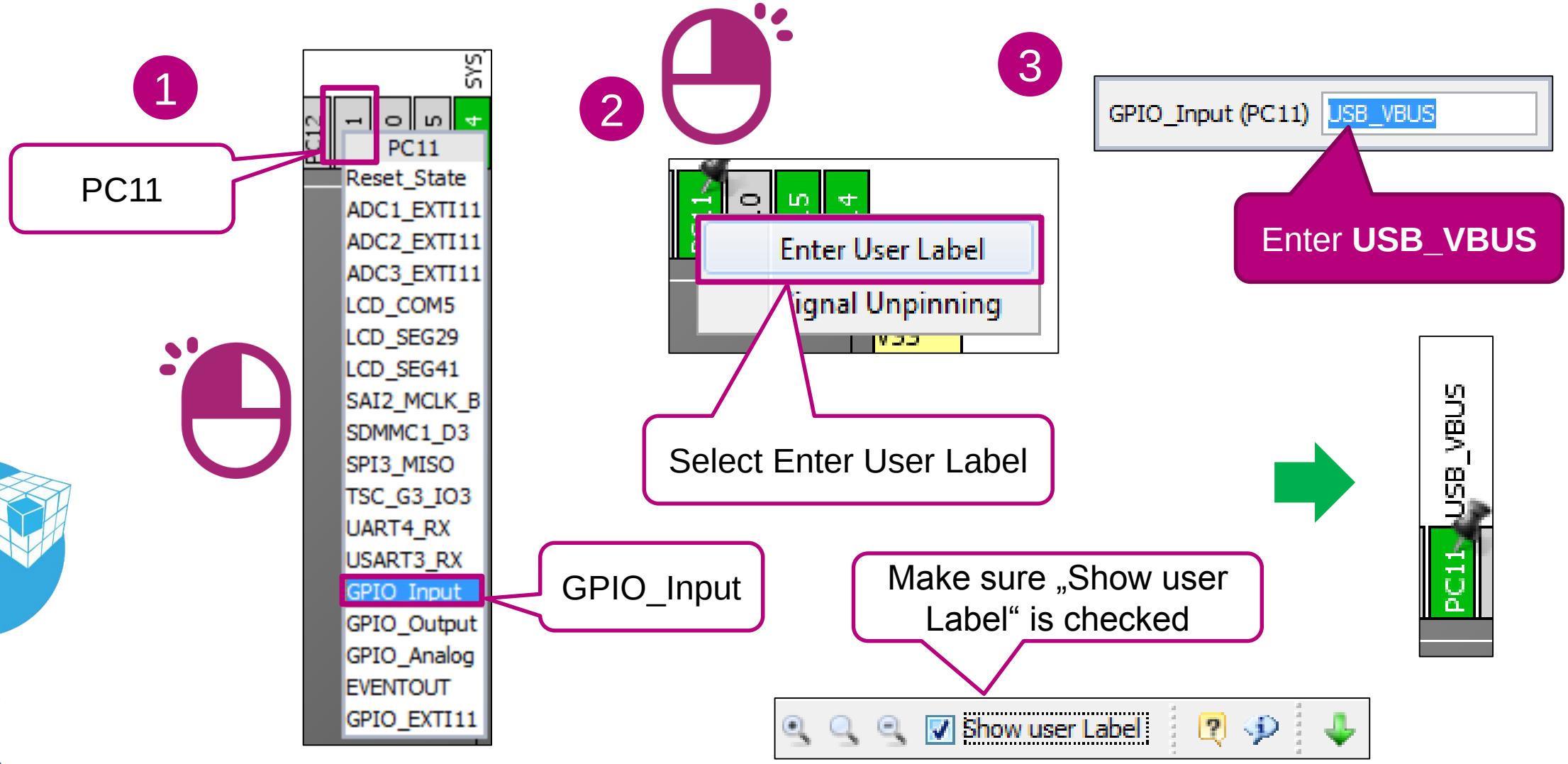
# 1 Configure USB_OTG_FS interface





# 1 Configure USB_OTG_FS interface



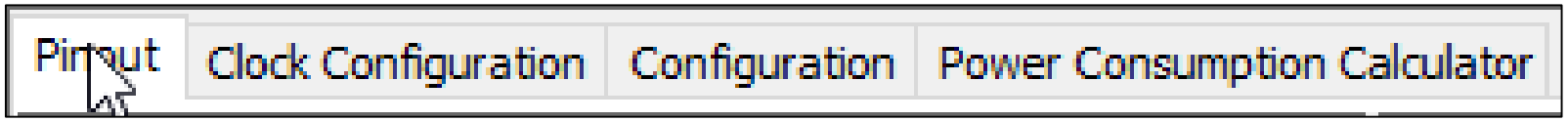


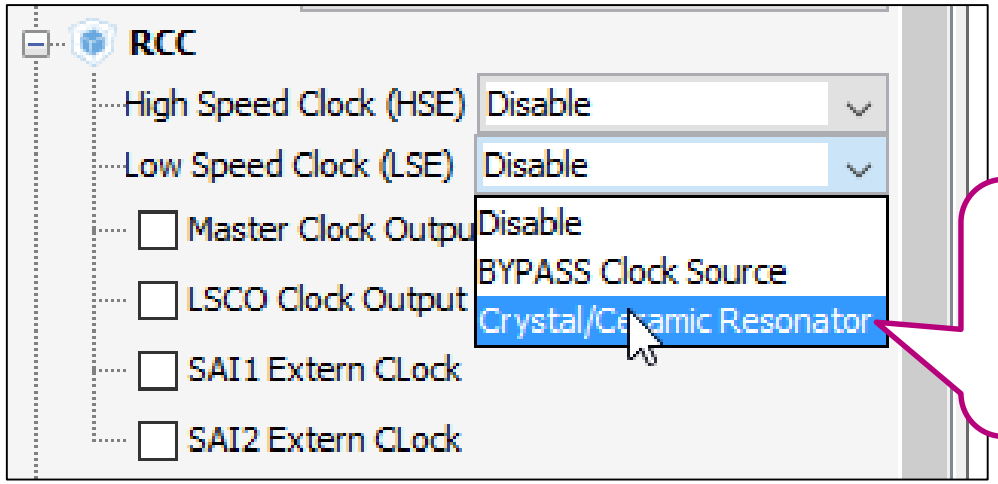

STM32 Cube

life.augmented

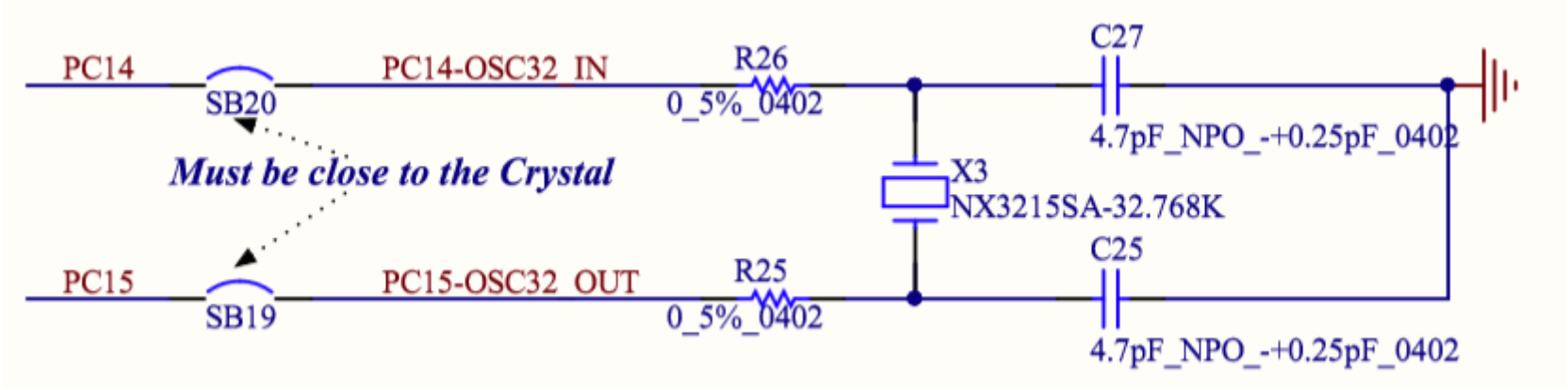
# 1 Configure LSE oscillator





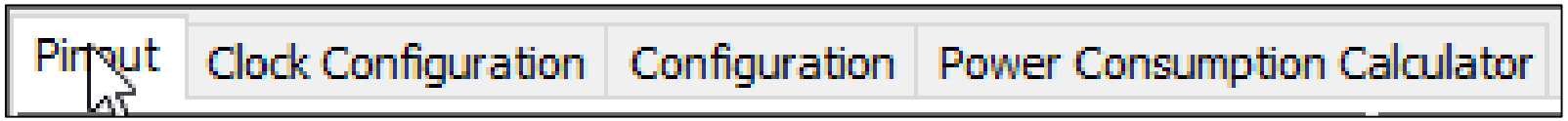



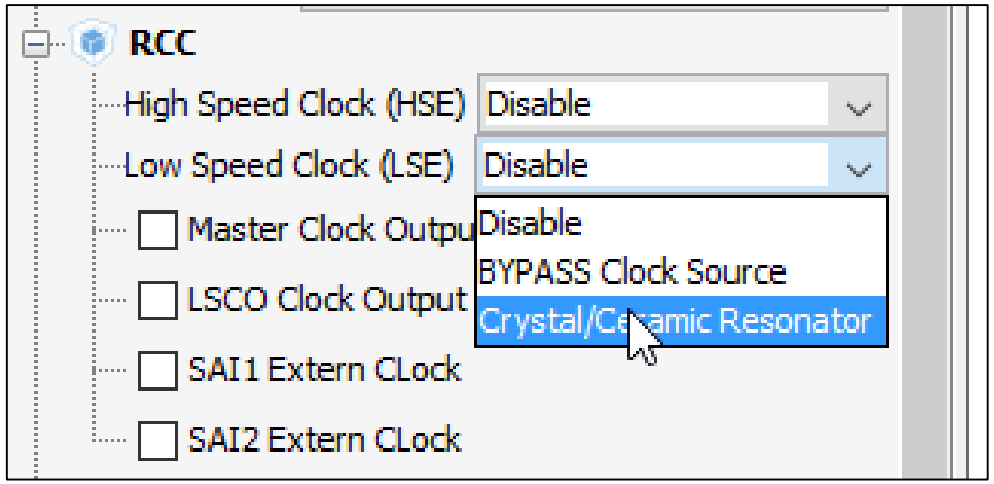

Under RCC Low Speed Clock (LSE)
select **Crystal/Ceramic Resonator**

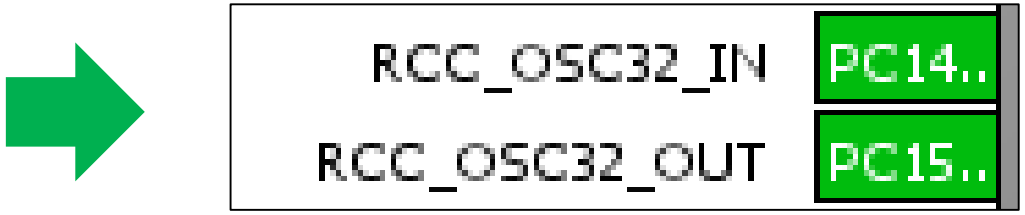


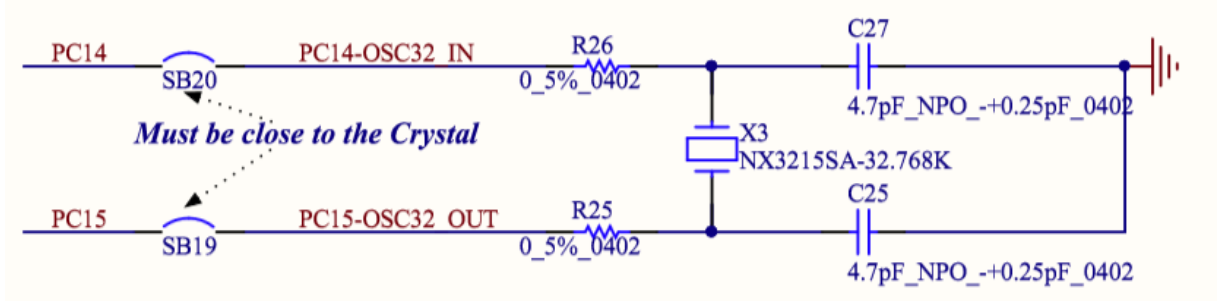
# 1 Configure LSE oscillator





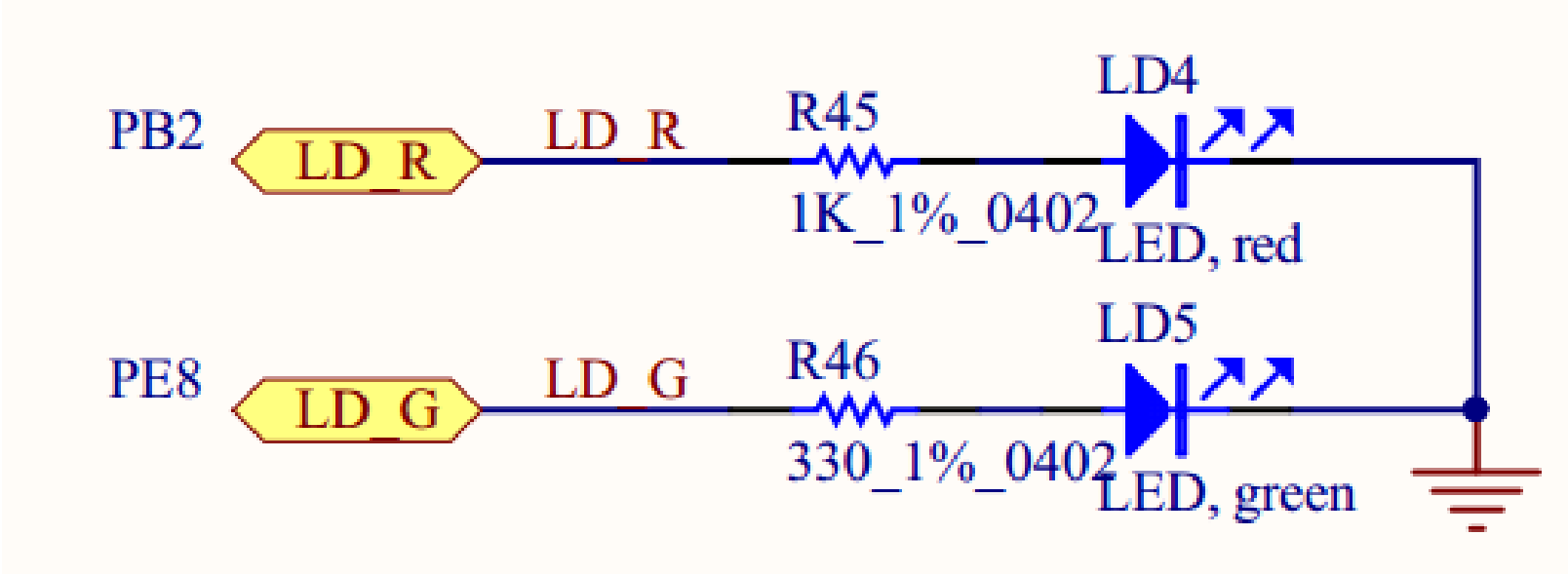




The corresponding pins are assigned and configured automatically!

life.augmented

# 1 Configure LED pins

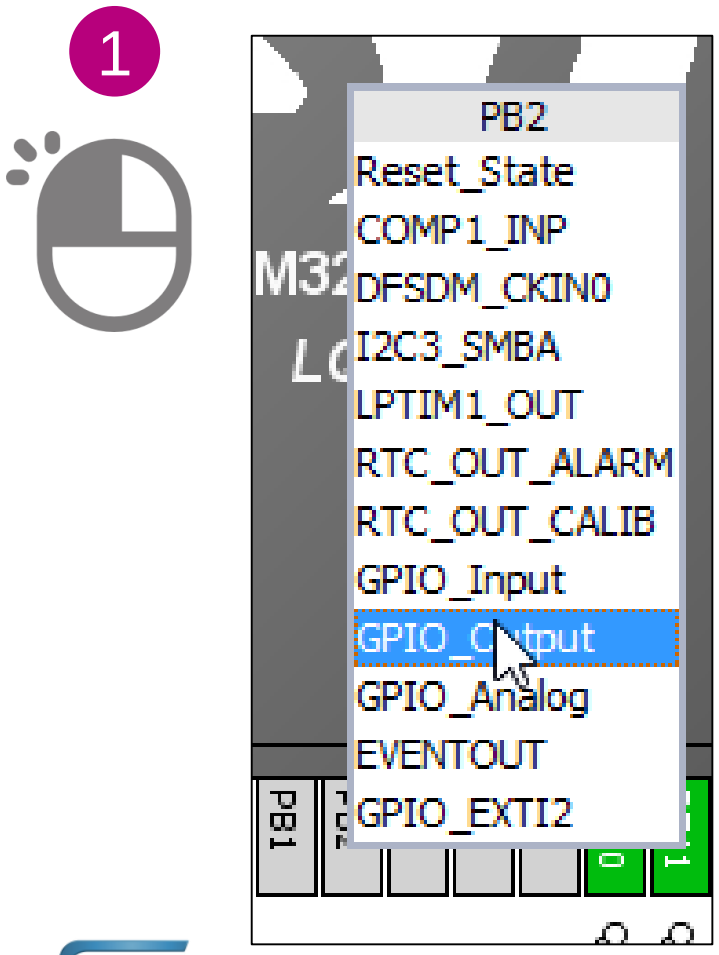


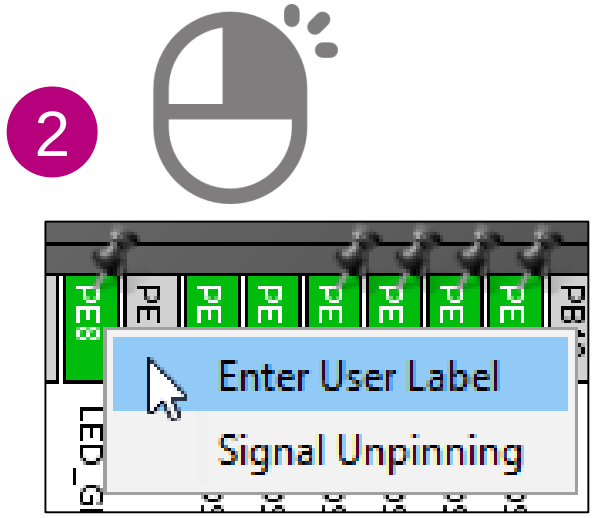


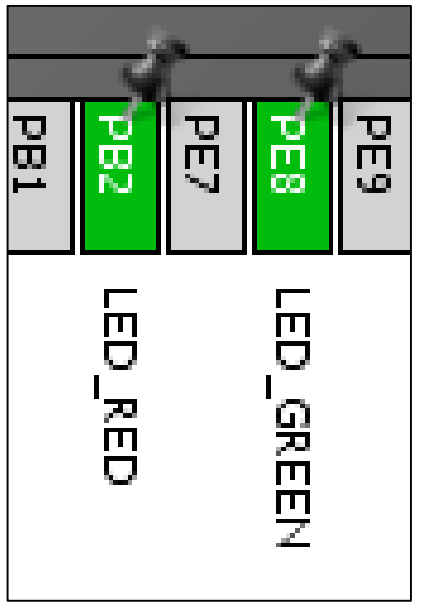
# 1

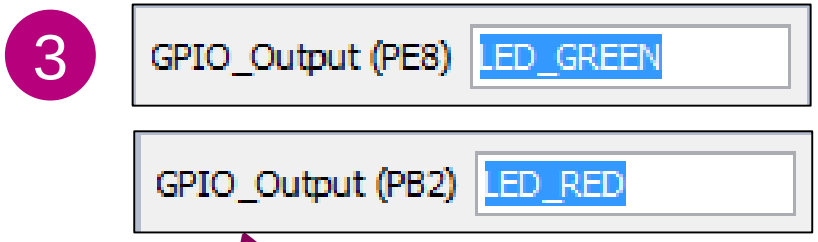
# Configure LED pins



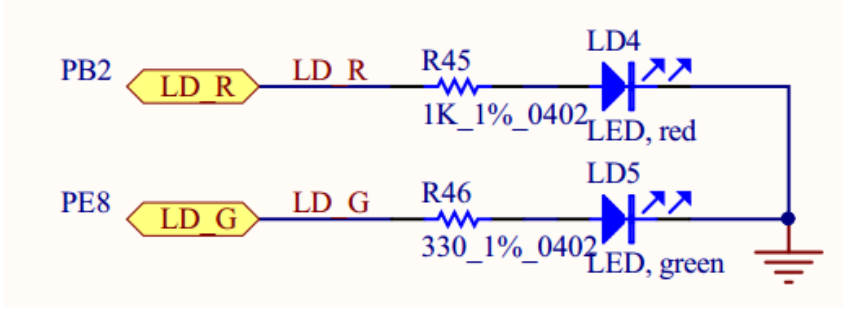



**LED_GREEN** for PE8
**LED_RED** for PB2



# Checkpoint #1



1

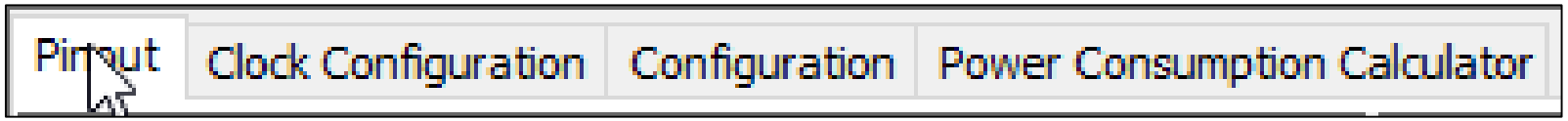


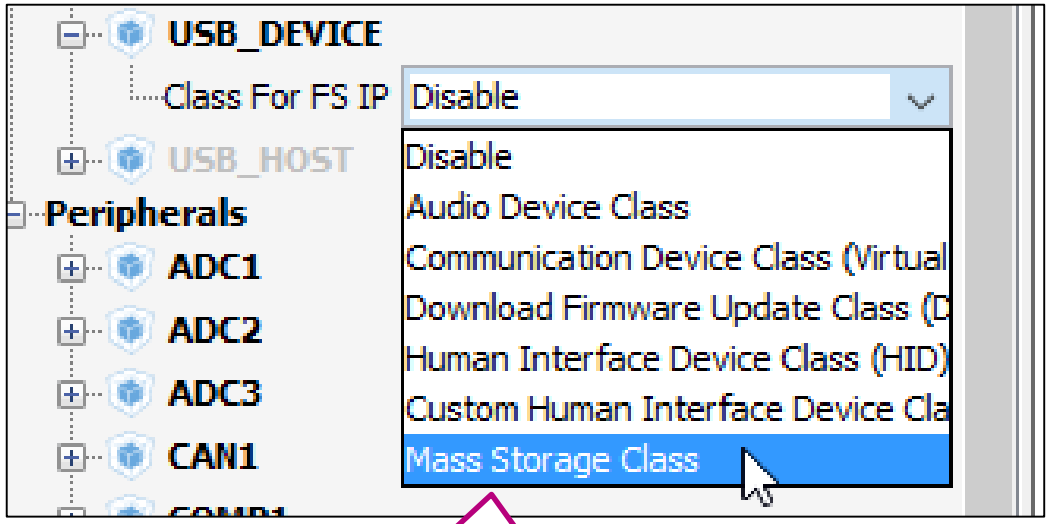

life.augmented

STM32 Cube

# 1 Configure USB_DEVICE stack





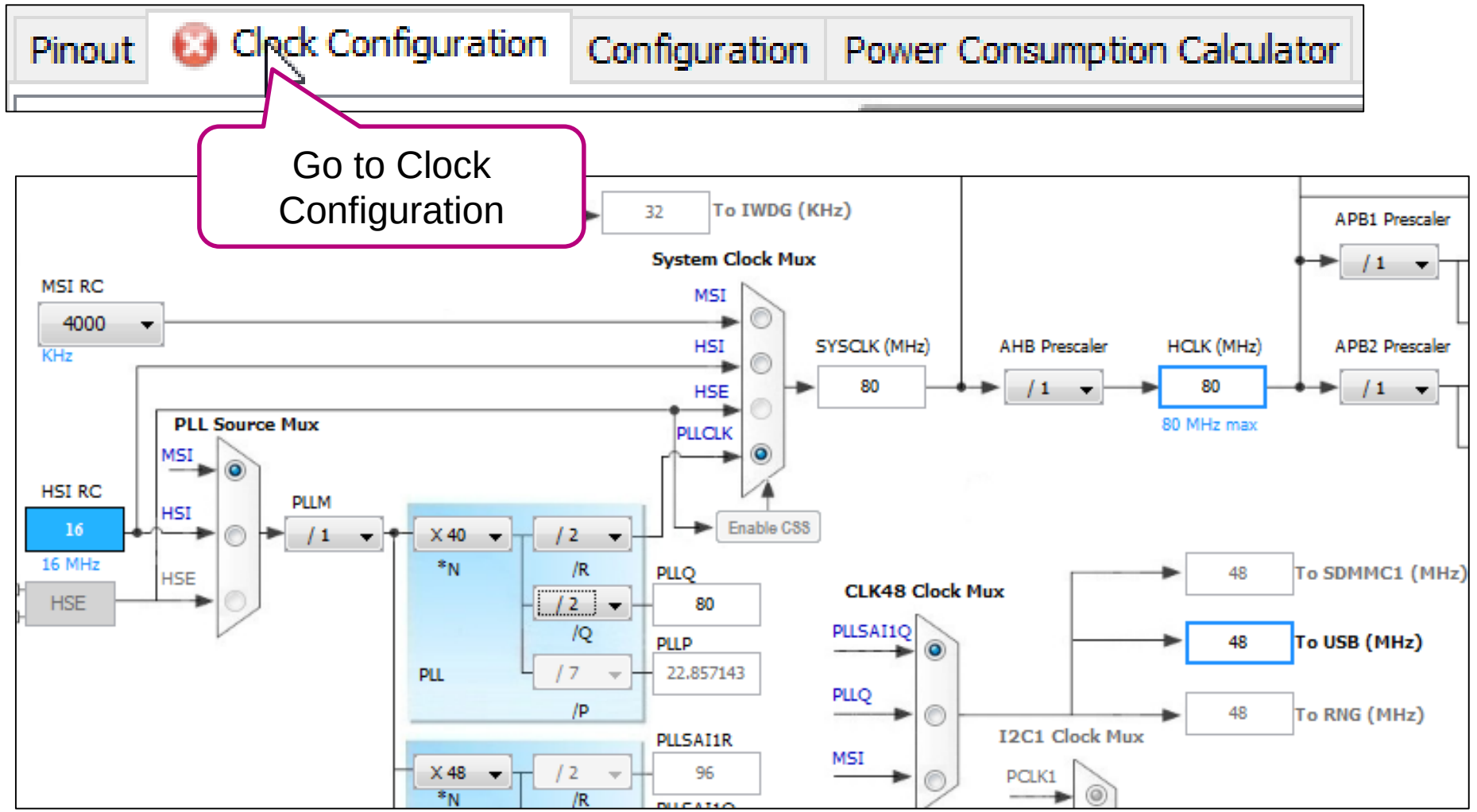

Under USB_DEVICE MiddleWare select **Mass Storage Class**

life.augmented

STM32 Cube

# 1 System Clock Configuration



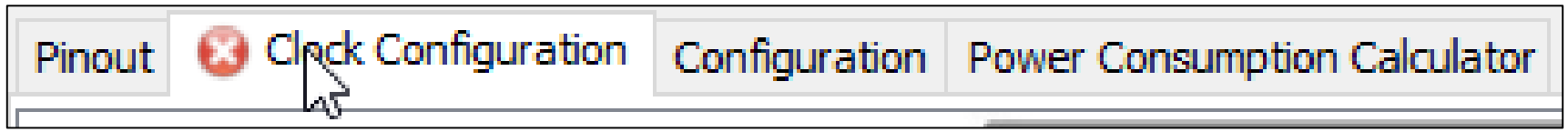



life.augmented

STM32 Cube

# System Clock Configuration






4000 kHz
(MSI RC)

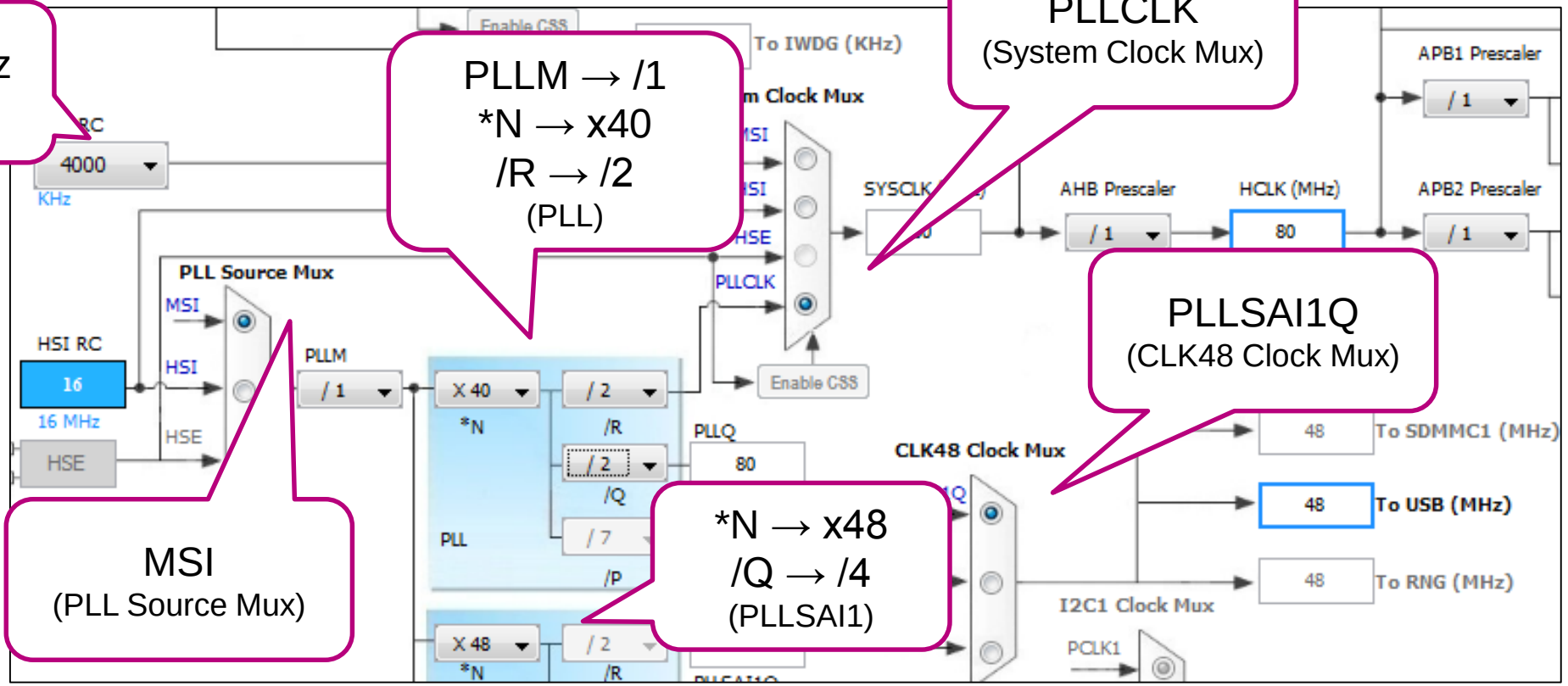
PLLM → /1
*N → x40
/R → /2
(PLL)

PLLCLK
(System Clock Mux)

PLLSAI1Q
(CLK48 Clock Mux)

MSI
(PLL Source Mux)

*N → x48
/Q → /4
(PLLSAI1)

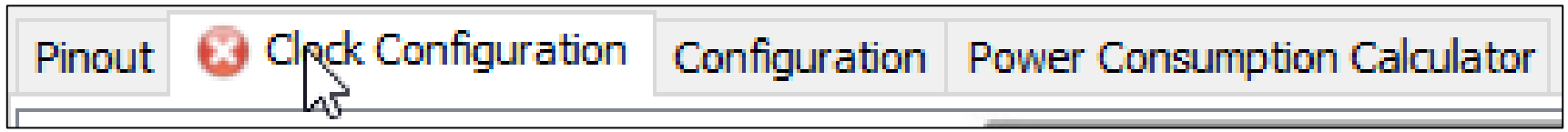


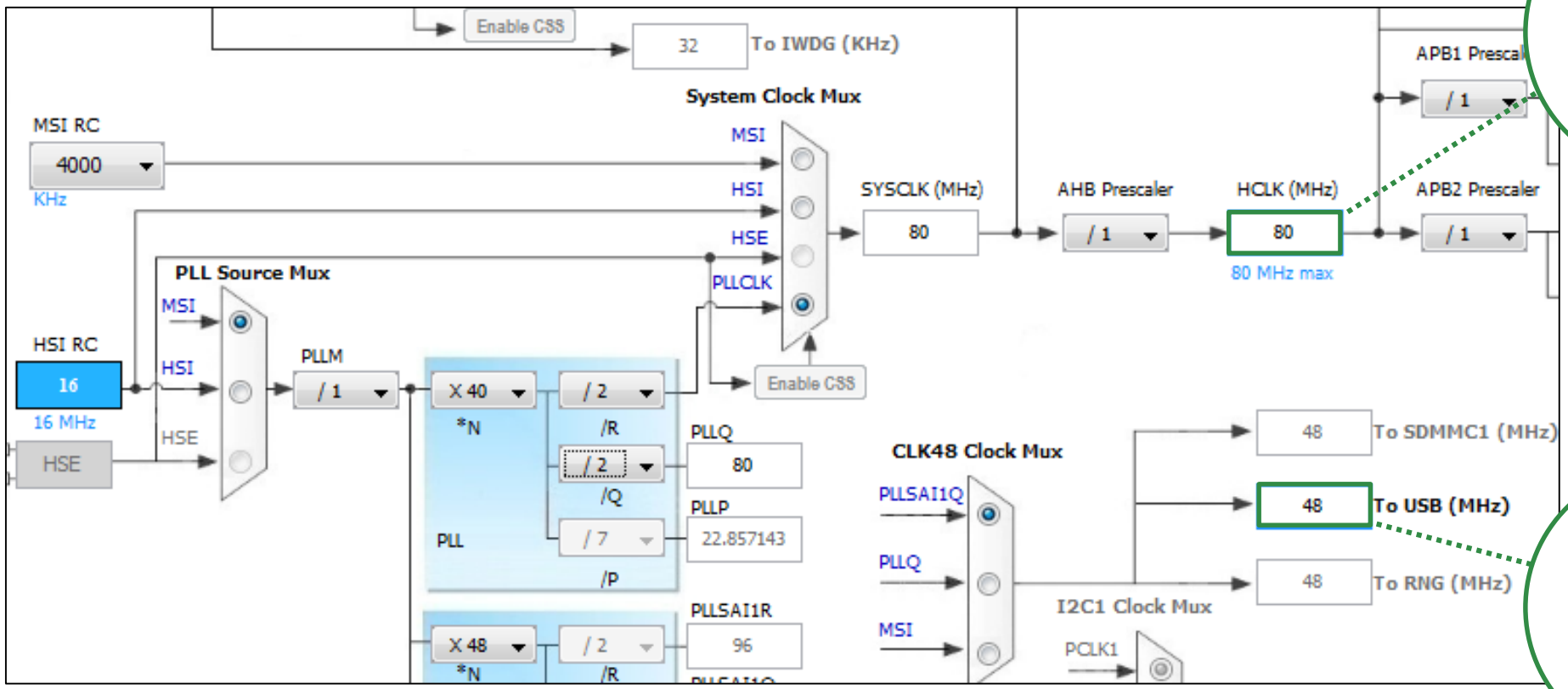

life.augmented

STM32 Cube

# Checkpoint #2






80MHz
(HCLK)

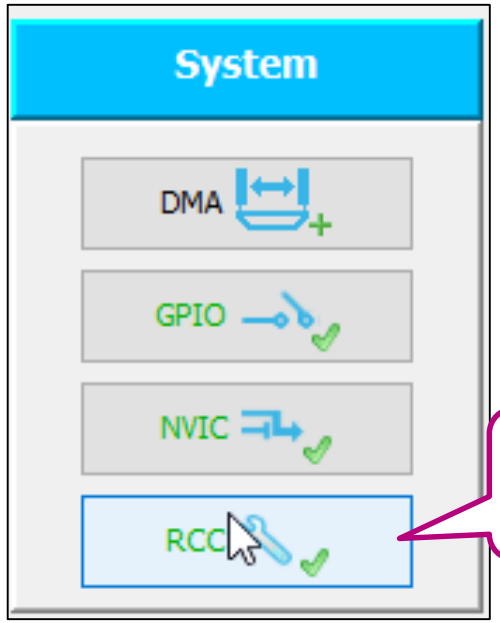
48MHz
(To USB)

life.augmented

# 1 RCC Configuration check



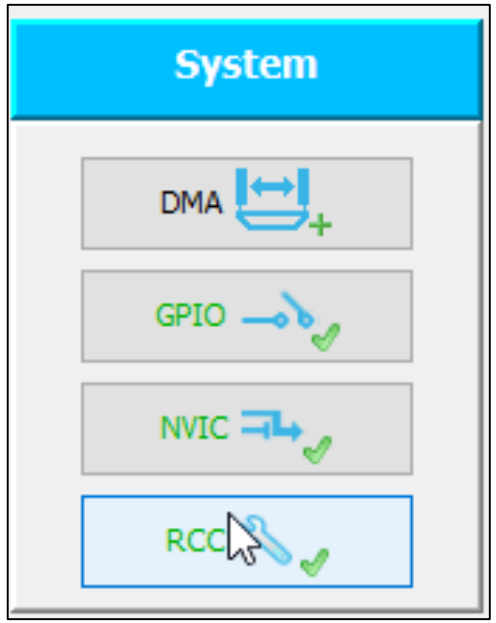

Go to Configuration

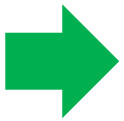


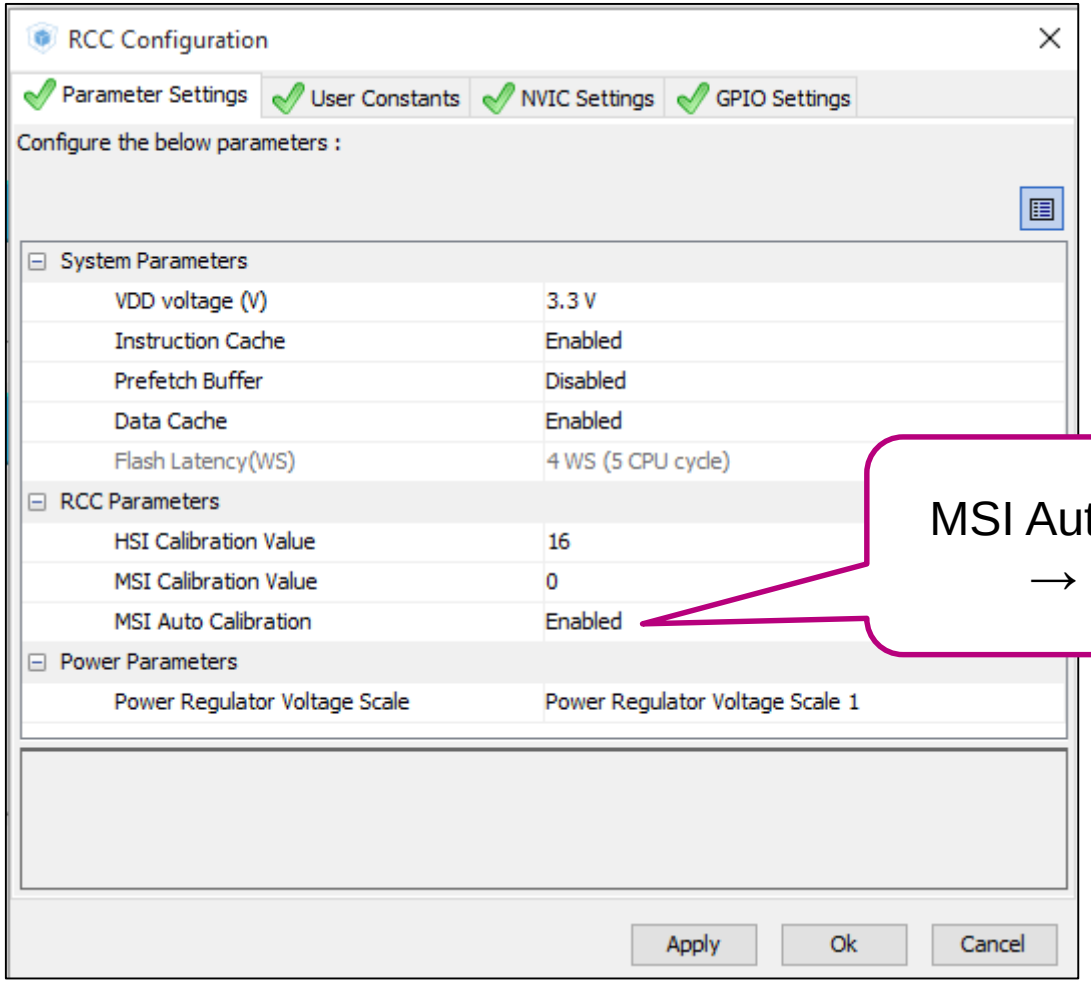

RCC in System

# 1 RCC Configuration check





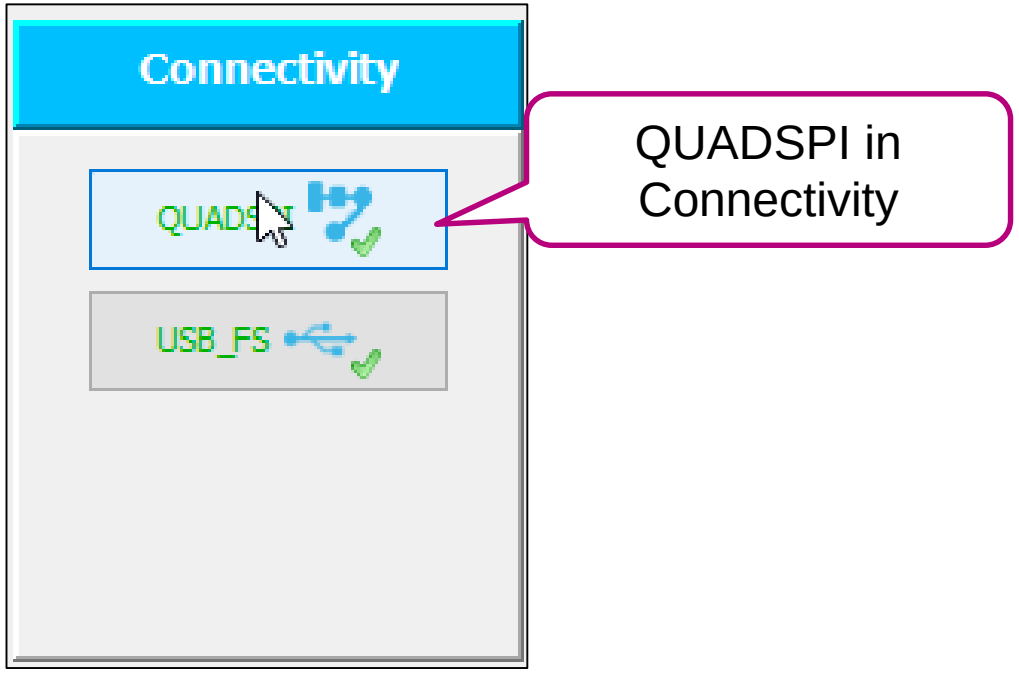

MSI Auto Calibration
→ Enabled

life.augmented

STM32 Cube

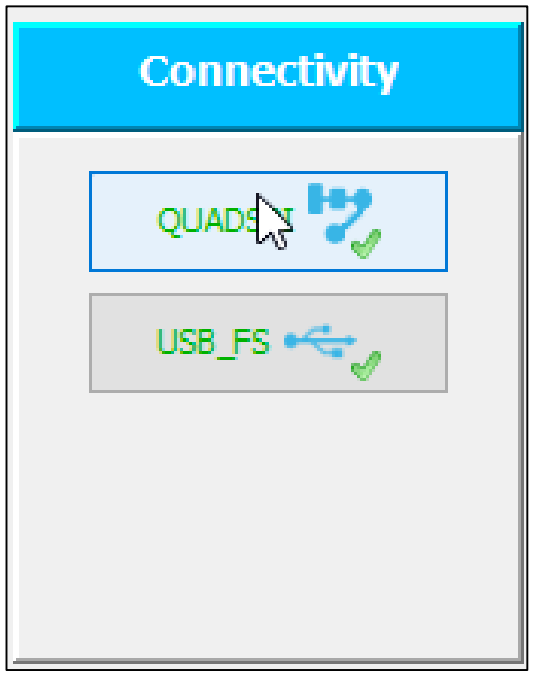
# 1 QUADSPI Configuration

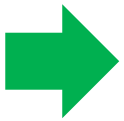


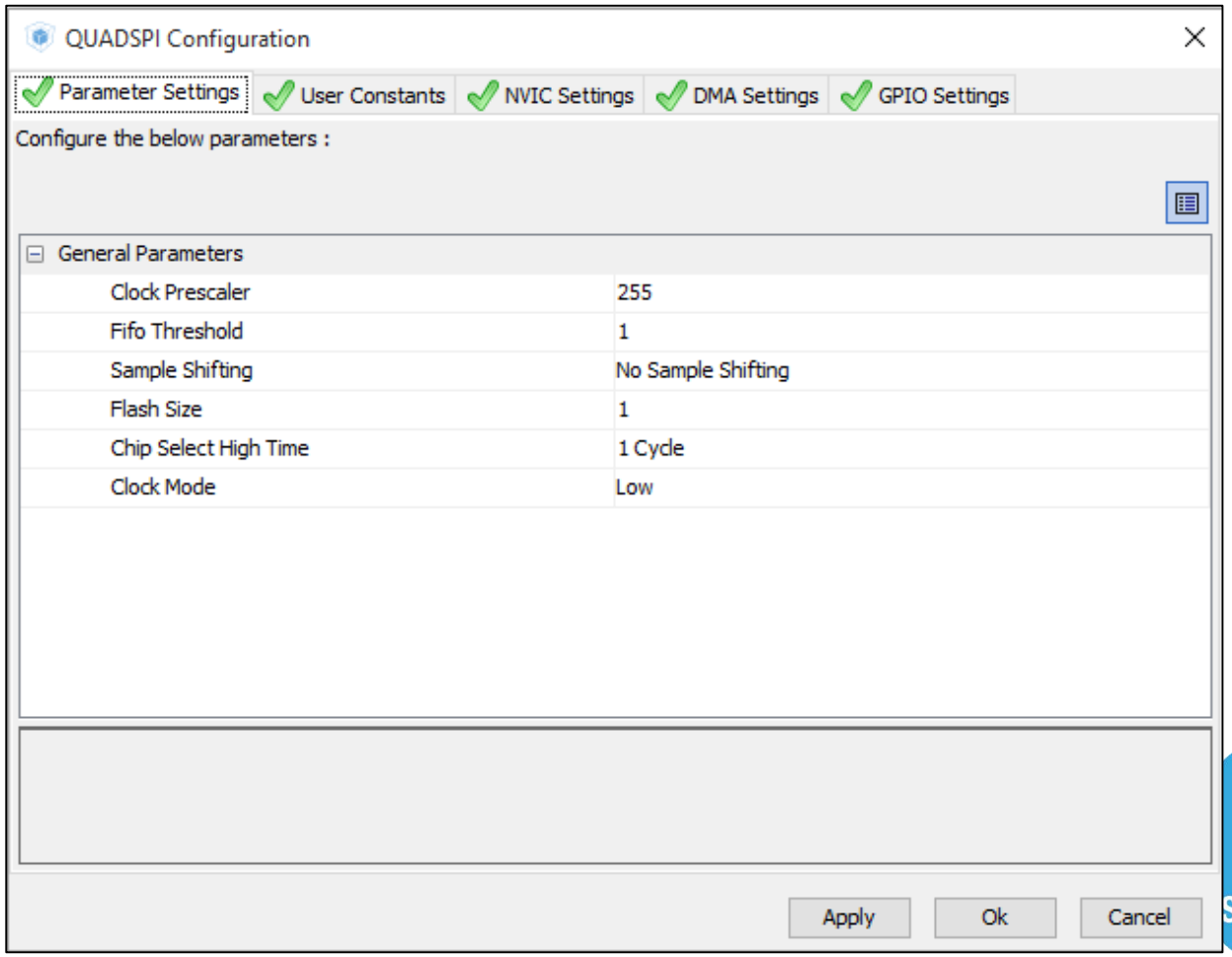



# QUADSPI Configuration



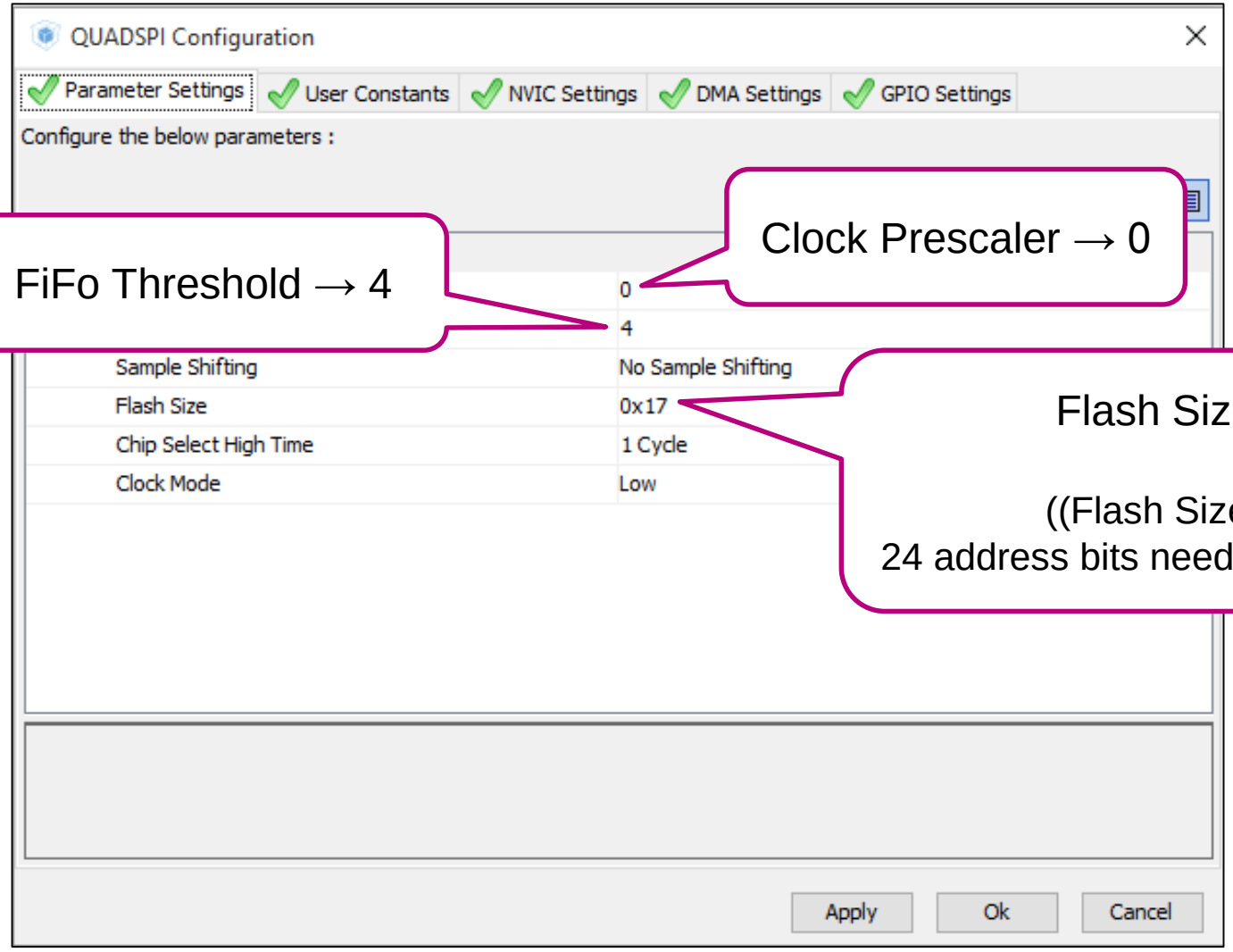



life.augmented

STM32 Cube

# QUADSPI Configuration






FiFo Threshold → 4

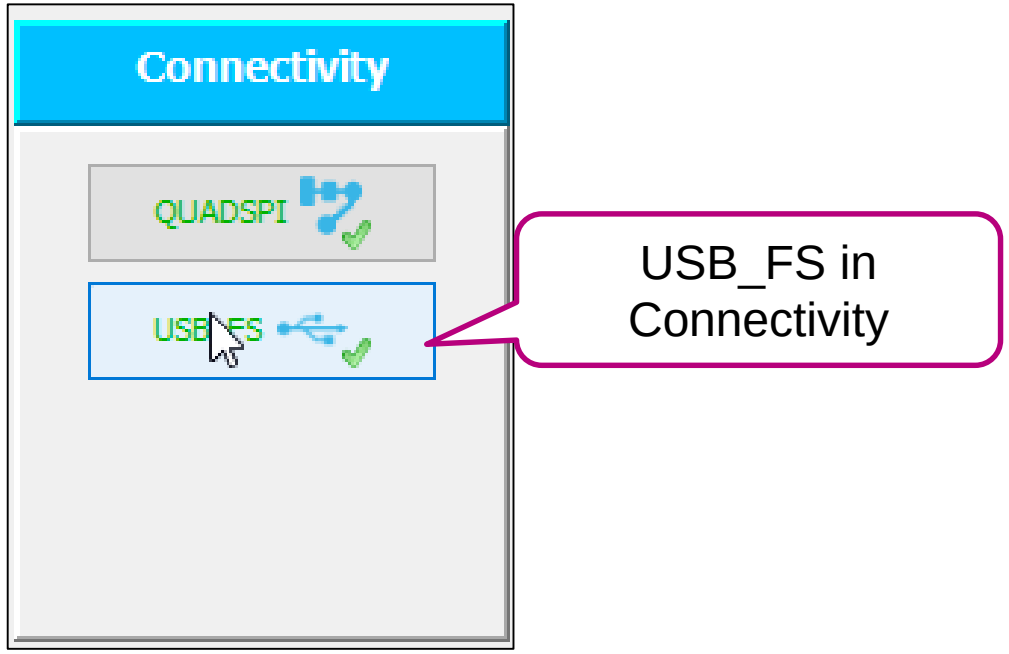
Clock Prescaler → 0

Flash Size → 0x17

((Flash Size) + 1) = 24
24 address bits needed for 16MB memory

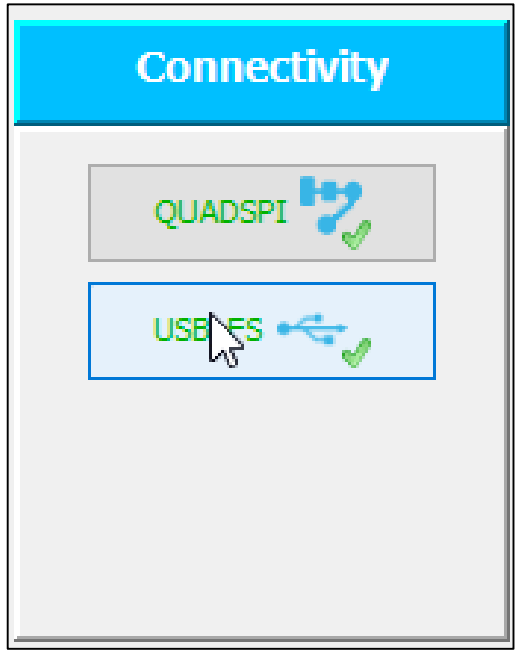
# 1 USB_OTF_FS Configuration

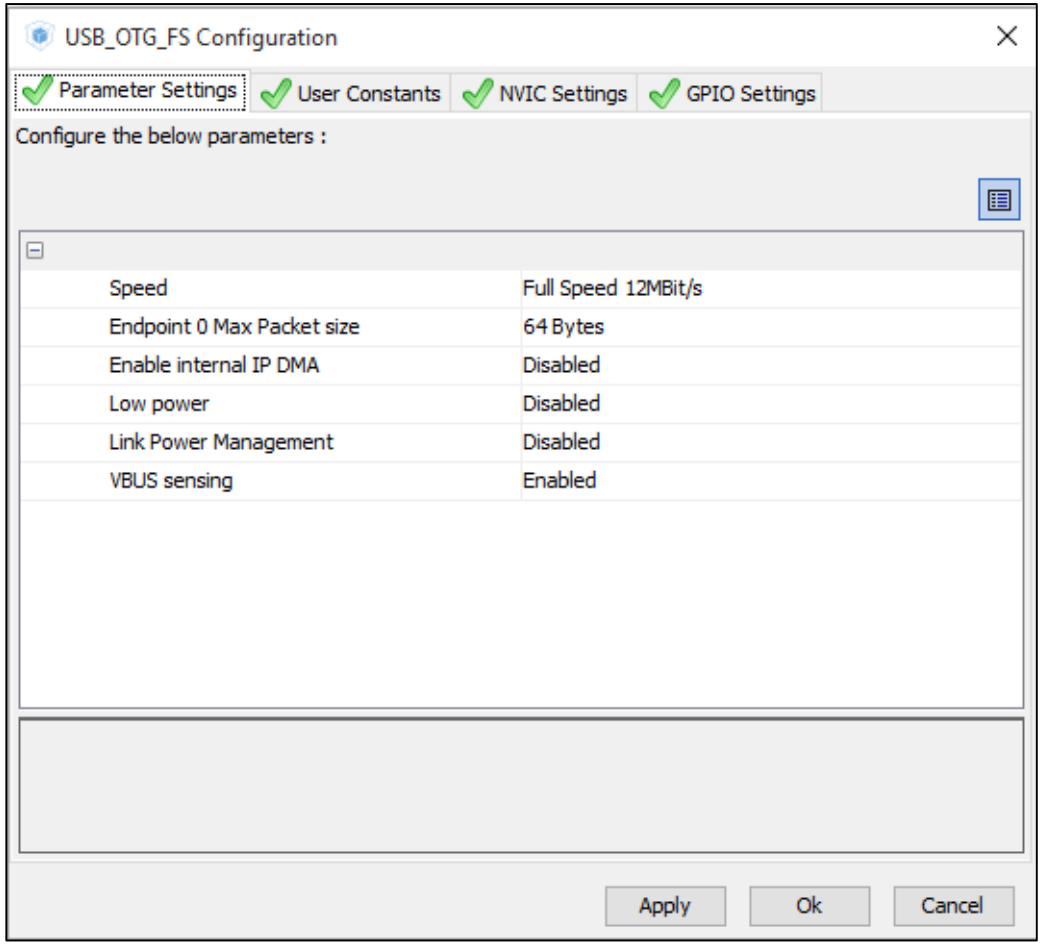


# 1 USB_OTF_FS Configuration



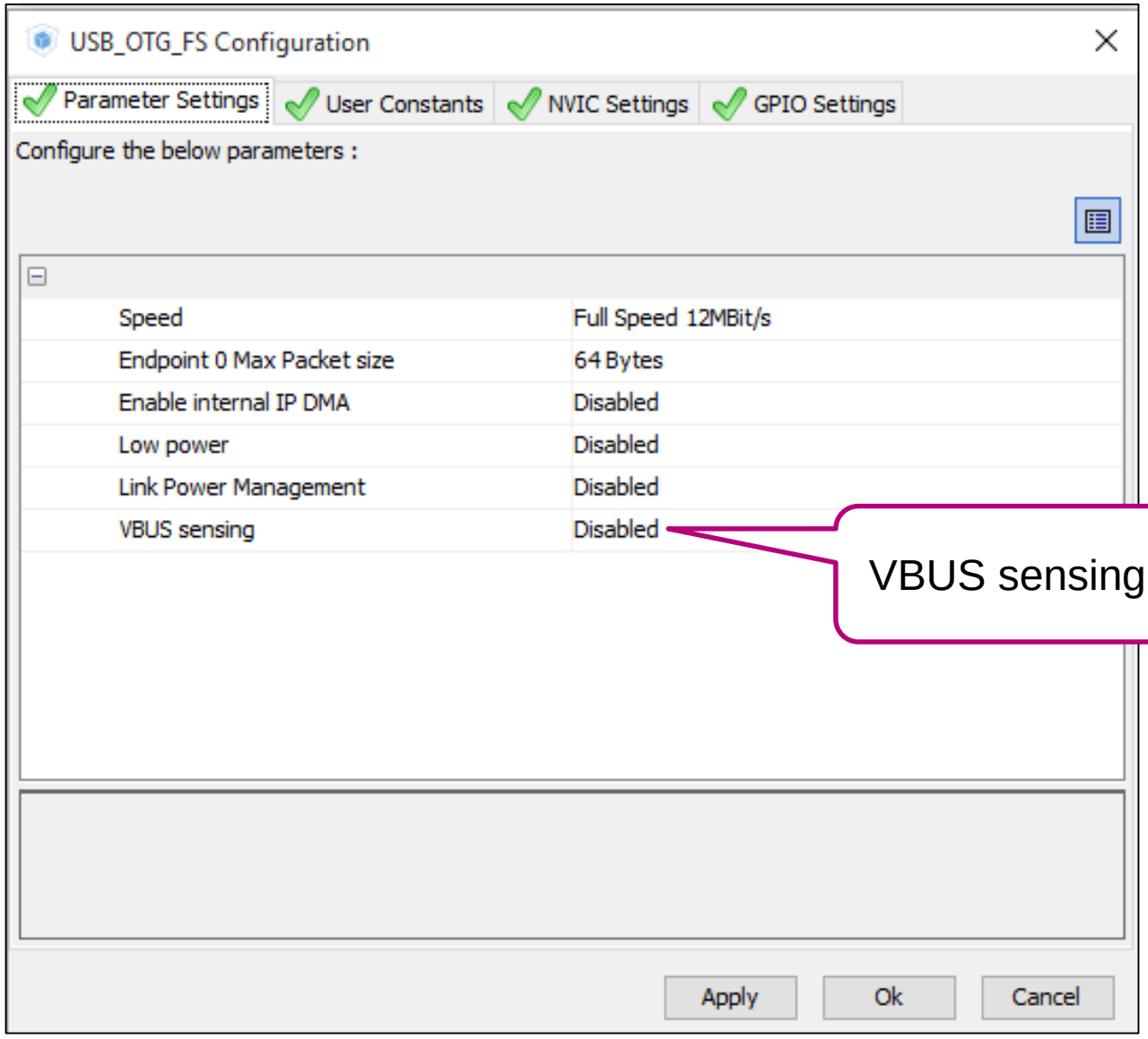




life.augmented

STM32 Cube

# USB_OTG_FS configuration

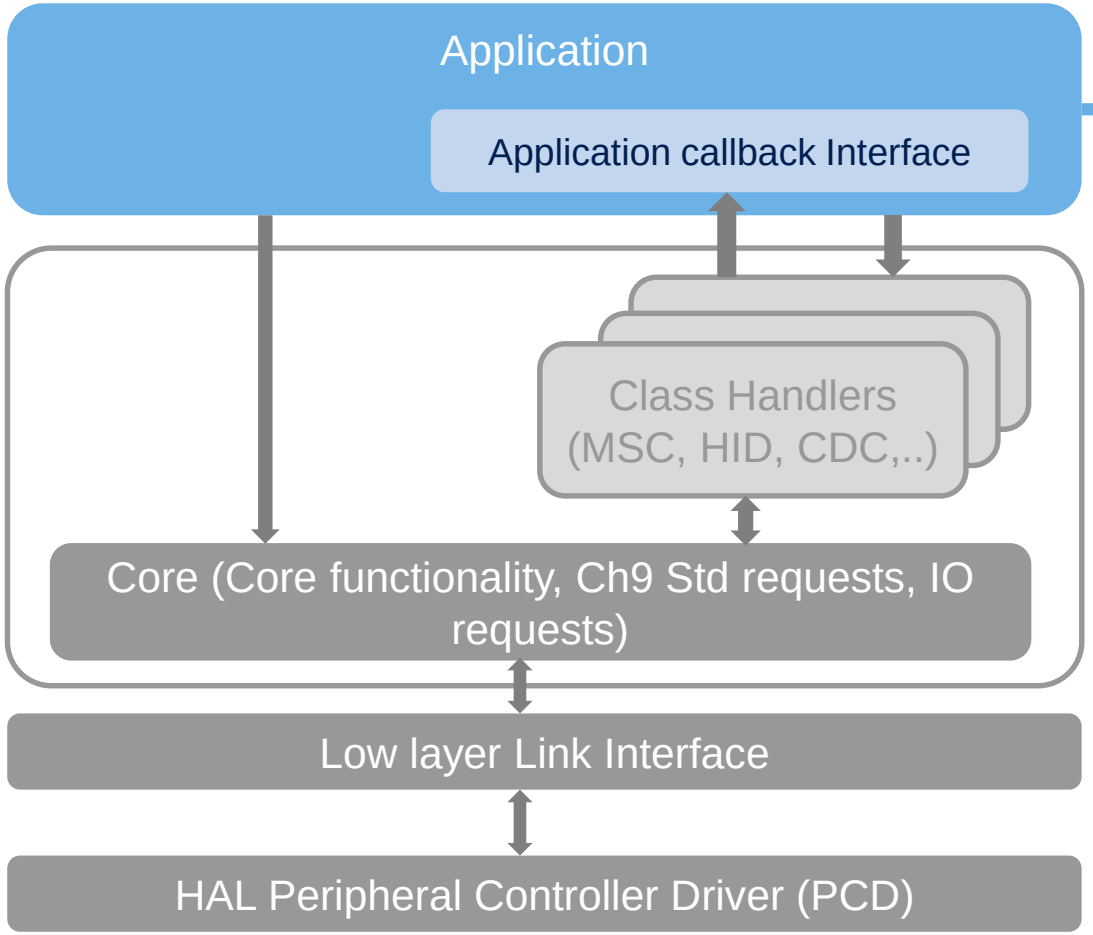





VBUS sensing → Disabled

# 1 USB Mass Storage Class Basics






- USB wrapper for SCSI commands

SCSI = Small Computer System Interface

**SCSI command subset for USB**

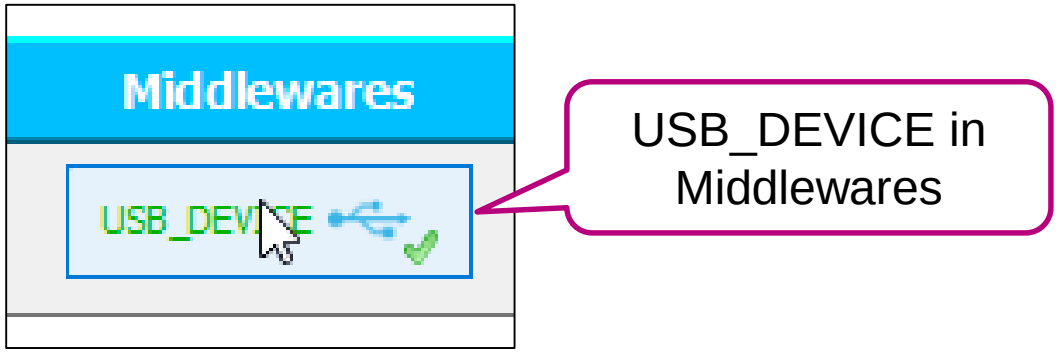
- Init
- Read
- Write
- Is Ready
- Get Capacity
- Get Max LUN
- Is Write Protected

life.augmented 15y since Windows Me® (2000), Linux kernel 2.4 (2001), Mac OS 9/X® (2000)

STM32 Cube

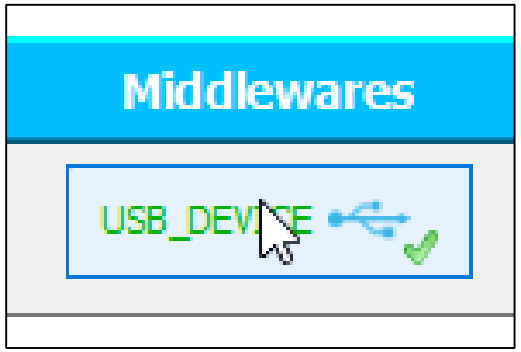
# 1 USB_DEVICE MiddleWare configuration

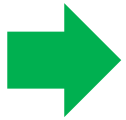


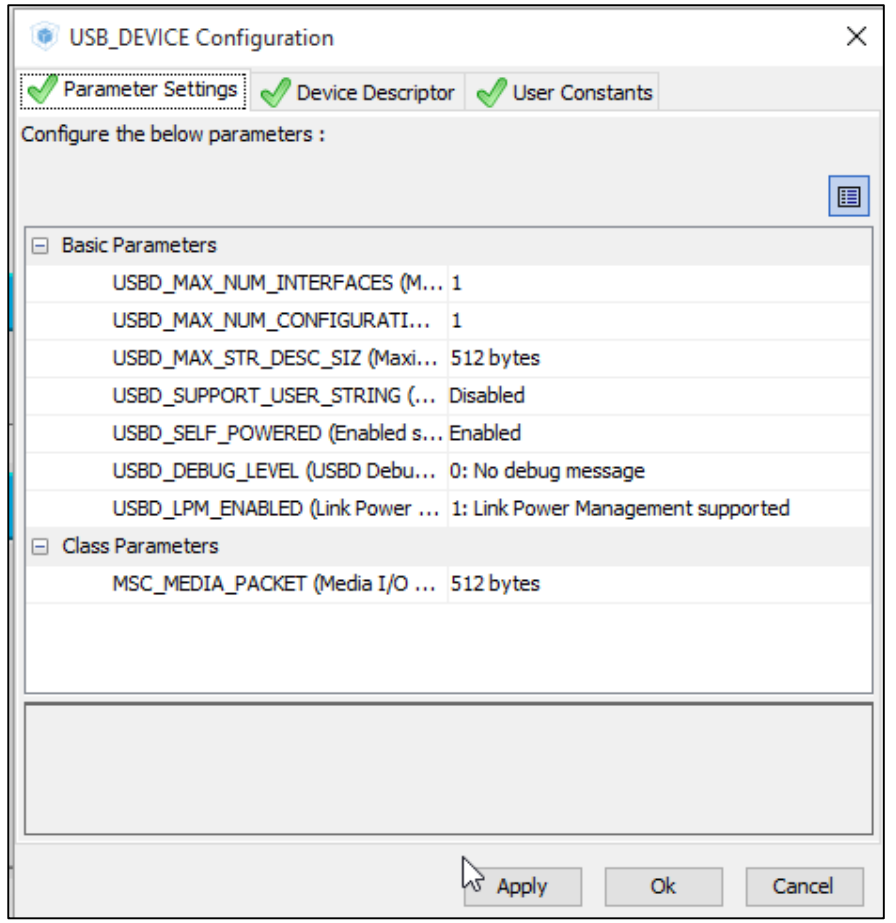



# 1 USB_DEVICE MiddleWare configuration



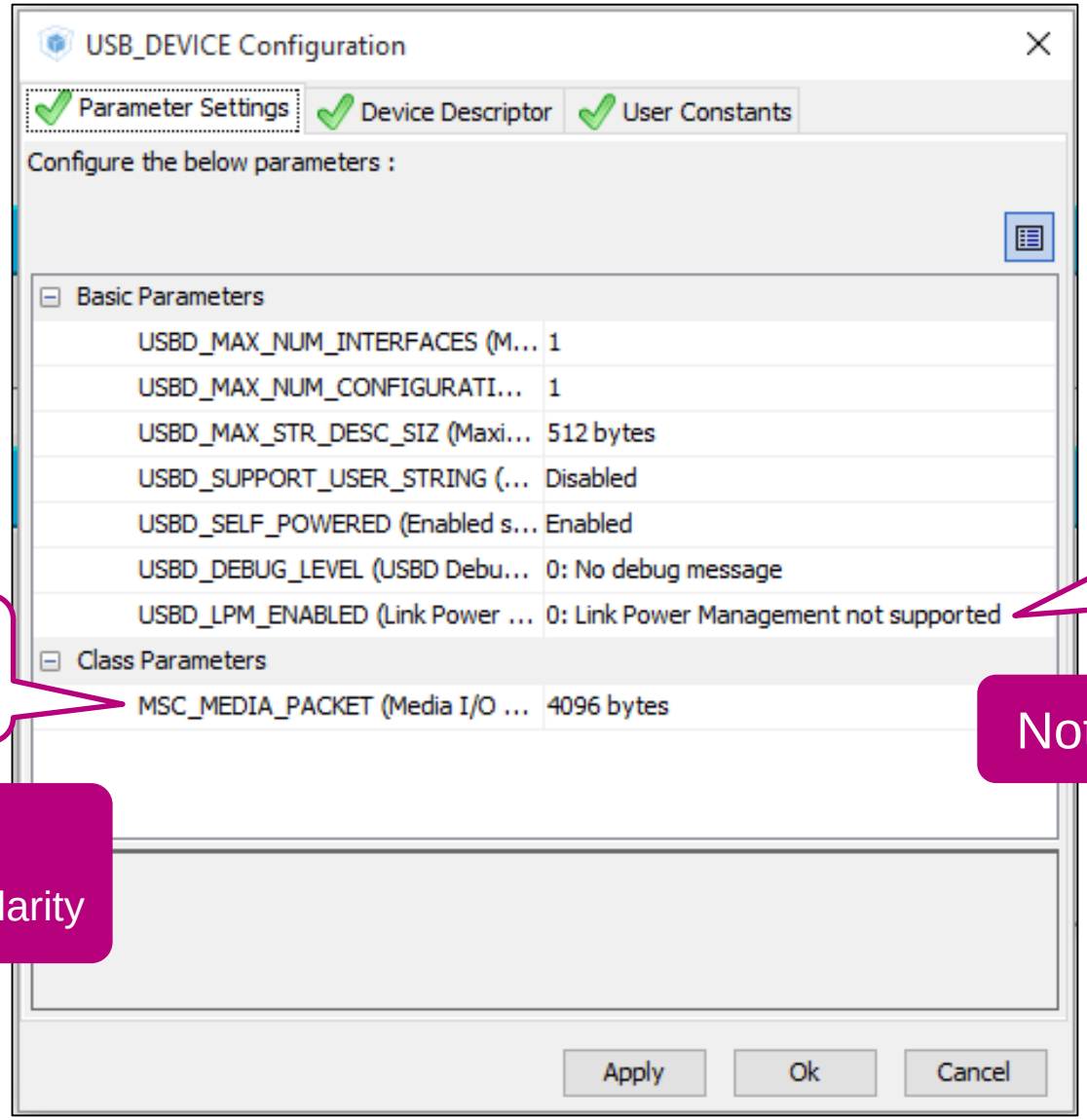



life.augmented

STM32 Cube

# 1 USB_DEVICE MiddleWare configuration






MSC_MEDIA_PACKET
→ 4096 bytes

USBD_LPM_ENABLED
→ 0

Not used in this demo app

N25Q128A memory
4kB Sub-Sector Erase Granularity

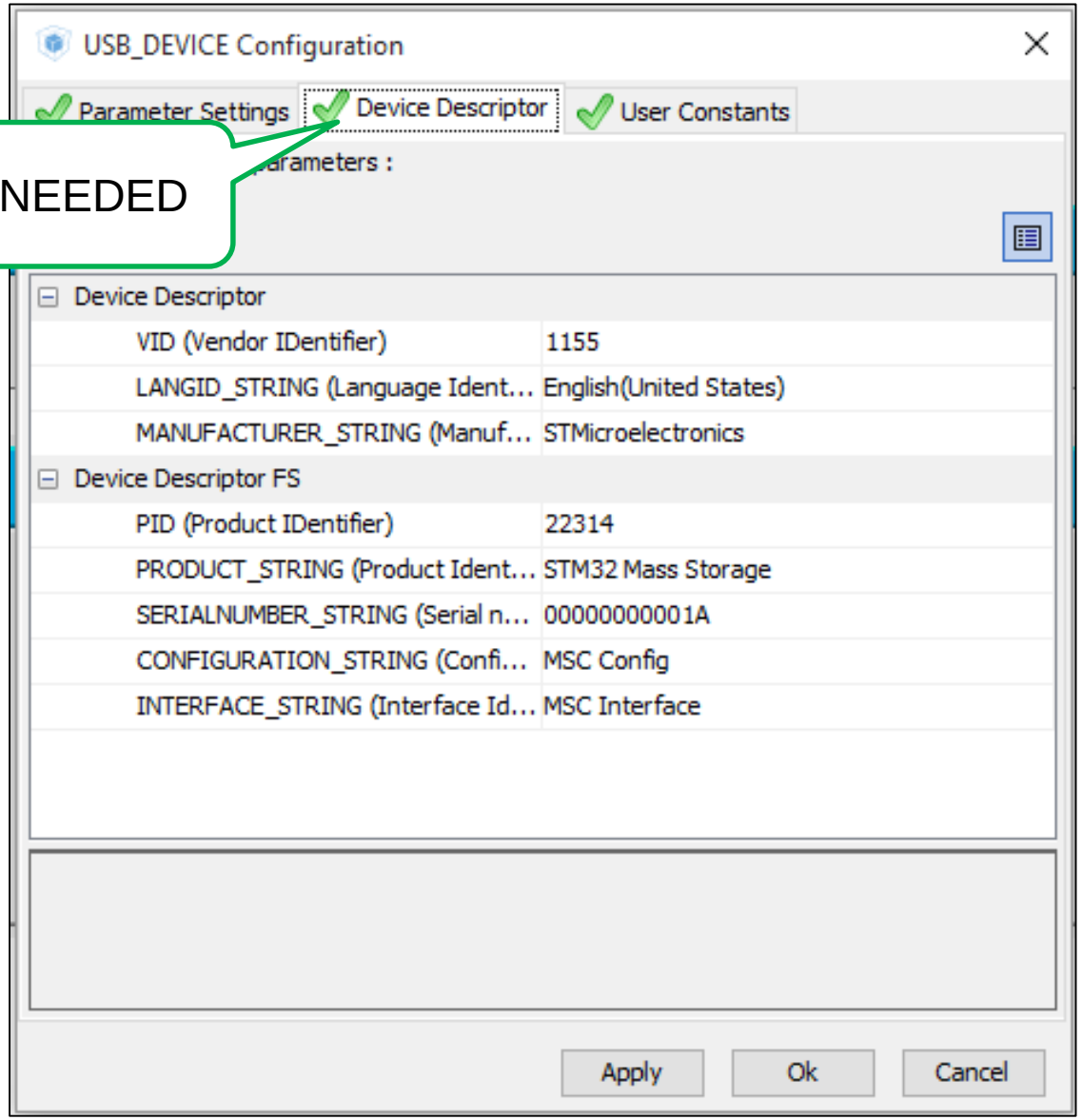
life.augmented

STM32 Cube

# 1 USB_DEVICE MiddleWare configuration



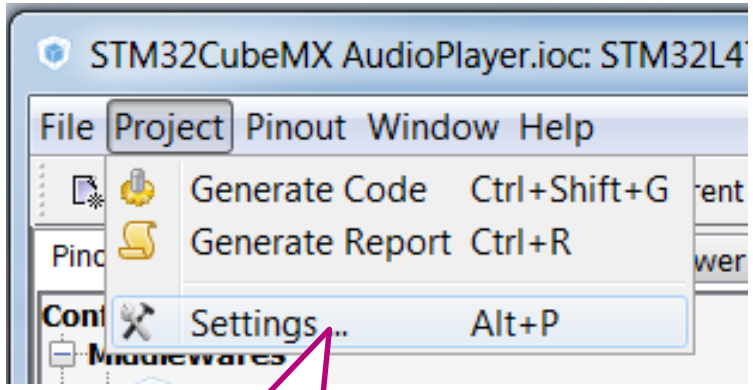
NO CHANGE NEEDED



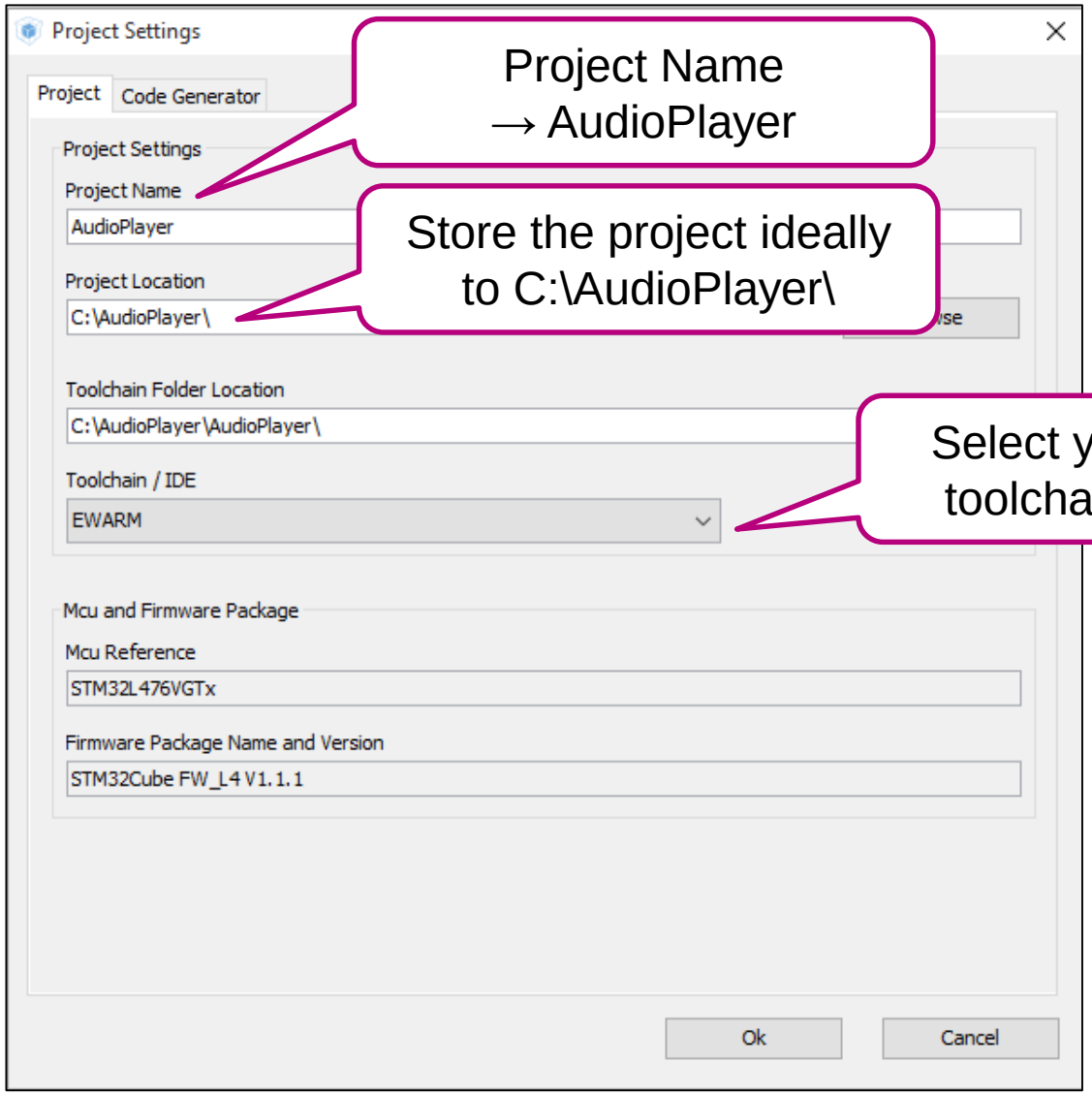
# 1

# Project Settings






Configure the project (Alt+P)




Project Name → AudioPlayer

Store the project ideally to C:\AudioPlayer\

Select your preferred toolchain (EWARM)

life.augmented

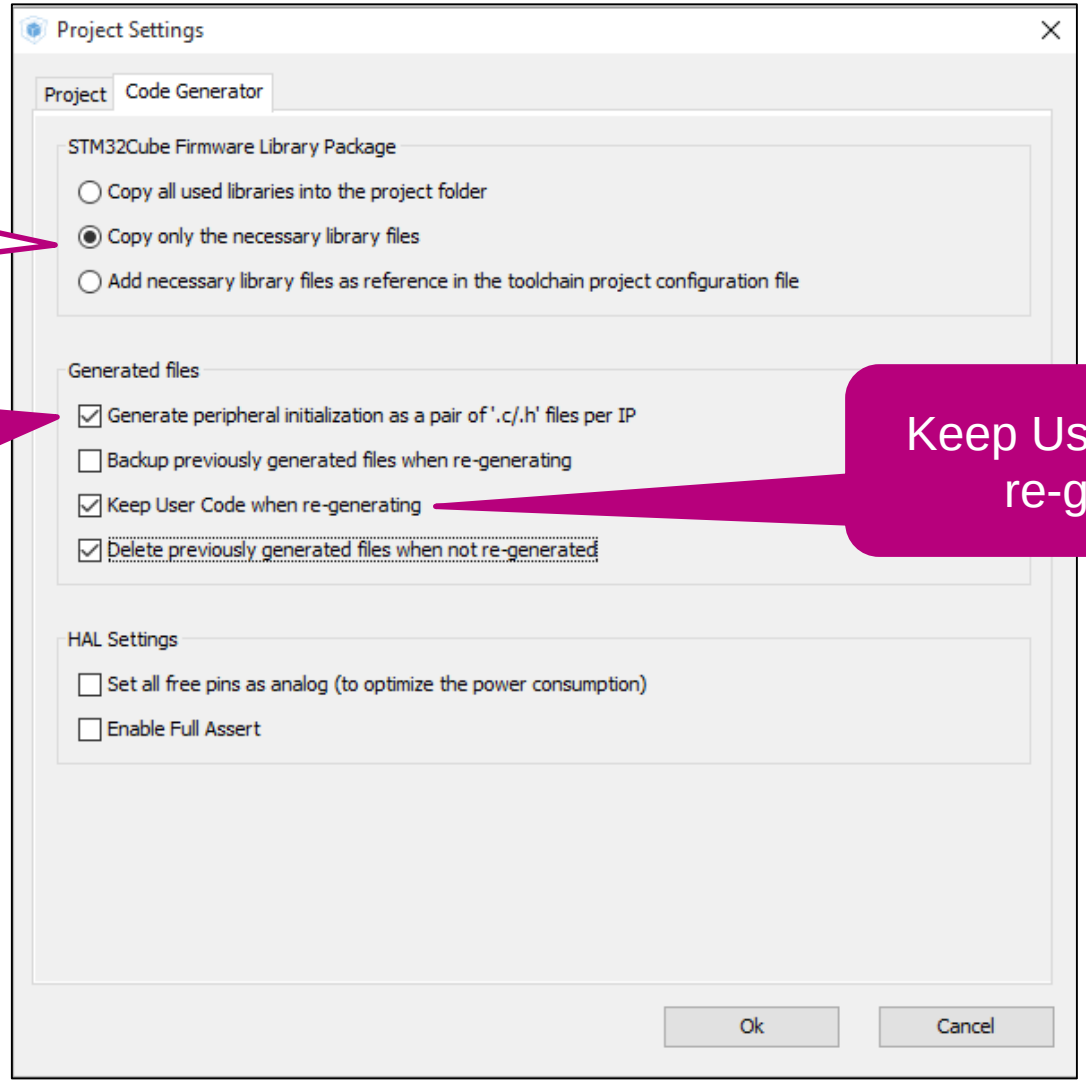
STM32 Cube

# Project Settings



Copy only the necessary library files

Generate peripheral initialization as a pair of .c/.h files per IP



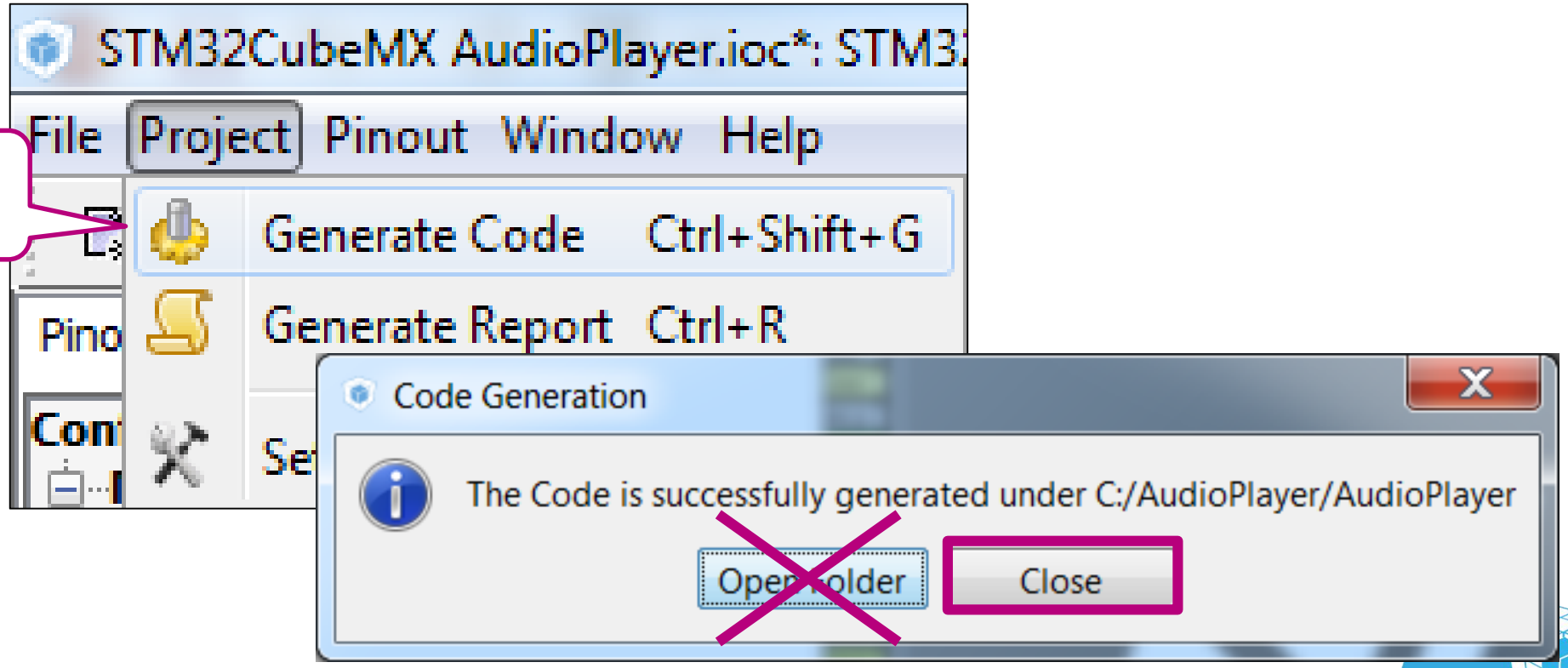

Keep User Code when re-generating

# 1 Save and generate the project



Do not open the project yet

Generate Code
(Ctrl+Shift+G)



# 1 Apply small patch



- Apply patch from folder

  **C:\STM32L4_Workshop\HandsOn\2_Putting_All_Together\Patch\STEP1_USB_MSC_Device**

  to project folder

  (C:\AudioPlayer\AudioPlayer\)

- It contains the following files:

  .\Inc\

  quadspi.h

  n25q128a.h

  .\Src\

  usbd_storage_if.c

  quadspi.c

This operation is STM32CubeMX non-intrusive. You can re-generate the project later and the modifications will be retained.

These files contains the main needed source code the user has to add to have the USB MSC device running together with QSPI FLASH storage

# 1 quadspi.c added content



- quadspi.c

QSPI_Init(void)
QSPI_DeInit(void)
} Initialization

QSPI_Read(…)
QSPI_Write(…)
} Read / Write access

QSPI_Erase_Block (…)
QSPI_Erase_Sector(…)
QSPI_Erase_Chip(void)
} Erase operations

QSPI_GetStatus(void)
QSPI_GetInfo(…)
} QUADSPI memory info / status

Based on BSP driver in STM32CubeL4 HAL

# 1 usbd_storage_if.c added content



- usbd_storage_if.c
  - STORAGE_Init_FS (…)
  - STORAGE_Read_FS (…)
  - STORAGE_Write_FS (…)
  - STORAGE_IsReady_FS (…)
  - STORAGE_GetCapacity_FS (…)

Most important and implemented
Support for 4kB sector size only
(due to 4kB block erase feature of used memory)

  - STORAGE_GetMaxLun_FS (void)
  - STORAGE_IsWriteProtected_FS (…)

Not so important

Templates generated by STM32CubeMX, just the bodies of the functions were implemented.

# 1 Let’s try it out

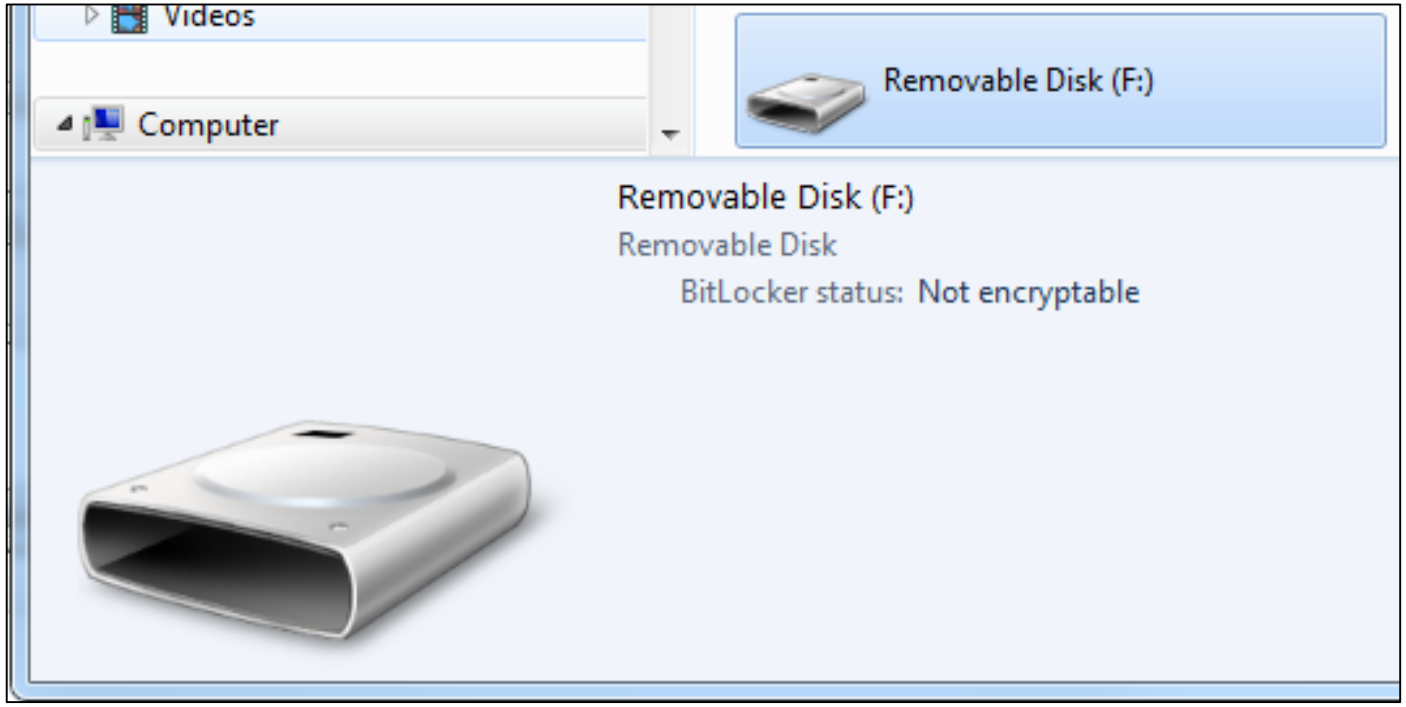


1. Open the project
2. Connect discovery kit over miniUSB cable (ST-Link)
3. Compile and download
4. Press the reset button on discovery once download has finished

# 1 Connect the new “FLASH stick”

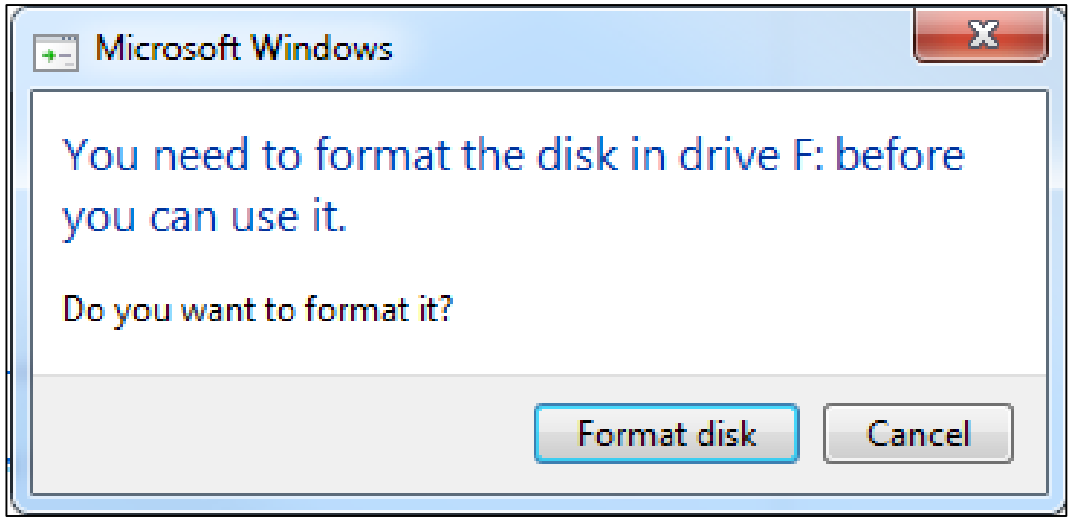


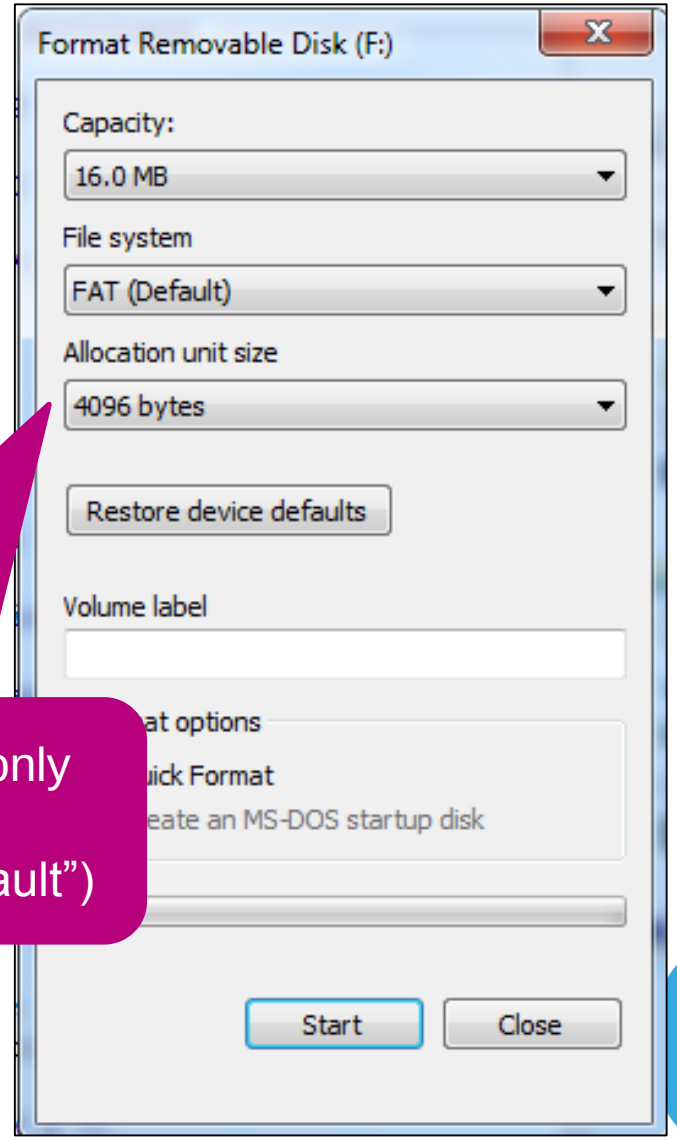
- Plug-In microUSB cable
- Check if new Mass Storage Class device appeared



# Storage formatting






Actual tiny implementation supports only native 4kB sector size
(Windows 10 users should select “default”)

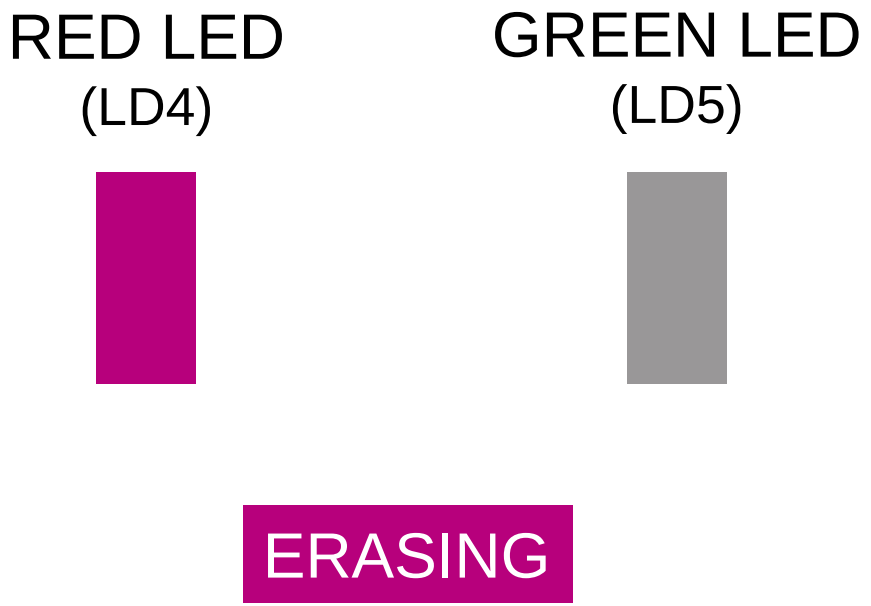
life.augmented

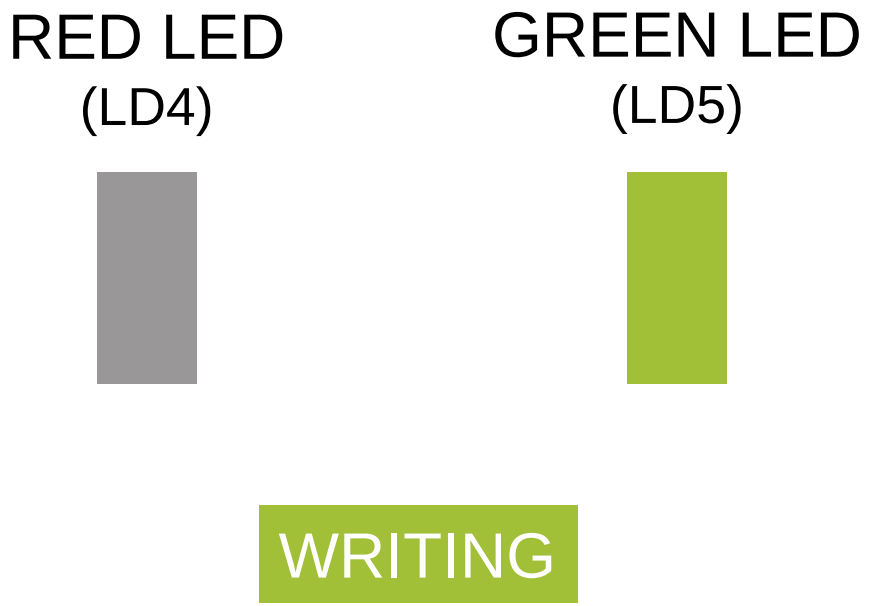
STM32 Cube

# 1 Test the functionality



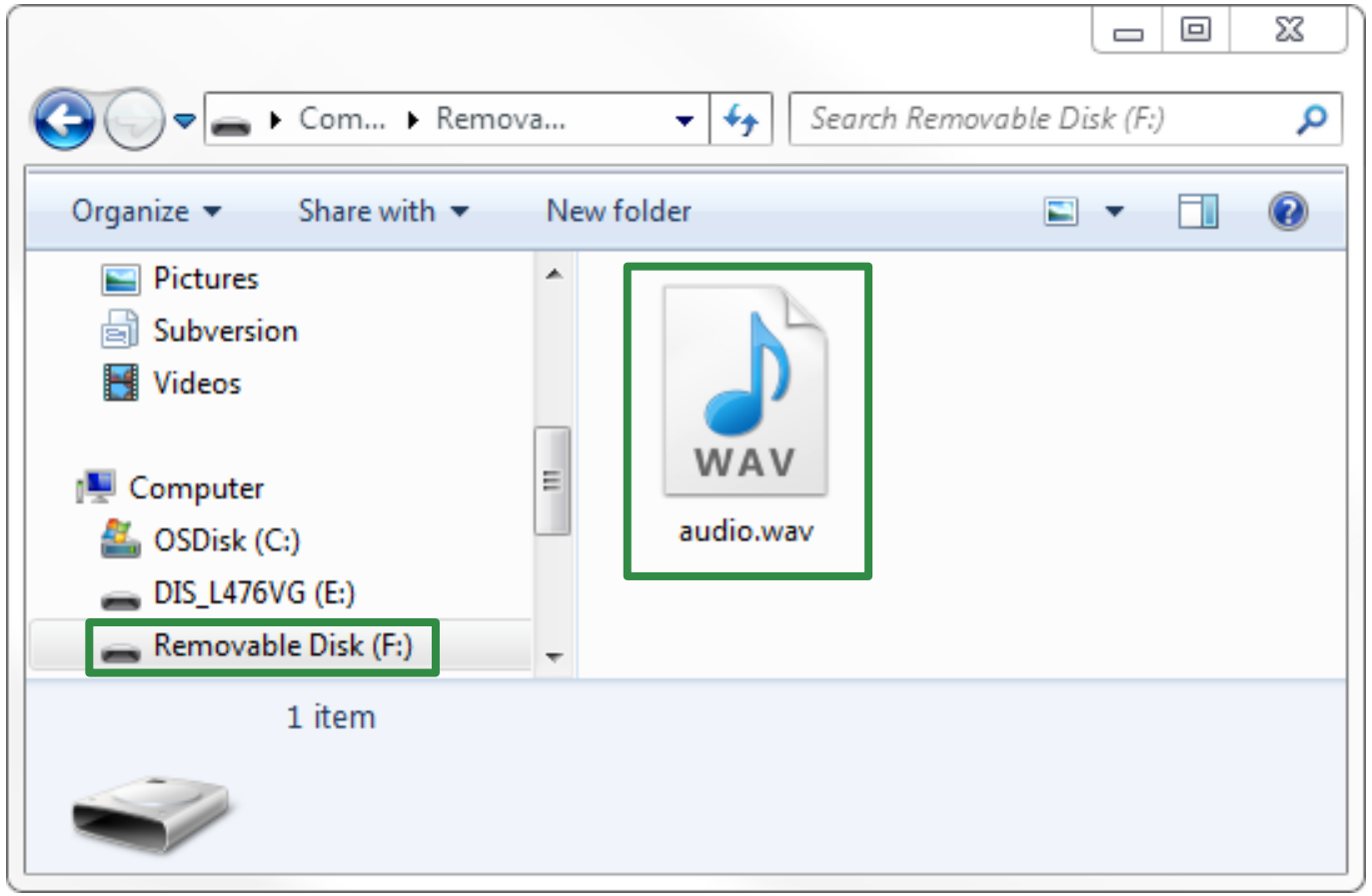
- Now try to copy the **audio.wav** file from the package to the new “USB flash stick” you created

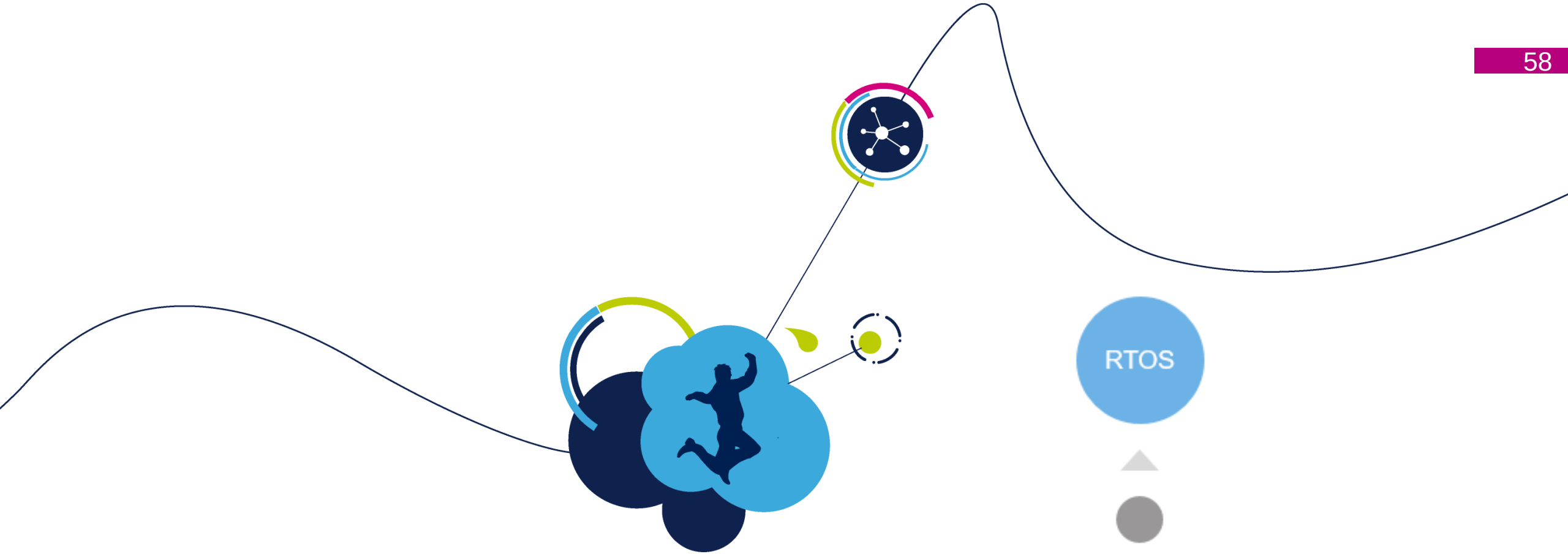


# Checkpoint #3



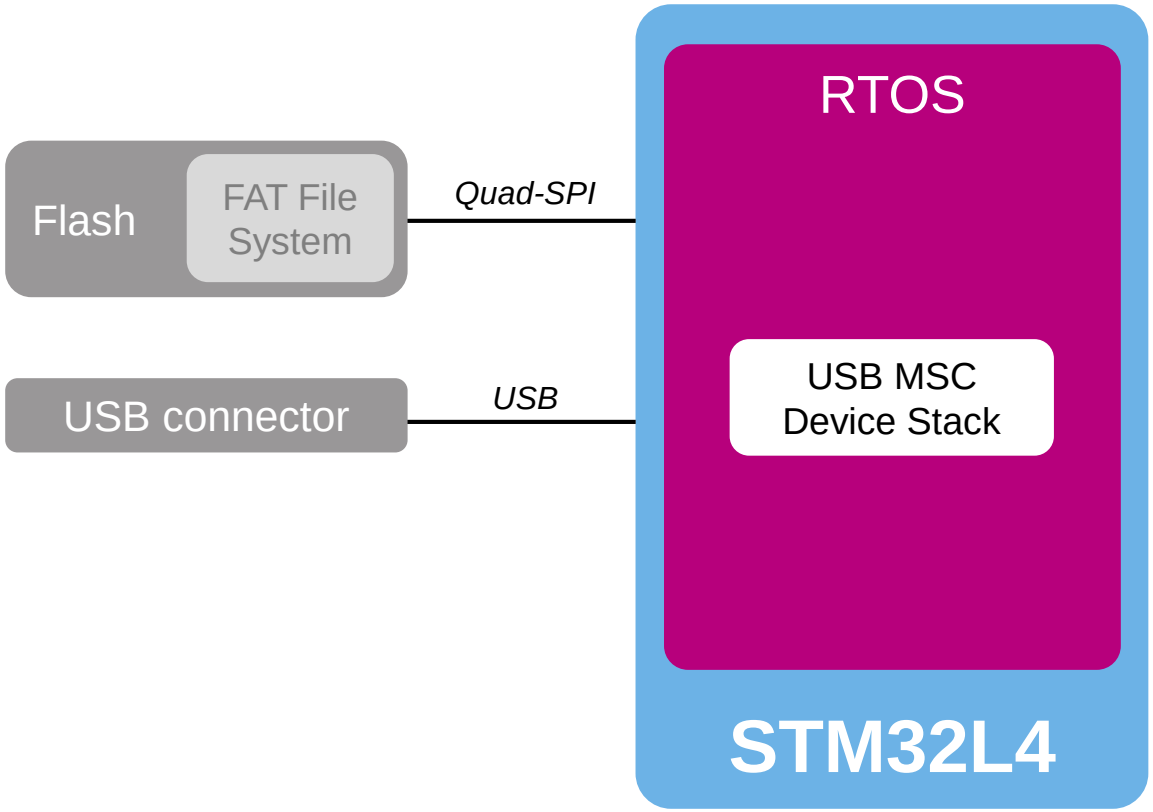



# STEP2 – RTOS

# Adding RTOS



- We want to have the application running on RTOS



# 2 Why RTOS?



- To have the application modular (design not closed yet)
  - add/remove functionality easily
- We want to implement application with multiple functionality quickly
  - we don’t want to care about the operations synchronization and flow management
- Because we can
  - far more than enough flash, RAM, performance
  - don’t need to limit ourselves

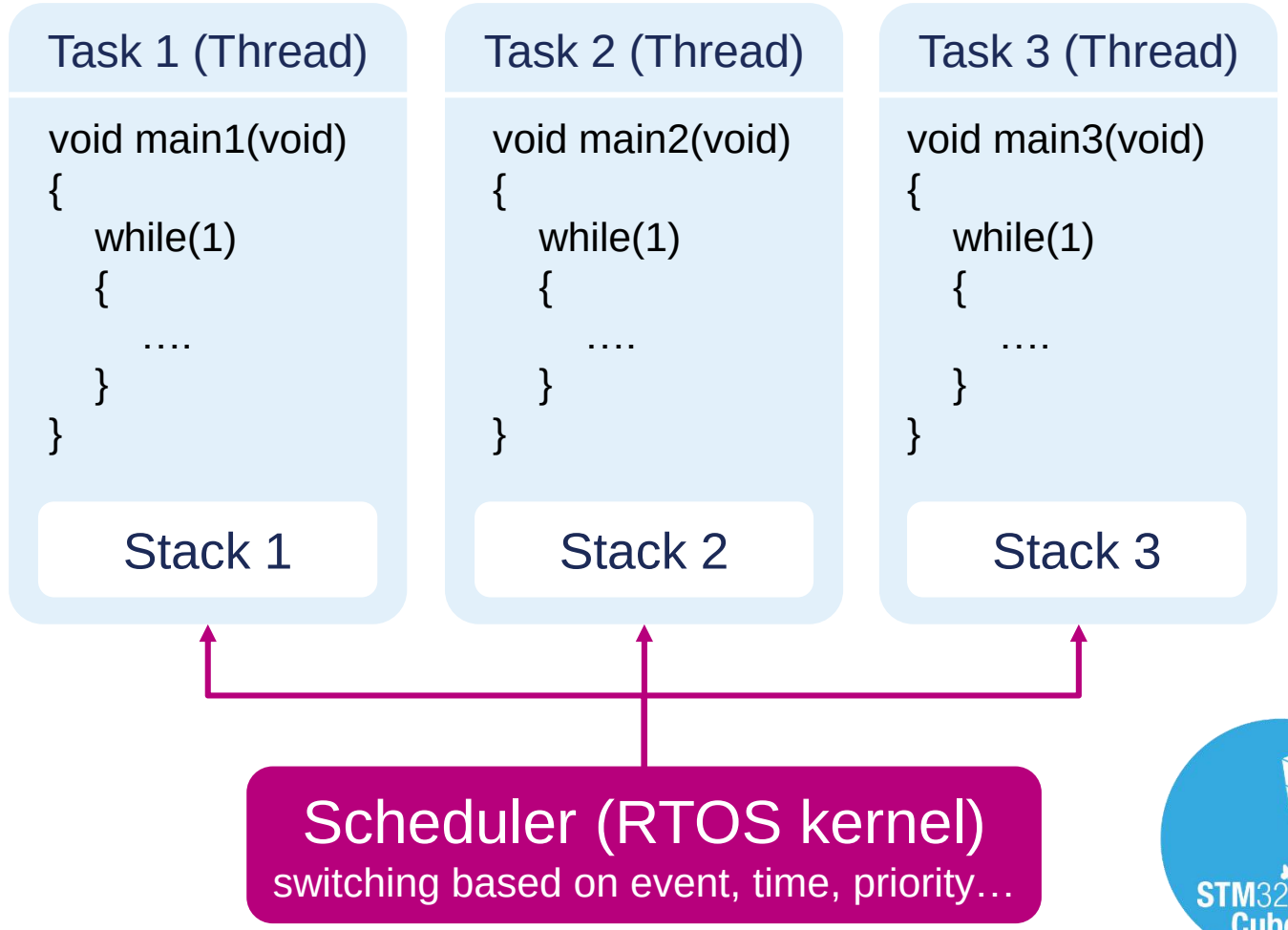
# RTOS basics



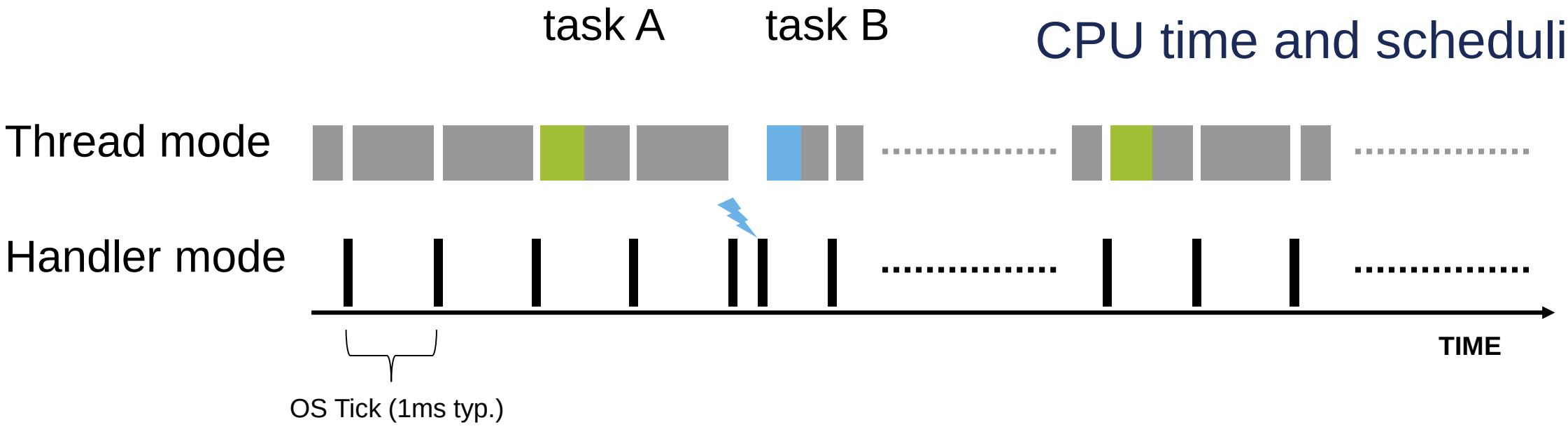
## Conventional app

```
void main(void)
{
   while(1)
   {
      switch task {
      case 1:
         ….
         break;
      case 2:
         ….
         break;
      case 3:
         ….
         break;
      }
   }
}
```

## RTOS app



## CPU time and scheduling



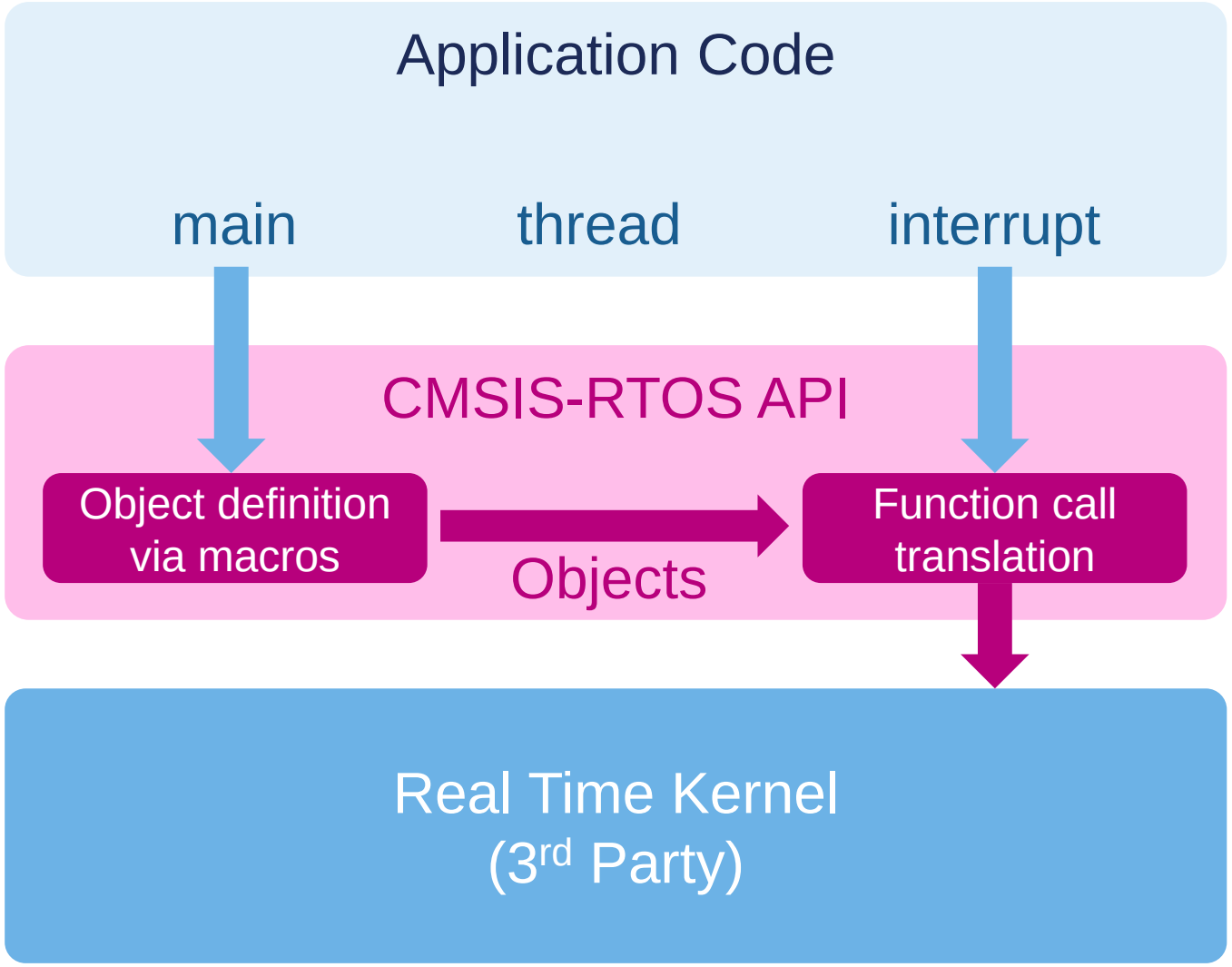

- A timeout parameter is incorporated in RTOS functions to avoid system lockup
- When a timeout is specified the system waits until a resource is available or event occurs
- While waiting, other threads are scheduled
- osDelay function puts a thread into the state WAITING for a specified period of time

## Threads and ISR communication

- Signal
  - flag that may be used to signal specific condition to a thread
  - can be modified in an ISR or set from other threads.
- Message
  - 32-bit value that can be sent to a thread or an ISR
  - buffered in a queue, type and queue size defined in a descriptor
- Mail
  - fixed-size memory block that can be sent to a thread or an ISR
  - buffered in a queue and memory allocation is provided
  - type and queue size are defined in a descriptor.

# CMSIS-RTOS introduction






FreeRTOS

# 2 Where to find more information?



- UM1722 – Developing Applications on STM32Cube with RTOS

  http://www.st.com/st-web-ui/static/active/en/resource/technical/document/user_manual/DM00105262.pdf

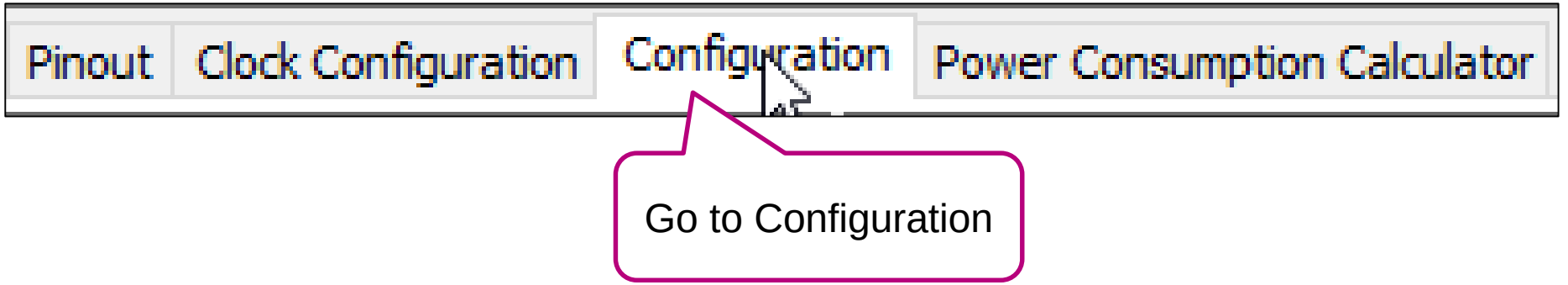
- http://www.freertos.org

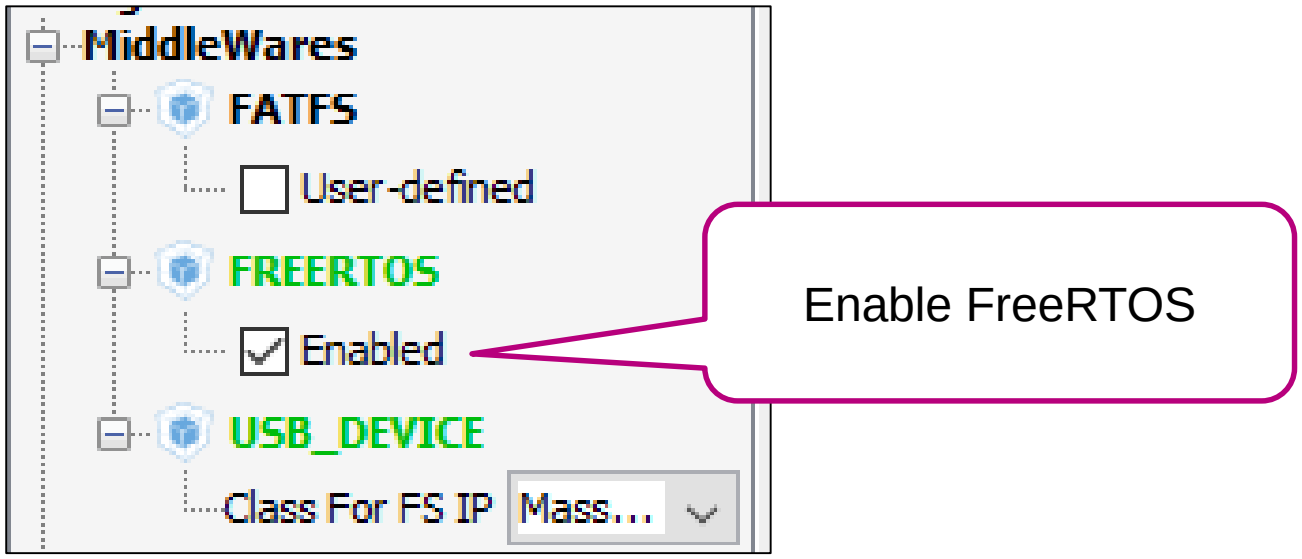
- Using The FreeRTOS Real Time Kernel - Cortex-M3 edition

  http://shop.freertos.org/FreeRTOS_Tutorial_Book_Generic_Cortex_M3_Edition_p/pdf_cortex-m3_tutorial_book.htm

# 2 FreeRTOS Configuration

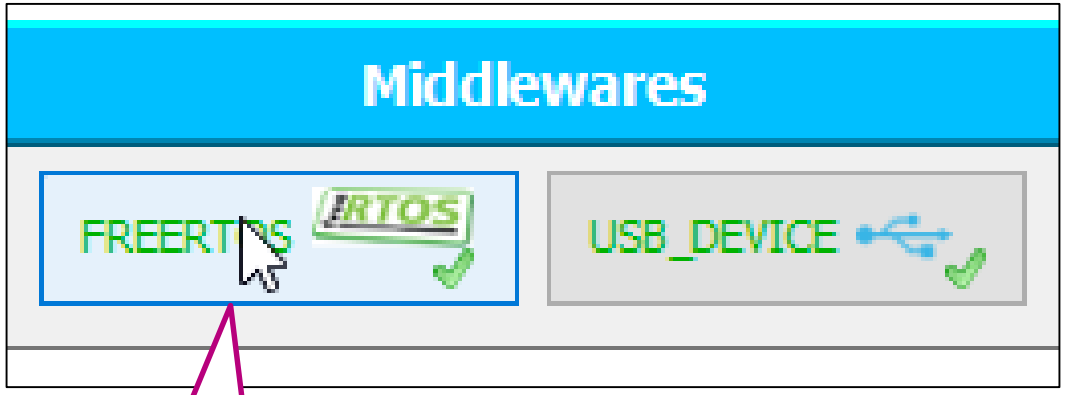




# 2 FreeRTOS Configuration

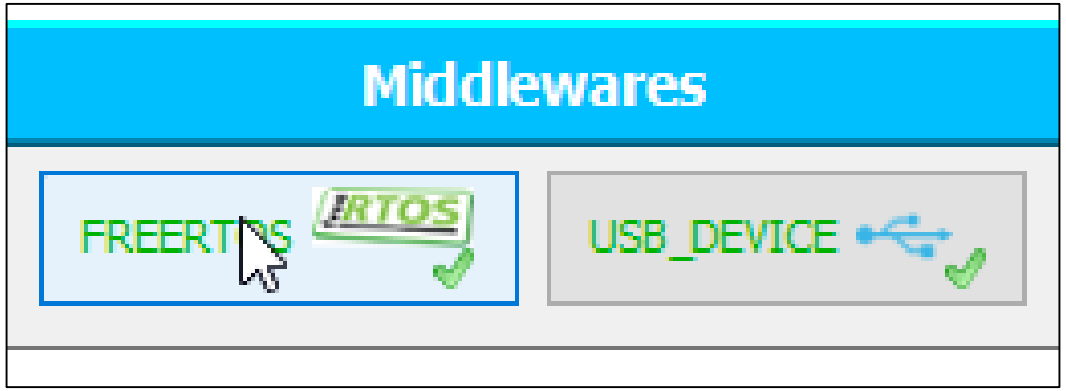


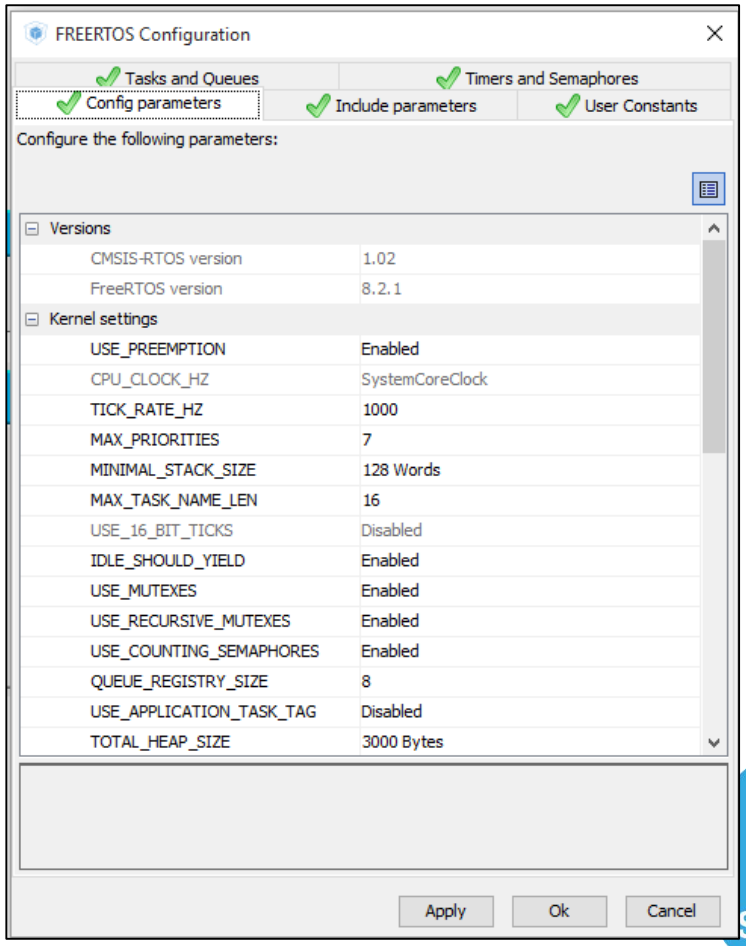




FREERTOS in Middlewares

# FreeRTOS Configuration



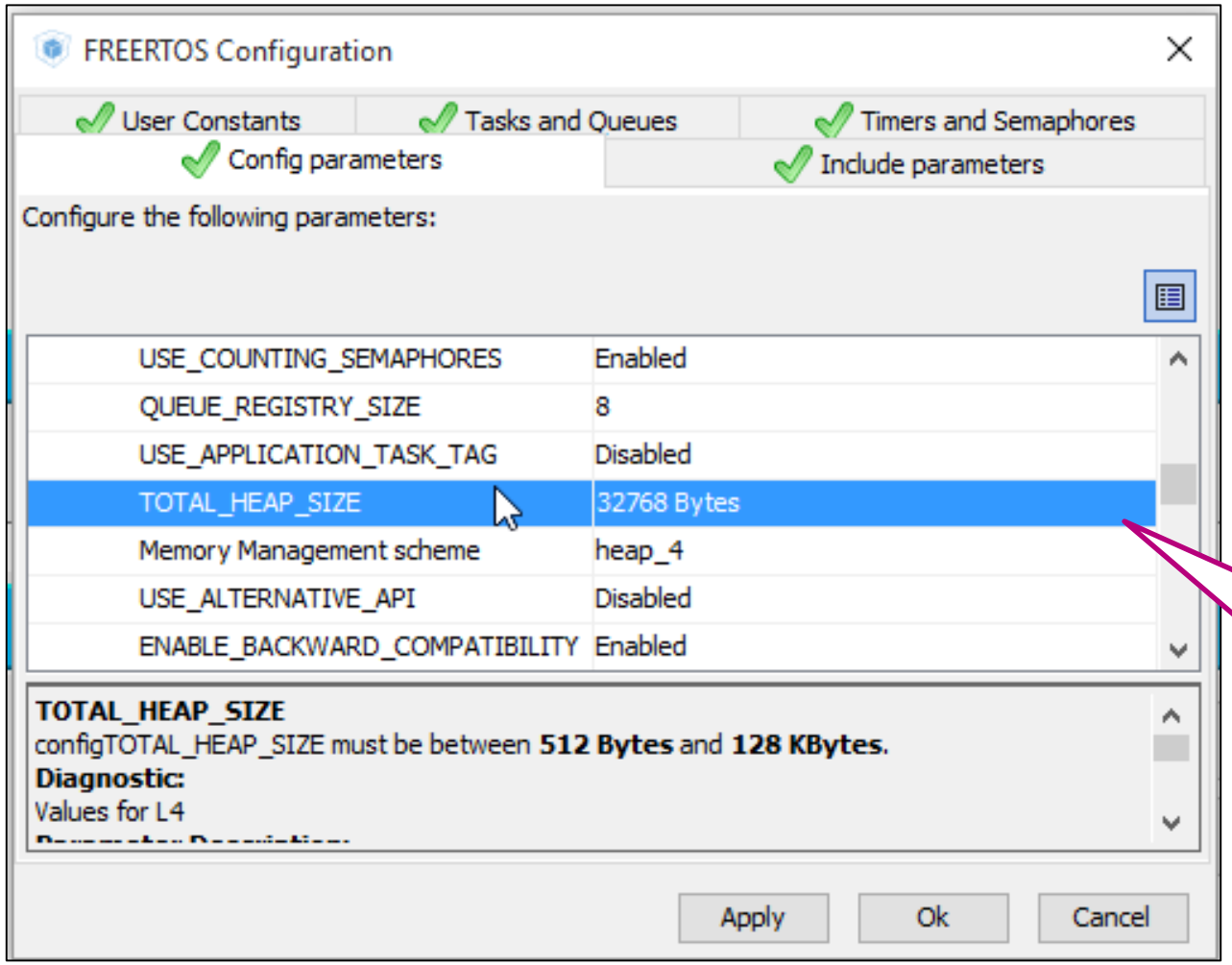




life.augmented

STM32 Cube

# FreeRTOS Configuration



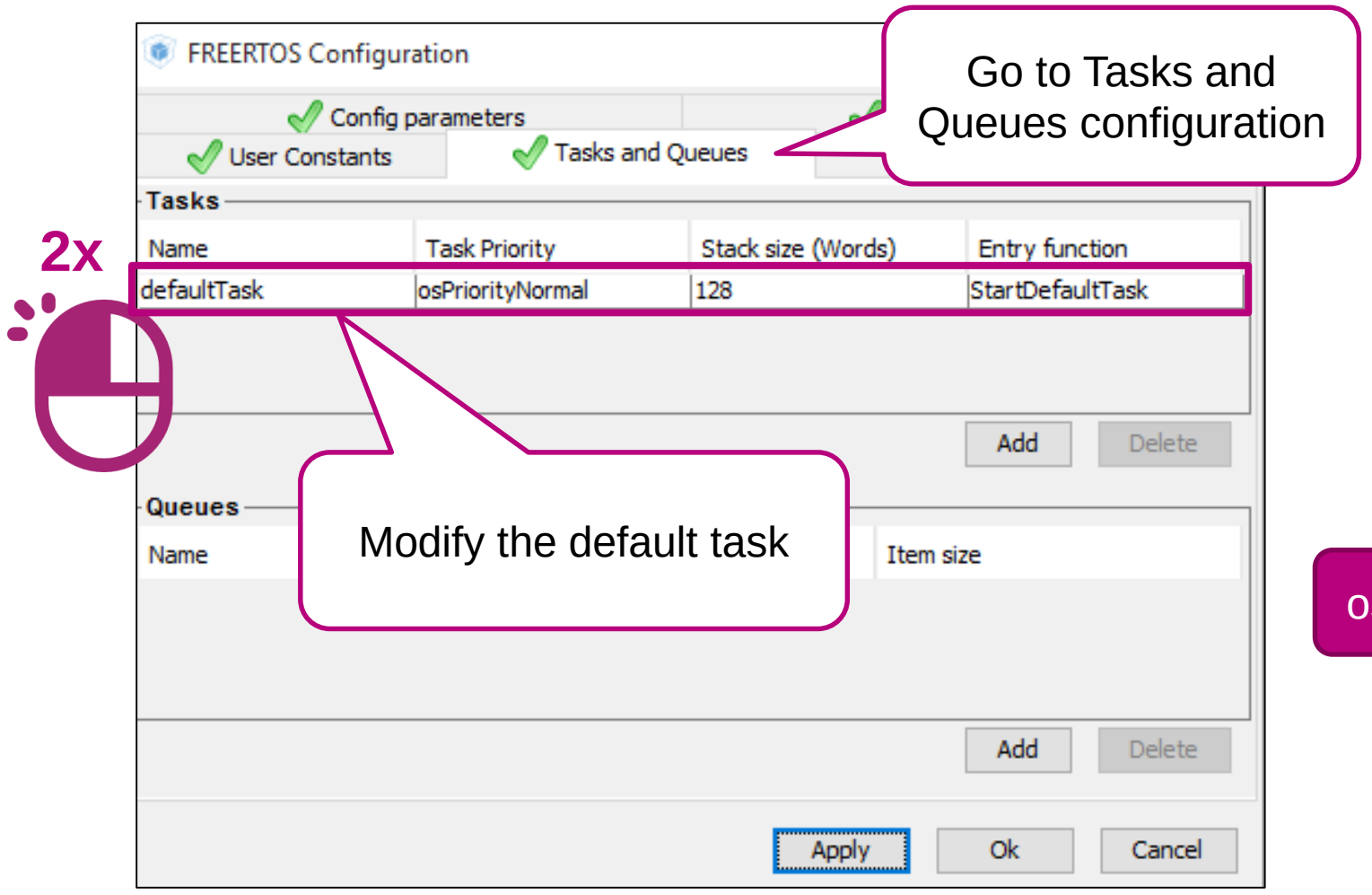


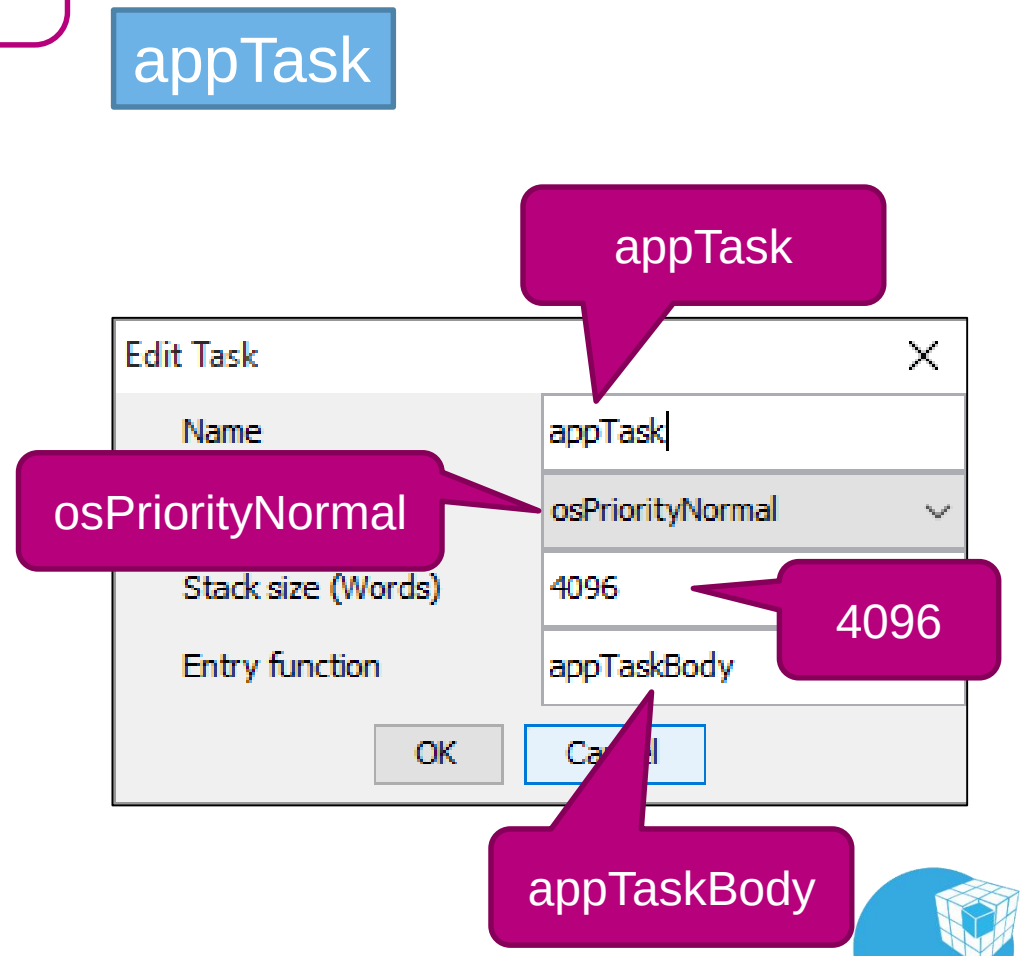

TOTAL_HEAP_SIZE
→ 32768 bytes

(to have some reserve)

# 2 FreeRTOS Configuration





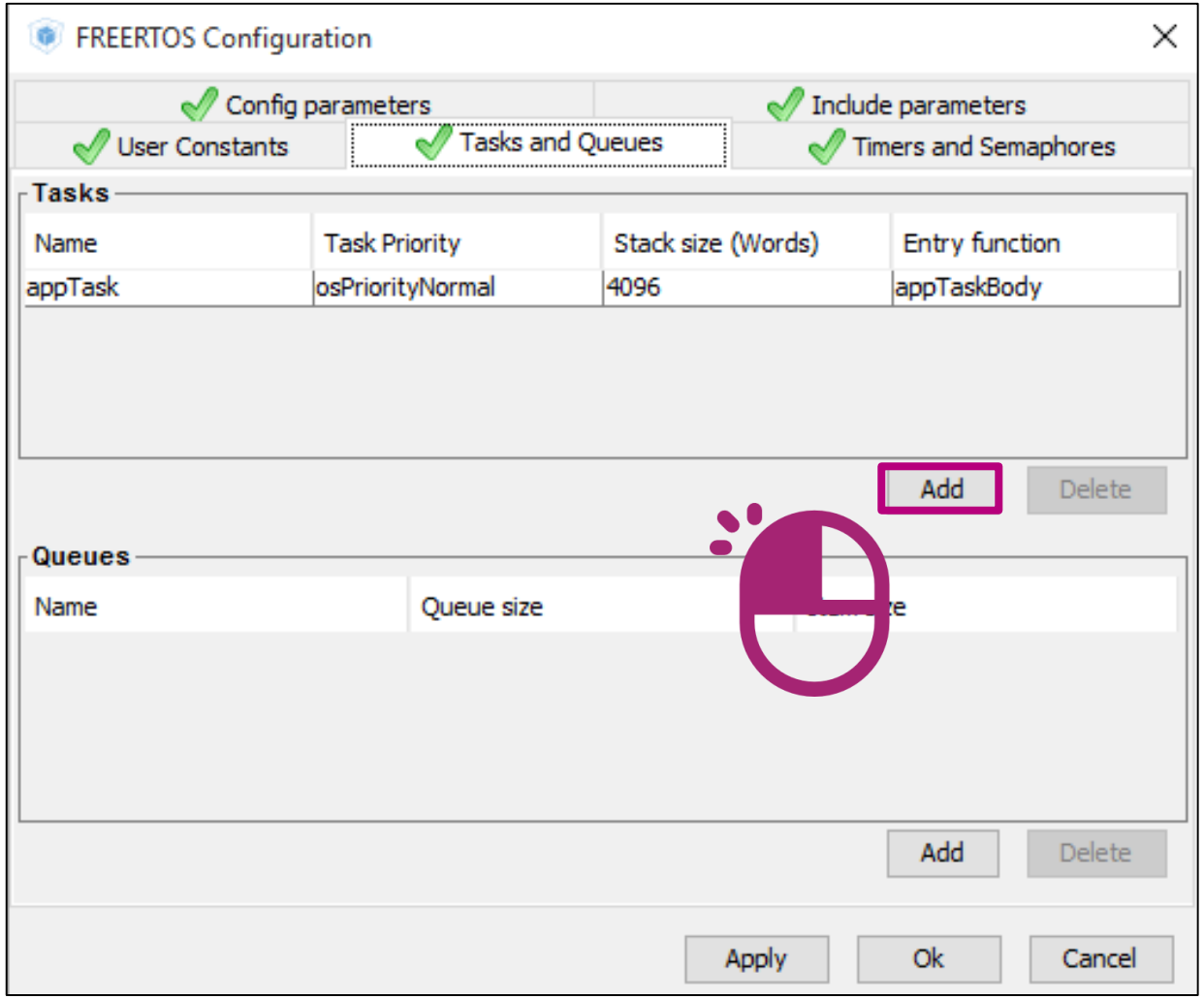




life.augmented

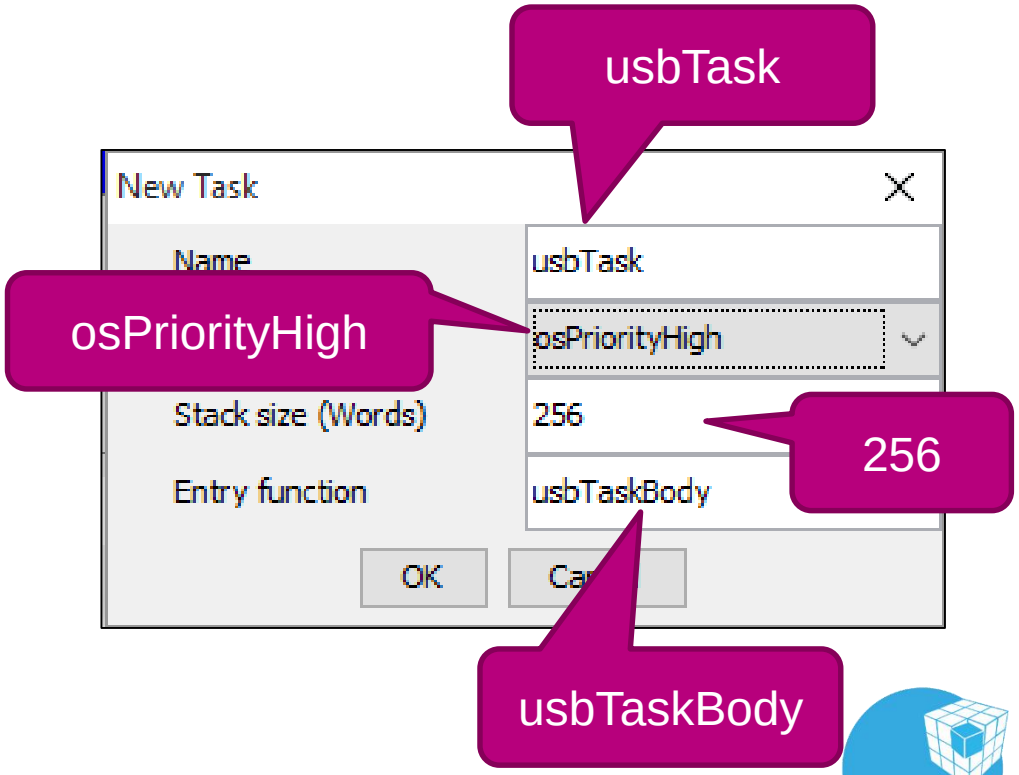
STM32 Cube

# FreeRTOS Configuration





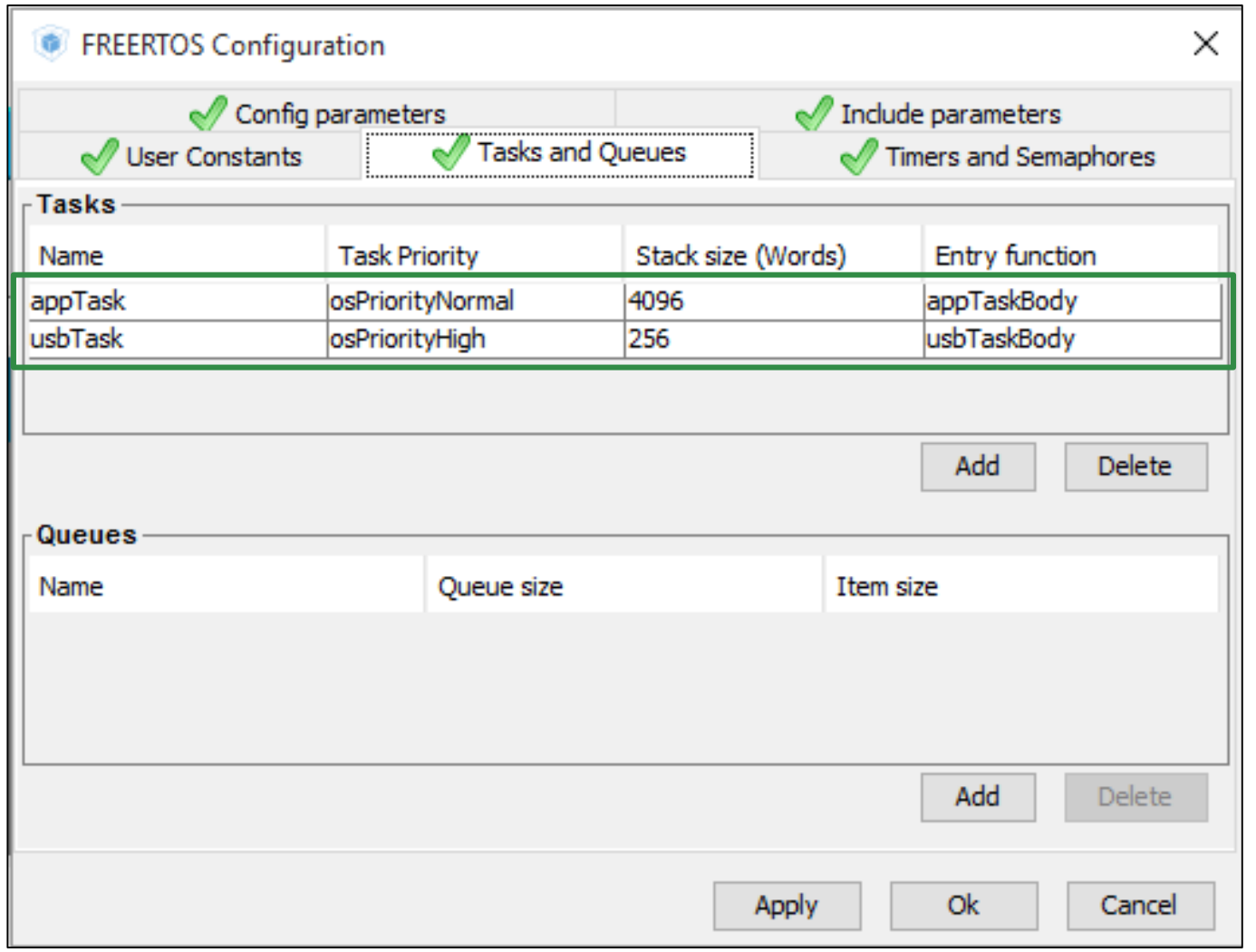

usbTask



# Checkpoint #4



USB

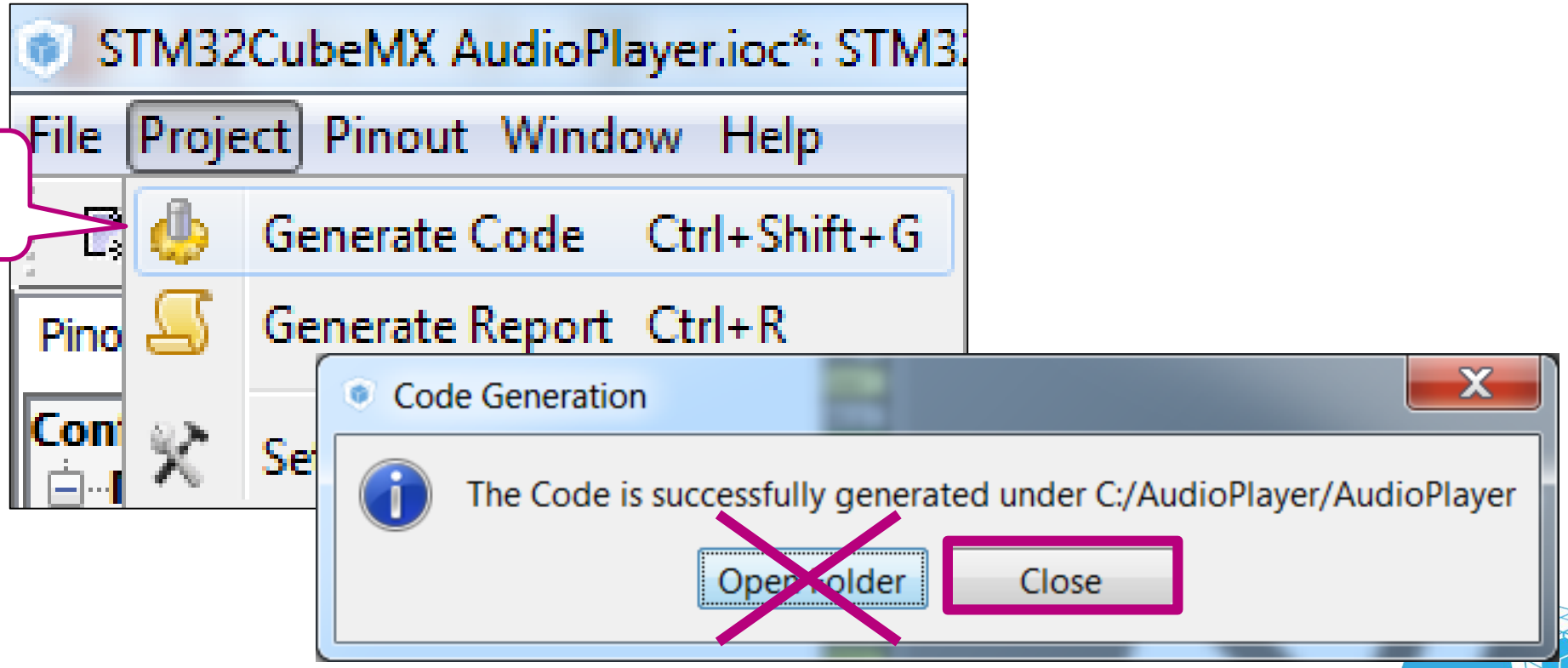



# 2 Save and generate the project



Do not open the project yet

Generate Code
(Ctrl+Shift+G)



# 2 Apply small patch



- Apply patch from folder
  **C:\STM32L4_Workshop\HandsOn\2_Putting_All_Together\Patch\STEP2_RTOS**
  to root folder of your project
  (C:\AudioPlayer\AudioPlayer\)

- It contains the following files:
  .\Src\
    freertos.c
    usbd_storage_if.c
    stm32l4xx_it.c

This operation is STM32CubeMX non-intrusive. You can re-generate the project later and the modifications will be retained.

- usbd_storage_if.c – change in LEDs use, we need them for other purpose

# 2

# Let’s try it out



1. Open the project

2. Compile and download

3. Press the reset button on discovery once download has finished

# 2

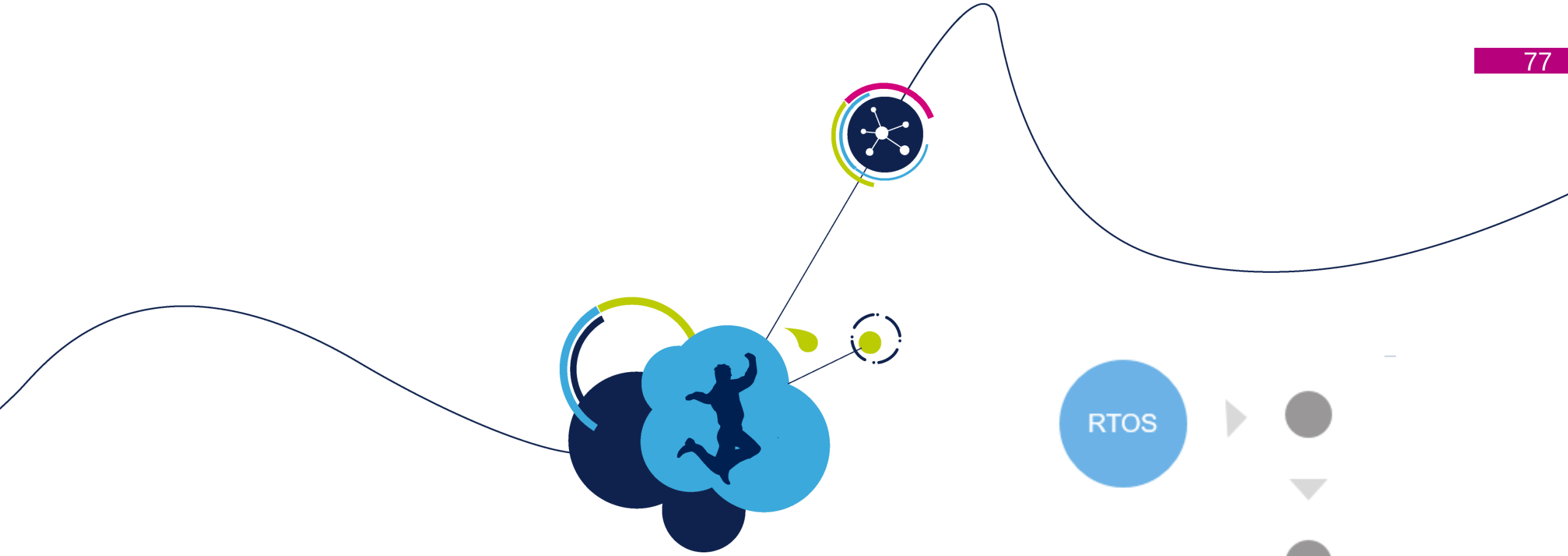
# Checkpoint #5

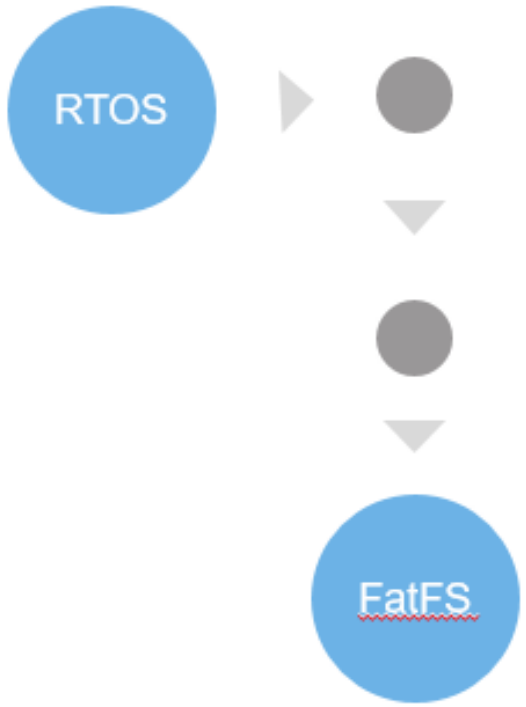


- appTask – GREEN LED blinking
- usbTask – USB connected → RED LED ON
- usbTask – USB connected → RED LED OFF


life.augmented

STM32 Cube



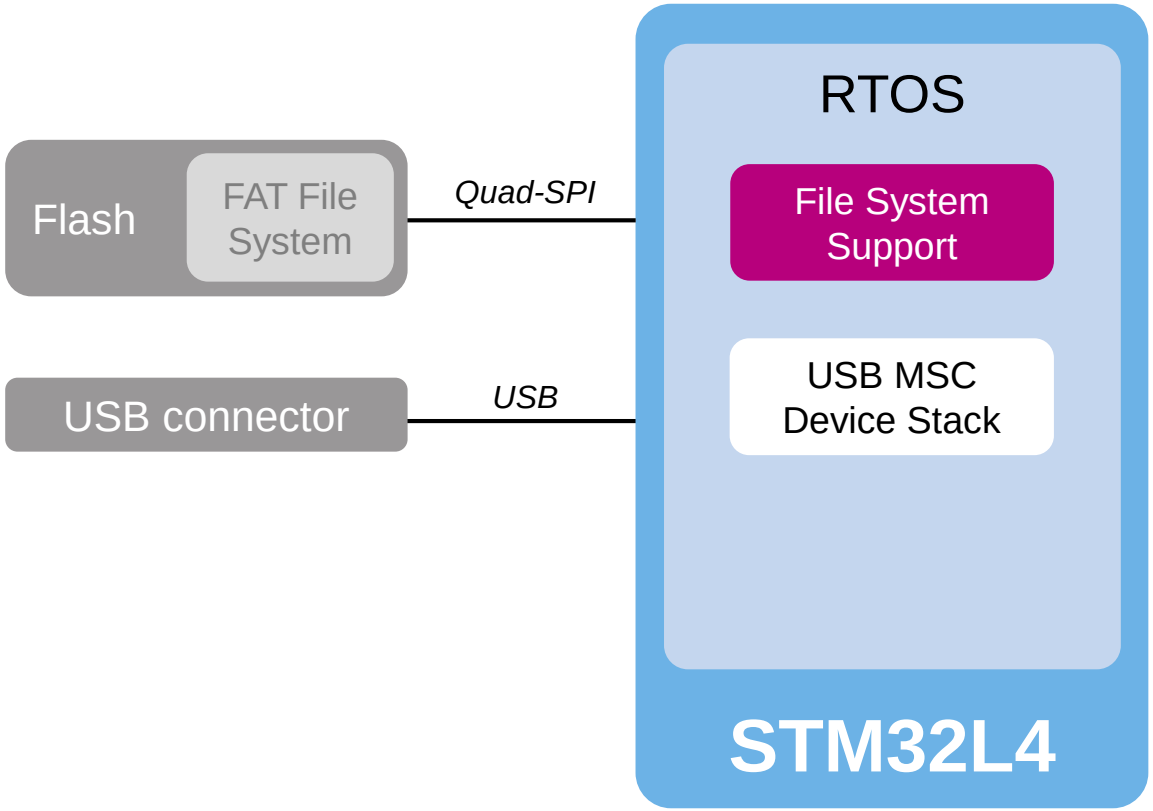



# STEP3 – File System

# 3 Adding File System support



- We want to have the access to our “audio.wav” file from RTOS



# File System Support Basics



| PC application in C language | MCU application in C language |
|---|---|

C standard libraries (stdio.h)

- fopen(…) – opens a file
- fclose(…) – closes a file
- fread(…) – reads from a file
- fwrite(…) – writes to a file
- fprintf(…) – formatted output to a file
- fscanf(…) – formatted input from a file

C standard libraries (stdio.h)

- ~~fopen(…) – opens a file~~
- ~~fclose(…) – closes a file~~
- ~~fread(…) – reads from a file~~
- ~~fwrite(…) – writes to a file~~
- ~~fprintf(…) – formatted output to a file~~
- ~~fscanf(…) – formatted input from a file~~

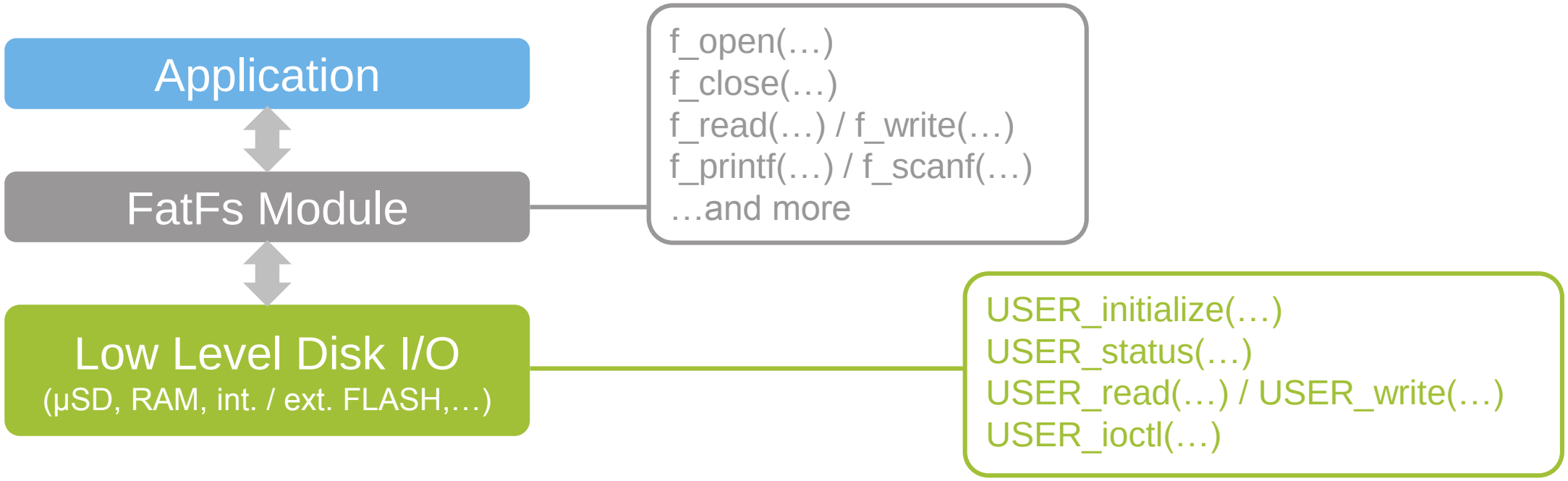
No OS + No standard File System = No native support by C standard IO library

Let's add "special" library to support that

# FatFs Module Basics



- Windows compatible FAT file system support




UM1721 – Developing Applications on STM32Cube with FatFs
http://www.st.com/st-web-ui/static/active/en/resource/technical/document/user_manual/DM00105259.pdf

FatFs project home page
http://elm-chan.org/fsw/ff/00index_e.html

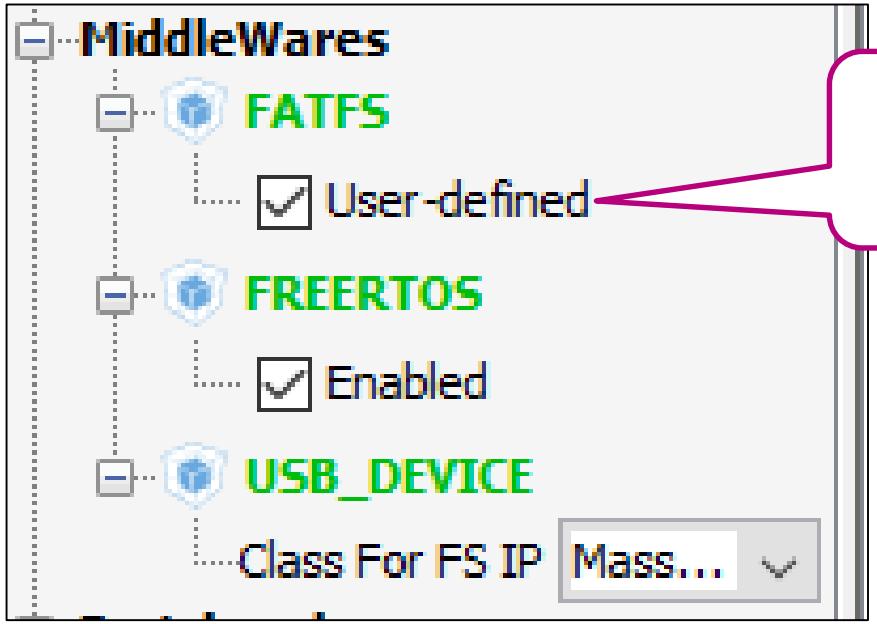
life.augmented

STM32 Cube

# 3 FatFs Configuration




Go to Configuration




Enable User-defined FatFs

# 3 FatFs Configuration

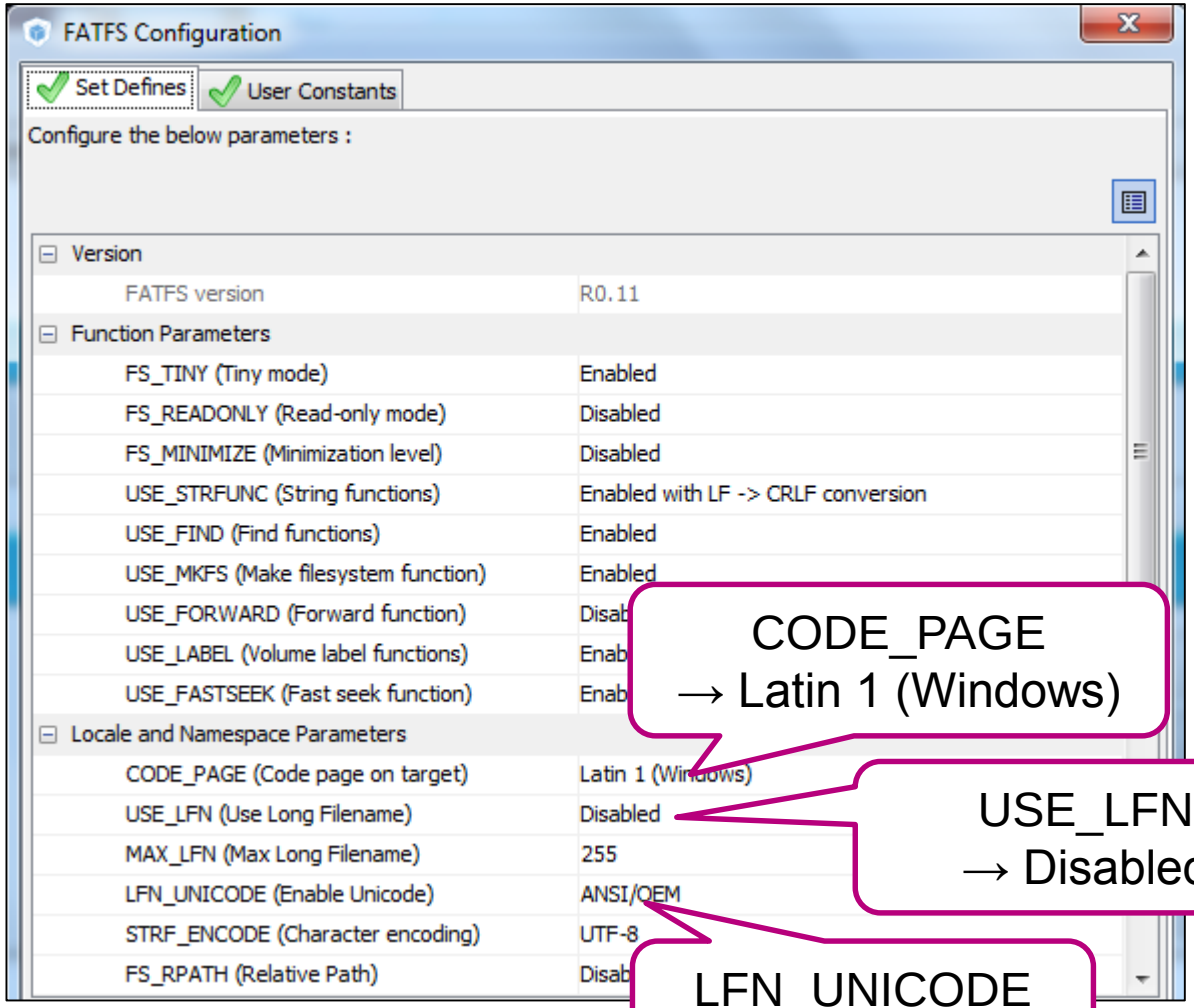


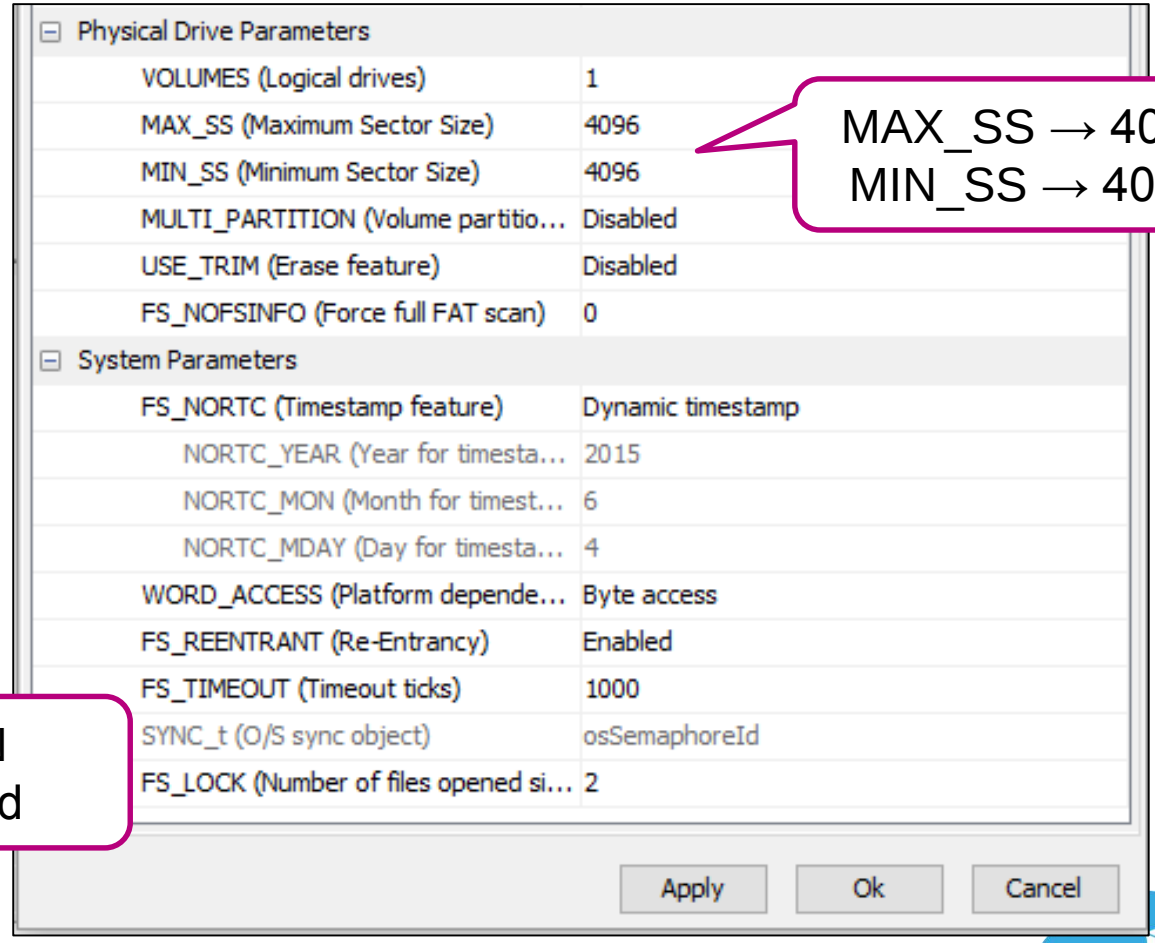


FATFS in Middlewares

# FatFs Configuration






CODE_PAGE
→ Latin 1 (Windows)

USE_LFN
→ Disabled

LFN_UNICODE
→ ANSI/OEM

MAX_SS → 4096
MIN_SS → 4096

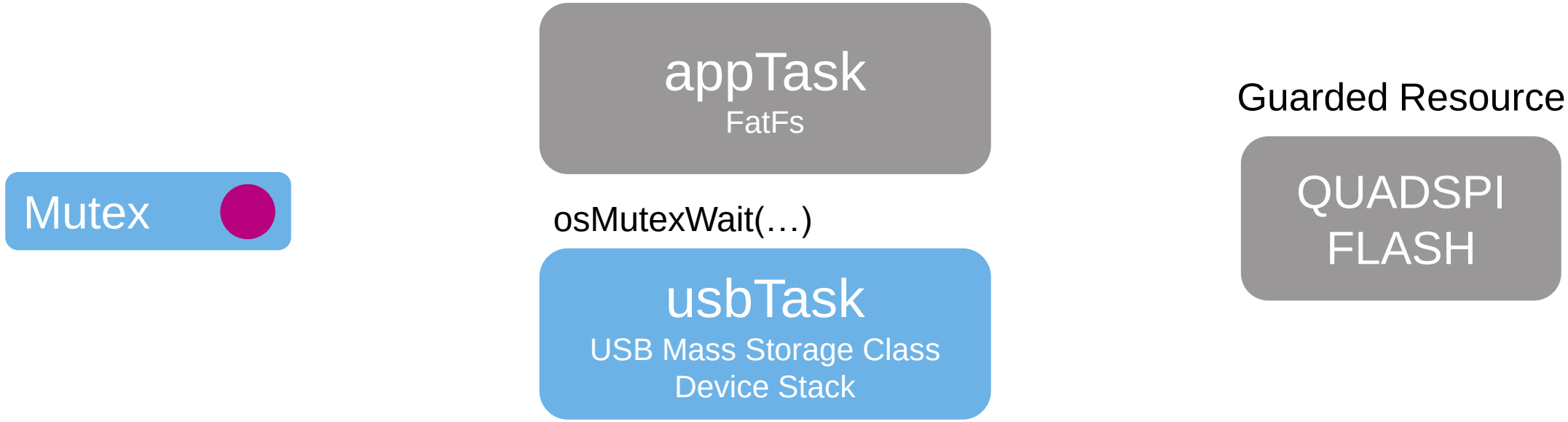
life.augmented

STM32 Cube

# Mutex



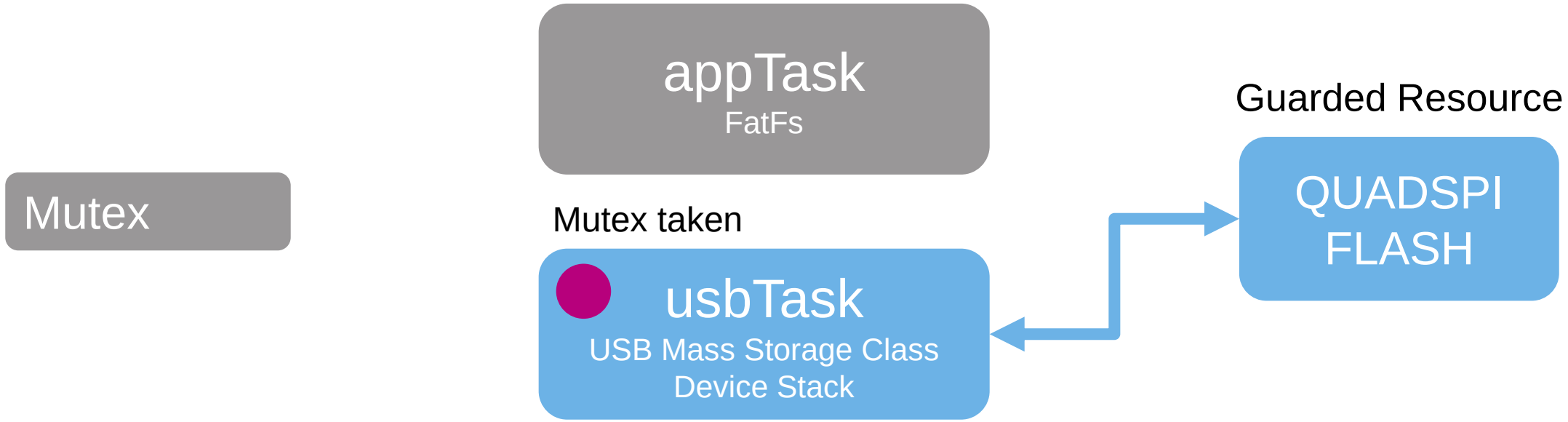
- Mutex needed as we can’t have concurrent access to the QSPI FLASH



# Mutex



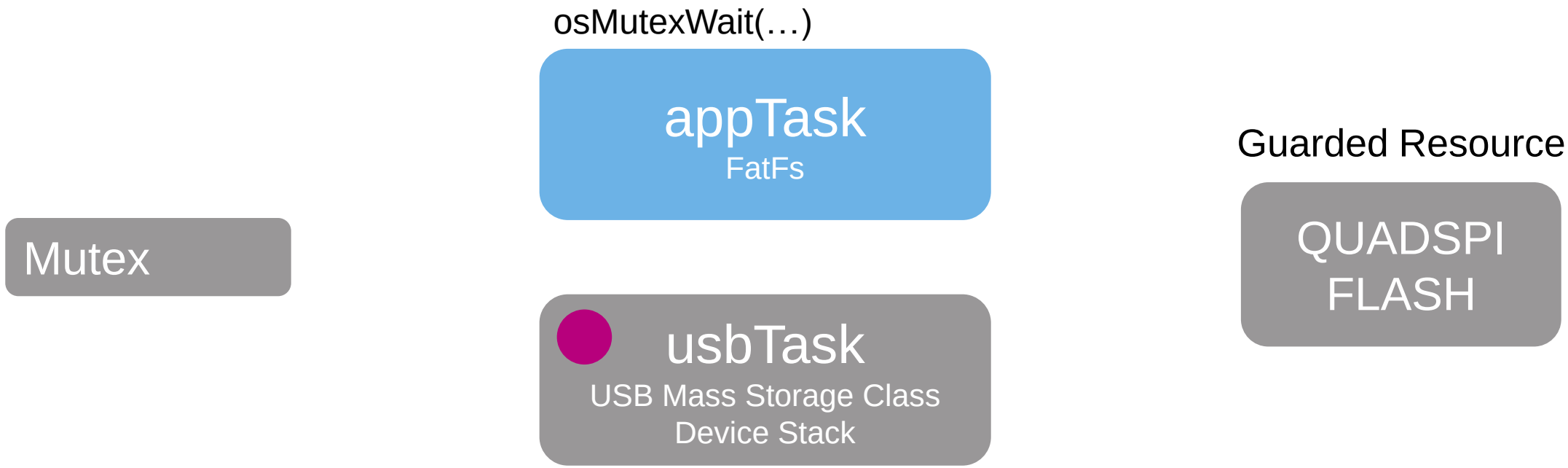
- Mutex needed as we can’t have concurrent access to the QSPI FLASH



# Mutex



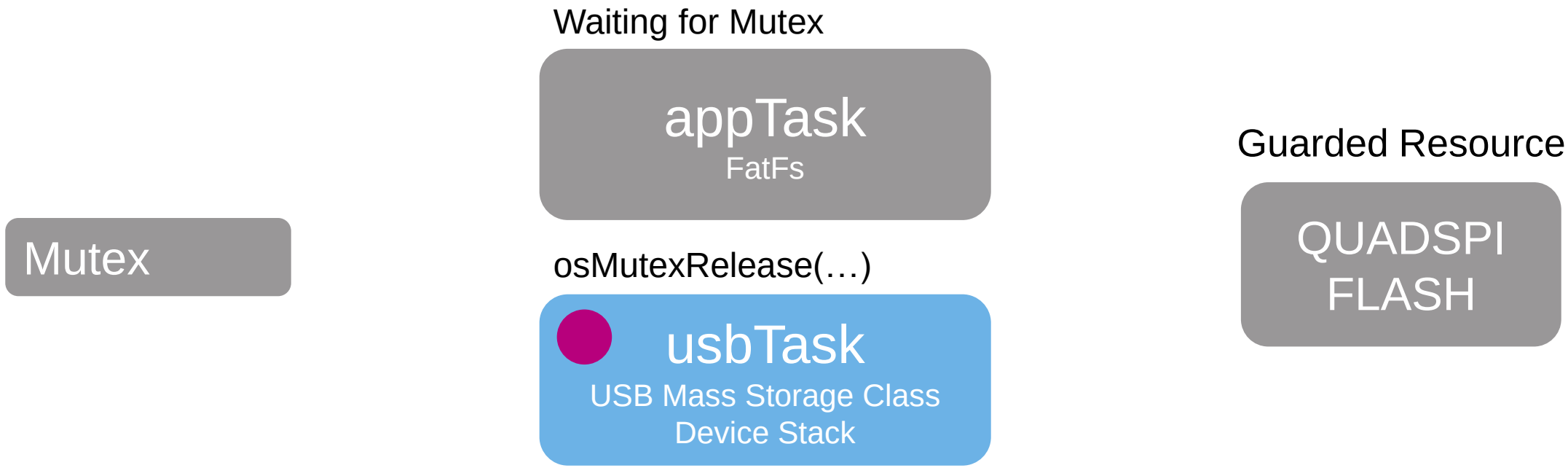
- Mutex needed as we can’t have concurrent access to the QSPI FLASH



# Mutex



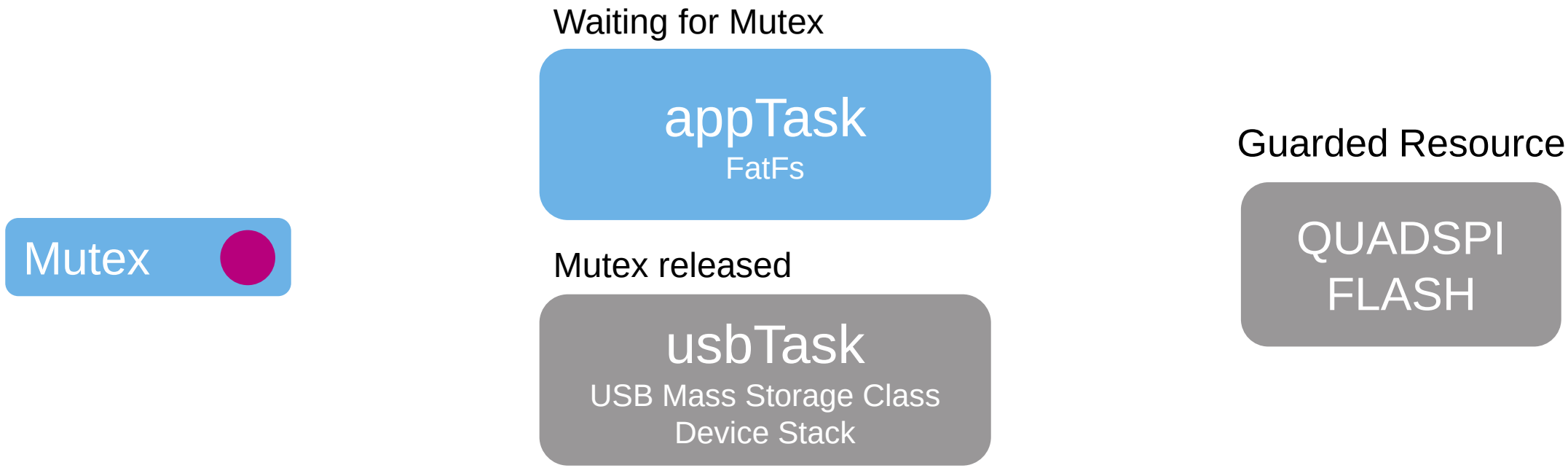
- Mutex needed as we can’t have concurrent access to the QSPI FLASH



# Mutex



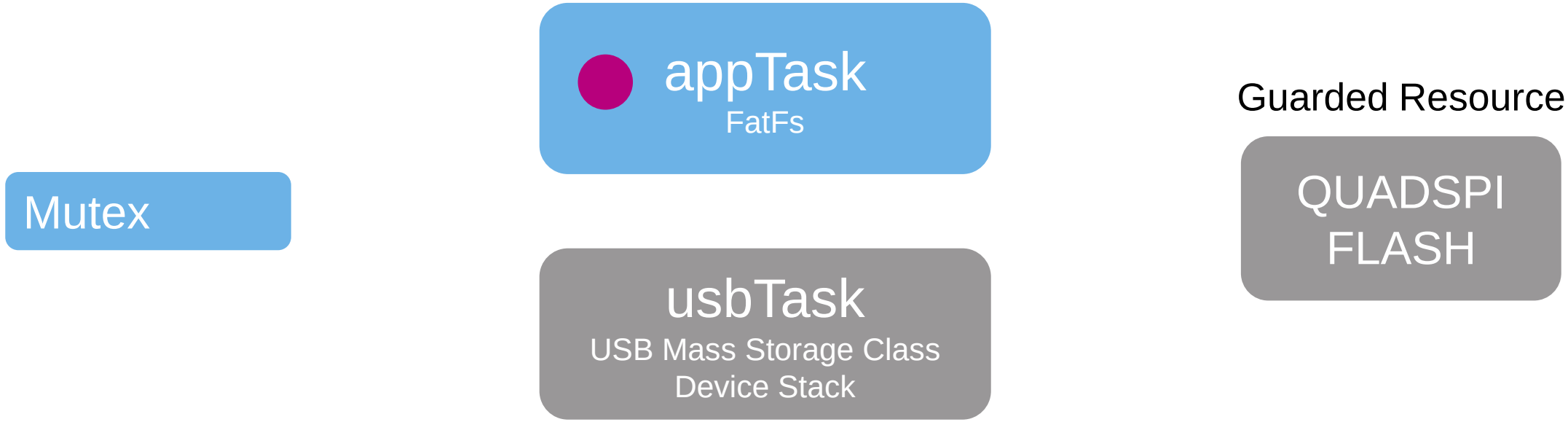
- Mutex needed as we can’t have concurrent access to the QSPI FLASH



# Mutex



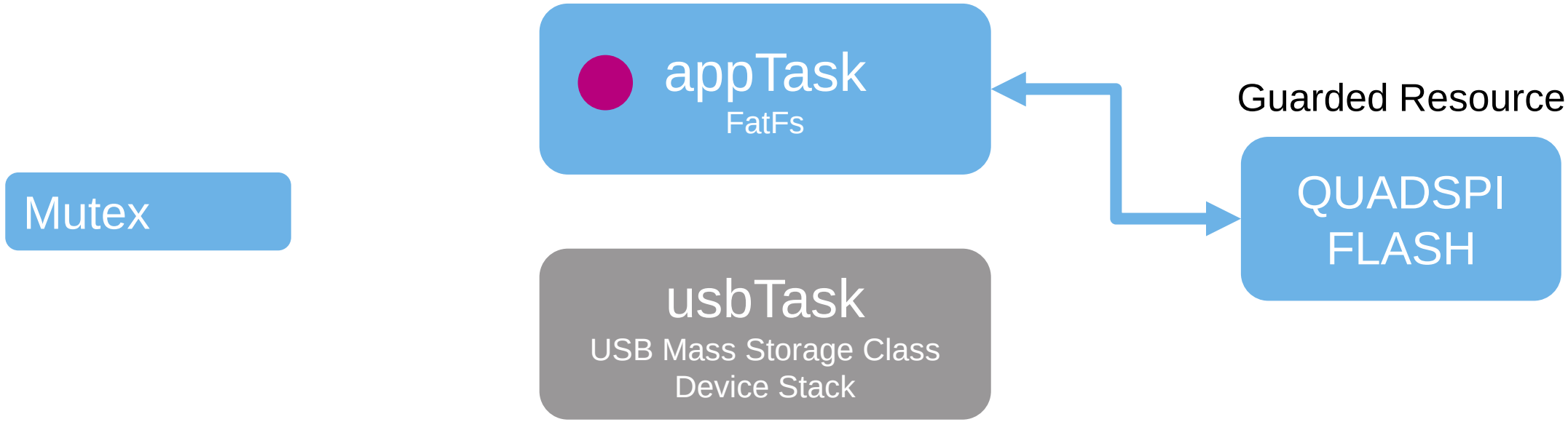
- Mutex needed as we can’t have concurrent access to the QSPI FLASH



# Mutex



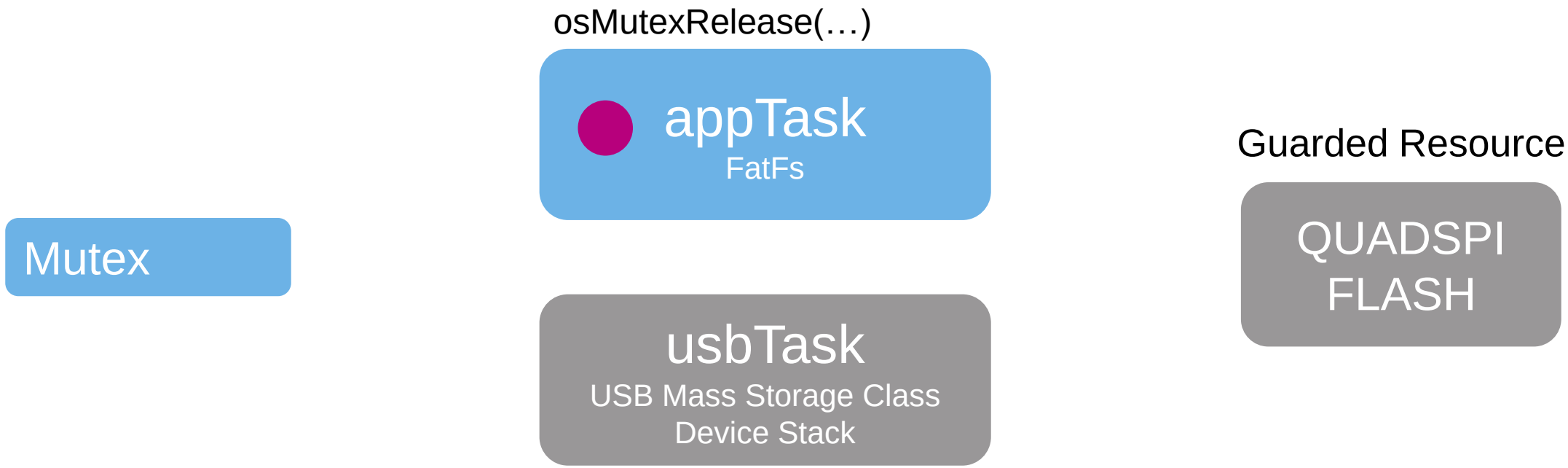
- Mutex needed as we can’t have concurrent access to the QSPI FLASH



# Mutex



- Mutex needed as we can’t have concurrent access to the QSPI FLASH



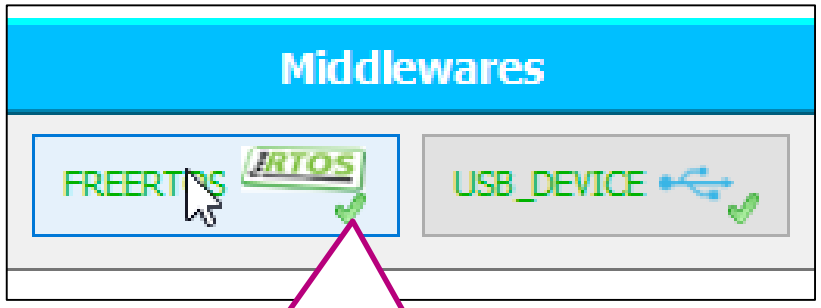
# Mutex

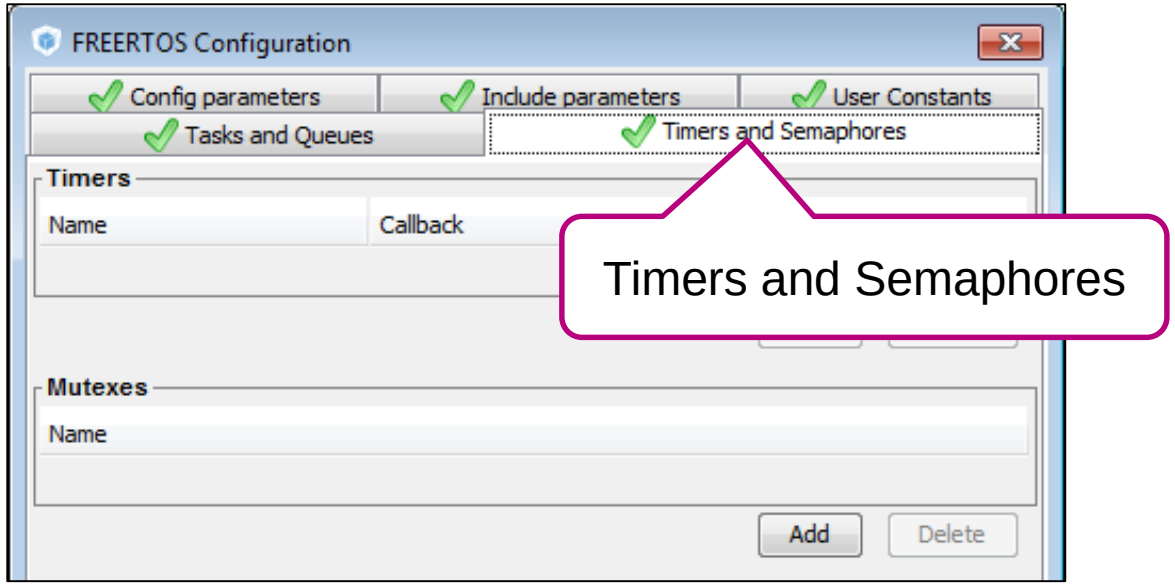


- Mutex needed as we can’t have concurrent access to the QSPI FLASH

# 3 FreeRTOS Configuration Change

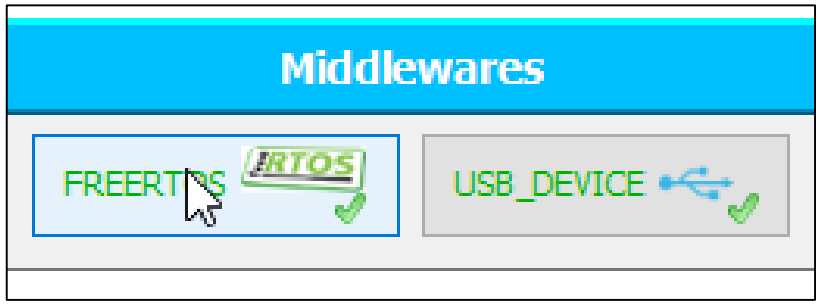


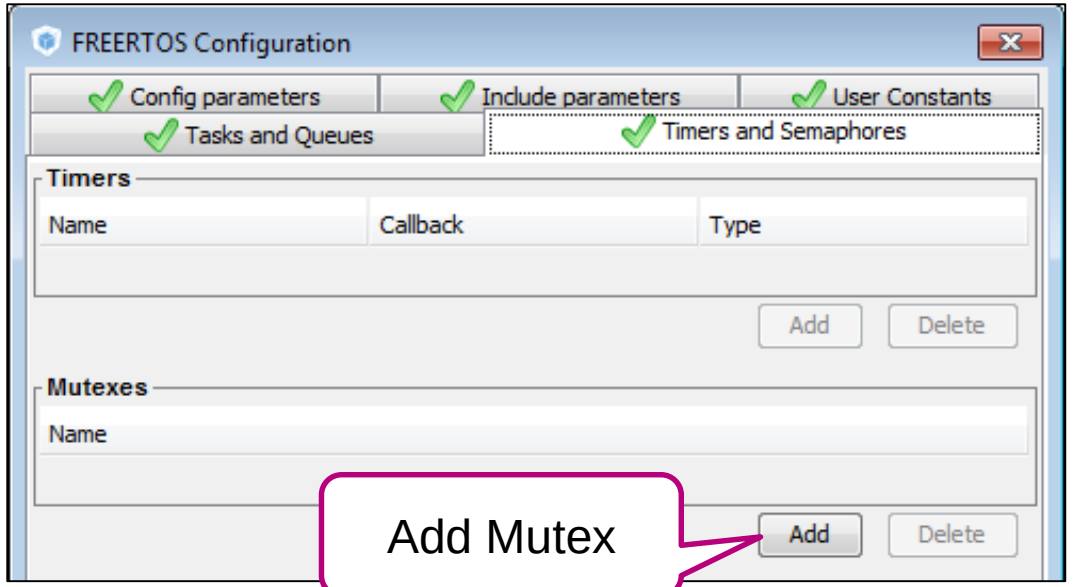


# 3 FreeRTOS Configuration Change

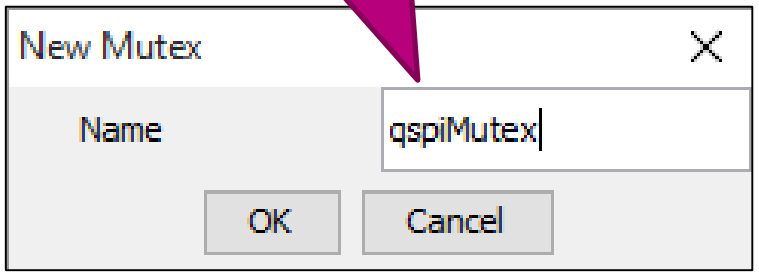





Add Mutex

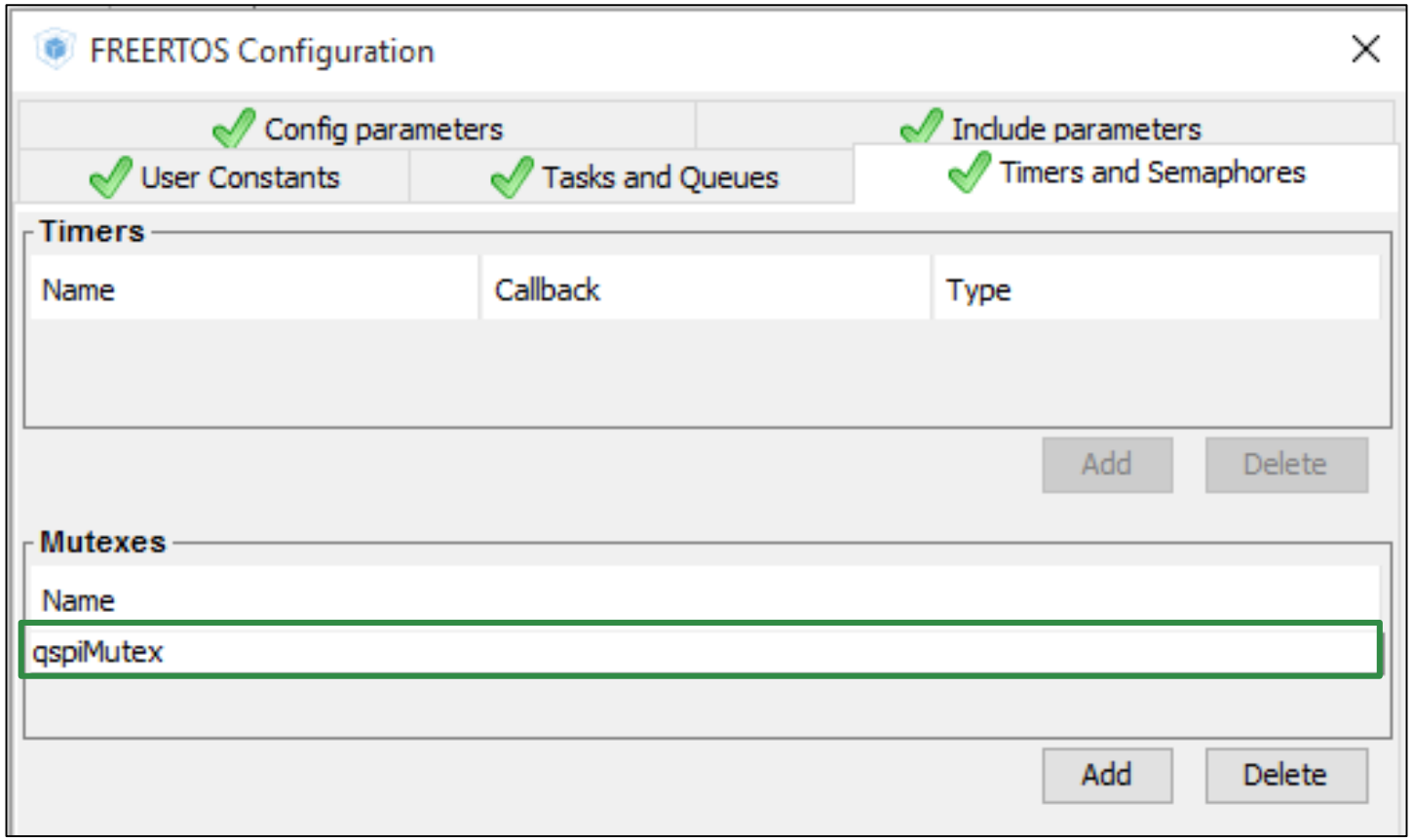
Name → qspiMutex



# Checkpoint #6



RTOS

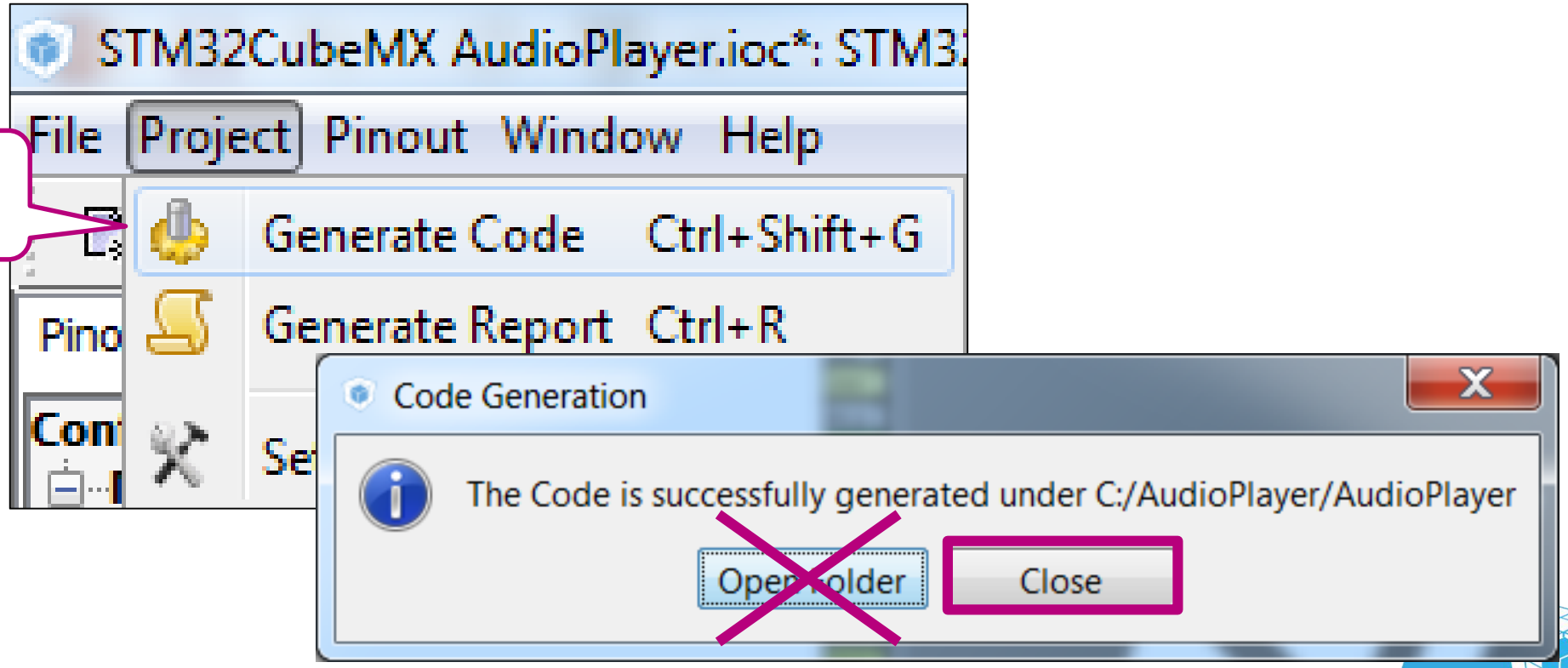



# 3 Save and generate the project



Do not open the project yet

Generate Code
(Ctrl+Shift+G)



# 3 Apply small patch



- Apply patch from folder
  **C:\STM32L4_Workshop\HandsOn\2_Putting_All_Together\Patch\STEP3_FileSystem**
  to root folder of your project
  (C:\AudioPlayer\AudioPlayer\)

- It contains the following files:
  .\Src\
    freertos.c
    user_diskio.c

This operation is STM32CubeMX non-intrusive. You can re-generate the project later and the modifications will be retained.

- These files contains the main needed source code the user has to add to have the FatFS working together with QSPI FLASH storage

# 3 user_diskio.c added content



- user_diskio.c
  - USER_initialize (…)
  - USER_status (…)

  } Storage initialization and Status info

  - USER_read (…)
  - USER_write (…)

  } Read / Write operations

  - USER_ioctl (…)

  } IO control

Based on BSP driver in STM32CubeL4 HAL

# 3 FatFS test flow



- f_mount(…) – mount the storage
  - f_mkfs(…) – create filesystem if not existing
- f_open(…) – open/create file with name FATFSOK
- f_printf(…) – write “FatFS is working properly.“ into FATFSOK file
- f_close(…) – close the file, no more needed

**FATFSOK file in the root of the connected storage**
**Content should be (FatFS is working properly.)**

# 3

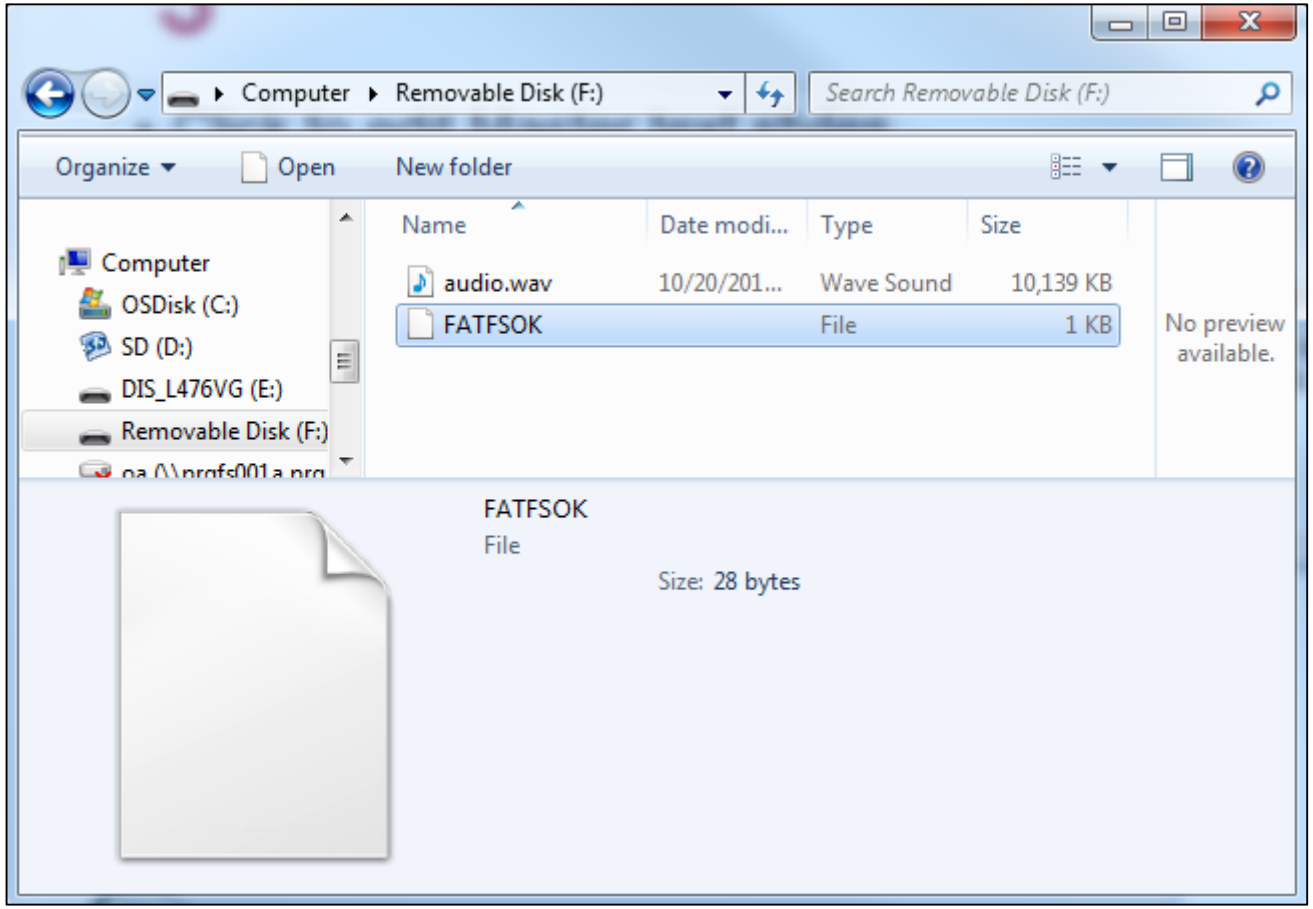
# Let’s try it out

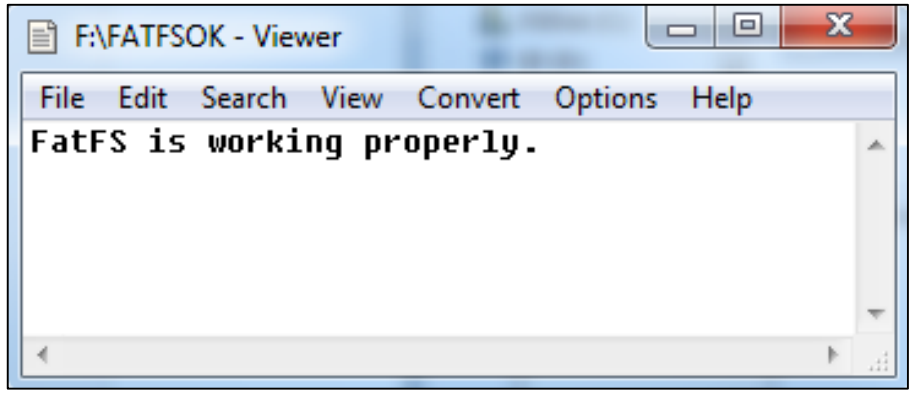


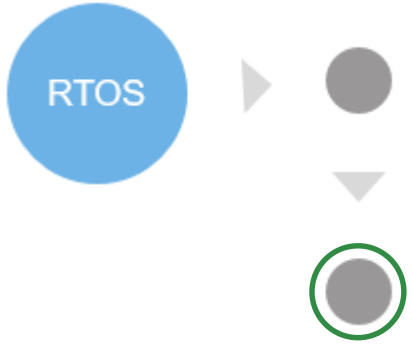
1. Open the project
2. Compile and download
3. Press the reset button on discovery once download has finished
4. Try to connect microUSB to PC

# Checkpoint #7





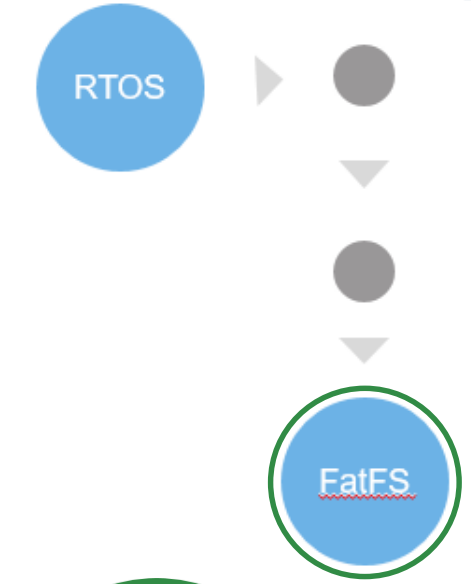

life.augmented

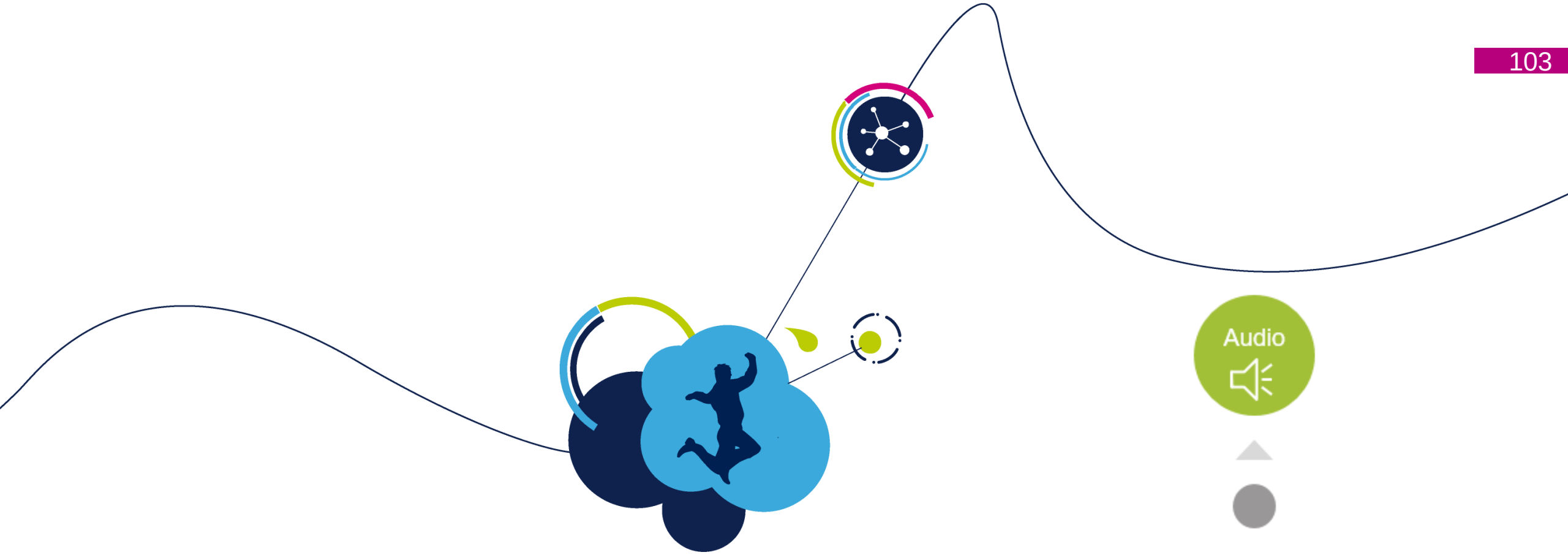
STM32 Cube

# Checkpoint #8



- LED_GREEN ON before qspiMutex take in appTask
- LED_GREEN OFF after qspiMutex release in appTask
- LED_RED ON on qspiMutex take in usbTask
- LED_RED OFF on qspiMutex release in usbTask



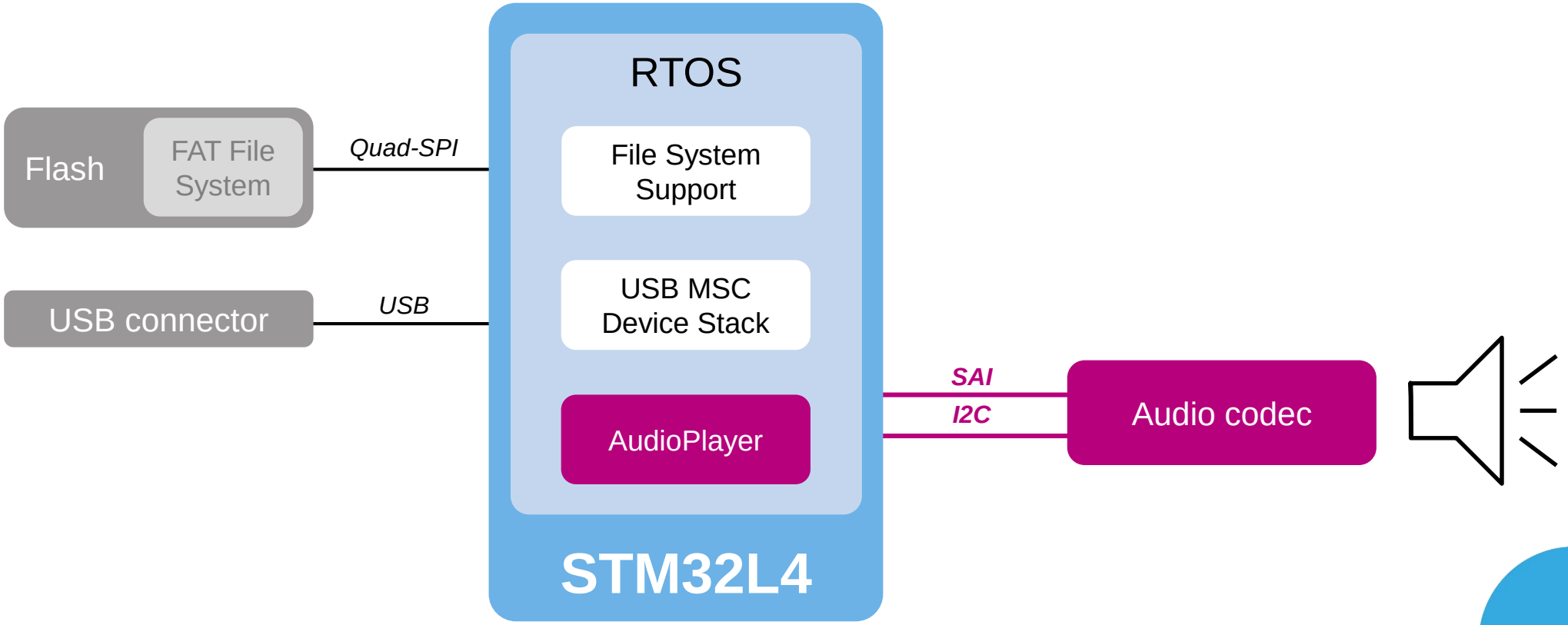



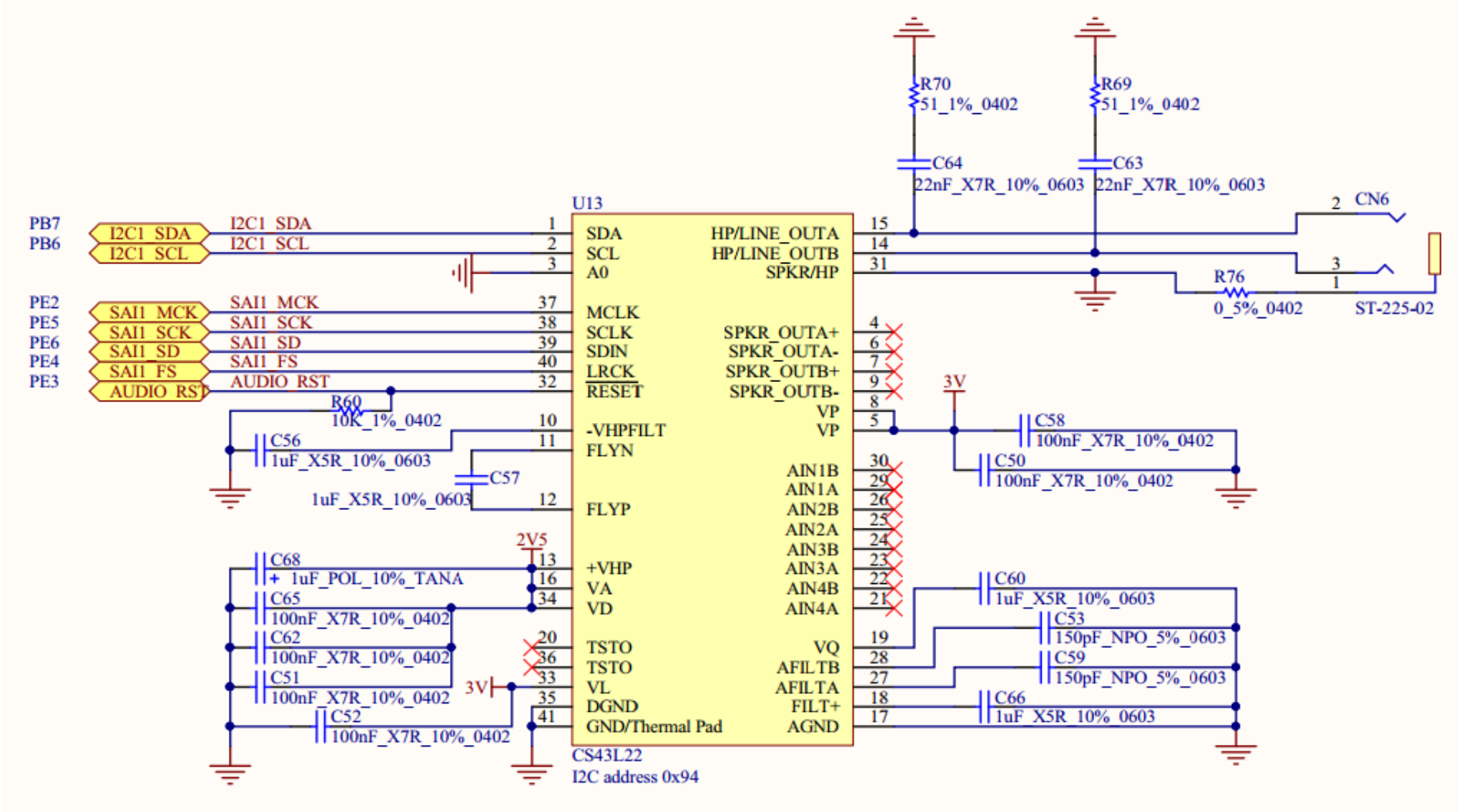
# STEP4 – Audio

# 4 Adding “playback” functionality



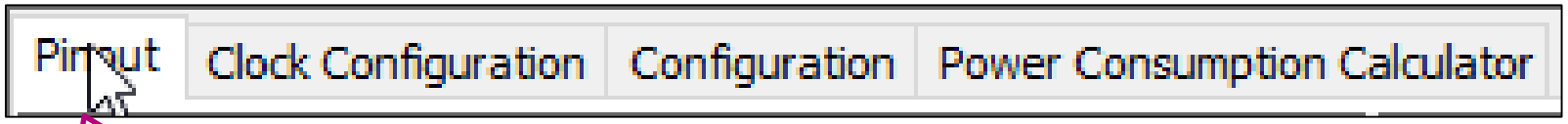
- We want to hear our “audio.wav” file in headphones



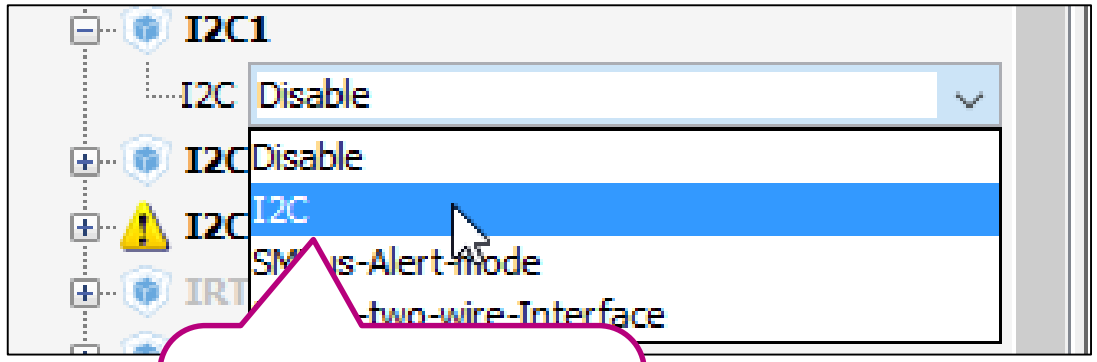

U13
CS43L22
I2C address 0x94
SDA
SCL
A0
MCLK
SCLK
SDIN
LRCK
RESET
-VHPFILT
FLYN
FLYP
+VHP
VA
VD
TSTO
TSTO
VL
DGND
GND/Thermal Pad
HP/LINE_OUTA
HP/LINE_OUTB
SPKR/HP
SPKR_OUTA+
SPKR_OUTA-
SPKR_OUTB+
SPKR_OUTB-
VP
VP
AIN1B
AIN1A
AIN2B
AIN2A
AIN3B
AIN3A
AIN4B
AIN4A
VQ
AFILTB
AFILTA
FILT+
AGND
PB7
PB6
I2C1_SDA
I2C1_SCL
PE2
PE5
PE6
PE4
PE3
SAI1_MCK
SAI1_SCK
SAI1_SD
SAI1_FS
AUDIO_RST
R70
51_1%_0402
R69
51_1%_0402
C64
22nF_X7R_10%_0603
C63
22nF_X7R_10%_0603
CN6
R76
0_5%_0402
ST-225-02
R60
10K_1%_0402
C56
1uF_X5R_10%_0603
C57
1uF_X5R_10%_0603
3V
C58
100nF_X7R_10%_0402
C50
100nF_X7R_10%_0402
2V5
C68
1uF_POL_10%_TANA
C65
100nF_X7R_10%_0402
C62
100nF_X7R_10%_0402
C51
100nF_X7R_10%_0402
C52
100nF_X7R_10%_0402
C60
1uF_X5R_10%_0603
C53
150pF_NPO_5%_0603
C59
150pF_NPO_5%_0603
C66
1uF_X5R_10%_0603

# 4 Audio Configuration

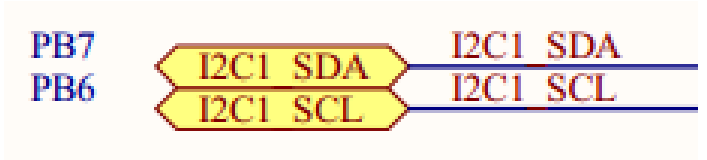





Go to Pinout




Under I2C1 peripheral select **I2C** interface

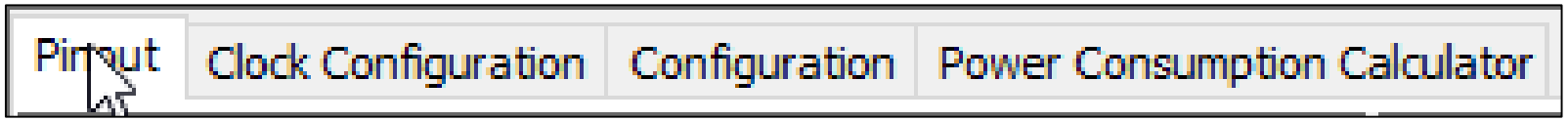


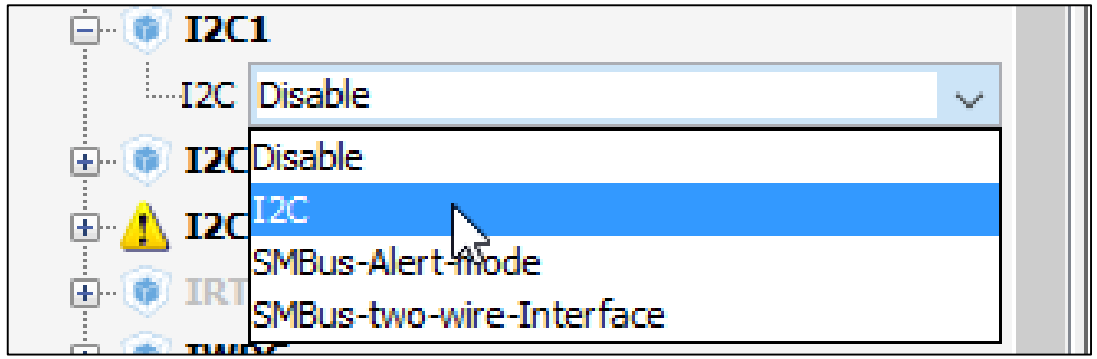

life.augmented

STM32 Cube

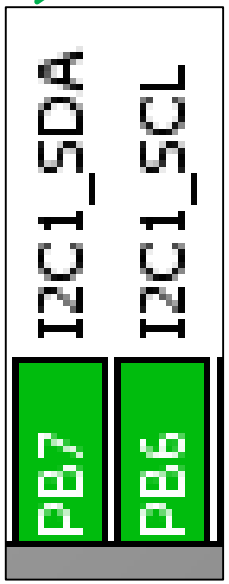
# Audio Configuration

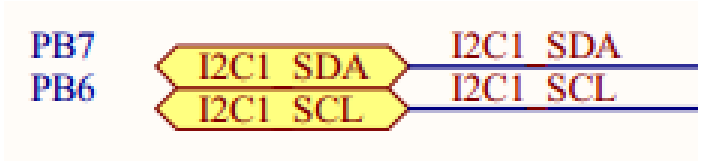





The corresponding pins are assigned and configured automatically!



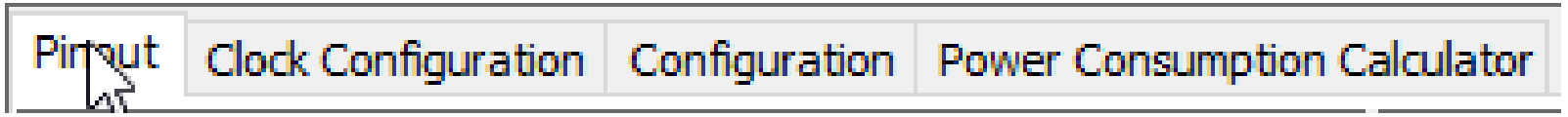



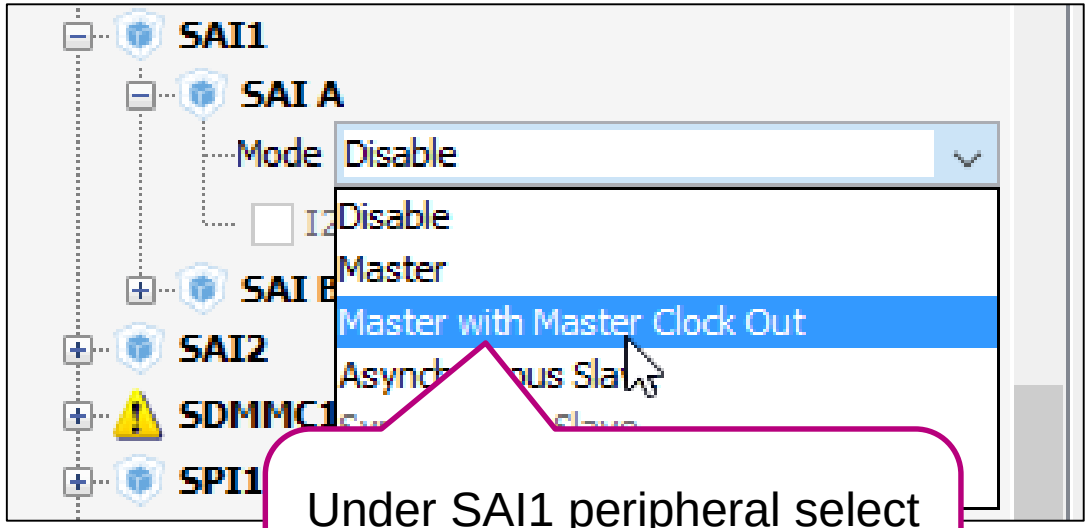

life.augmented

STM32 Cube

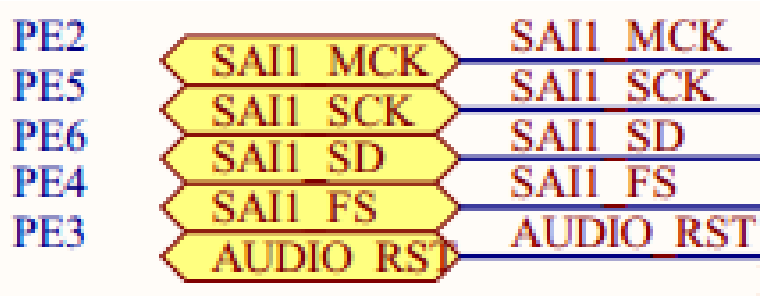
# Audio Configuration






Under SAI1 peripheral select **SAI A** interface to **Master with Master Clock Out**

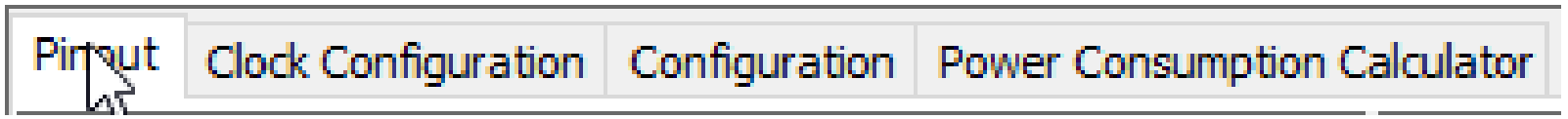


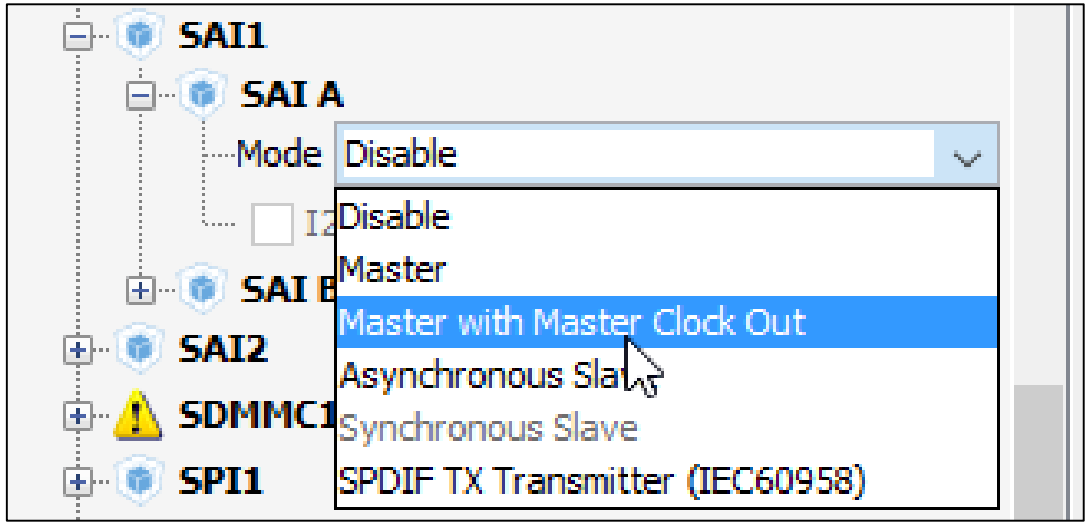

life.augmented

STM32 Cube

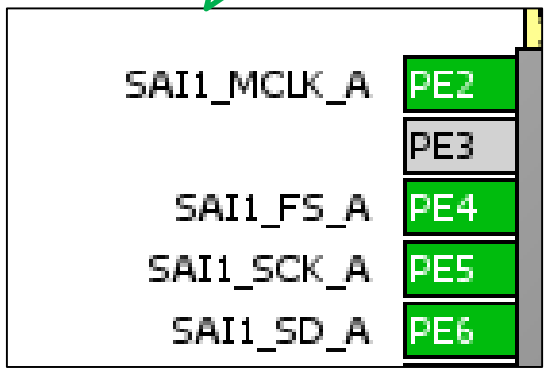
# Audio Configuration

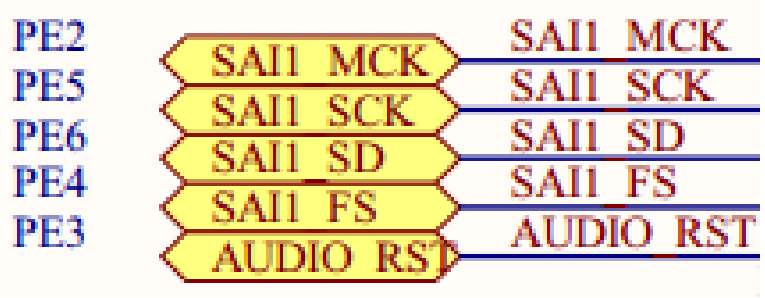





The corresponding pins are assigned and configured automatically!



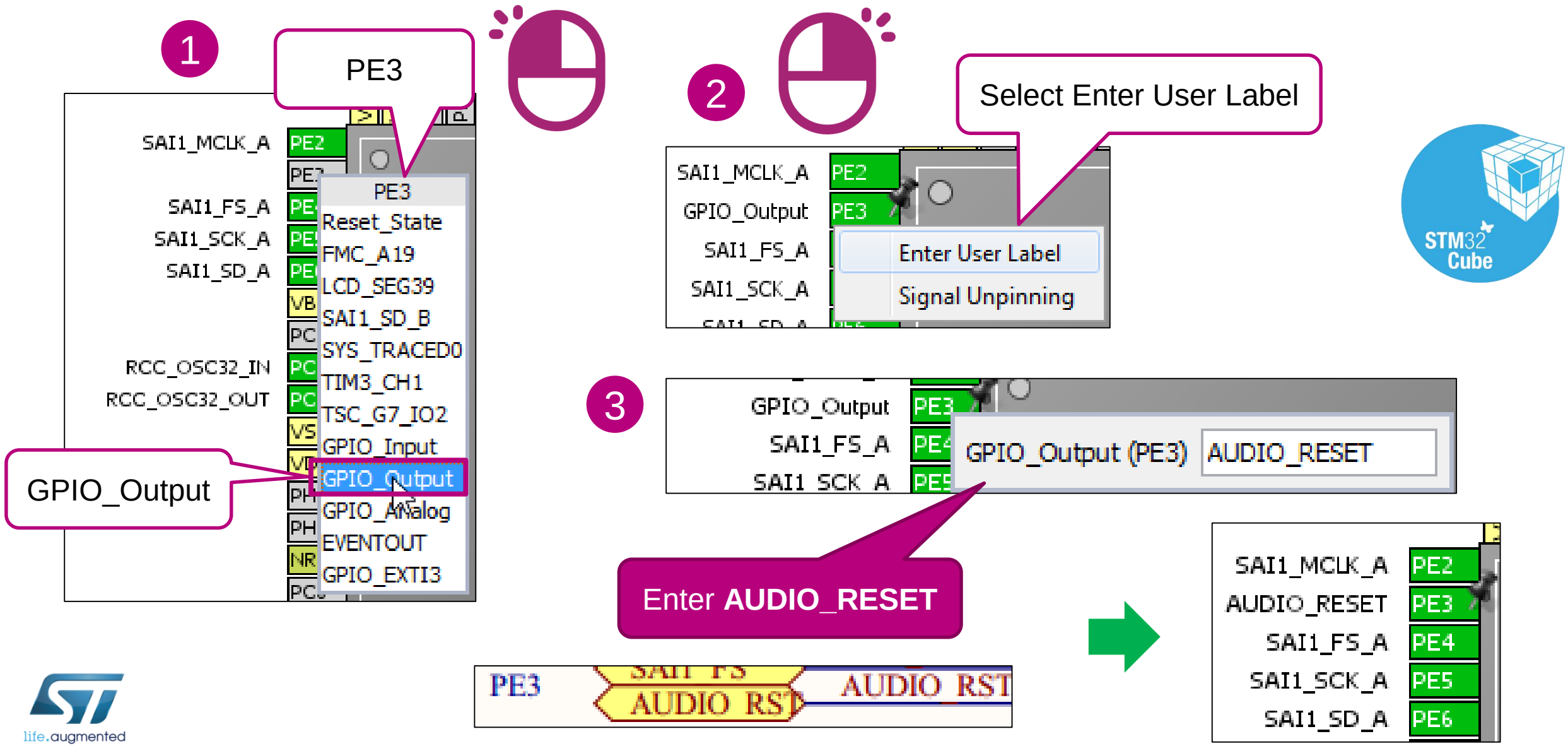




life.augmented

STM32 Cube

# 4 Audio Configuration



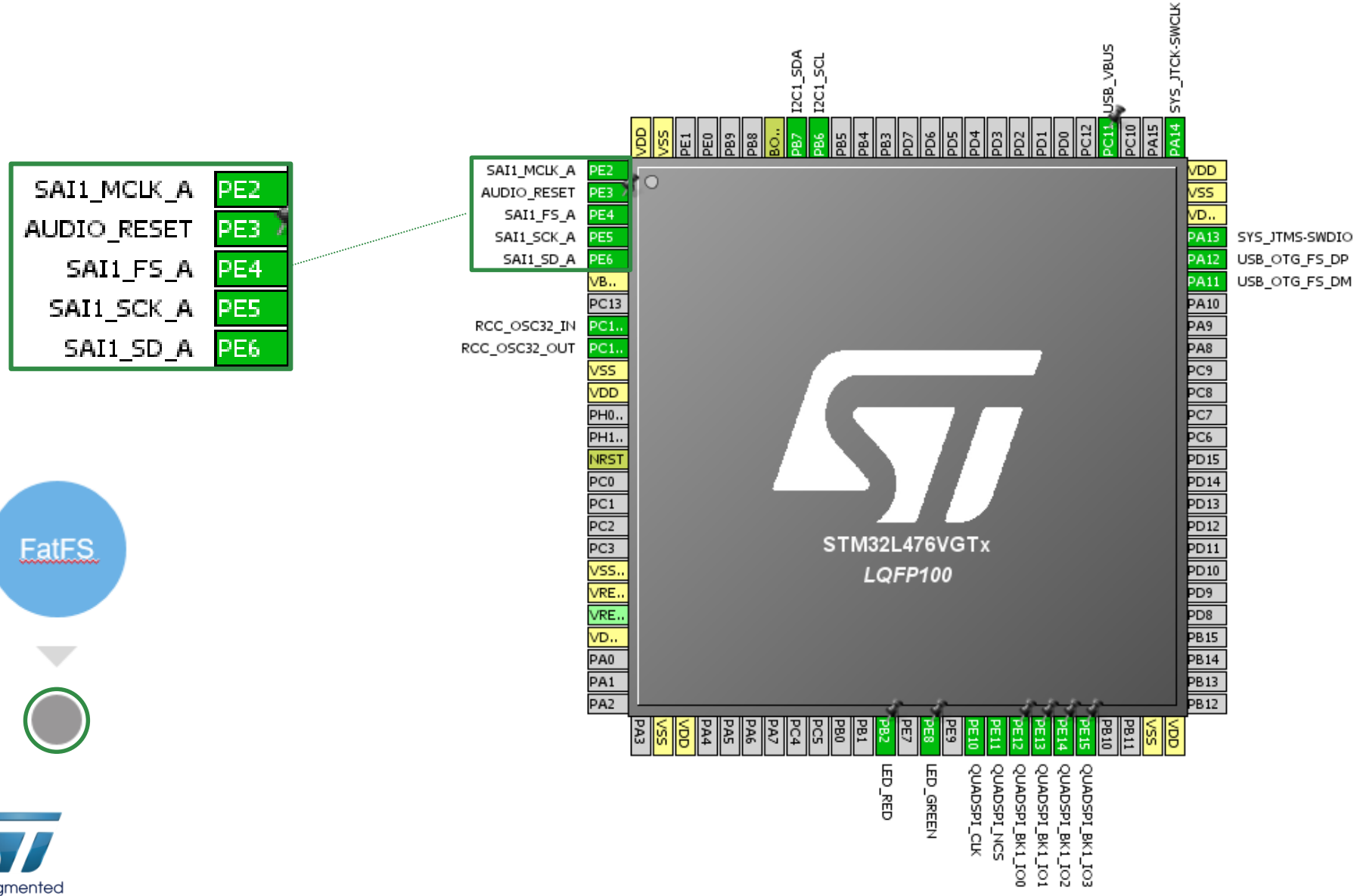



life.augmented

# 4

# Checkpoint #9






FatFS

life.augmented

STM32 Cube

# Audio Configuration



## SAI MCLK configuration

| audio.wav | |
|---|---|
| **WAVE Information** | |
| Length | 00:58 |
| Compression | PCM |
| Bitrate (bps) | 16 |
| Frequency (Hz) | 44 100 |
| Mode | Stereo |

CIRRUS LOGIC

**CS43L22**

### 4.6 Serial Port Clocking

The CS43L22 serial audio interface port operates either as a slave or master, determined by the M/$\overline{S}$ bit. It accepts externally generated clocks in Slave Mode and will generate synchronous clocks derived from an input master clock (MCLK) in Master Mode. Refer to the tables below for the required setting in register 05h and 06h associated with a given MCLK and sample rate.

| Referenced Control | Register Location |
|---|---|
| M/$\overline{S}$ | "Master/Slave Mode" on page 40 |
| Register 05h | "Clocking Control (Address 05h)" on page 38 |
| Register 06h | "Interface Control 1 (Address 06h)" on page 40 |

| MCLK (MHz) | Sample Rate, Fs (kHz) | SPEED[1:0] (AUTO='0'b) | 32kGROUP | VIDEOCLK | RATIO[1:0] | MCLKDIV2 |
|---|---|---|---|---|---|---|
| 12.2880 | 8.0000 | 11 | 1 | 0 | 00 | 0 |
| | 12.0000 | 11 | 0 | 0 | 00 | 0 |
| | 16.0000 | 10 | 1 | 0 | 00 | 0 |
| | 24.0000 | 10 | 0 | 0 | 00 | 0 |
| | 32.0000 | 01 | 1 | 0 | 00 | 0 |
| | 48.0000 | 01 | 0 | 0 | 00 | 0 |
| | 96.0000 | 00 | 0 | 0 | 00 | 0 |
| 11.2896 | 11.0250 | 11 | 0 | 0 | 00 | 0 |
| | 22.0500 | 10 | 0 | 0 | 00 | 0 |
| | 44.1000 | 01 | 0 | 0 | 00 | 0 |
| | 88.2000 | 00 | 0 | 0 | 00 | 0 |

life.augmented

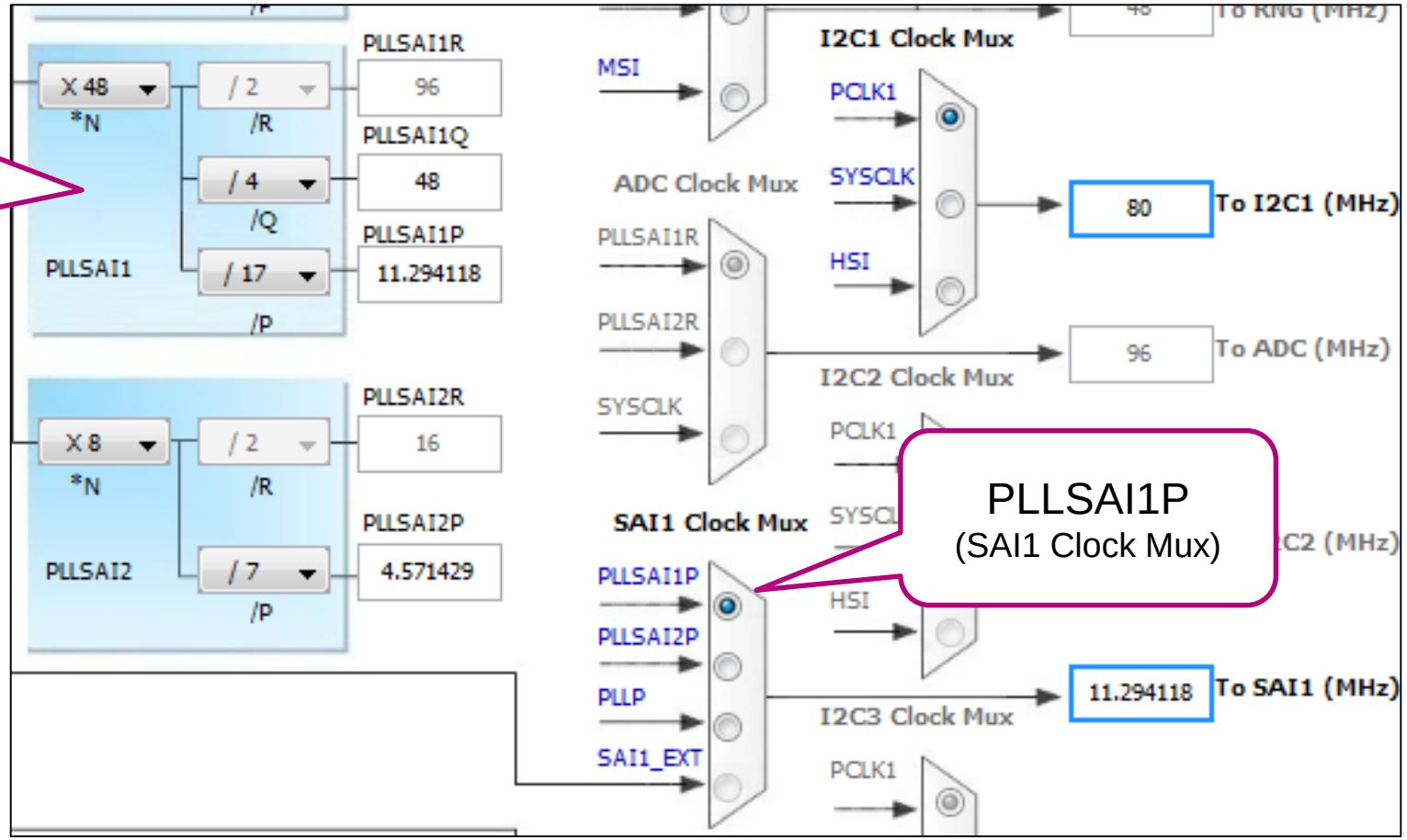
STM32 Cube

# Audio Configuration



Pinout | Clock Configuration | Configuration | Power Consumption Calculator

*N → x48
/Q → /4
/P → /17
(PLLSAI1)

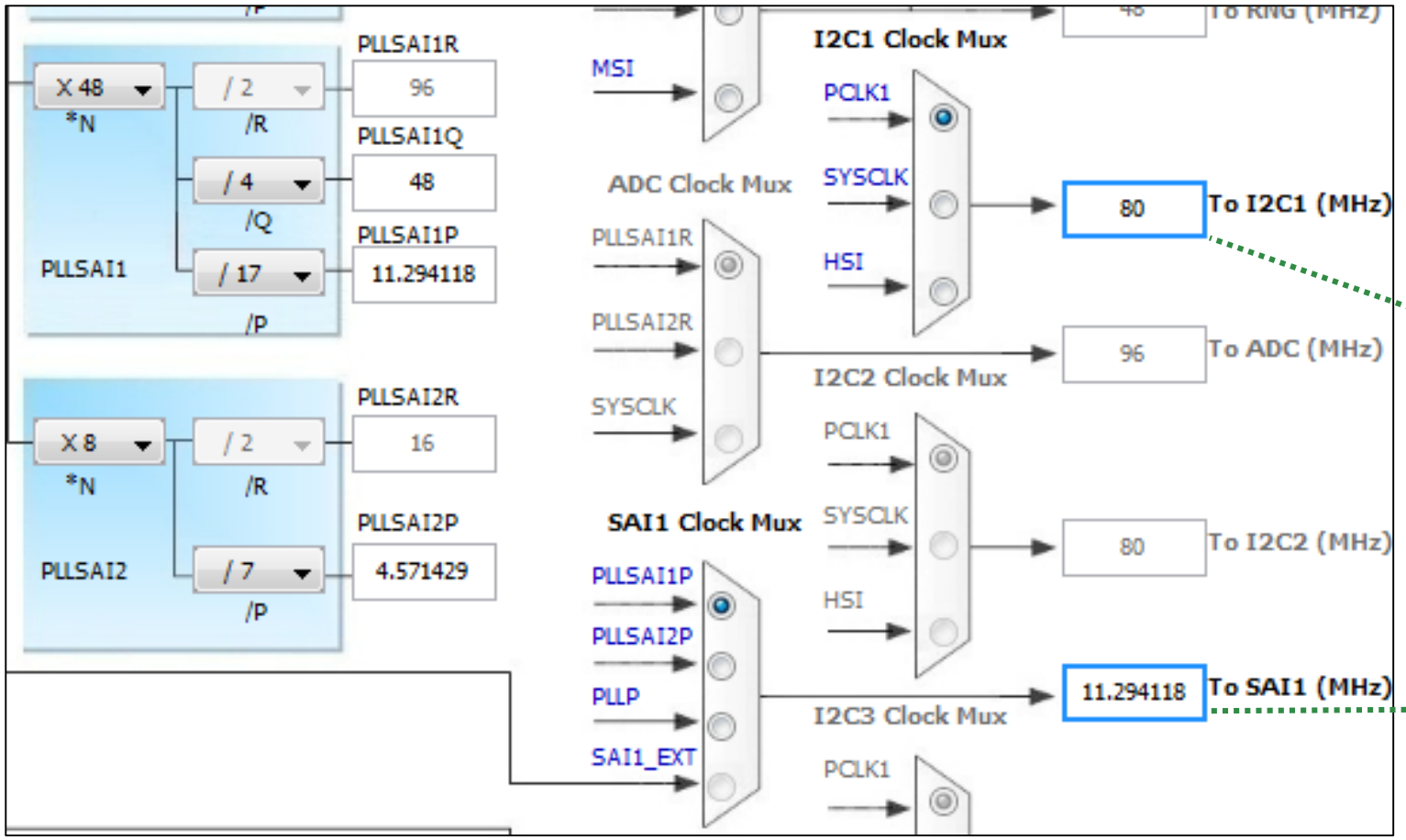



PLLSAI1P
(SAI1 Clock Mux)

# Checkpoint #10



FatFS



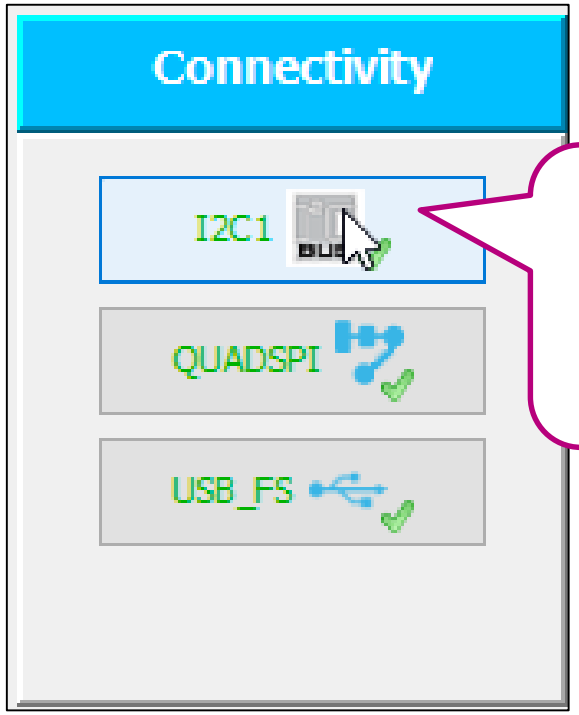

80MHz
(To I2C1)

11.294118
MHz
(To SAI1)

# Audio Configuration



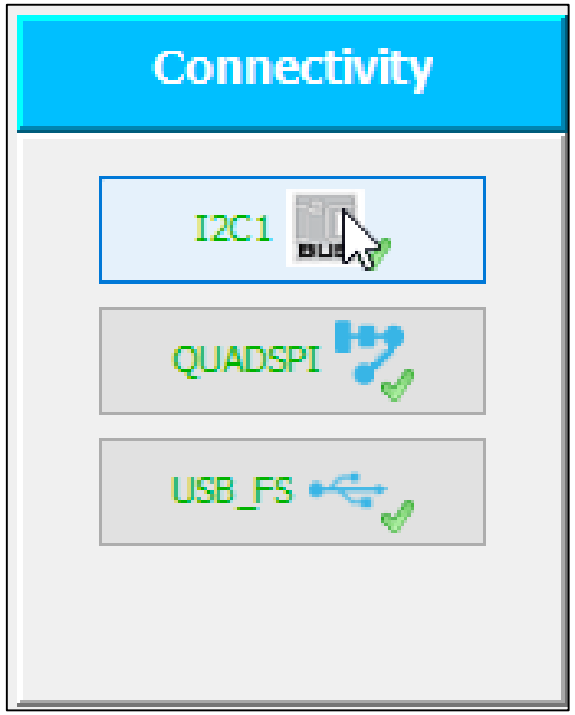

Go to Configuration

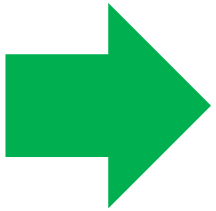


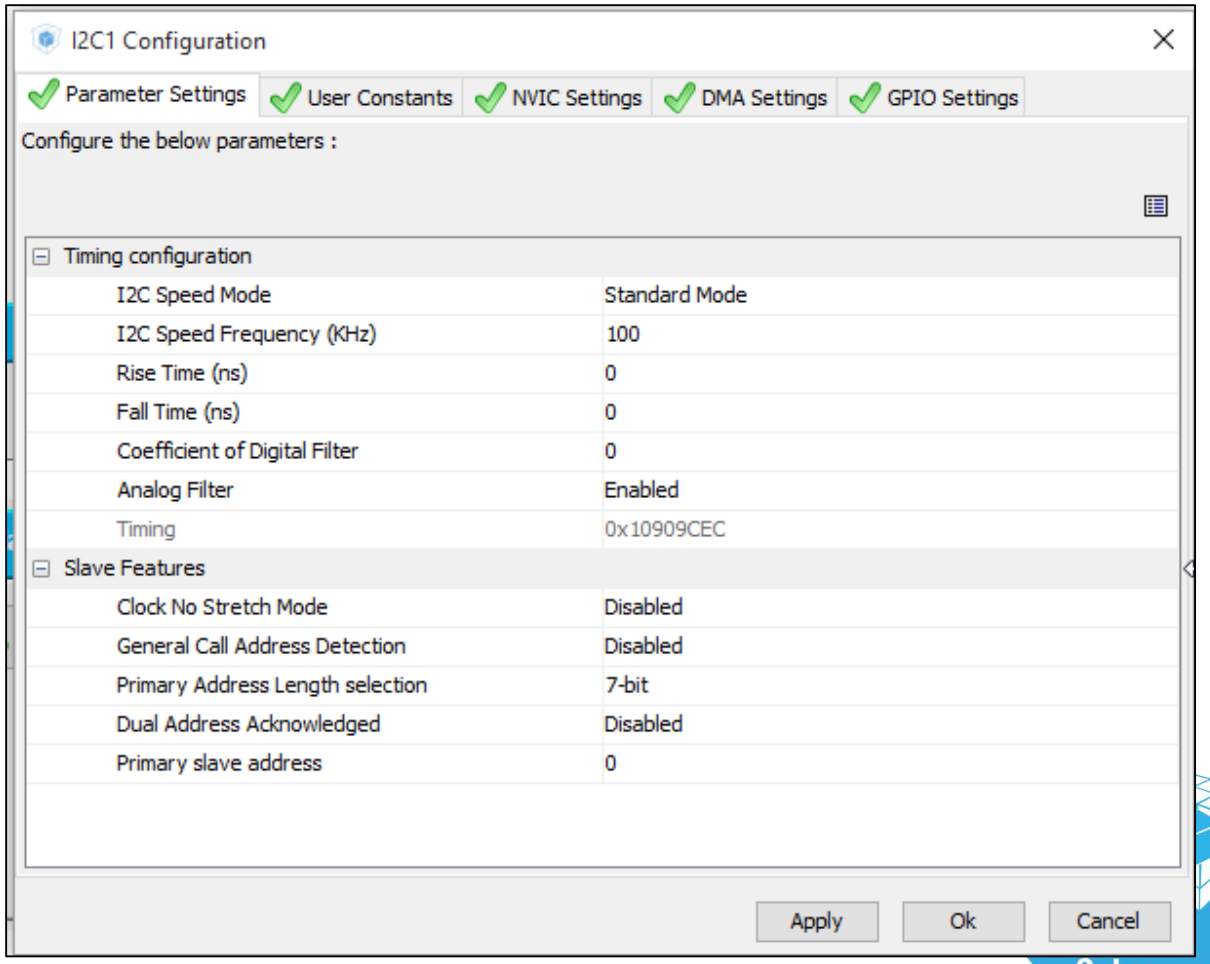

I2C1 in Connectivity

# Audio Configuration





CIRRUS LOGIC®

CS43L22

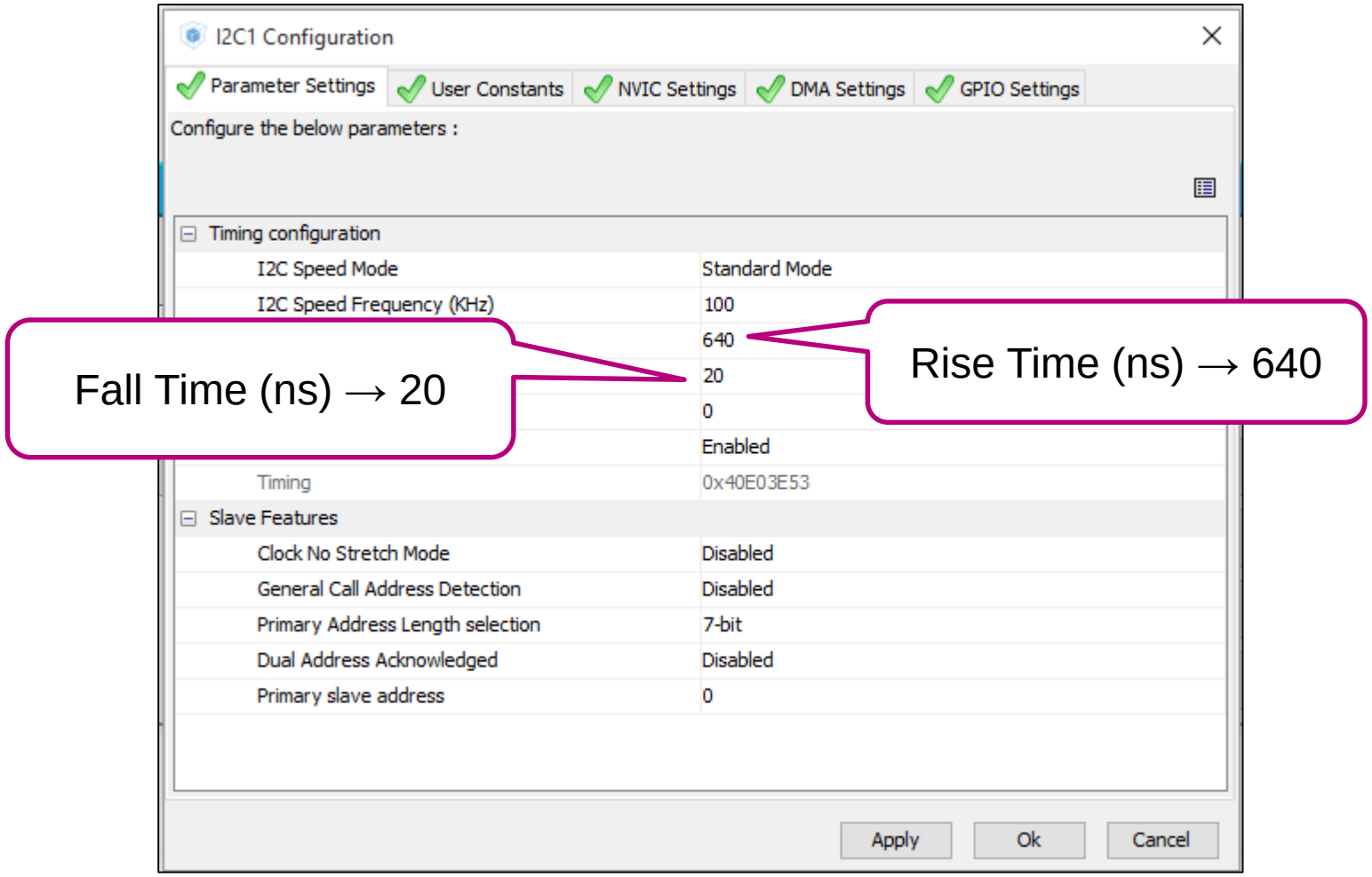
## SWITCHING SPECIFICATIONS - I²C CONTROL PORT

Inputs: Logic 0 = DGND; Logic 1 = V; SDA $C_L$ = 30 pF.

| Parameters | Symbol | Min | Max | Unit |
|---|---|---|---|---|
| SCL Clock Frequency | $f_{scl}$ | - | 100 | kHz |
| RESET Rising Edge to Start | $t_{irs}$ | 550 | - | ns |
| Bus Free Time Between Transmissions | $t_{buf}$ | 4.7 | - | µs |
| Start Condition Hold Time (prior to first clock pulse) | $t_{hdst}$ | 4.0 | - | µs |
| Clock Low time | $t_{low}$ | 4.7 | - | µs |
| Clock High Time | $t_{high}$ | 4.0 | - | µs |
| Setup Time for Repeated Start Condition | $t_{sust}$ | 4.7 | - | µs |
| SDA Hold Time from SCL Falling (Note 13) | $t_{hdd}$ | 0 | - | µs |
| SDA Setup time to SCL Rising | $t_{sud}$ | 250 | - | ns |
| Rise Time of SCL and SDA | $t_{rc}$ | - | 1 | µs |
| Fall Time SCL and SDA | $t_{fc}$ | - | 300 | ns |
| Setup Time for Stop Condition | $t_{susp}$ | 4.7 | - | µs |
| Acknowledge Delay from SCL Falling | $t_{ack}$ | 300 | 1000 | ns |

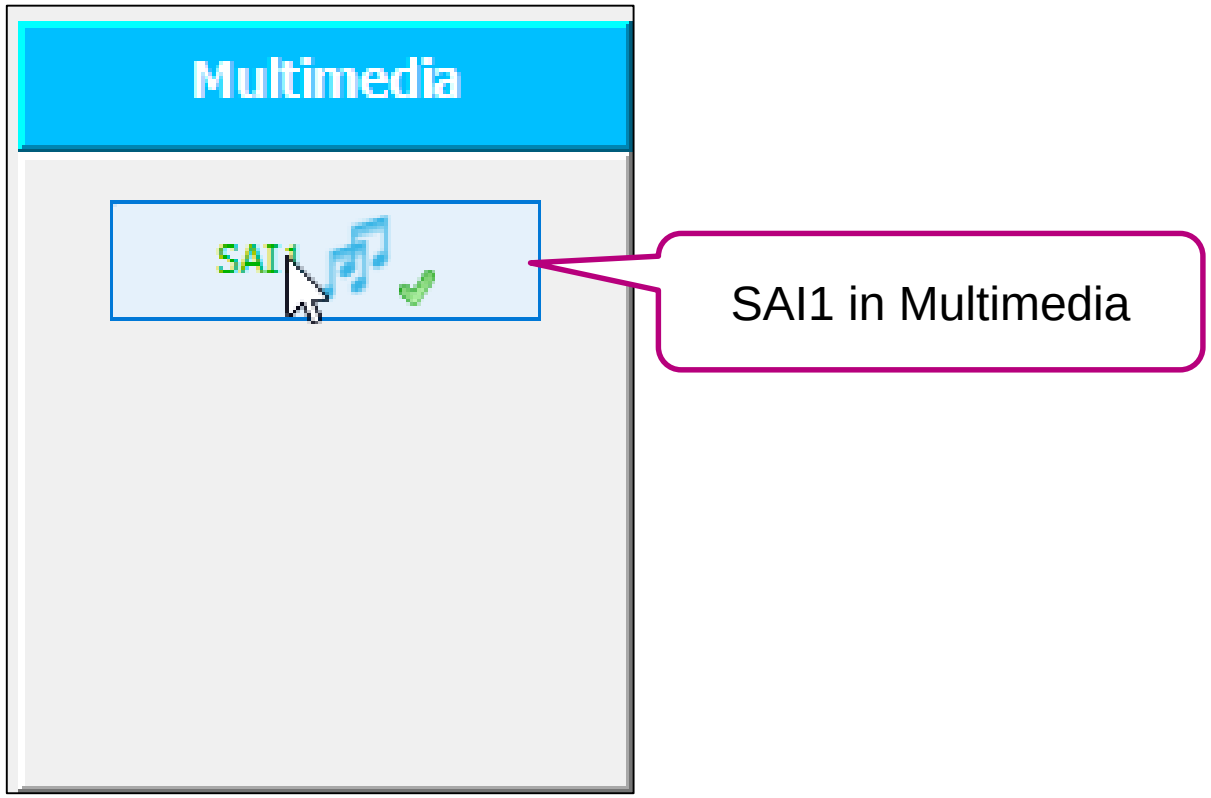
# Audio Configuration





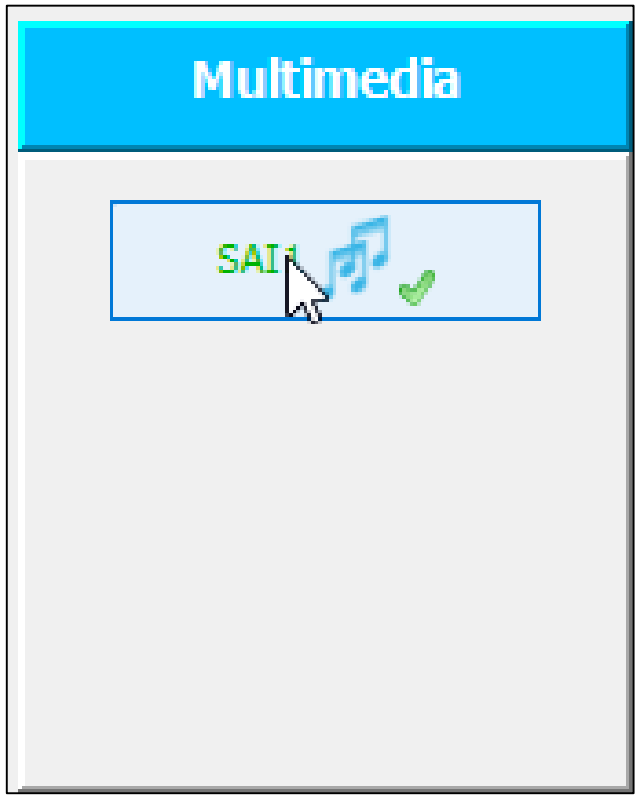
# Audio Configuration

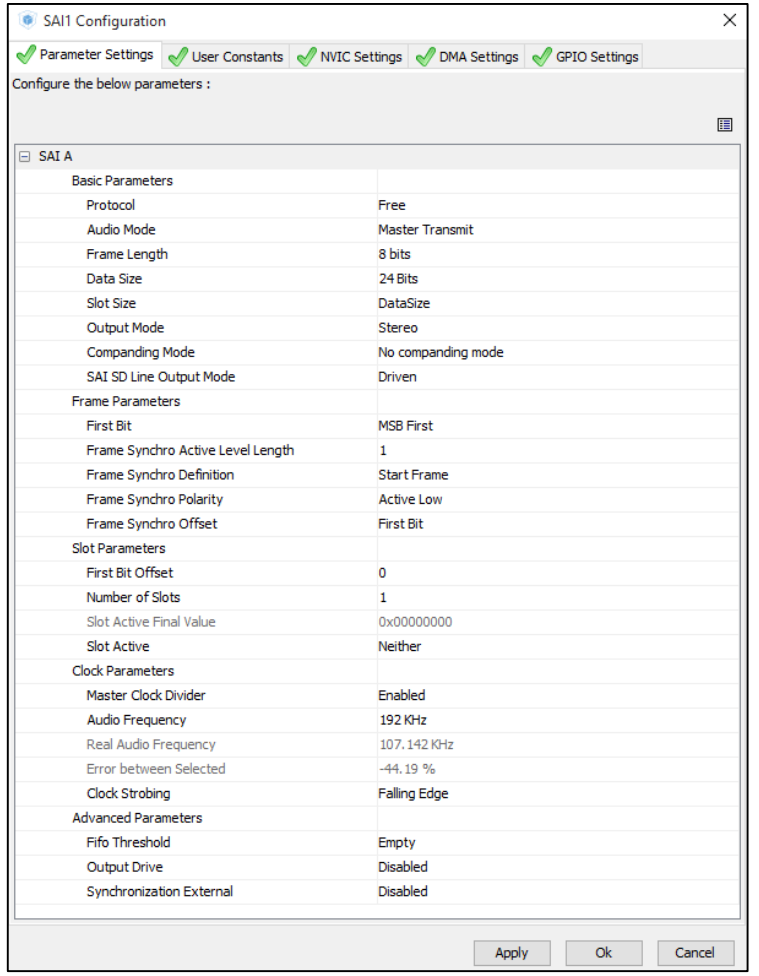


# Audio Configuration

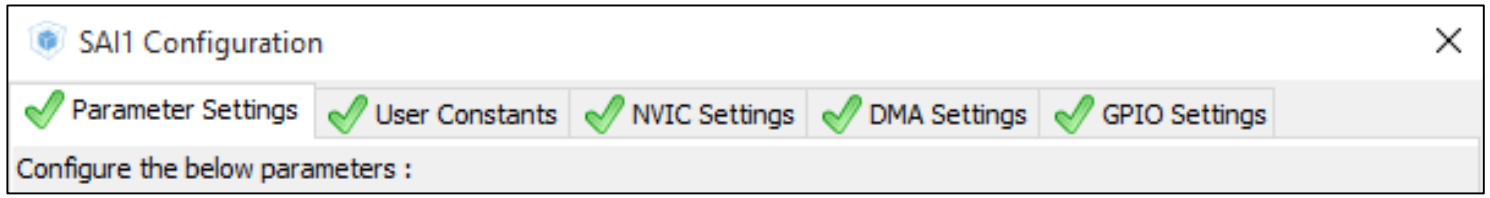


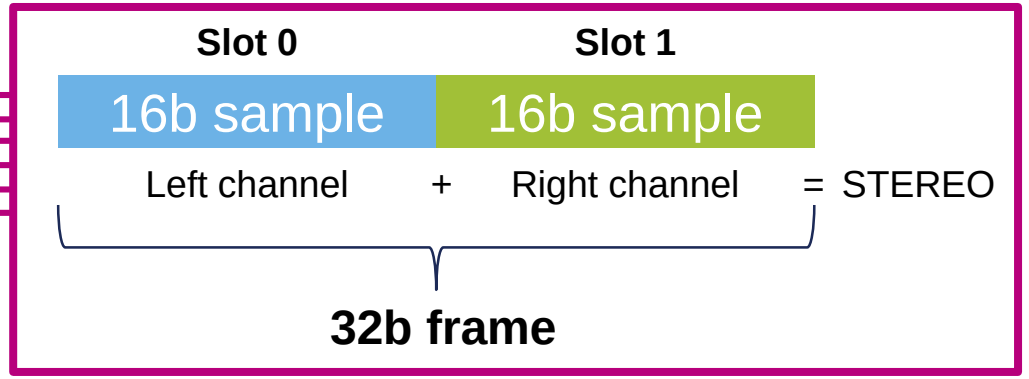



# Audio Configuration






| | | |
|---|---|---|
| 1 | Frame Length | 32 bits |
| 2 | Data Size | 16 Bits |
| 3 | Frame Synchro Active Level Length | 16 |
| 4 | Frame Synchro Offset | Before First Bit |
| 5 | Number of Slots (only Even Values) | 2 |
| 6 | Frame Synchro Definition | Channel Identification |
| | Slot Active | User Setting |
| | Slot 0 Active | ☑ |
| | Slot 1 Active | ☑ |
| | Audio Frequency | 44.1 KHz |
| | Clock Strobing | Rising Edge |
| | Fifo Threshold | One Quarter Full |
| | Output Drive | Enabled |



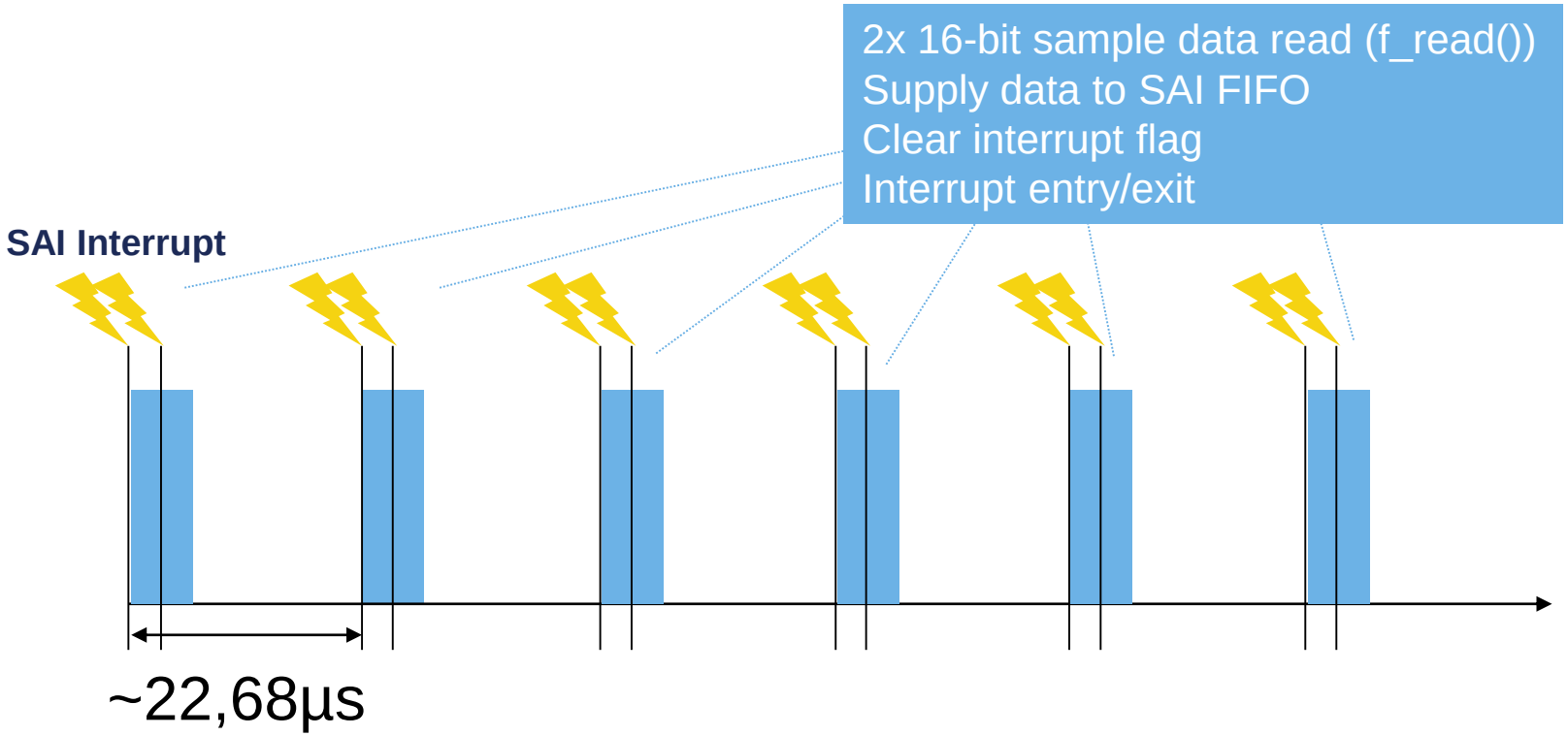

**audio.wav**

| WAVE Information | |
|---|---|
| Length | 00:58 |
| Compression | PCM |
| Bitrate (bps) | 16 |
| Frequency (Hz) | 44 100 |
| Mode | Stereo |

# 4 How to supply audio data to SAI



| audio.wav | |
|---|---|
| **WAVE Information** | |
| Length | 00:58 |
| Compression | PCM |
| Bitrate (bps) | 16 |
| Frequency (Hz) | 44 100 |
| Mode | Stereo |

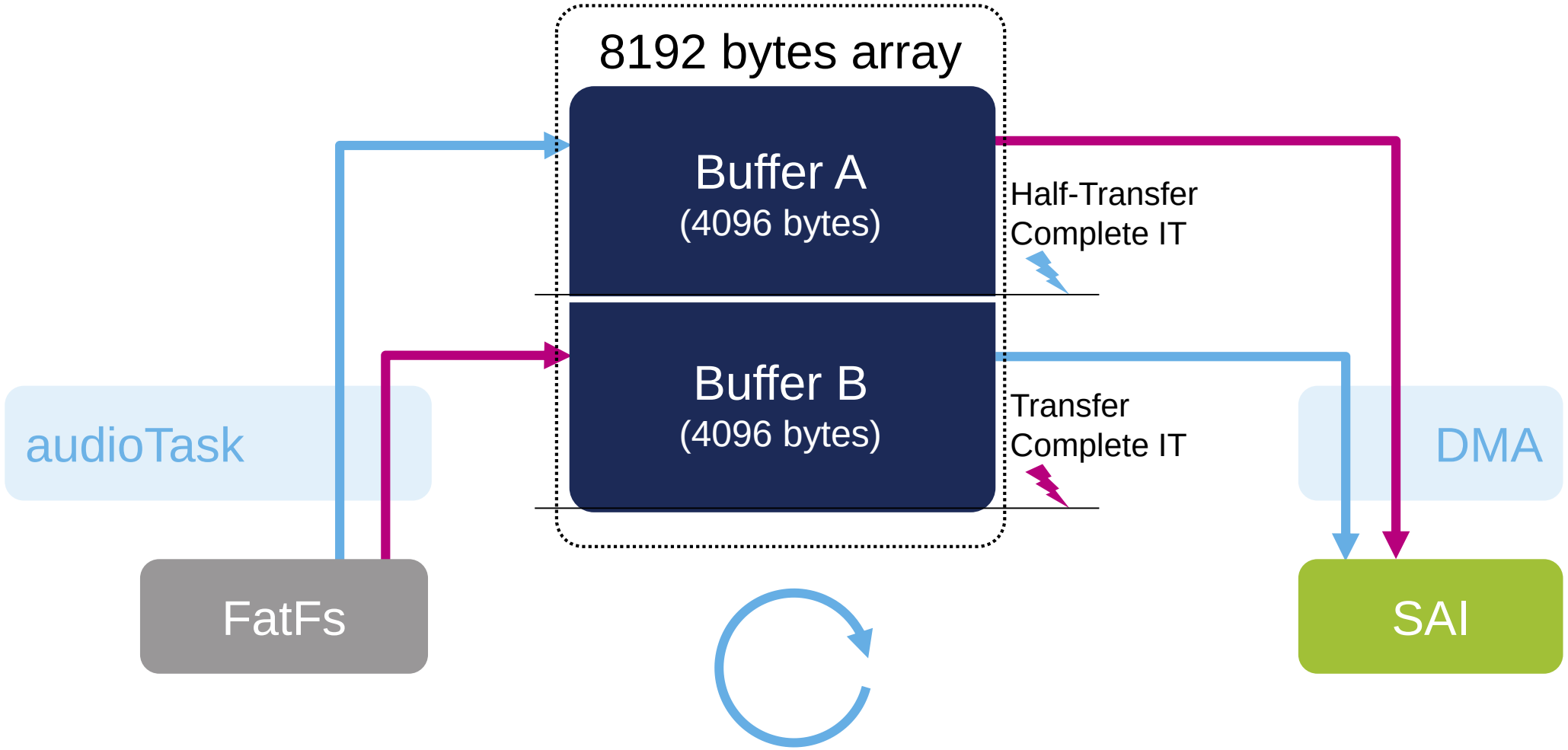



~1814 clk cycles @80 MHz → **far enough for the task**

But, is it “best practices” implementation of such task? NO!

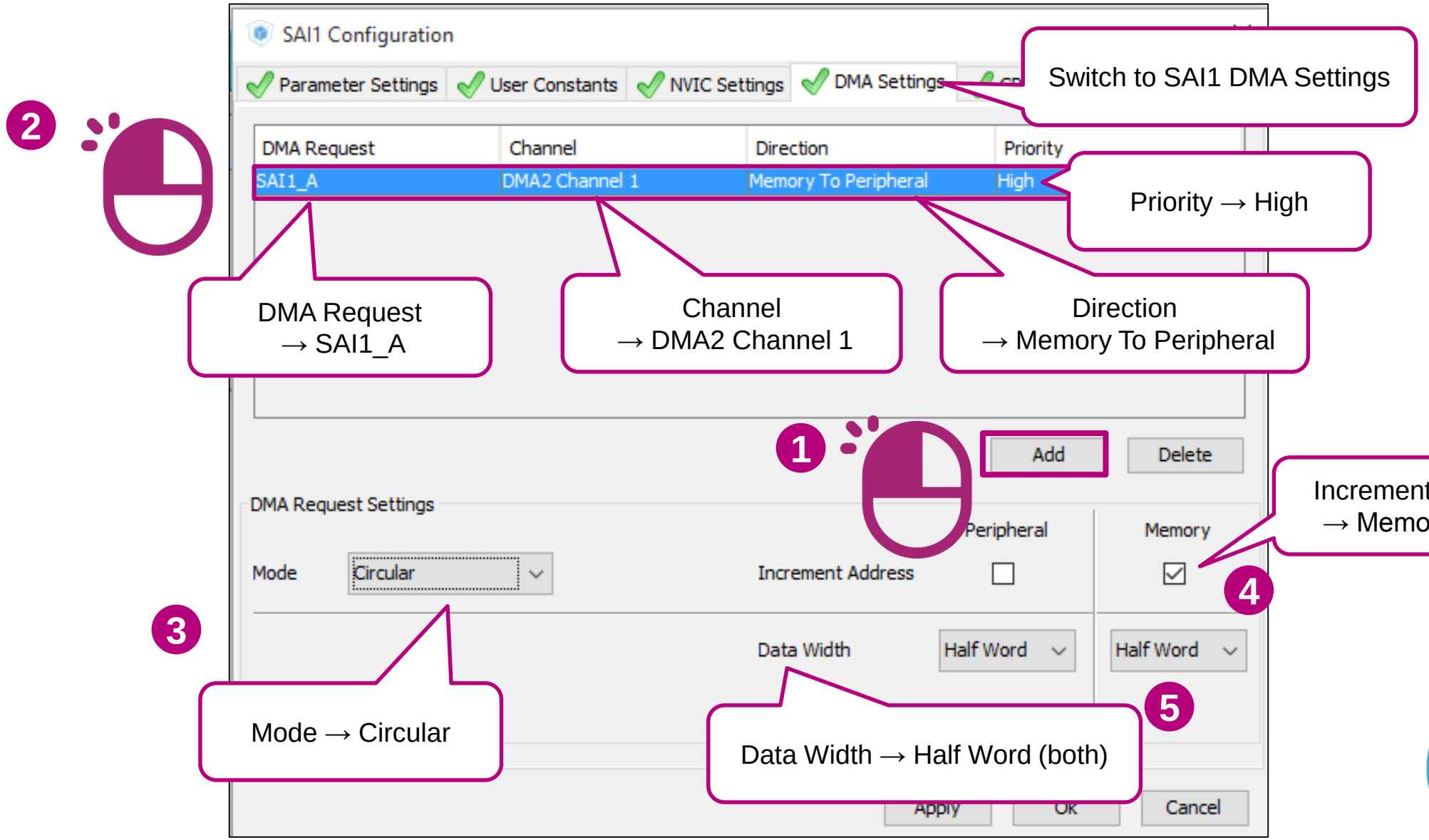
# 4 Audio data Dual-Buffer with DMA





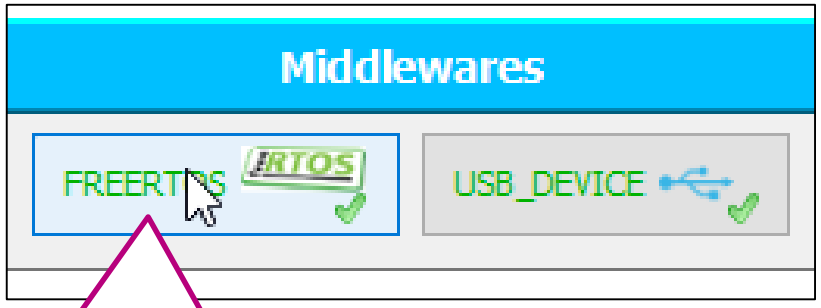
# Audio Configuration



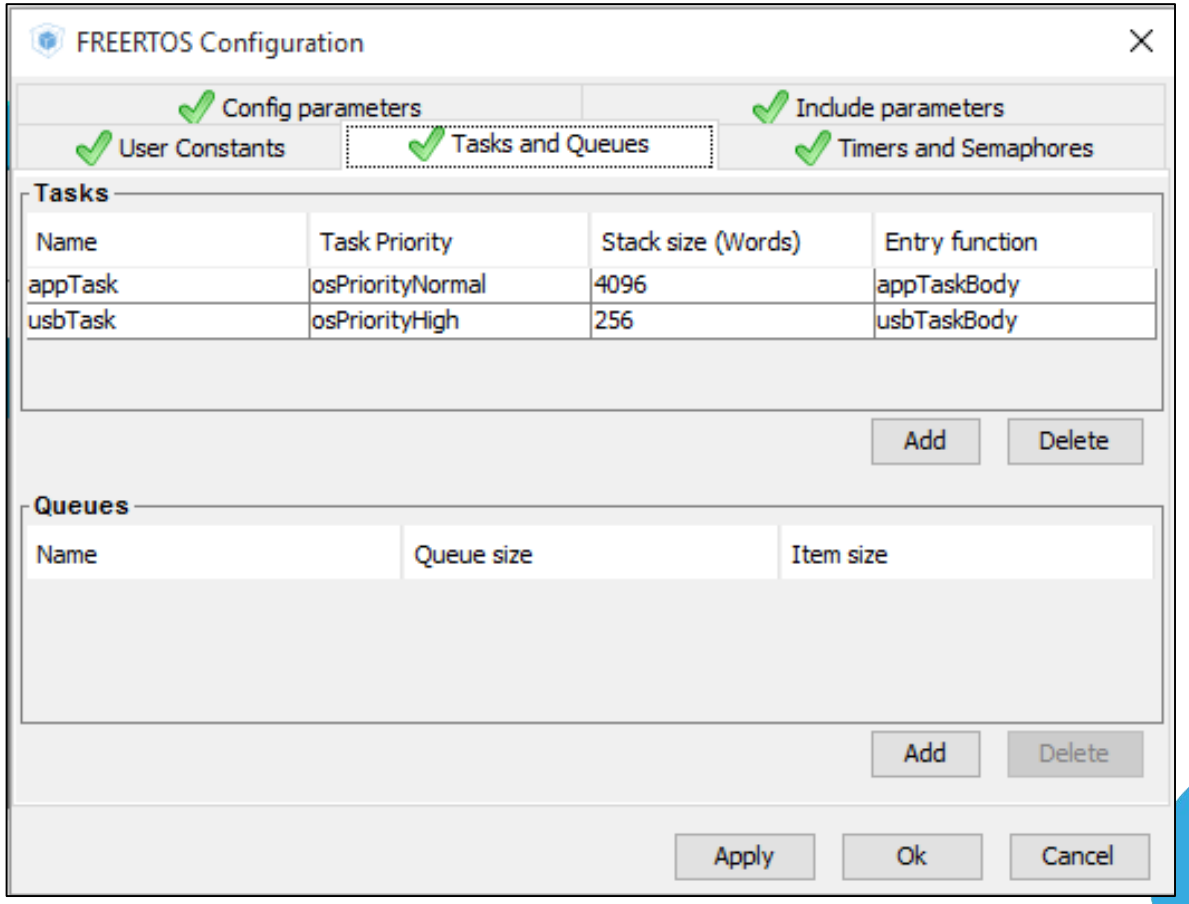


# 4 FreeRTOS Configuration Change




FREERTOS in Middlewares

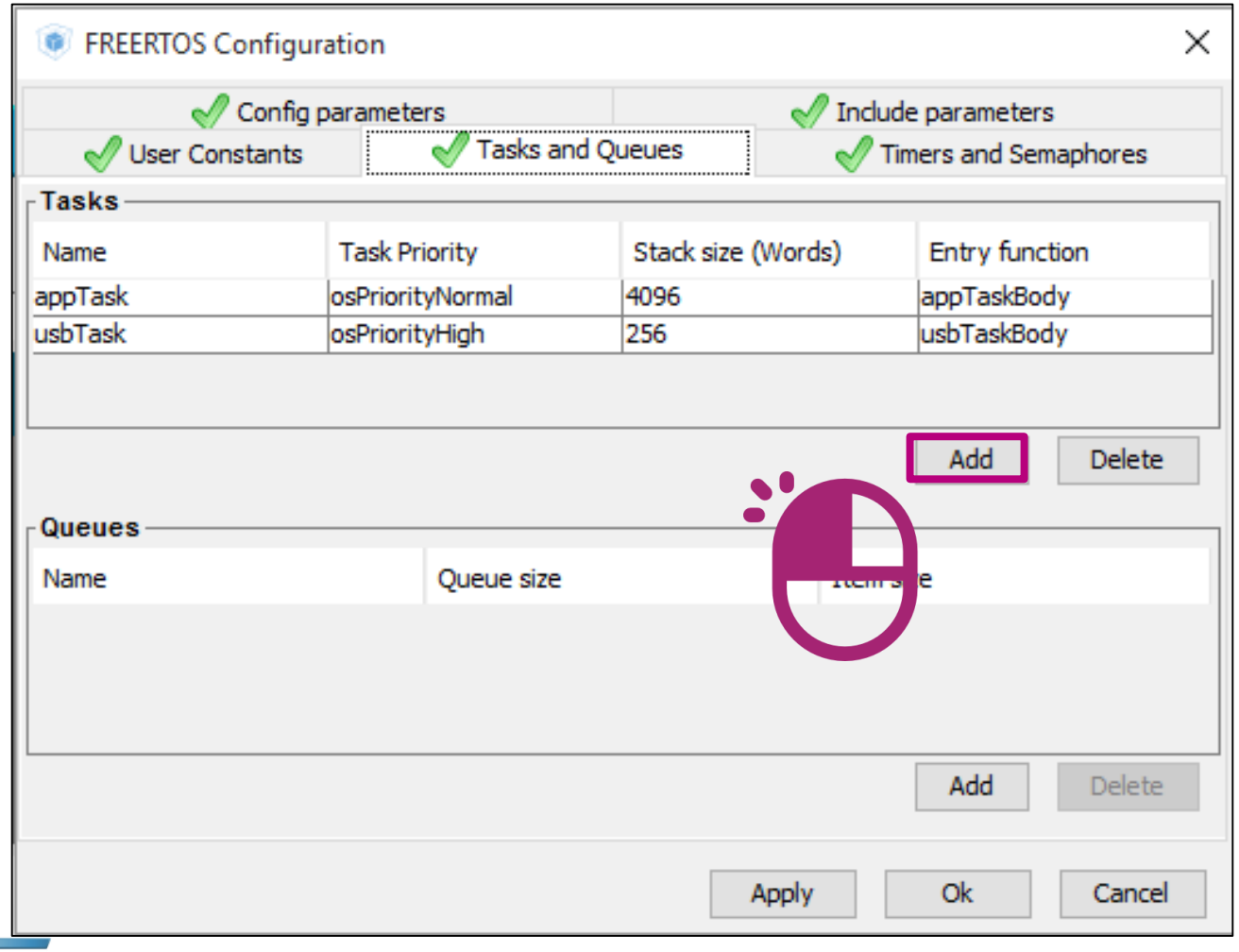



life.augmented

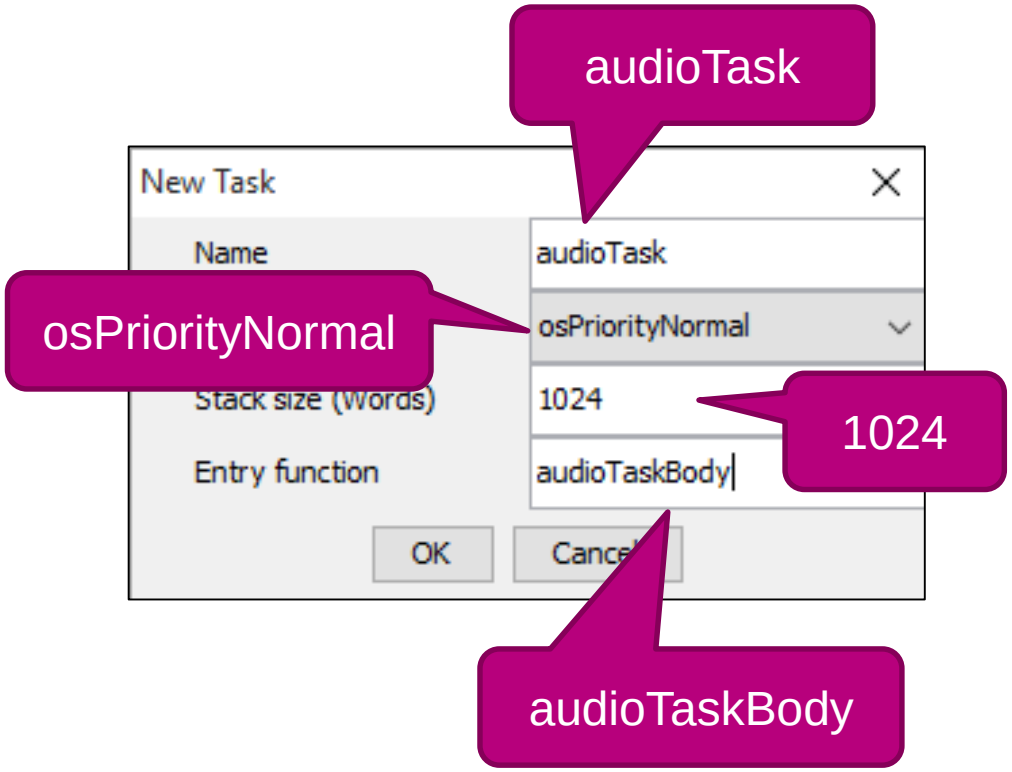
STM32 Cube

# 4 Audio Configuration





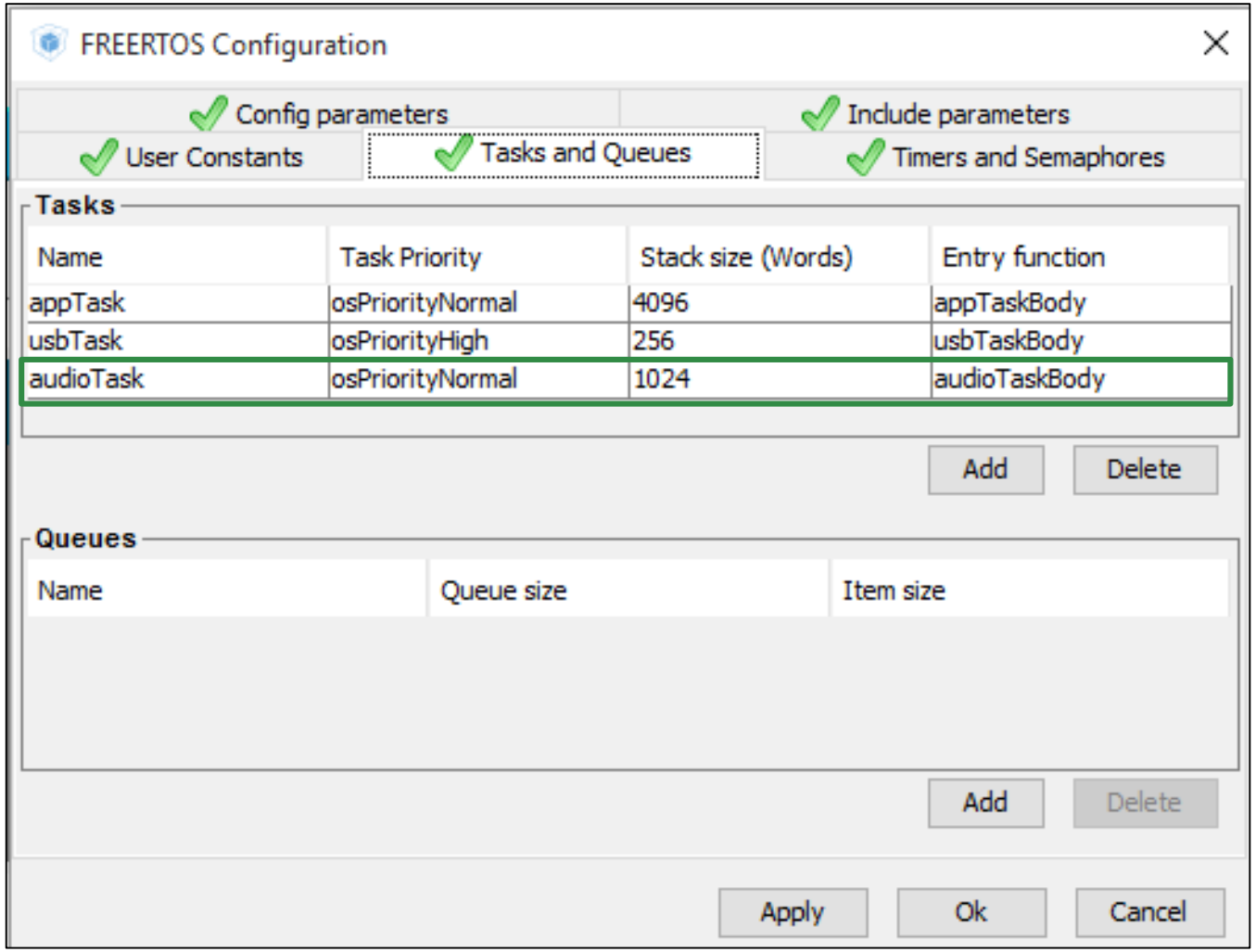

audioTask



# Checkpoint #11

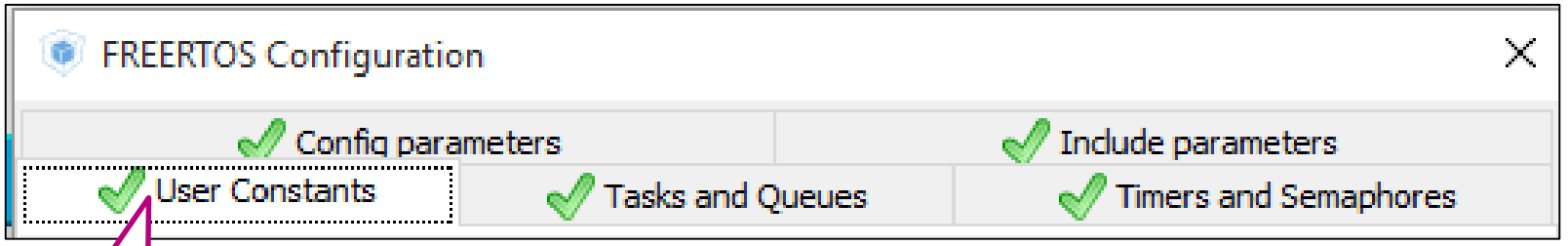


FatFS

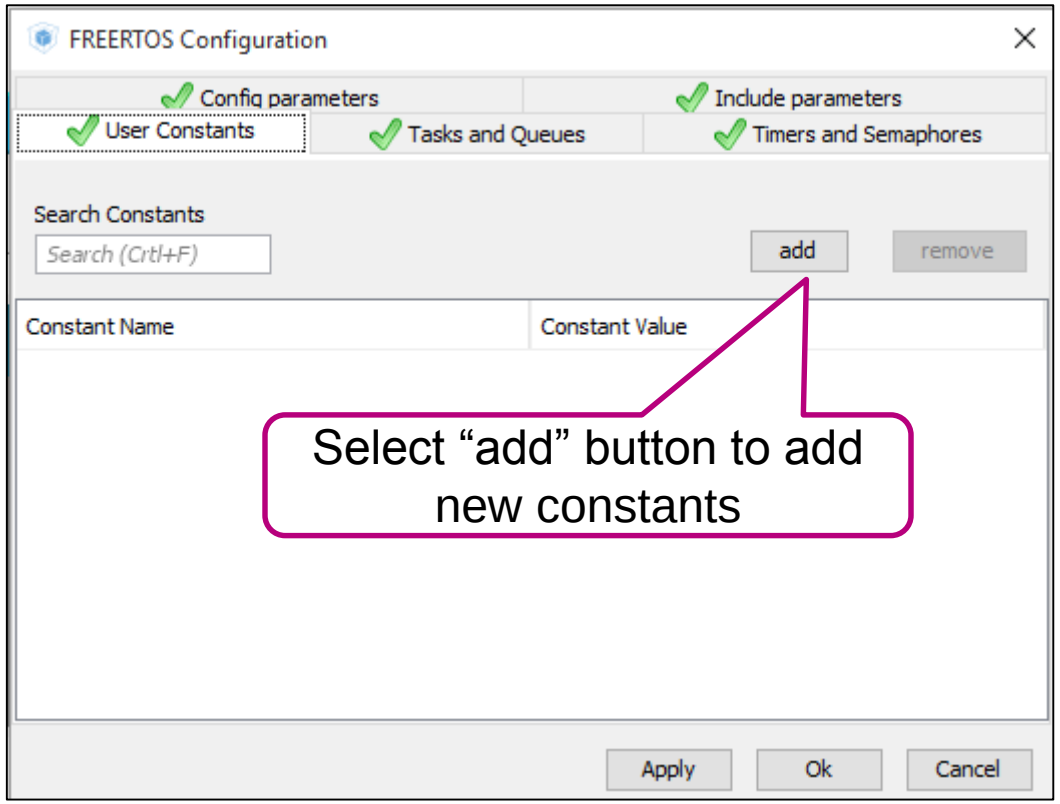



# FreeRTOS configuration





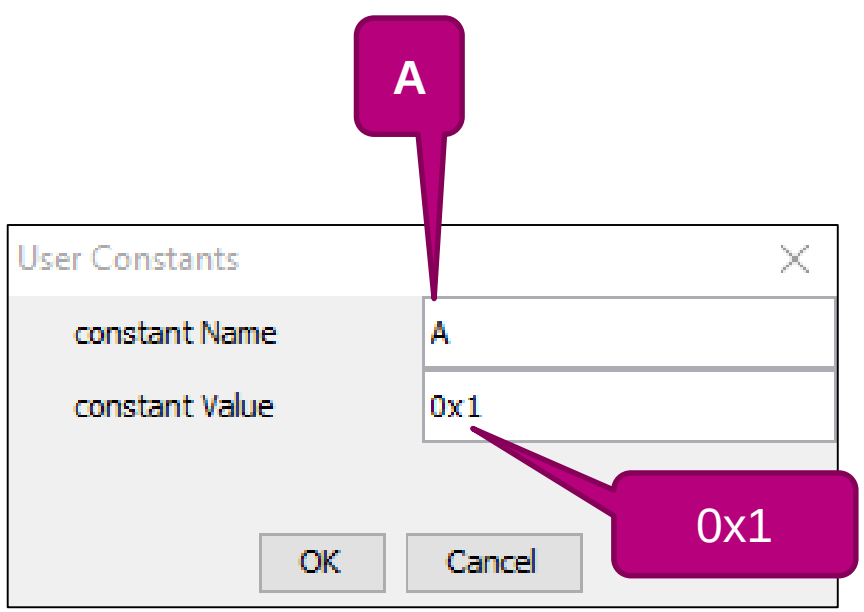

Go to User Constants in FreeRTOS Configuration

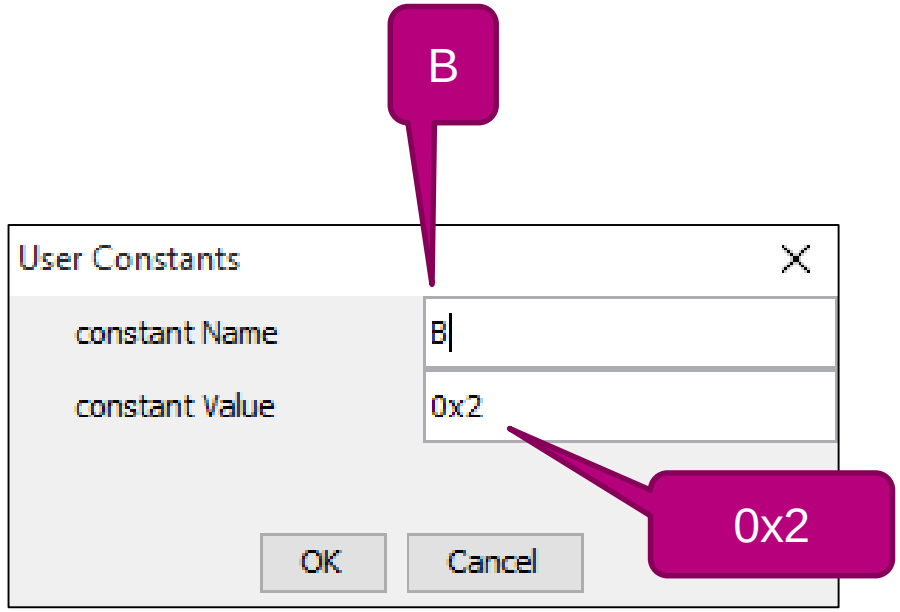



Select “add” button to add new constants

# 4 FreeRTOS Configuration Change

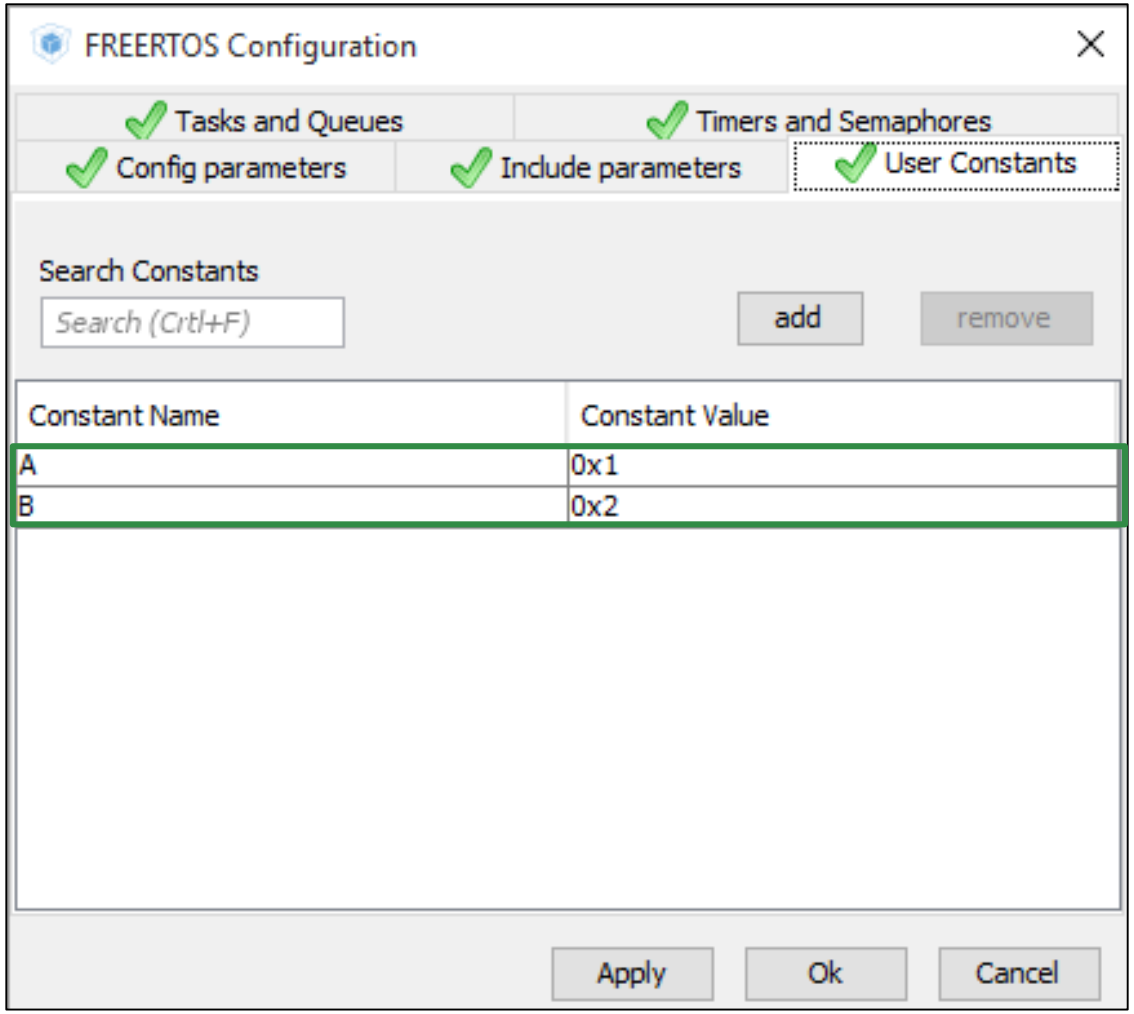


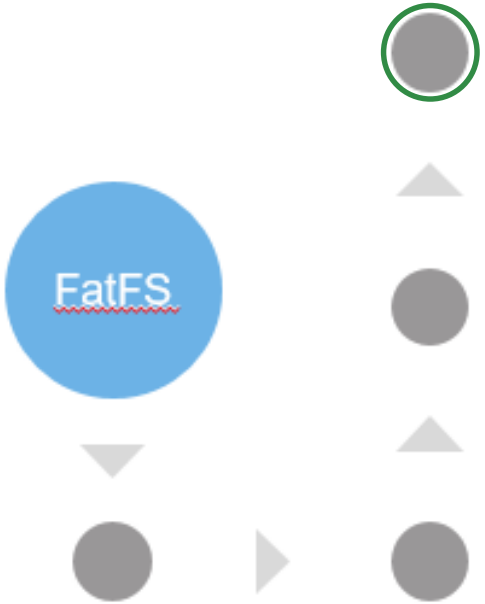


# Checkpoint #12





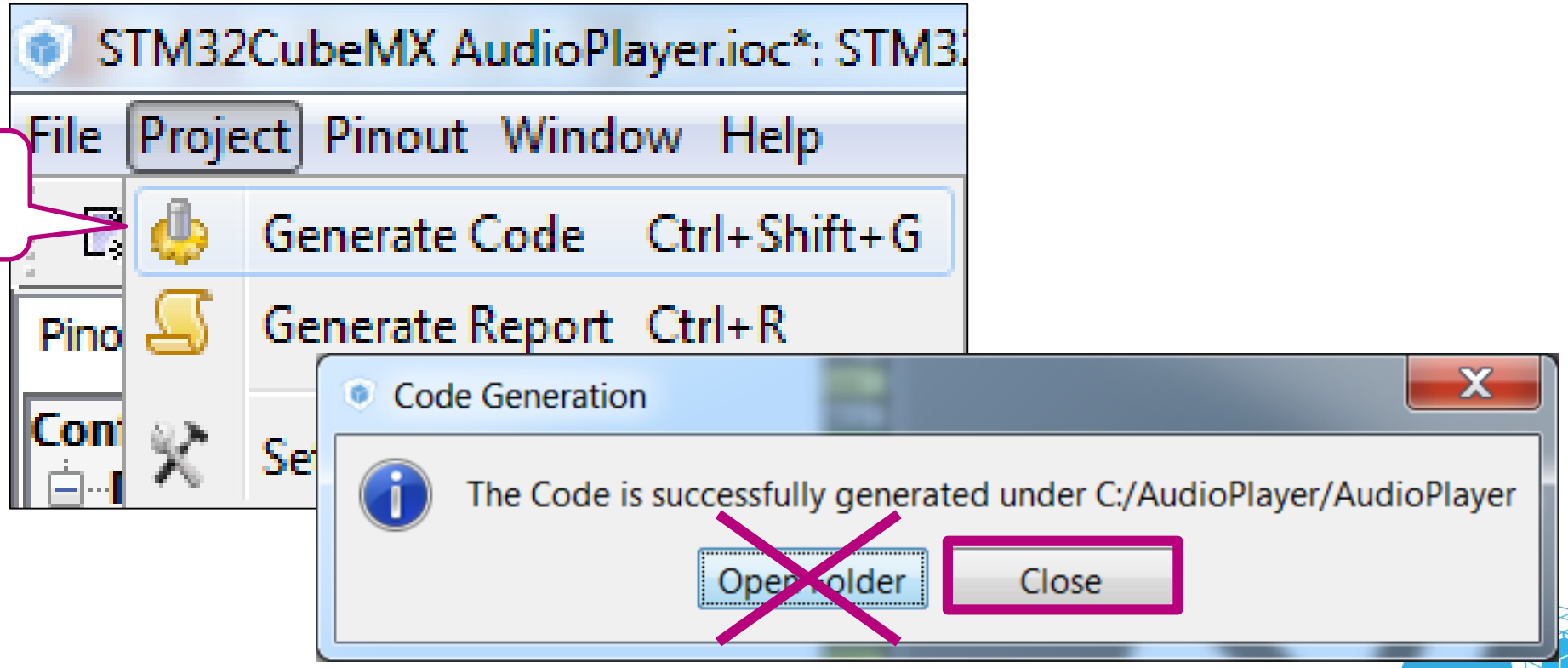



# 4 Save and generate the project



Do not open the project yet

Generate Code
(Ctrl+Shift+G)



# 4 Apply small patch



- Apply patch from folder
  **C:\STM32L4_Workshop\HandsOn\2_Putting_All_Together\Patch\STEP4_Audio**
  to root folder of your project (C:\AudioPlayer\AudioPlayer\)

- It contains the following files:

  .\Inc\

  cs43l22.h

  i2c.h

  .\Src\

  freertos.c

  i2c.c

This operation is STM32CubeMX non-intrusive. You can re-generate the project later and the modifications will be retained.

- These files contains the main needed source code the user has to add to be able to work with the external audio codec (cd43l22)

# 4 i2c.c added content



- i2c.c
  - AUDIO_Init(…)
  - AUDIO_DeInit(void)
  - AUDIO_ReadID(…)
  - AUDIO_Play(…)
  - AUDIO_Pause(…)
  - AUDIO_Resume(…)
  - AUDIO_Stop(…)
  - AUDIO_SetVolume(…)
  - AUDIO_SetFrequency(…)
  - AUDIO_SetMute(…)
  - AUDIO_SetOutputMode(…)

Based on BSP driver in STM32CubeL4 HAL

# 4 Let’s try it out



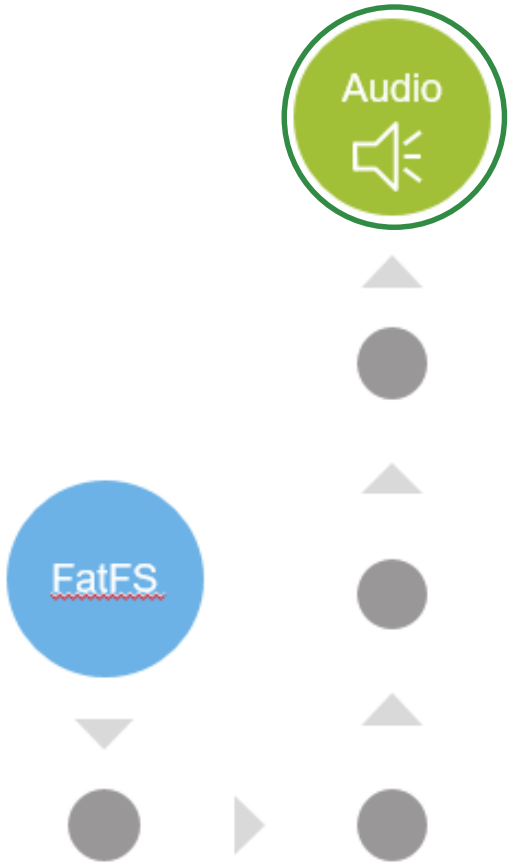
1. Open the project
2. Compile and download
3. Press the reset button to restart the application

# 4

# Checkpoint #13



- Connect headphones
- Do you hear music?
- LED_GREEN being toggled when playing audio file



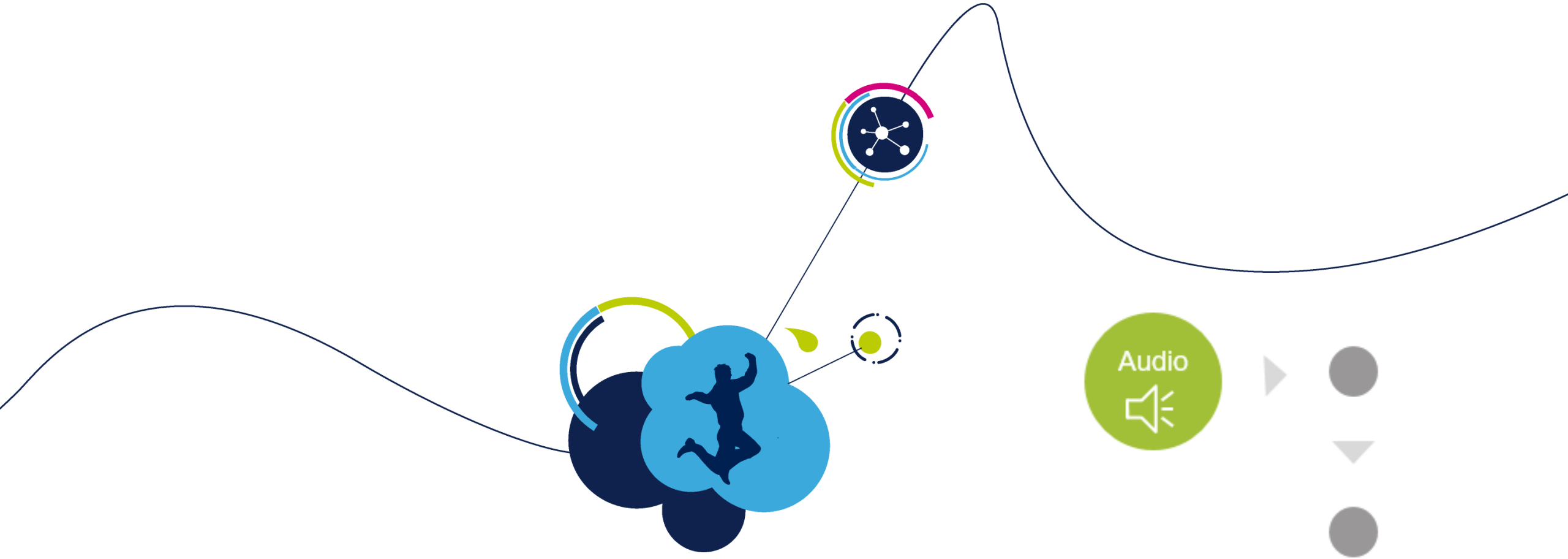

# Why Signal?



Synchronization of the dual buffer filling mechanism:

- if both at the same time → error, we are not fast enough maybe

- “Low cost” => fast management by Scheduler, 0 memory usage





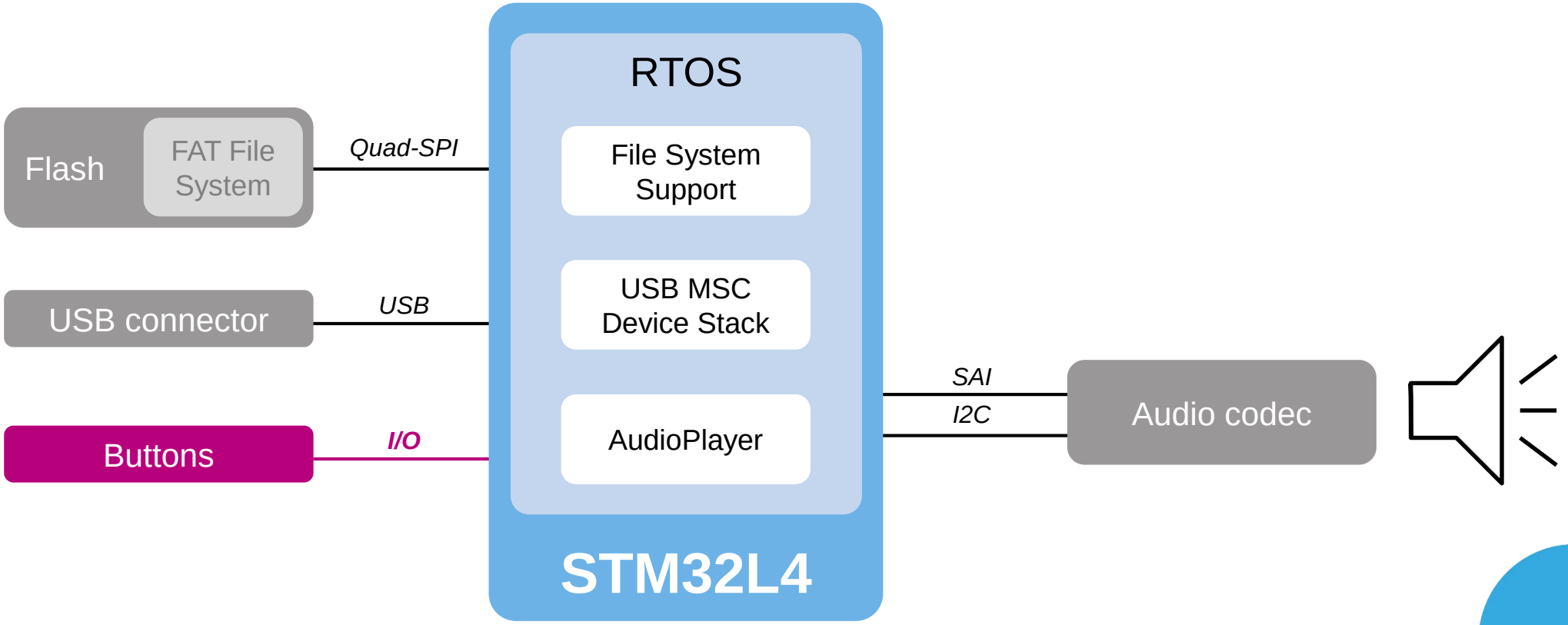

# STEP5 – Joystick Input

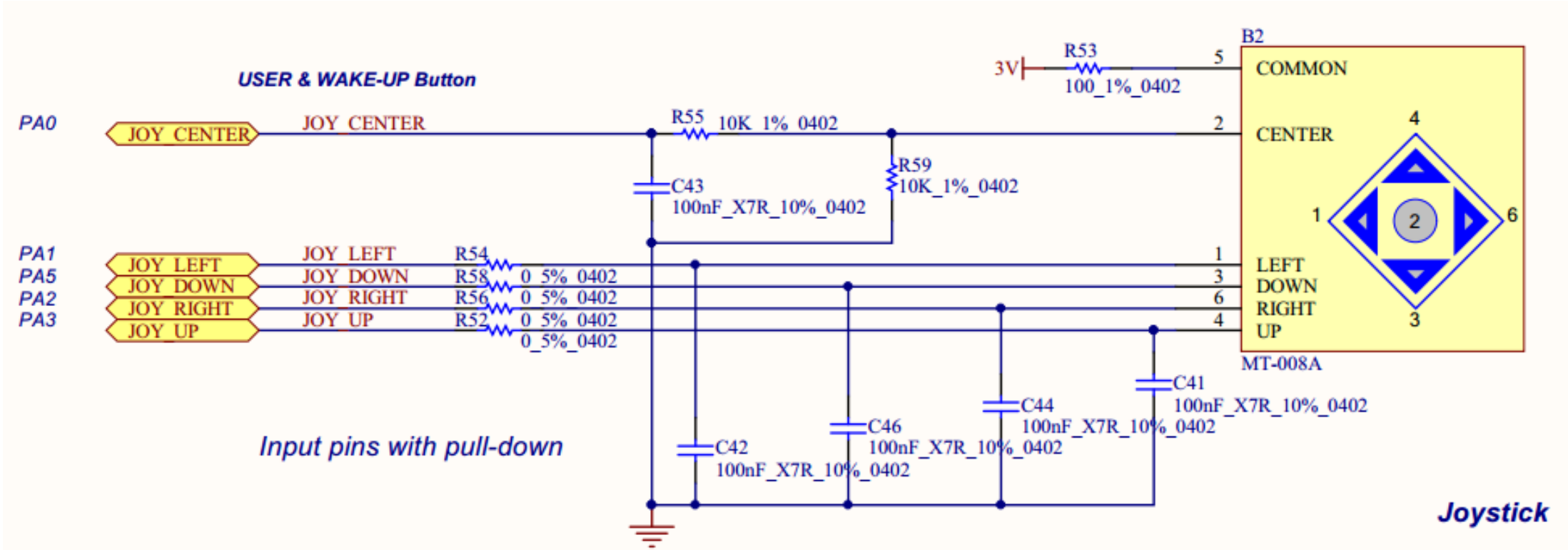
life.augmented

# 5 Adding Control Input functionality



- We want to control our application state
  - like “play”, “stop”, “pause” commands



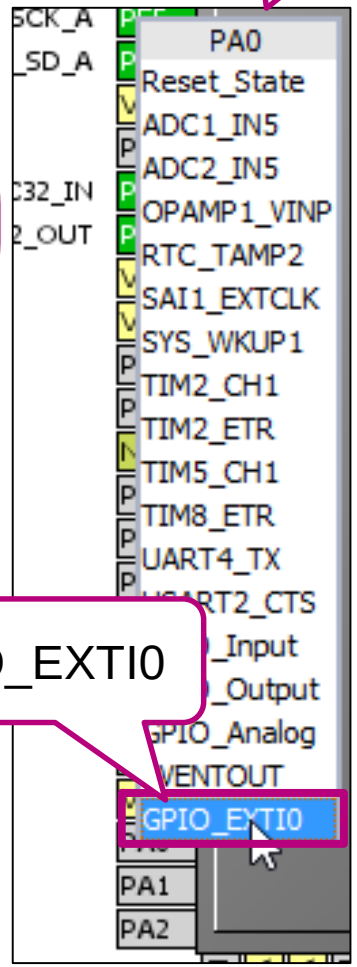

USER & WAKE-UP Button
PA0
JOY_CENTER
JOY_CENTER
R55 10K 1% 0402
C43
100nF_X7R_10%_0402
R59
10K_1%_0402
3V
R53
100_1%_0402
B2
COMMON
CENTER
LEFT
DOWN
RIGHT
UP
MT-008A
PA1
PA5
PA2
PA3
JOY_LEFT
JOY_DOWN
JOY_RIGHT
JOY_UP
R54
R58
R56
R52
0_5%_0402
C42
C46
C44
C41
Input pins with pull-down
Joystick

# Joystick Configuration



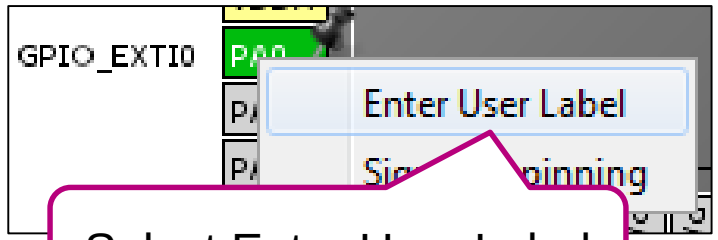
5 PA0

1



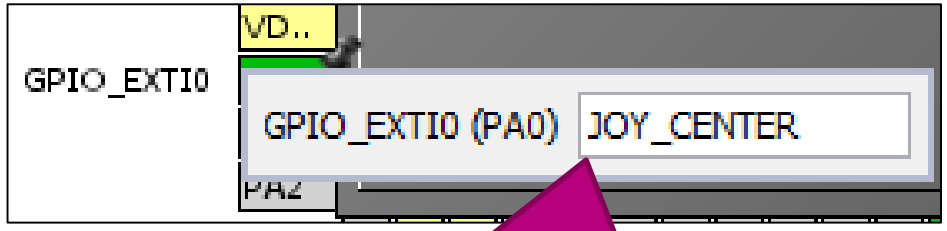

GPIO_EXTI0

2



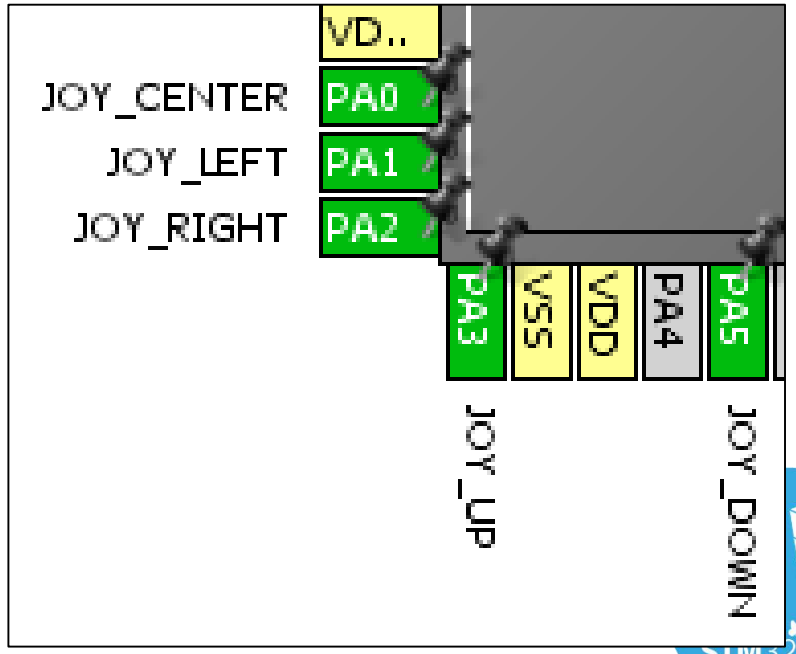

Select Enter User Label

3




Enter **JOY_CENTER**

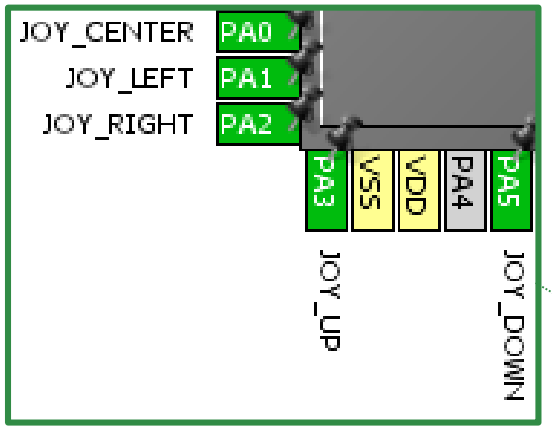
PA0 → JOY_CENTER
PA1 → JOY_LEFT
PA2 → JOY_RIGHT
PA3 → JOY_UP
PA5 → JOY_DOWN

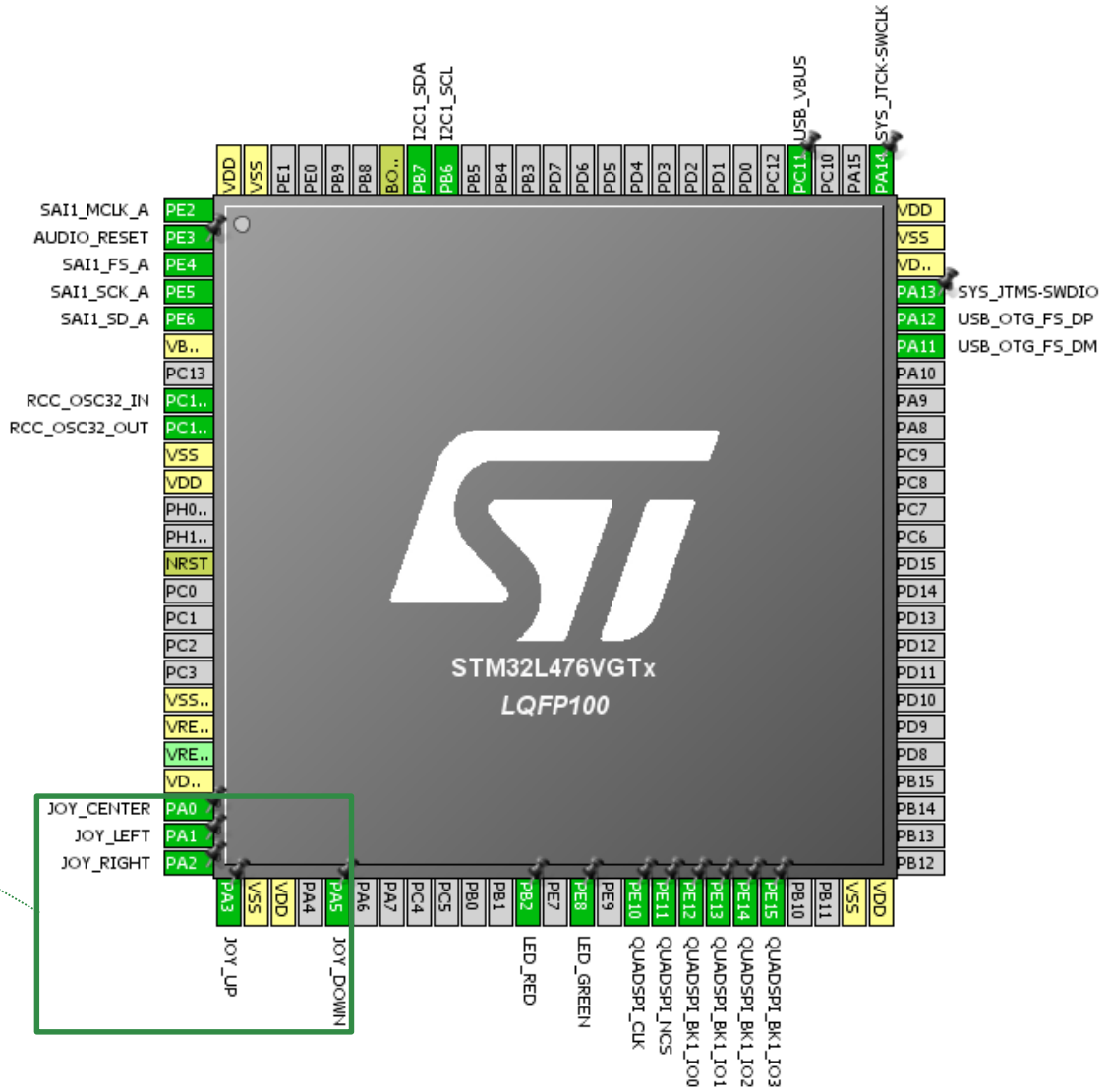



life.augmented

# Checkpoint #14



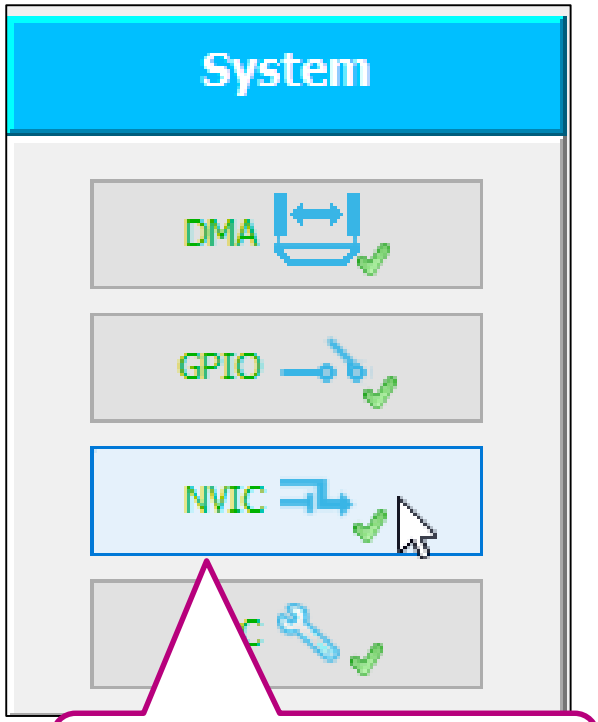

life.augmented

STM32 Cube

# Joystick Configuration



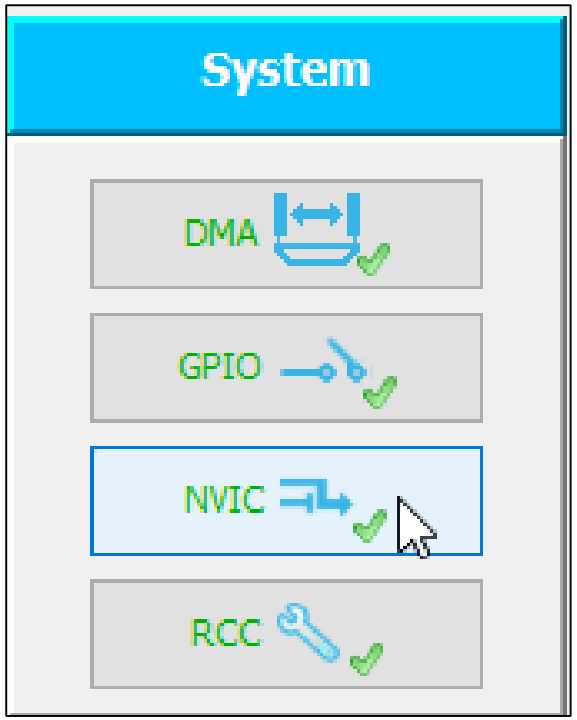

Go to Configuration

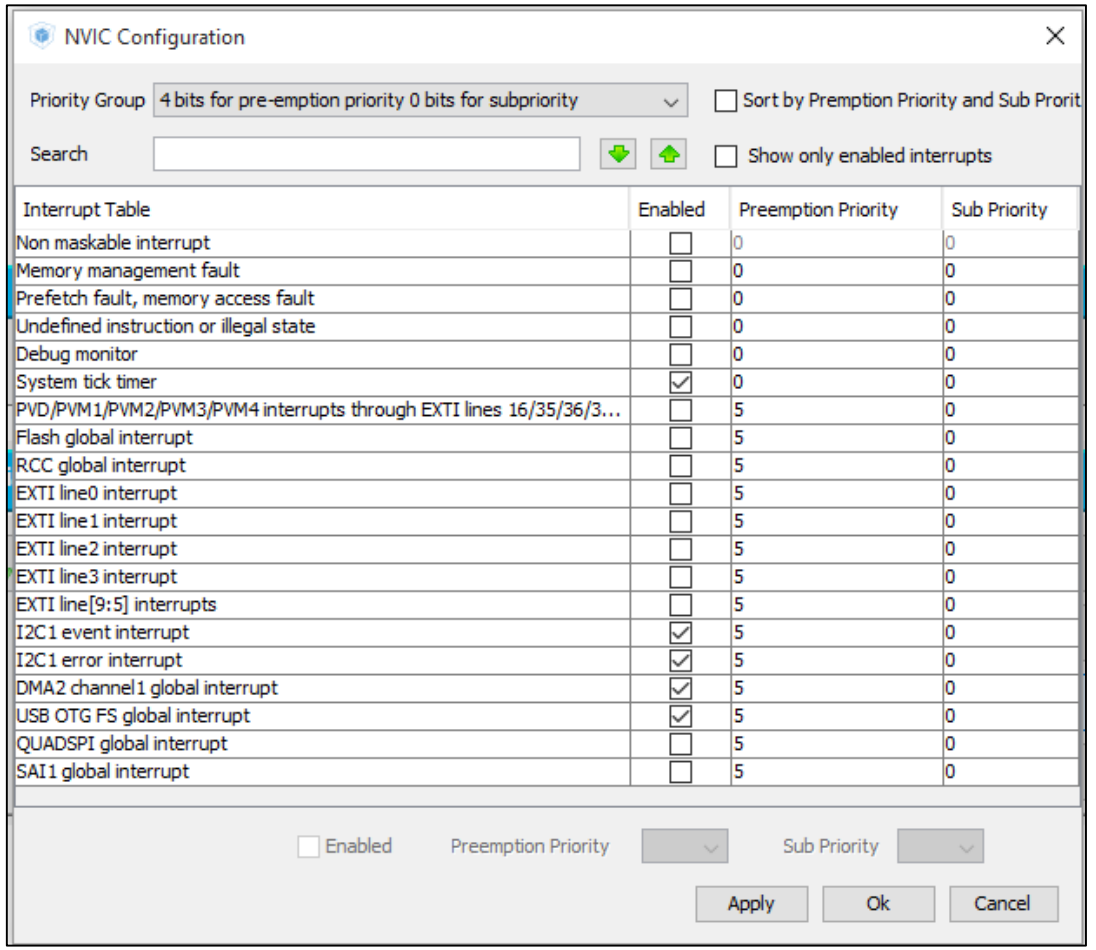



NVIC in System

# Joystick Configuration



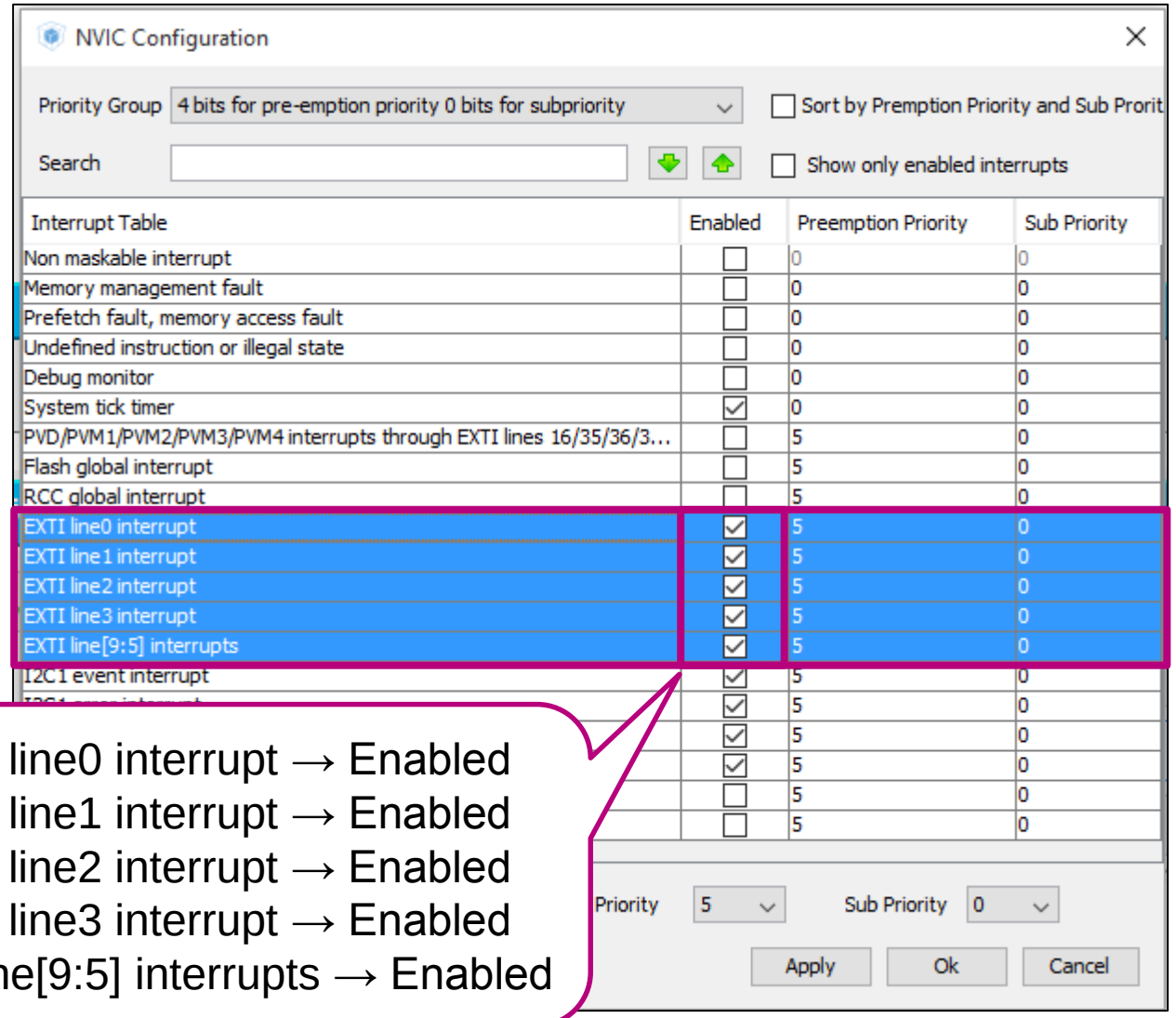




life.augmented

STM32 Cube

# Joystick Configuration

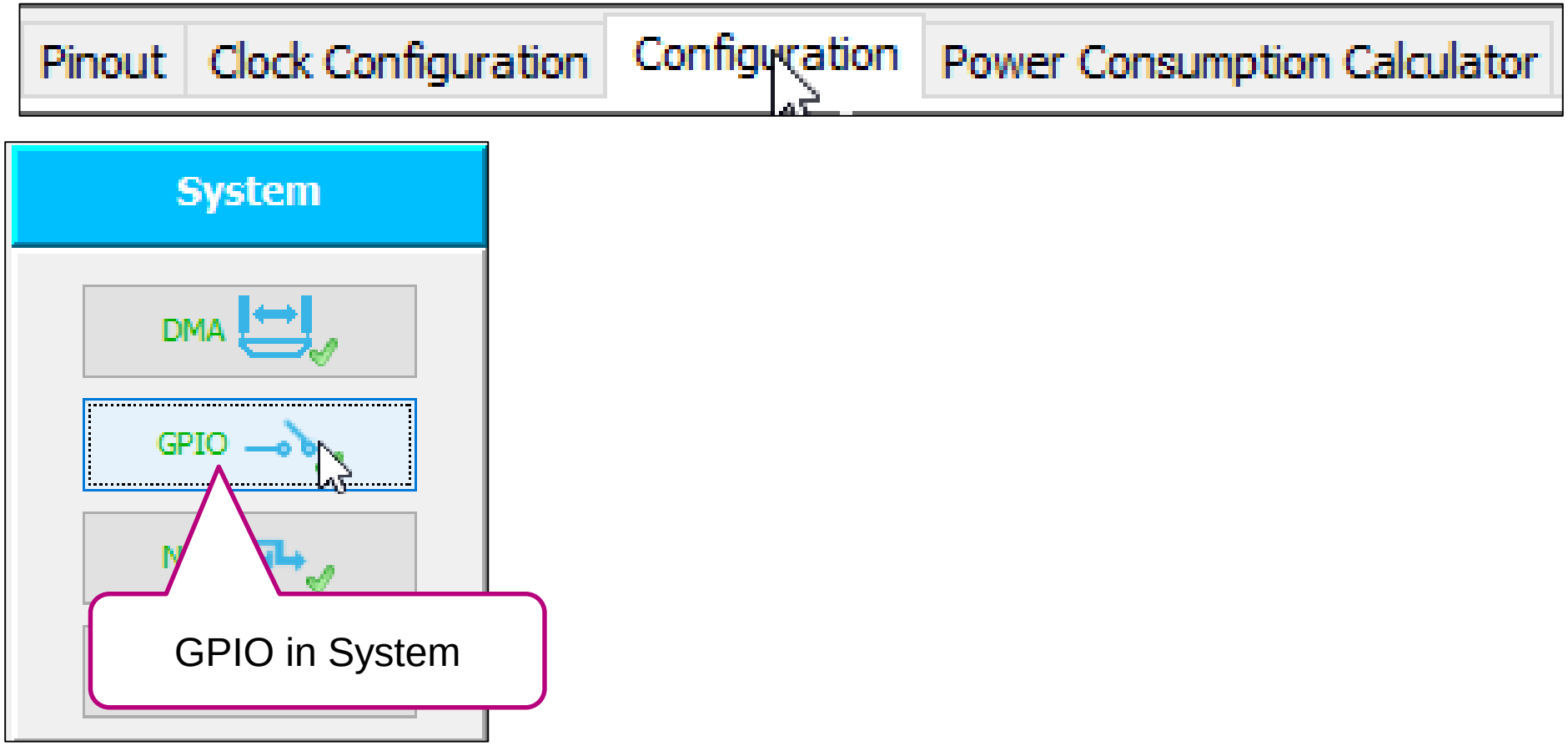





EXTI line0 interrupt → Enabled
EXTI line1 interrupt → Enabled
EXTI line2 interrupt → Enabled
EXTI line3 interrupt → Enabled
EXTI line[9:5] interrupts → Enabled

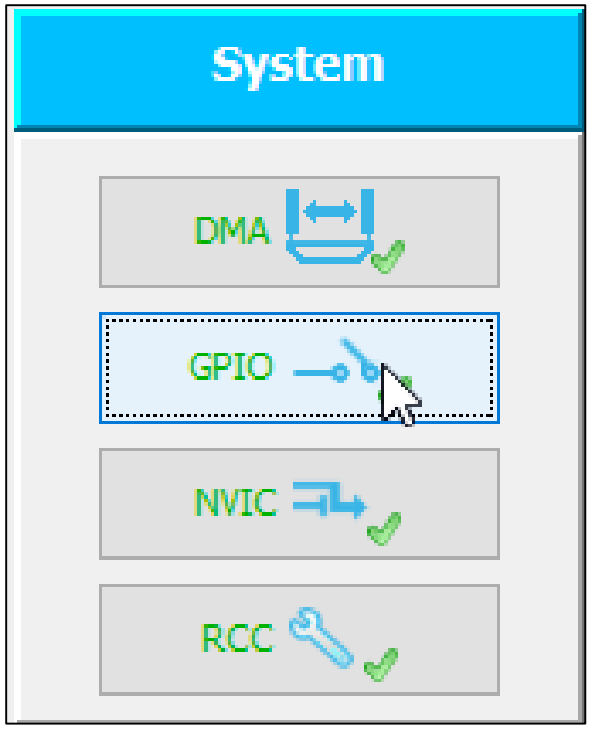
# Joystick Configuration

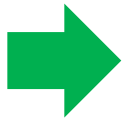


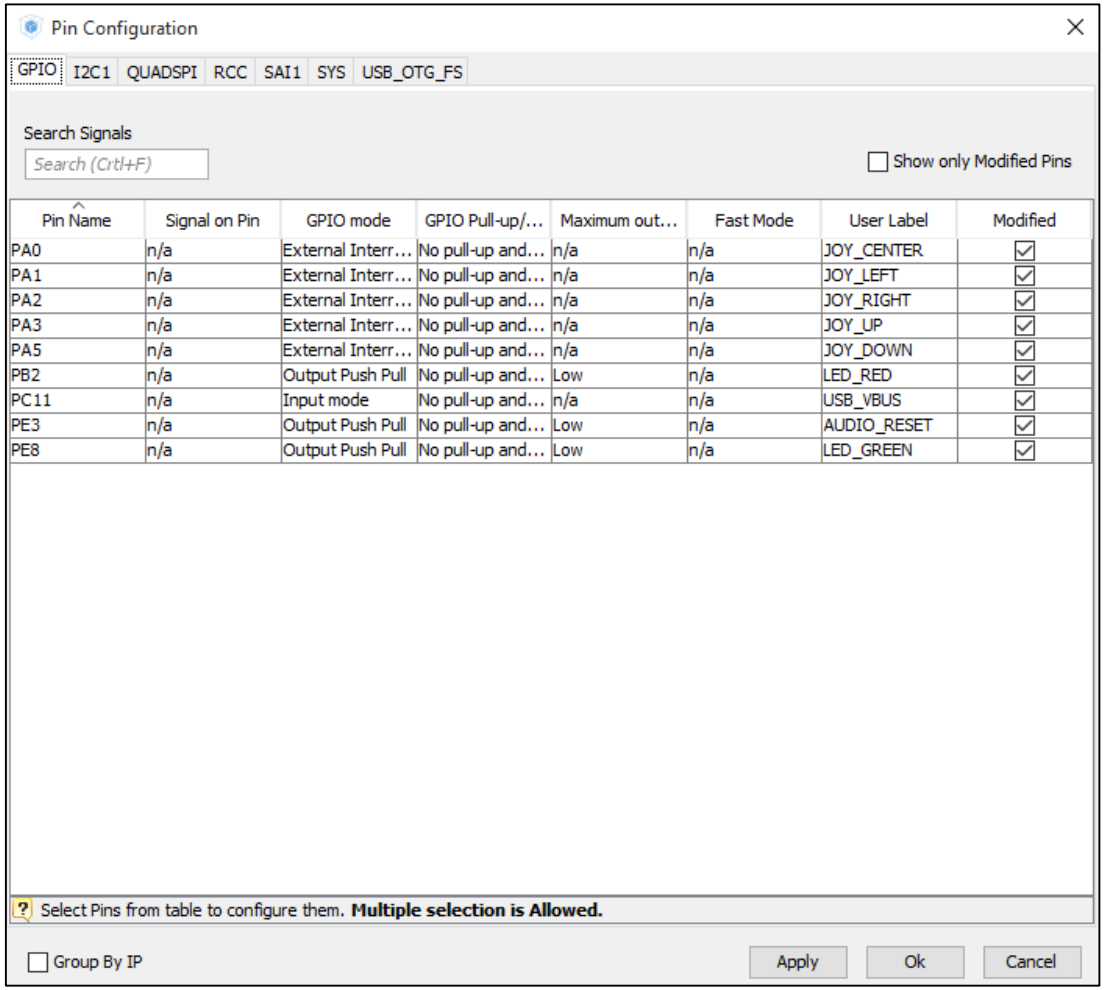


# Joystick Configuration



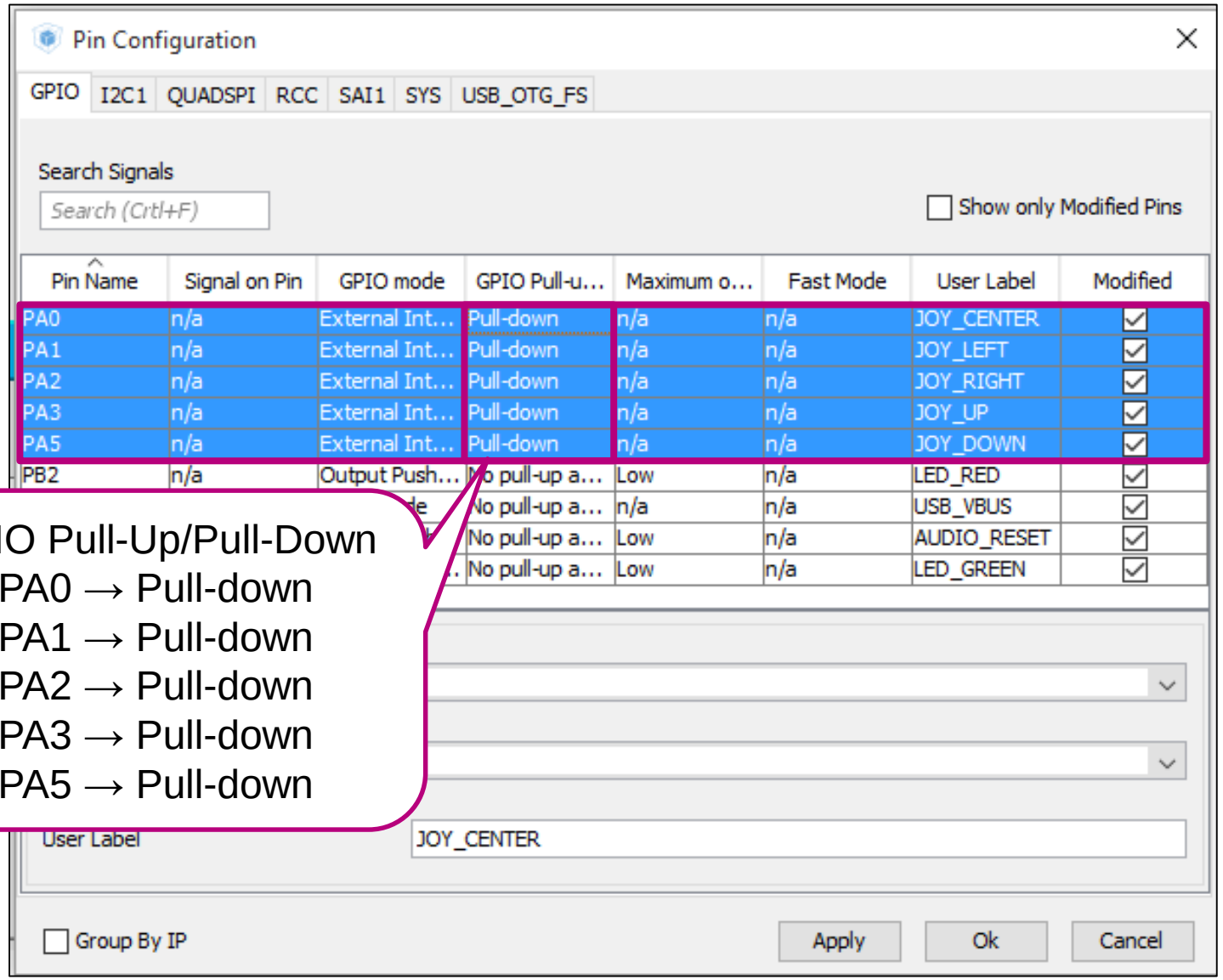





# Joystick Configuration



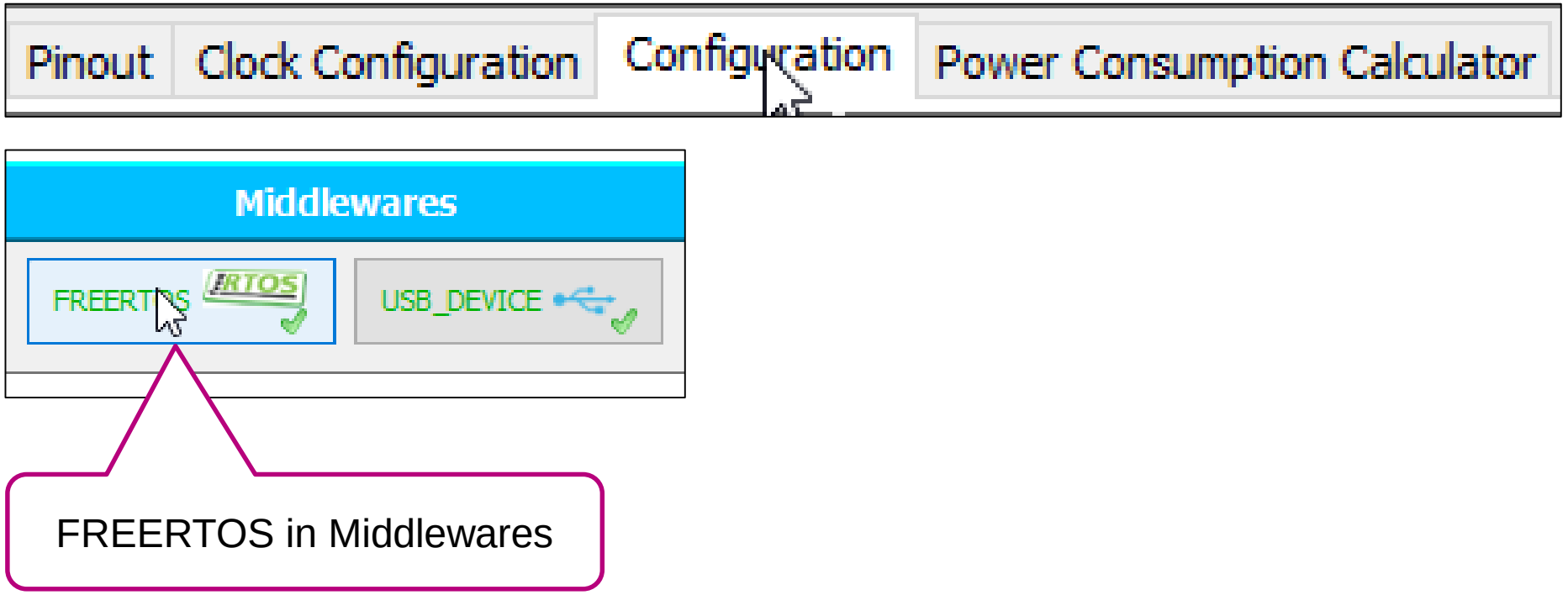



GPIO Pull-Up/Pull-Down

- PA0 → Pull-down
- PA1 → Pull-down
- PA2 → Pull-down
- PA3 → Pull-down
- PA5 → Pull-down

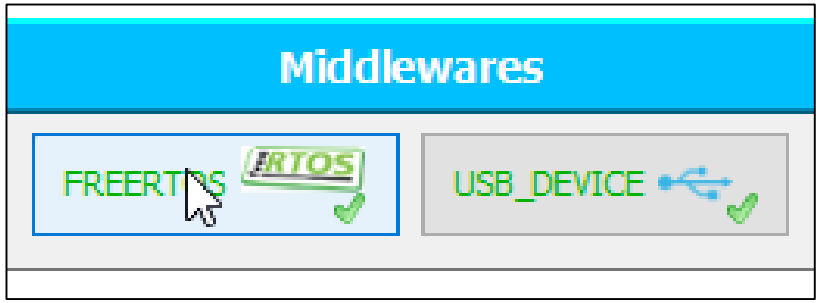
# 5 Joystick Configuration

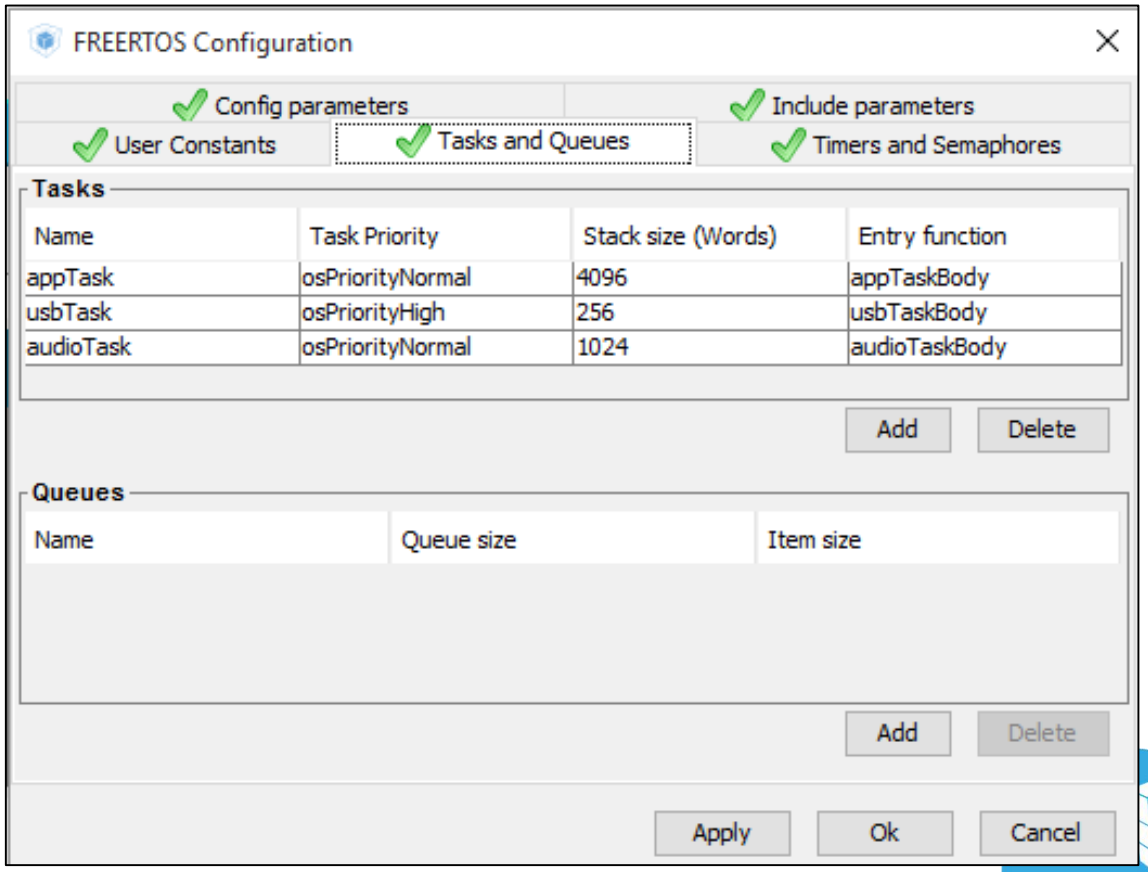




# Joystick Configuration



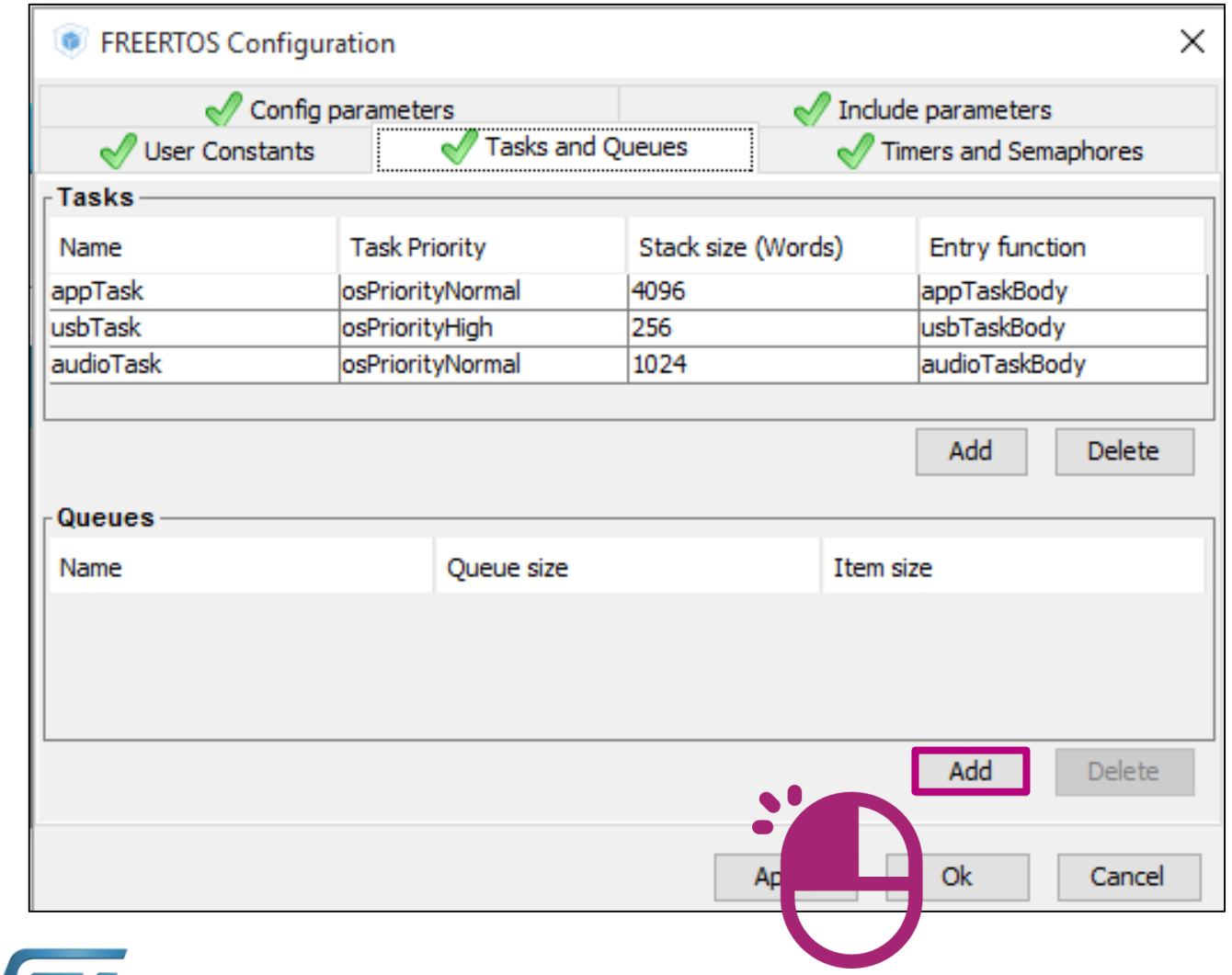



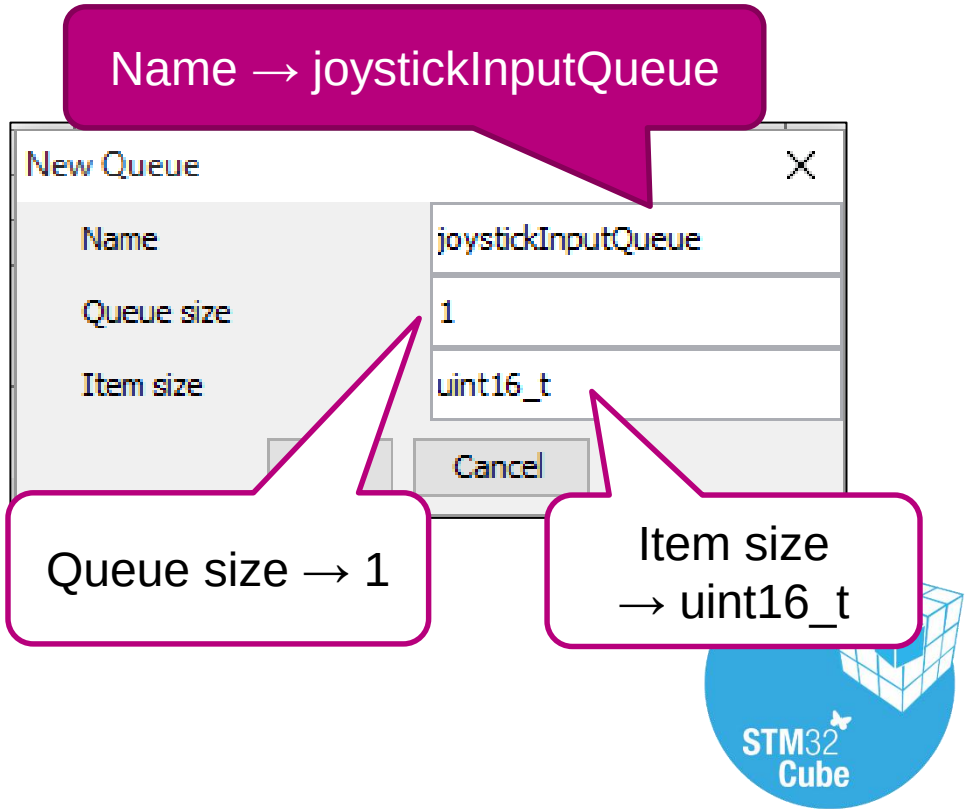

life.augmented

STM32 Cube

# Joystick Configuration






life.augmented

STM32 Cube

# Checkpoint #15



FREERTOS Configuration

Config parameters | Include parameters | User Constants | Tasks and Queues | Timers and Semaphores

**Tasks**

| Name | Task Priority | Stack size (Words) | Entry function |
|---|---|---|---|
| appTask | osPriorityNormal | 4096 | appTaskBody |
| usbTask | osPriorityHigh | 256 | usbTaskBody |
| audioTask | osPriorityNormal | 1024 | audioTaskBody |

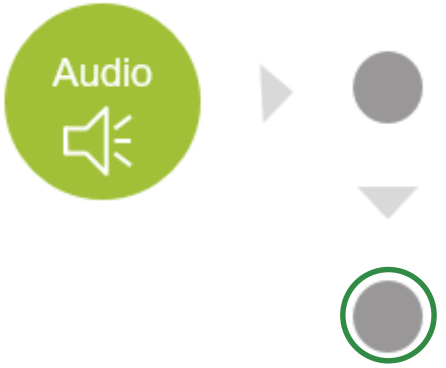
Add Delete

**Queues**

| Name | Queue size | Item size |
|---|---|---|
| joystickInputQueue | 1 | uint16_t |

Add Delete

Apply Ok Cancel




life.augmented

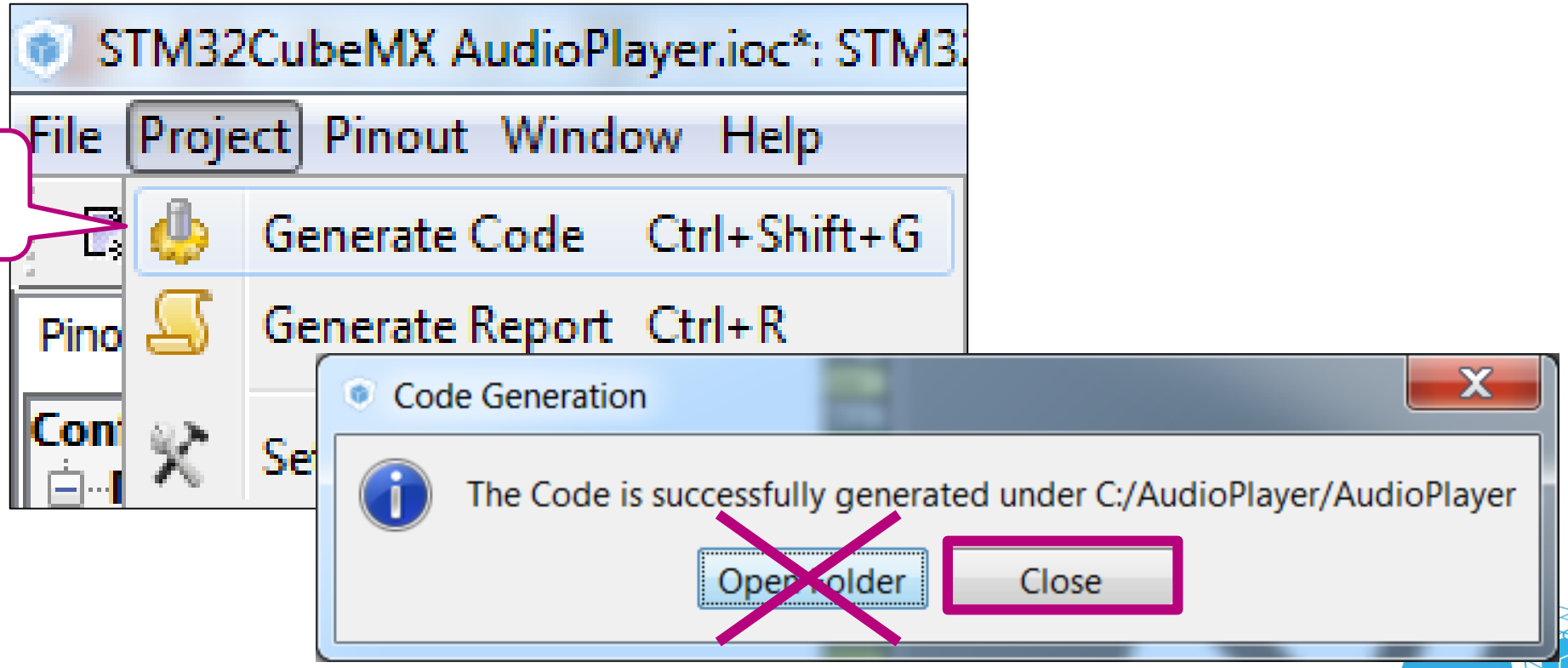
STM32 Cube

# 5 Save and generate the project



Do not open the project yet

Generate Code
(Ctrl+Shift+G)



# 5 Apply small patch



- Apply patch from folder
  **C:\STM32L4_Workshop\HandsOn\2_Putting_All_Together\Patch\STEP5_ControlInput**
  to root folder of your project

  (C:\AudioPlayer\AudioPlayer\)

- It contains the following files:

  .\Src\

  freertos.c

  usbd_conf.c

This operation is STM32CubeMX non-intrusive. You can re-generate the project later and the modifications will be retained.

- This operation is STM32CubeMX non-intrusive. You can re-generate the project later and the modifications will be retained.

# 5

# Let’s try it out



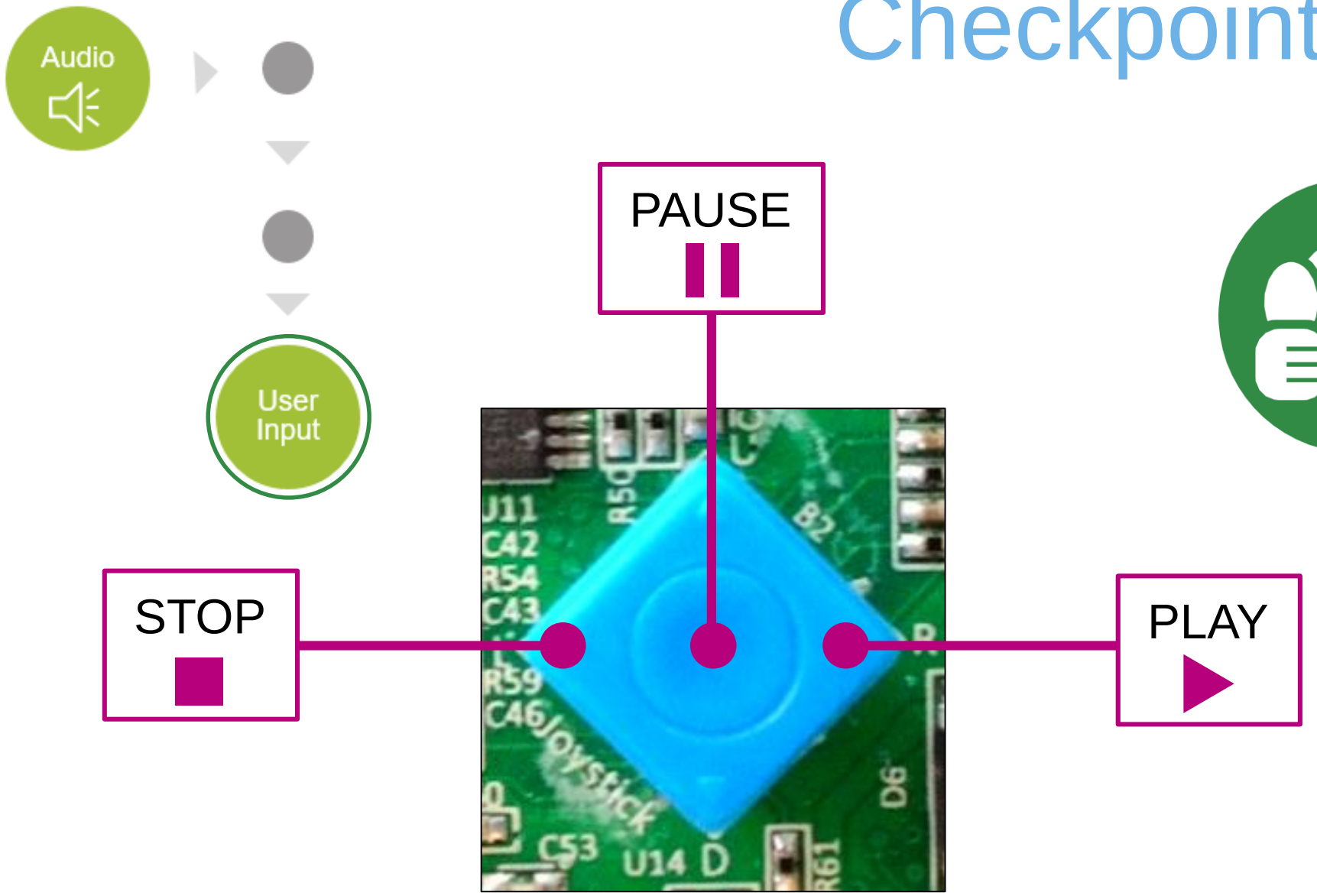
1. Open the project

2. Compile and download

3. Press the reset button on discovery once download has finished

# Checkpoint #16



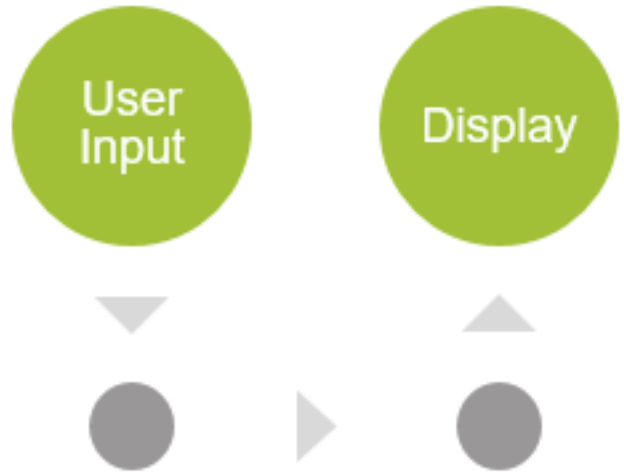



# Why Queue?



- Because we want to process just one input event at a time
- We are passing the command (queue item) from the EXTI ISR
- Why not signal?
  - Multiple signals can be set at a time

# STEP6 – LCD

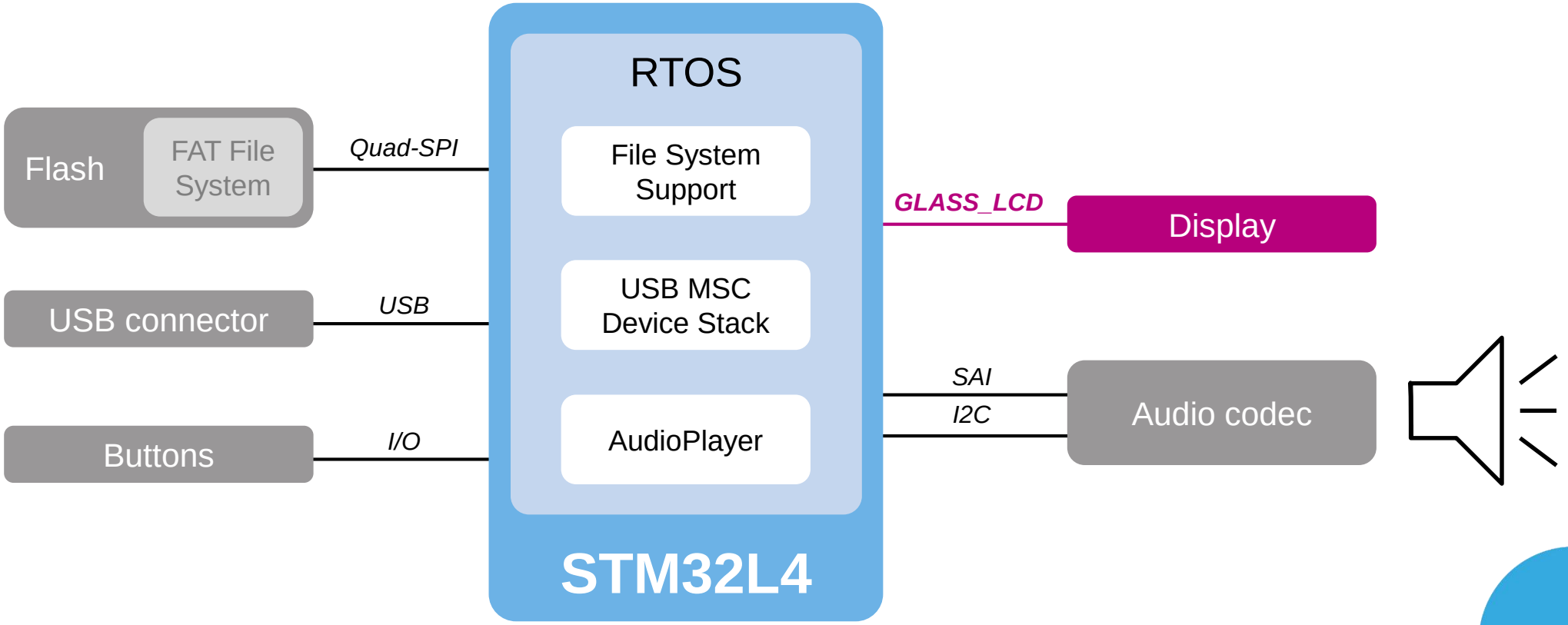



life.augmented

# Adding Display functionality



- We want to see the actual application state
  - like “PLAYING”, “STOPPED”, “PAUSED”



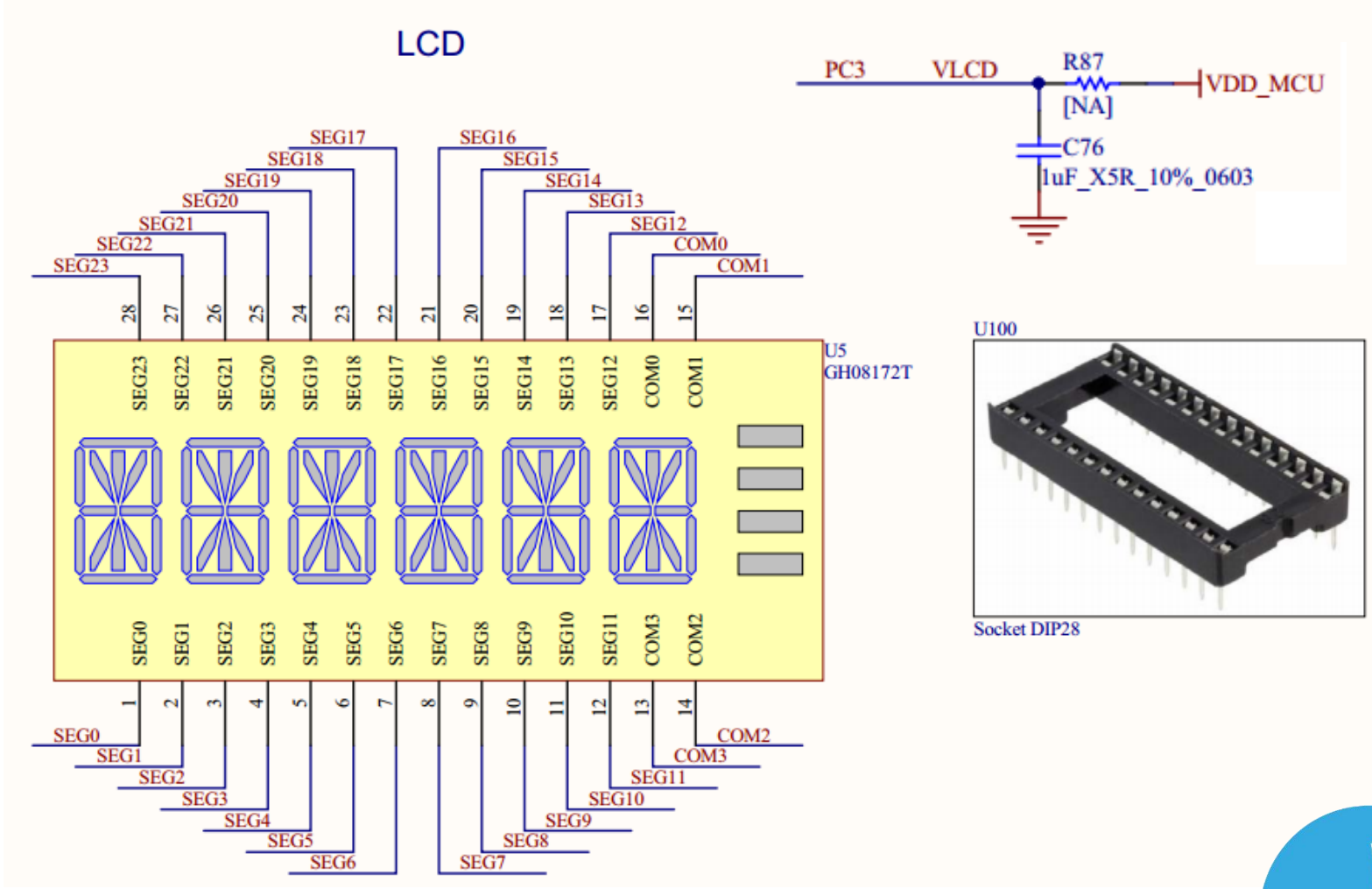
# 6 LCD Configuration




| | |
|---|---|
| PA8 | COM0 |
| PA9 | COM1 |
| PA10 | COM2 |
| PB9 | COM3 |

| | |
|---|---|
| PA7 | SEG0 |
| PC5 | SEG1 |
| PB1 | SEG2 |
| PB13 | SEG3 |
| PB15 | SEG4 |
| PD9 | SEG5 |
| PD11 | SEG6 |
| PD13 | SEG7 |
| PD15 | SEG8 |
| PC7 | SEG9 |
| PA15 | SEG10 |
| PB4 | SEG11 |
| PB5 | SEG12 |

| | |
|---|---|
| PC8 | SEG13 |
| PC6 | SEG14 |
| PD14 | SEG15 |
| PD12 | SEG16 |
| PD10 | SEG17 |
| PD8 | SEG18 |
| PB14 | SEG19 |
| PB12 | SEG20 |
| PB0 | SEG21 |
| PC4 | SEG22 |
| PA6 | SEG23 |

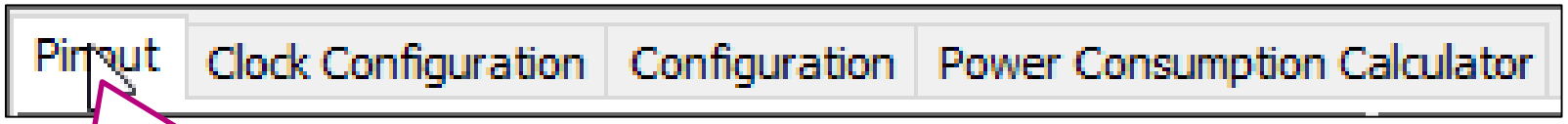



life.augmented

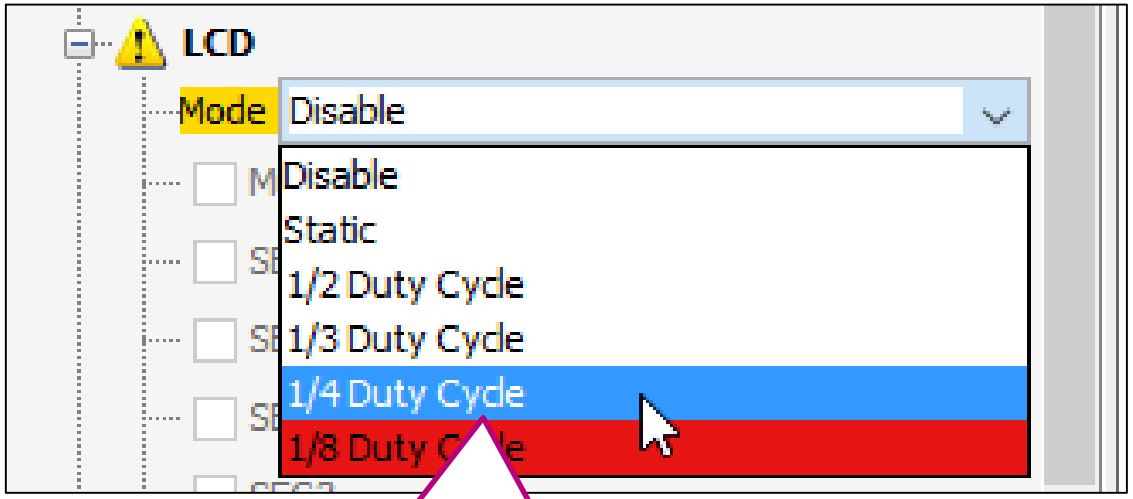
STM32 Cube

# 6 LCD Configuration





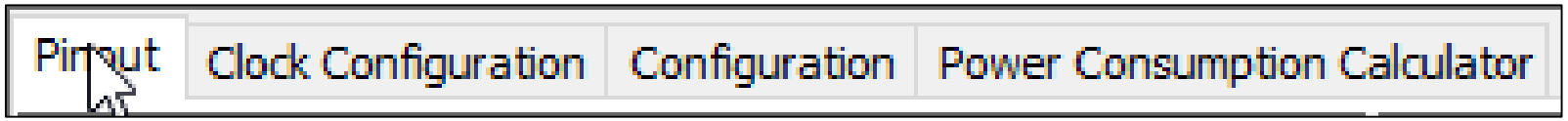

Go to Pinout

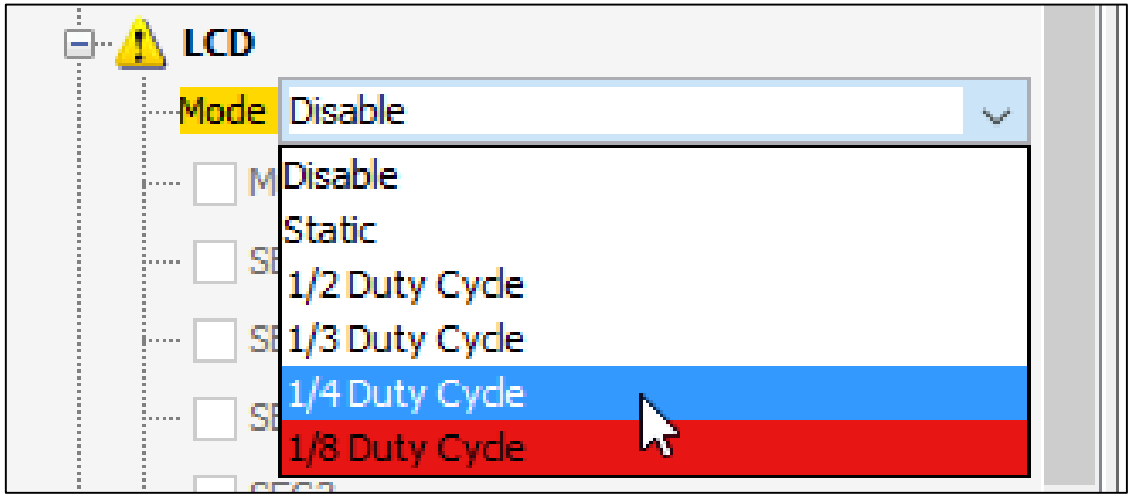


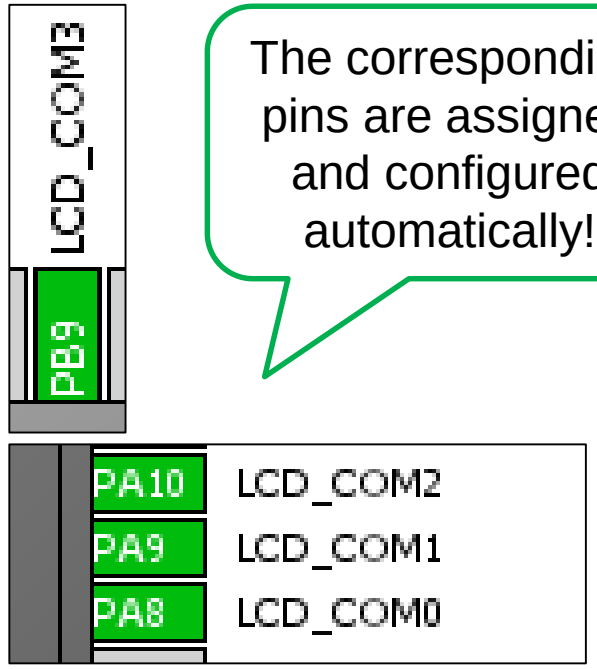

Under LCD peripheral select **1/4 Duty Cycle** mode (as we are using 4 commons)

# 6 LCD Configuration





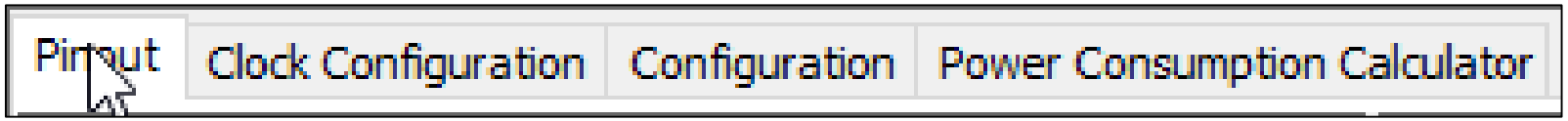



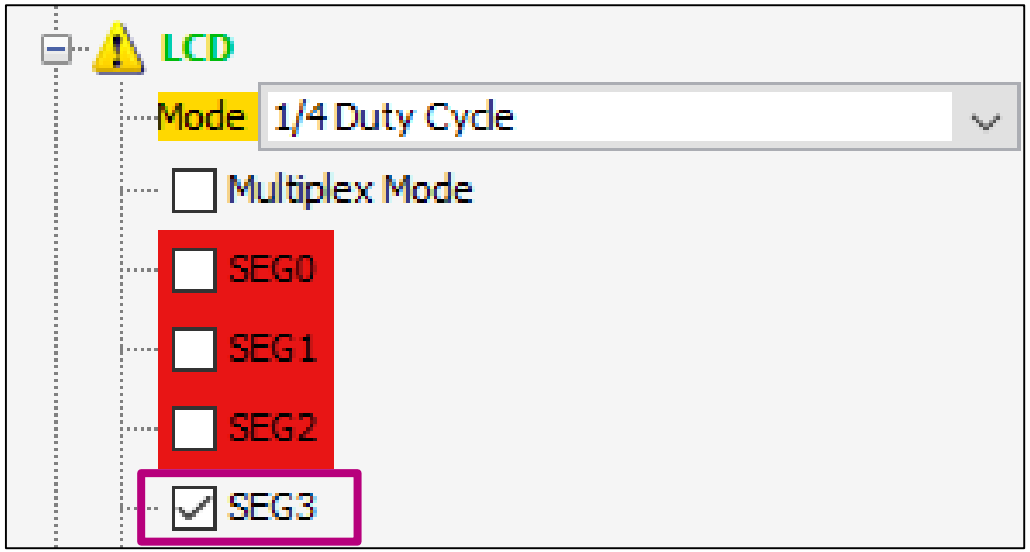

life.augmented

STM32 Cube

# 6 LCD Configuration






| | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| x | 3 | 4 | 5 | 6 | 8 | 9 | 12 | 13 | 14 | 15 | 17 | 22 |
| | 23 | 24 | 25 | 26 | 28 | 29 | 30 | 31 | 32 | 33 | 34 | 35 |

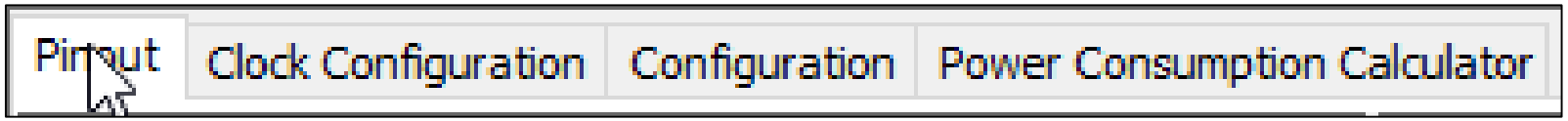
Under LCD peripheral check all the above **SEGx**

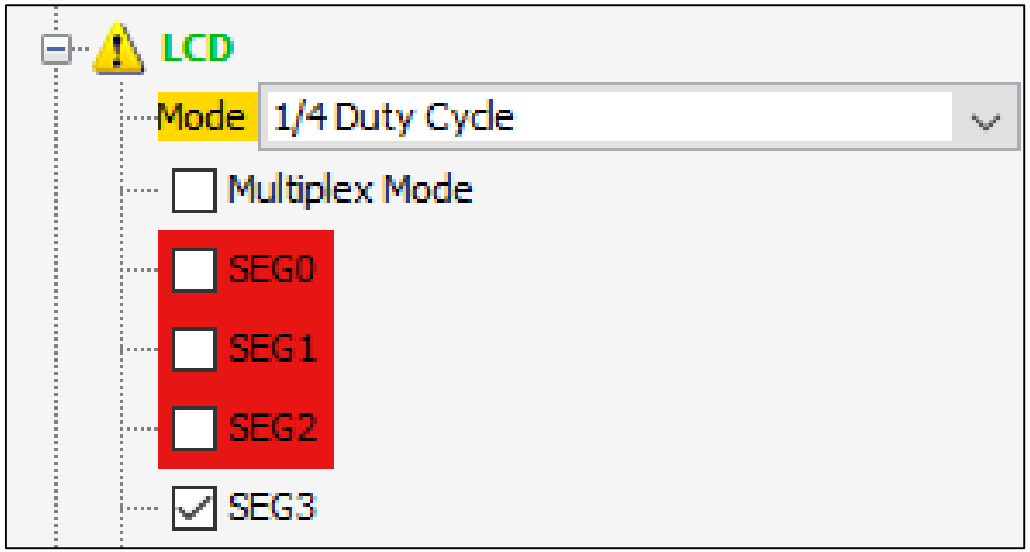
life.augmented

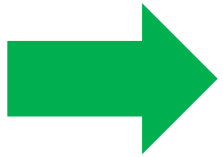
STM32 Cube

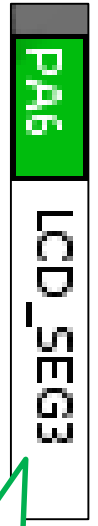
# LCD Configuration








| | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| x | 3 | 4 | 5 | 6 | 8 | 9 | 12 | 13 | 14 | 15 | 17 | 22 |
| | 23 | 24 | 25 | 26 | 28 | 29 | 30 | 31 | 32 | 33 | 34 | 35 |

The corresponding pins are assigned and configured automatically!

life.augmented

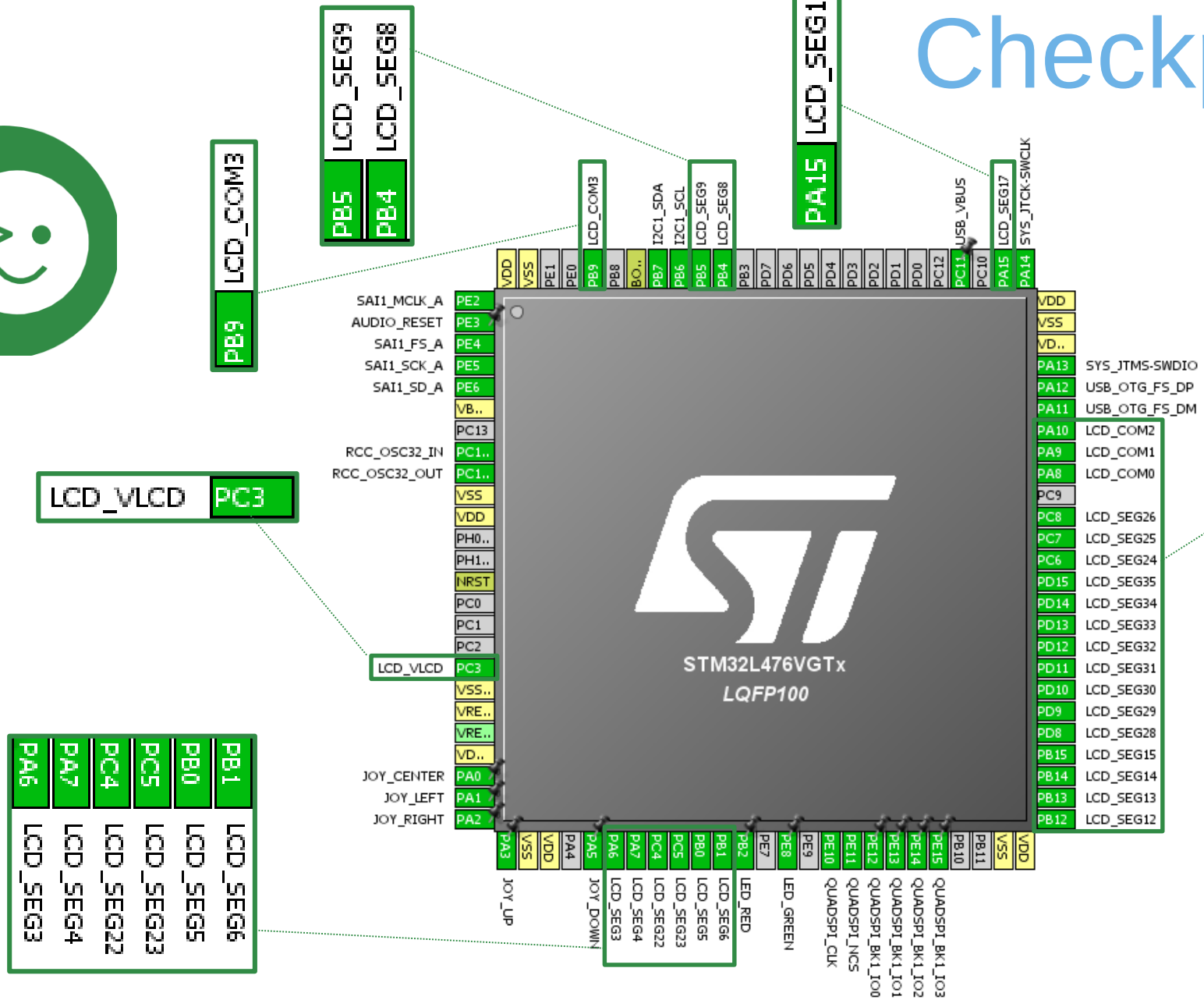
STM32 Cube

# 6

# Checkpoint #17



User Input

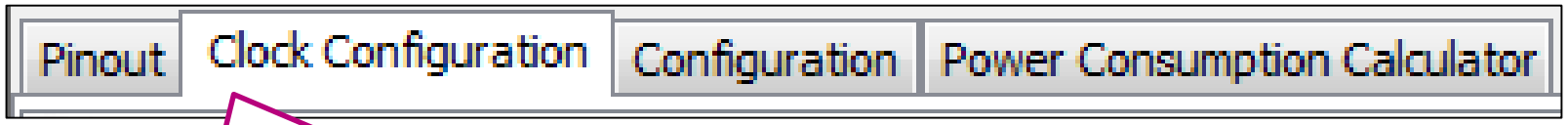



life.augmented

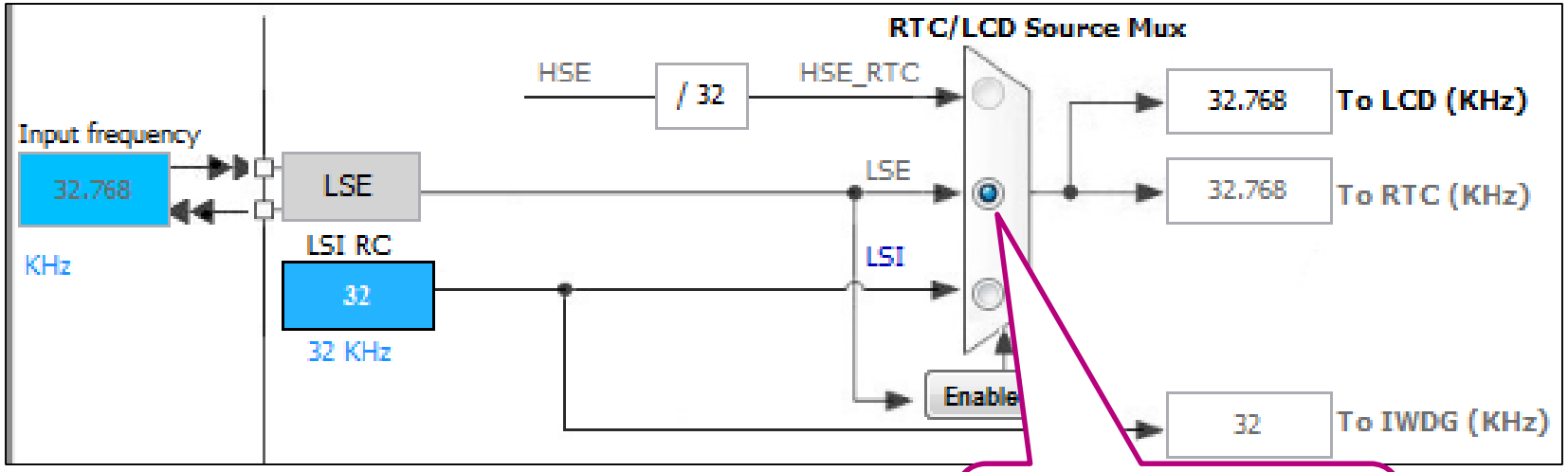
STM32 Cube

# LCD Configuration





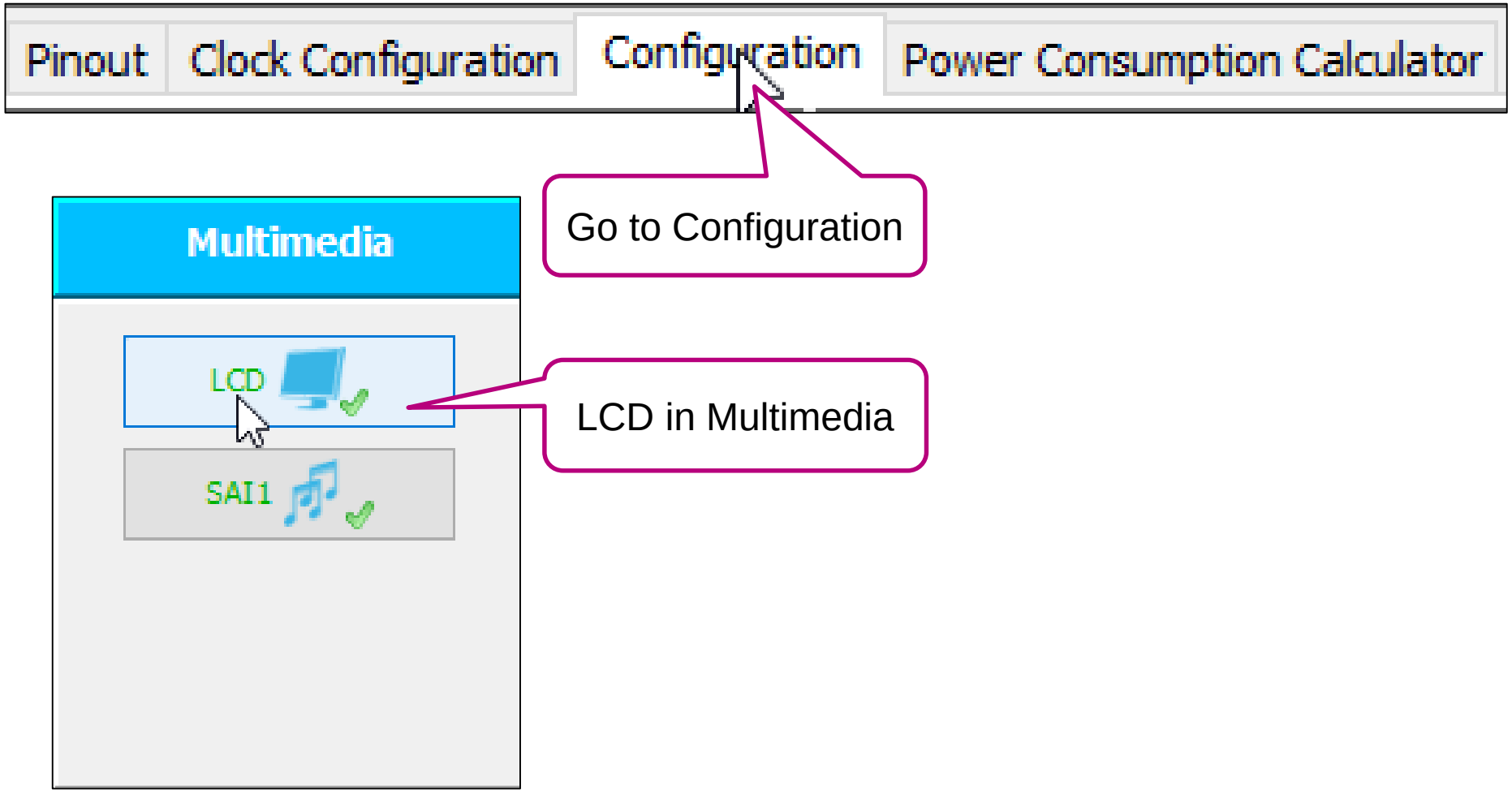

Go to Clock Configuration




LSE
(RTC/LCD Source Mux)

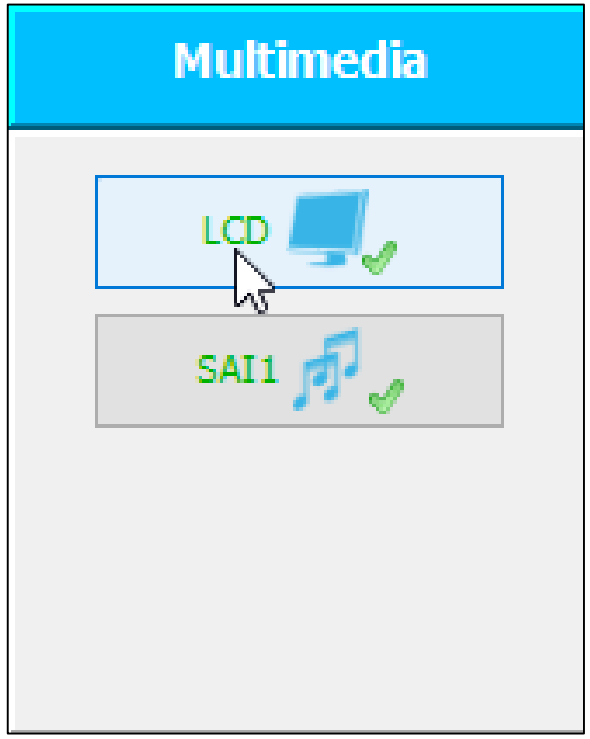
# LCD Configuration



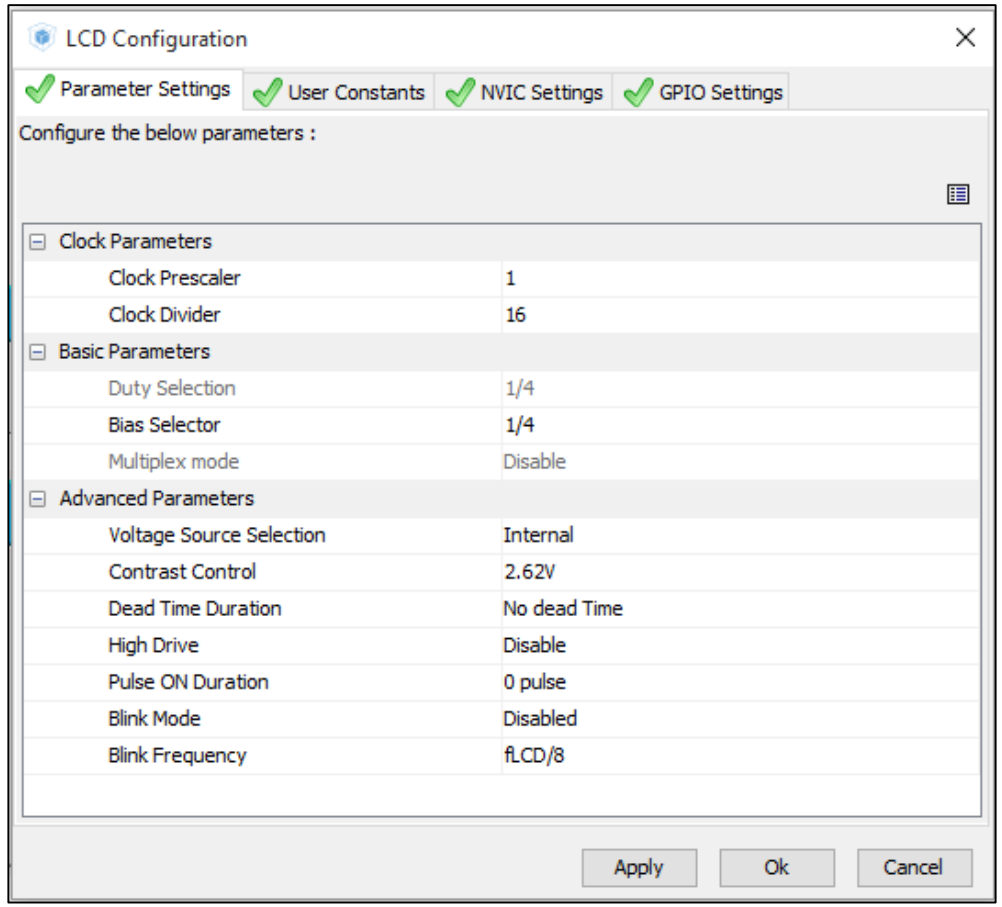


# LCD Configuration



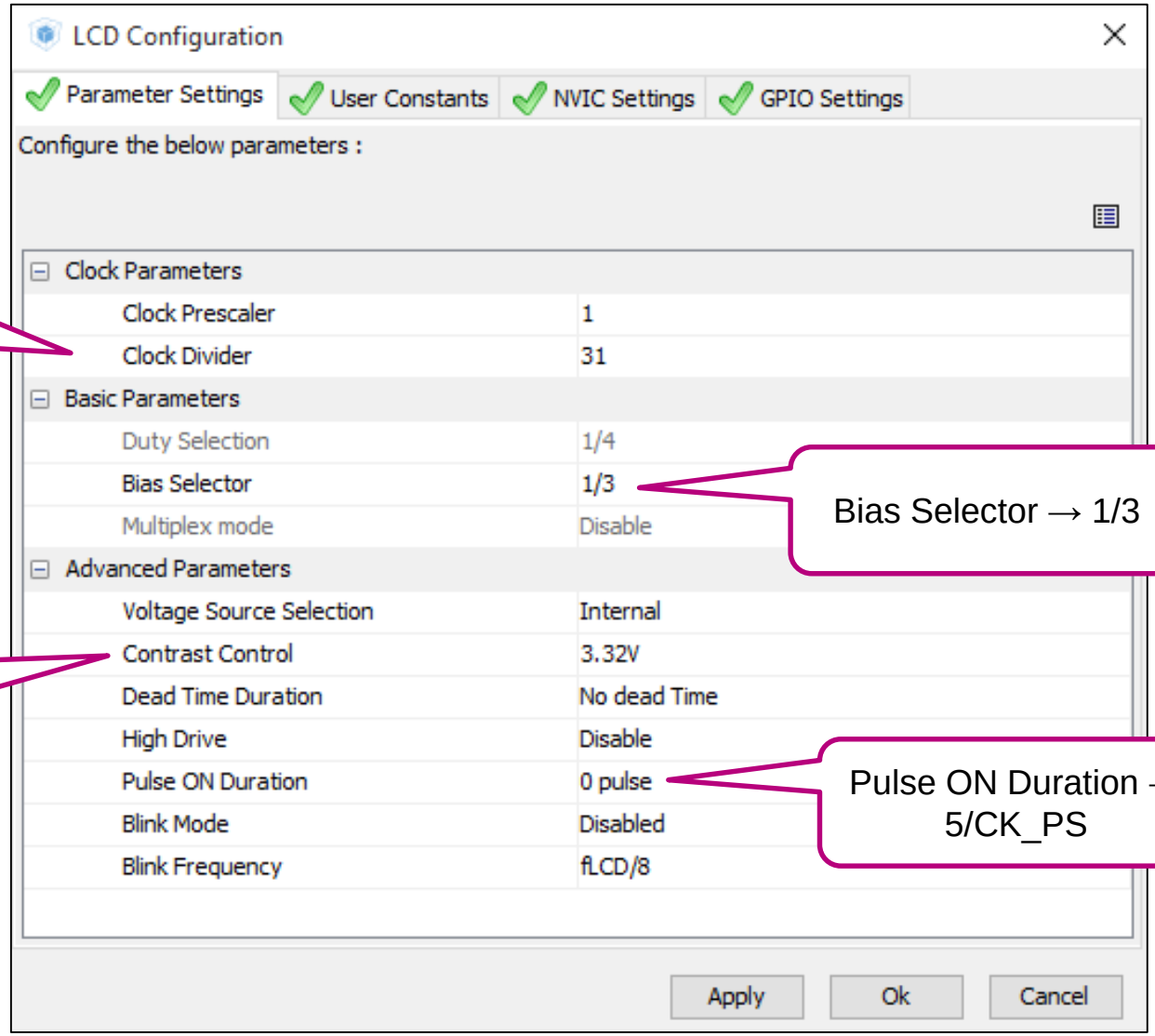





# LCD Configuration






Clock Divider → 31

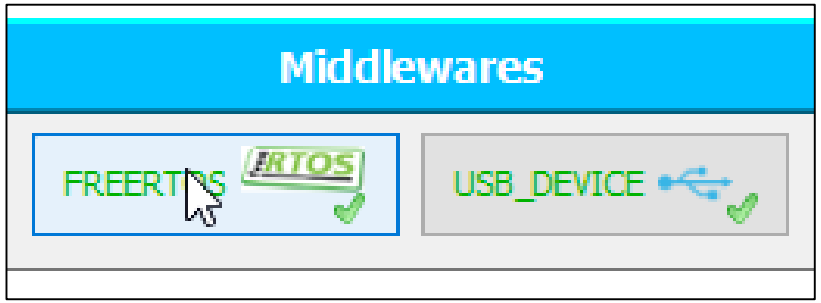
Contrast Control → 3.32V

Bias Selector → 1/3

Pulse ON Duration → 5/CK_PS

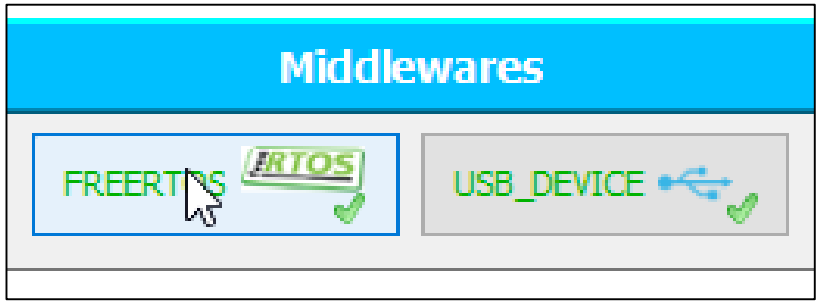
# LCD Configuration

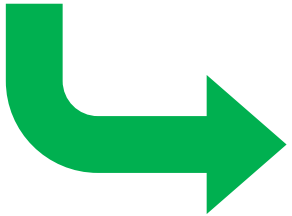


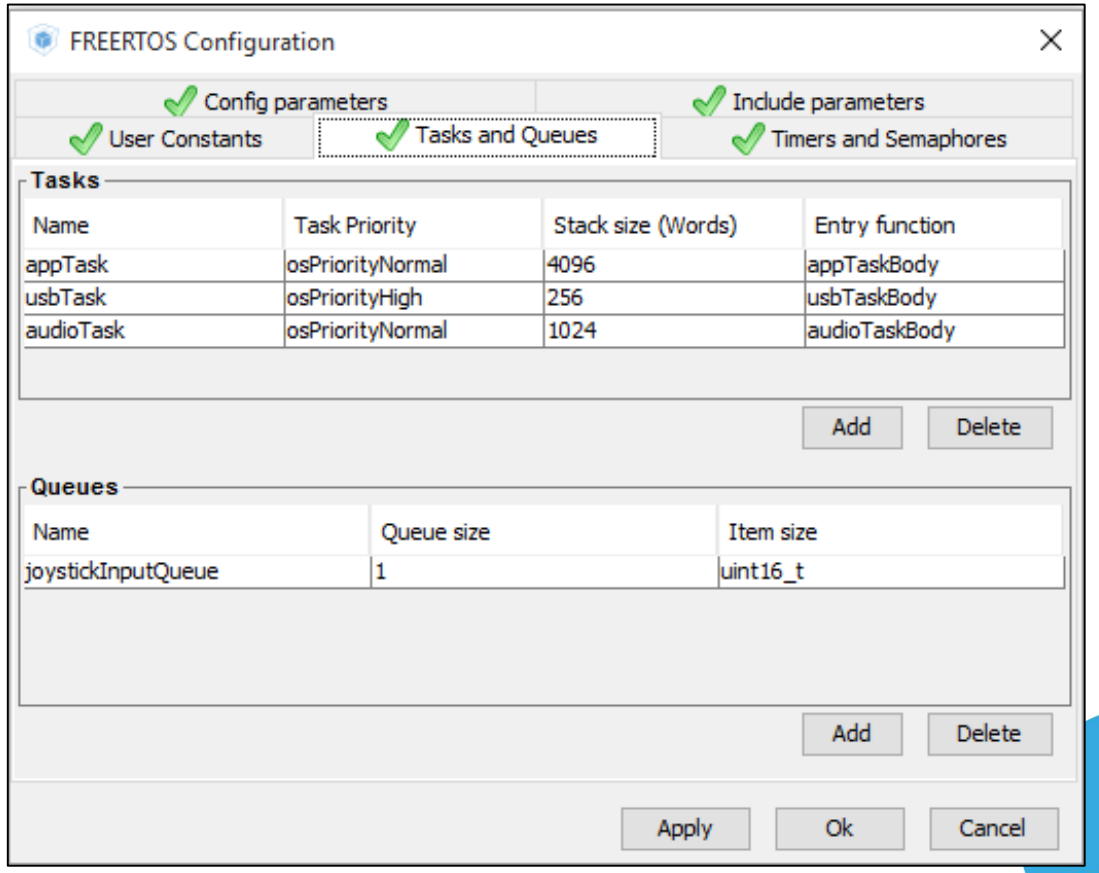



# LCD Configuration



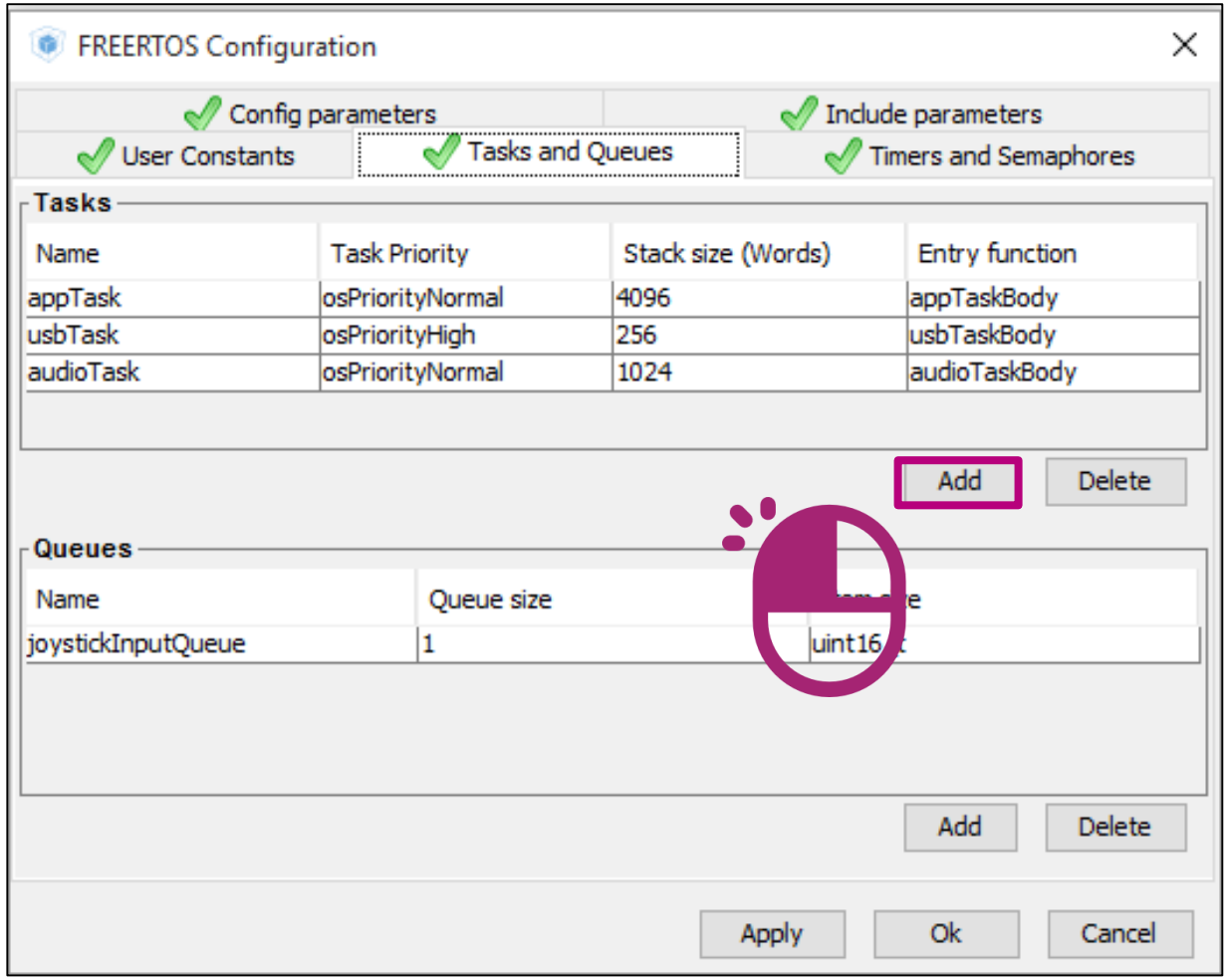



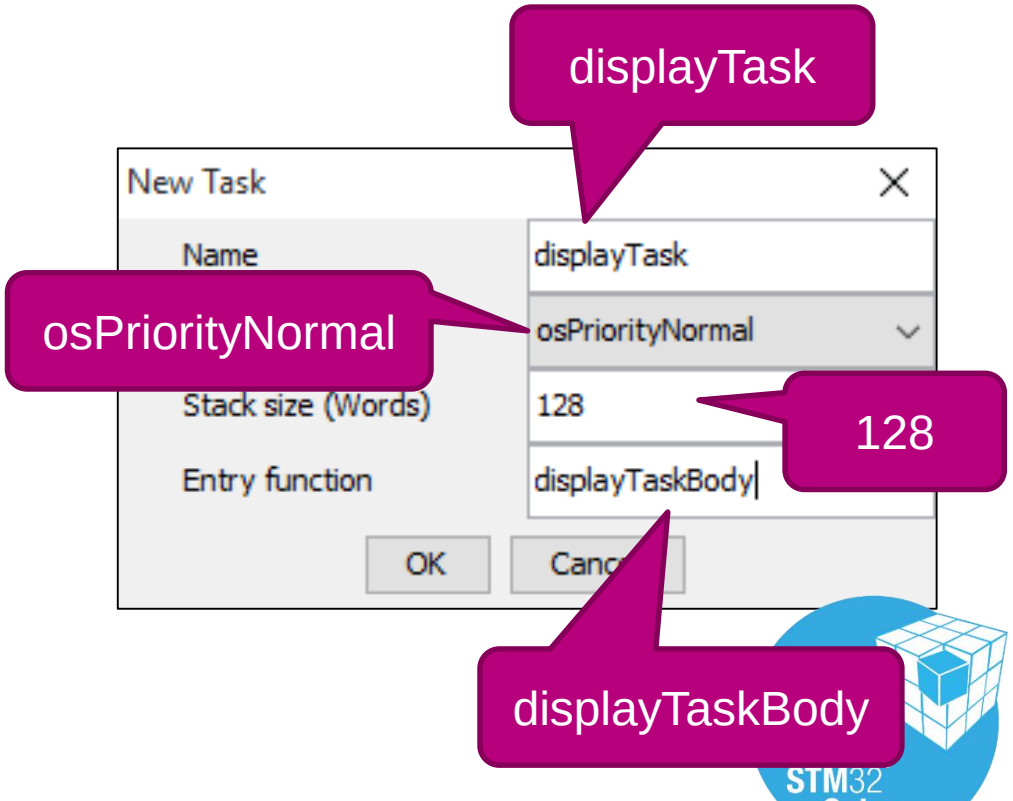


# LCD Configuration






displayTask



# Checkpoint #18



User Input

FREERTOS Configuration

Config parameters | Include parameters | User Constants | Tasks and Queues | Timers and Semaphores

**Tasks**

| Name | Task Priority | Stack size (Words) | Entry function |
|---|---|---|---|
| appTask | osPriorityNormal | 4096 | appTaskBody |
| usbTask | osPriorityHigh | 256 | usbTaskBody |
| audioTask | osPriorityNormal | 1024 | audioTaskBody |
| displayTask | osPriorityNormal | 128 | displayTaskBody |

Add Delete

**Queues**

| Name | Queue size | Item size |
|---|---|---|
| joystickInputQueue | 1 | uint16_t |

Add Delete

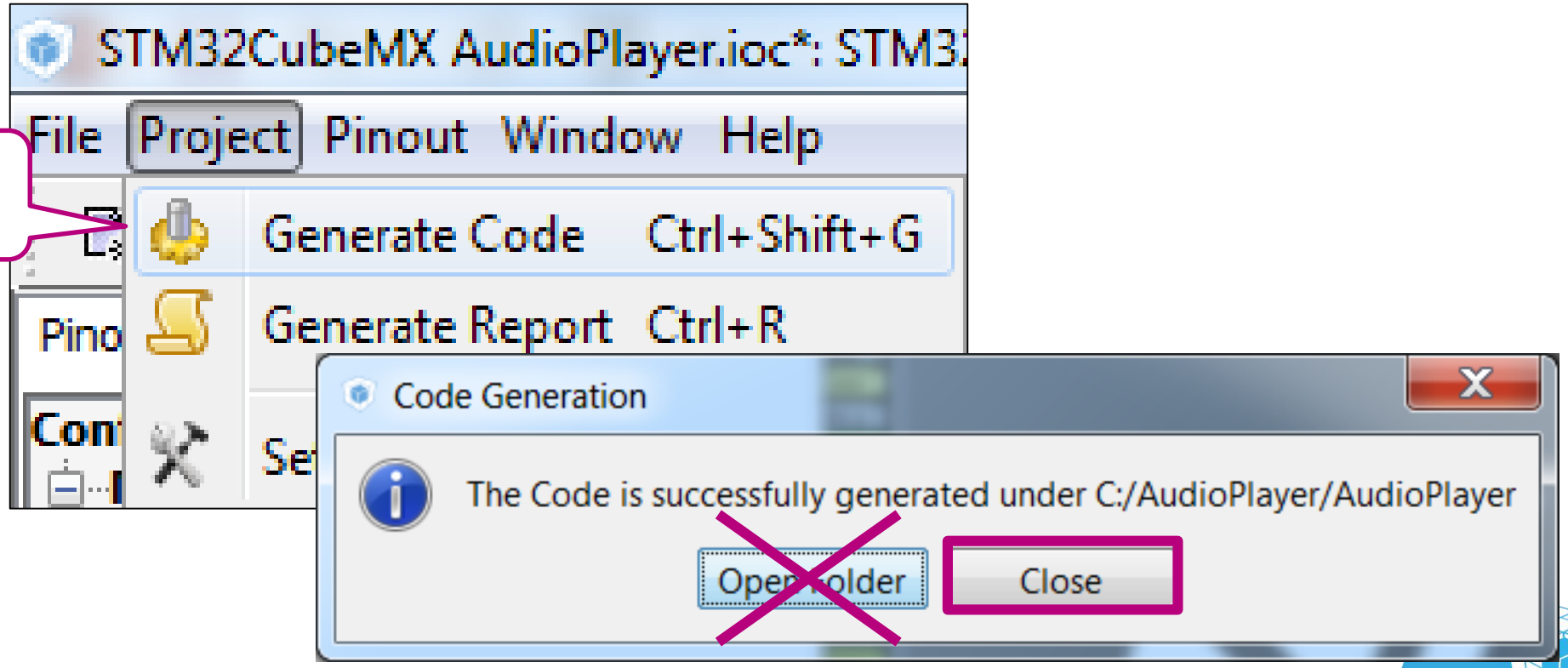
Apply Ok Cancel

# 6 Save and generate the project



Do not open the project yet

Generate Code
(Ctrl+Shift+G)



# 6 Apply small patch



- Apply patch from folder
  **C:\STM32L4_Workshop\HandsOn\2_Putting_All_Together\Patch\STEP6_Display**
  to root folder of your project
  (C:\AudioPlayer\AudioPlayer\)

- It contains the following files:
  .\Inc\
    lcd.h
  .\Src\
    freertos.c
    lcd.c
    usbd_conf.c

This operation is STM32CubeMX non-intrusive. You can re-generate the project later and the modifications will be retained.

- These files contains the main needed source code the user has to add to work with the particular glass LCD available on the STM32L4 discovery kits

# lcd.c content



- lcd.c

```
LCD_GLASS_Clear(void);
LCD_GLASS_DisplayString(…);
```



Based on BSP driver in STM32CubeL4 HAL

# 6

# Let’s try it out



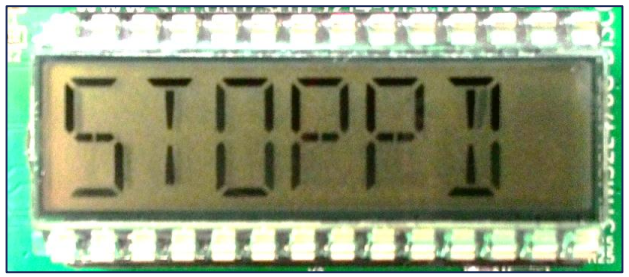
1. Open the project

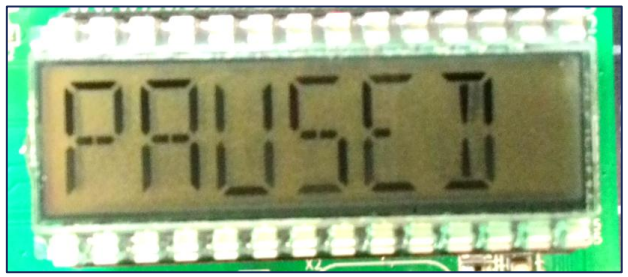
2. Compile and download

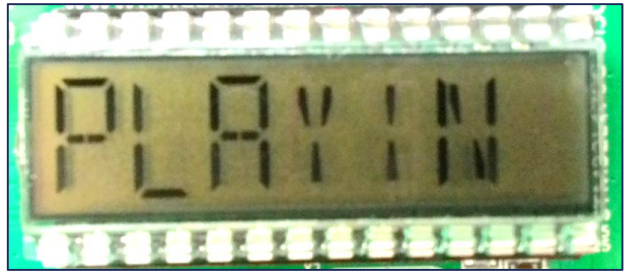
3. Press the reset button on discovery once download has finished

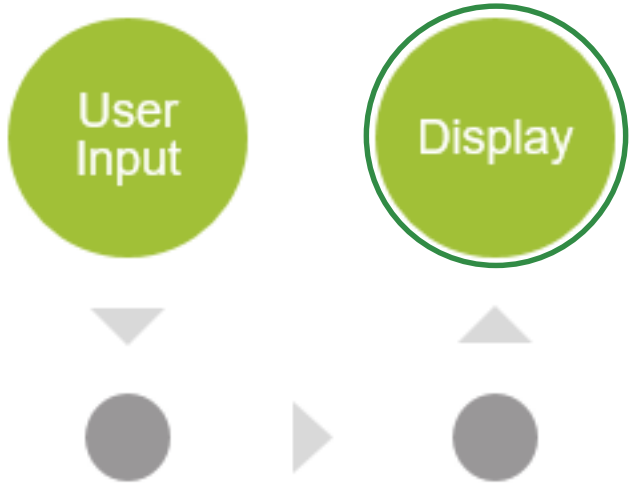
# Checkpoint #19





# Why new Thread?



Because it can run completely independent on the other threads in the application

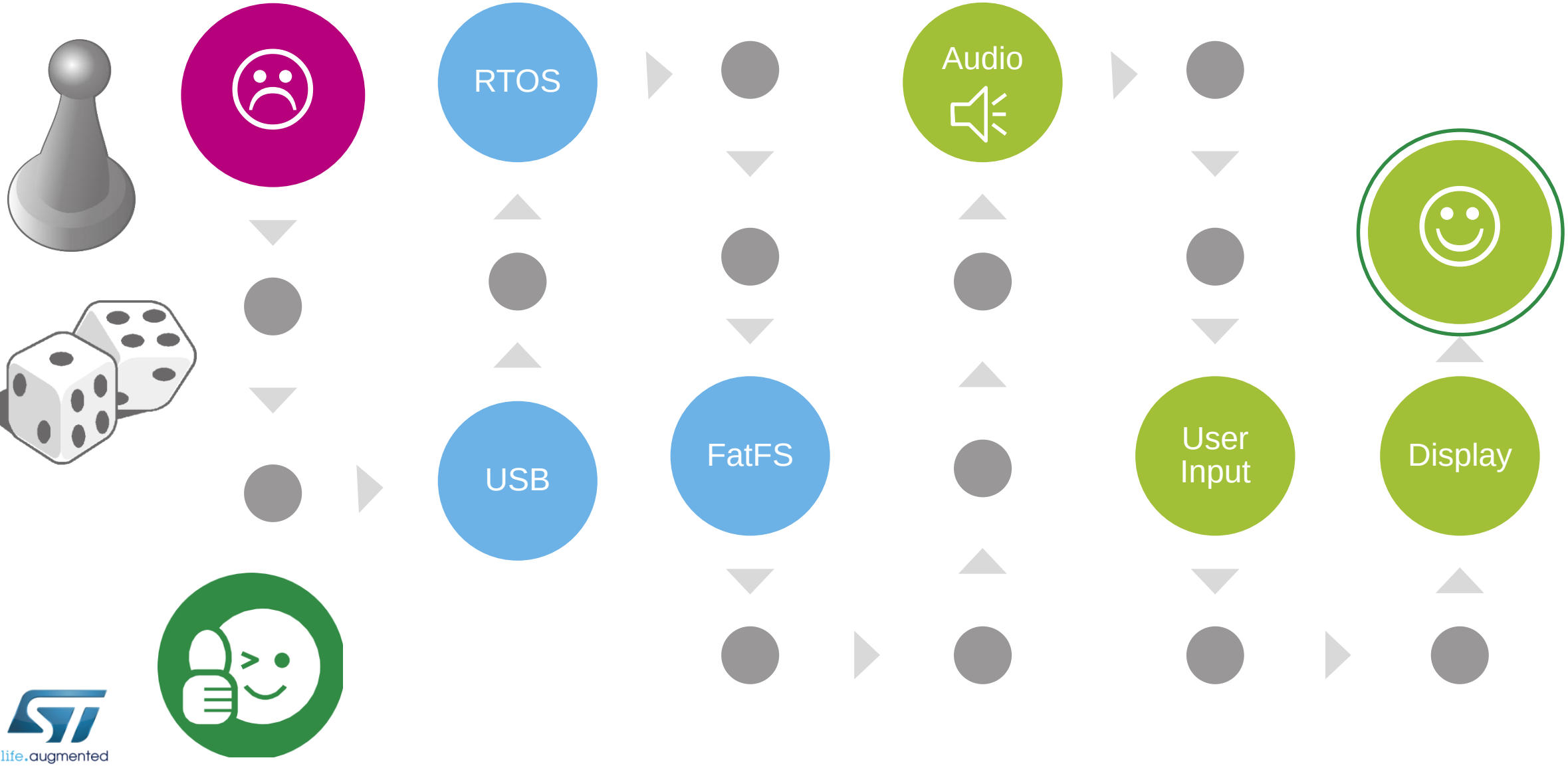
- You can implement previous state change checking mechanism

**EXAMPLE USE-CASE**

You may want to add animations or blinking segments with changing refresh rate related to the timeout variable used as argument of the osDelay(…) call.

# Finish





# Enjoy!




www.st.com/stm32l4