

STM32L4 – Putting All Together

STM32L4 Workshop





Goal of this part

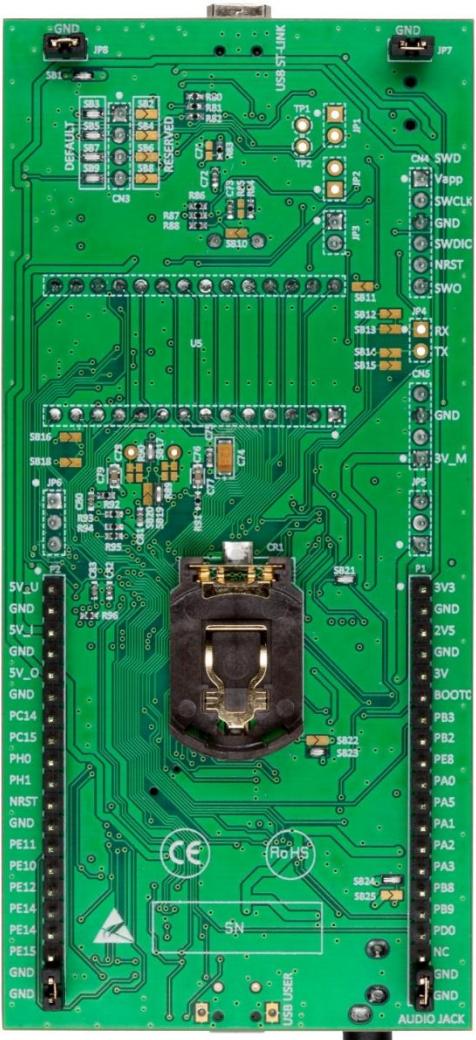
2

- 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

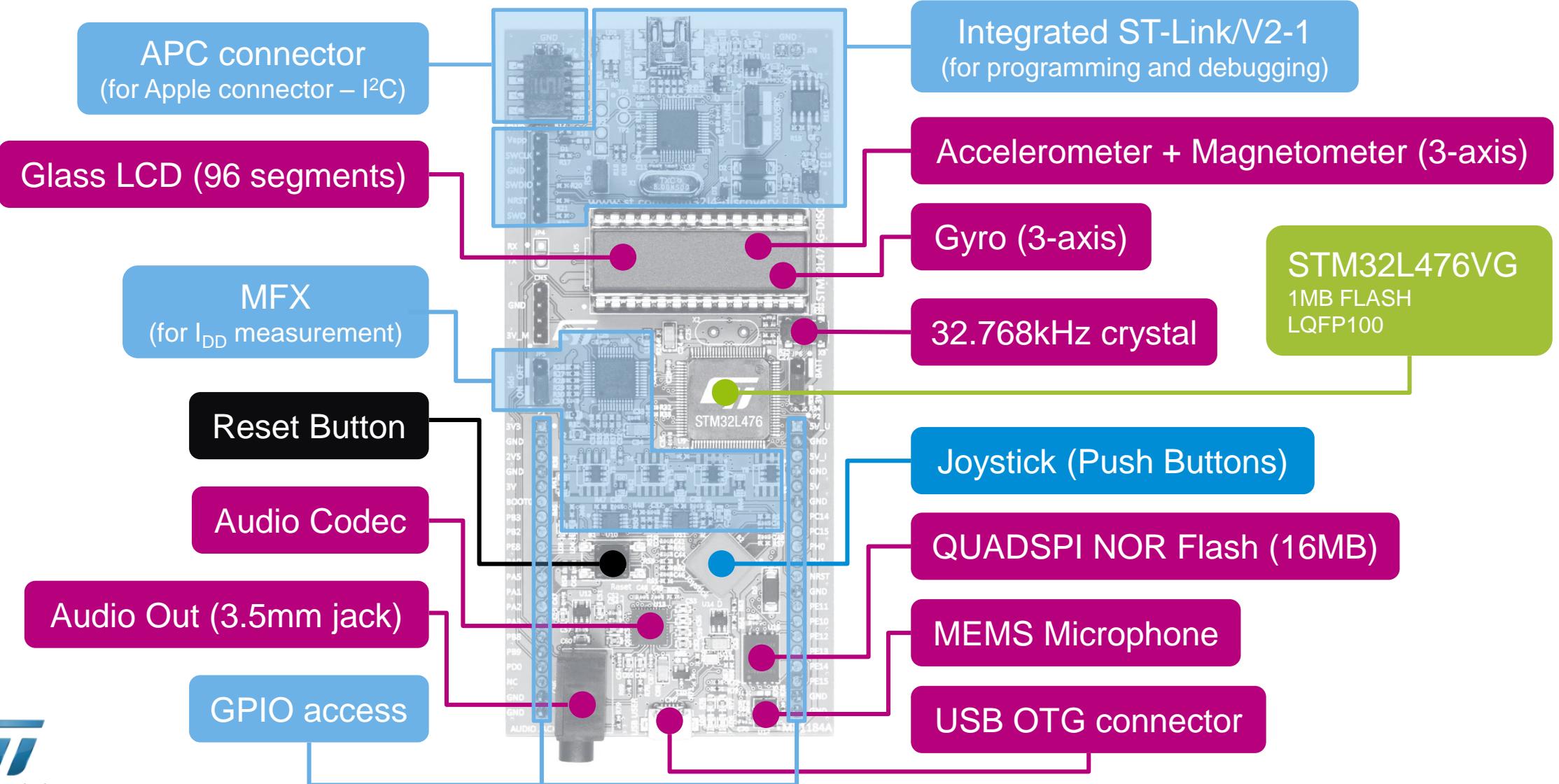
3





STM32L476 Discovery

4

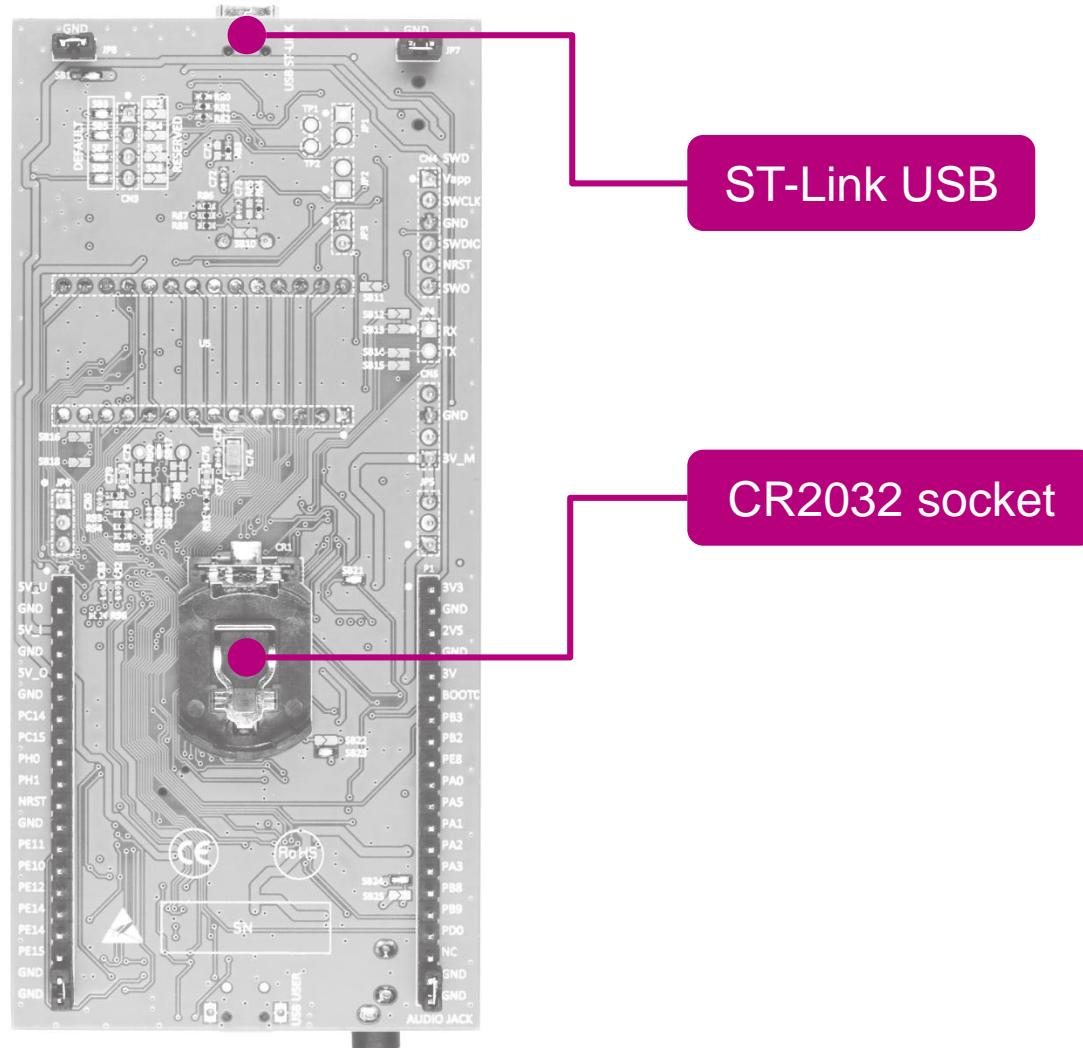




STM32L476 Discovery

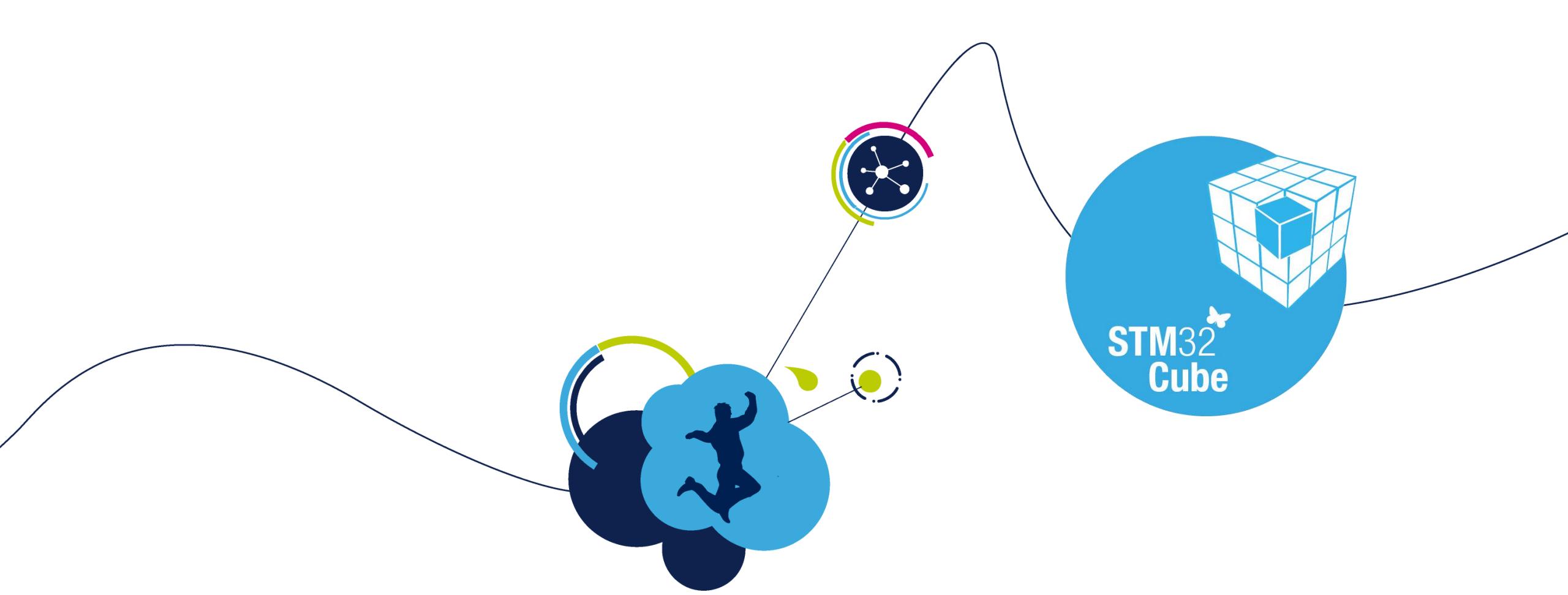
5

Flexible board
power supply



ST-Link USB

CR2032 socket



Let's start to work!

Application idea

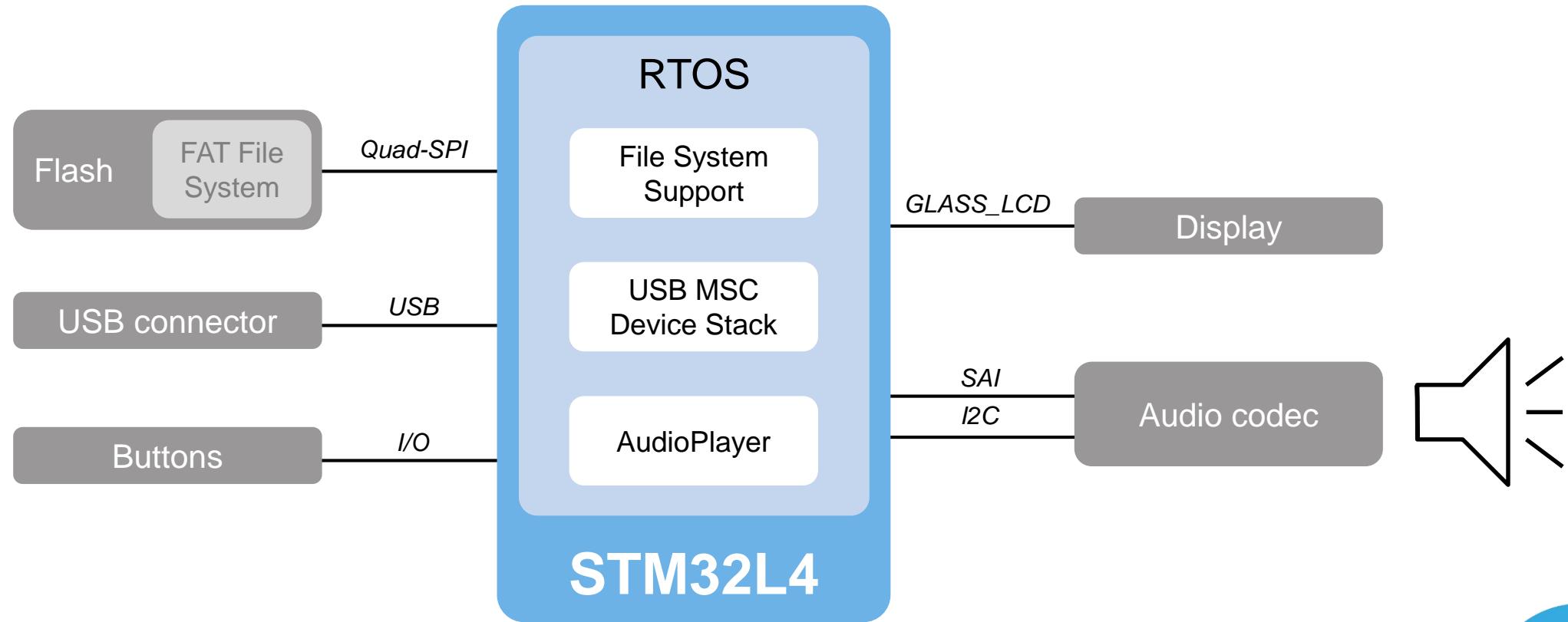
7

- Create simple Audio Player



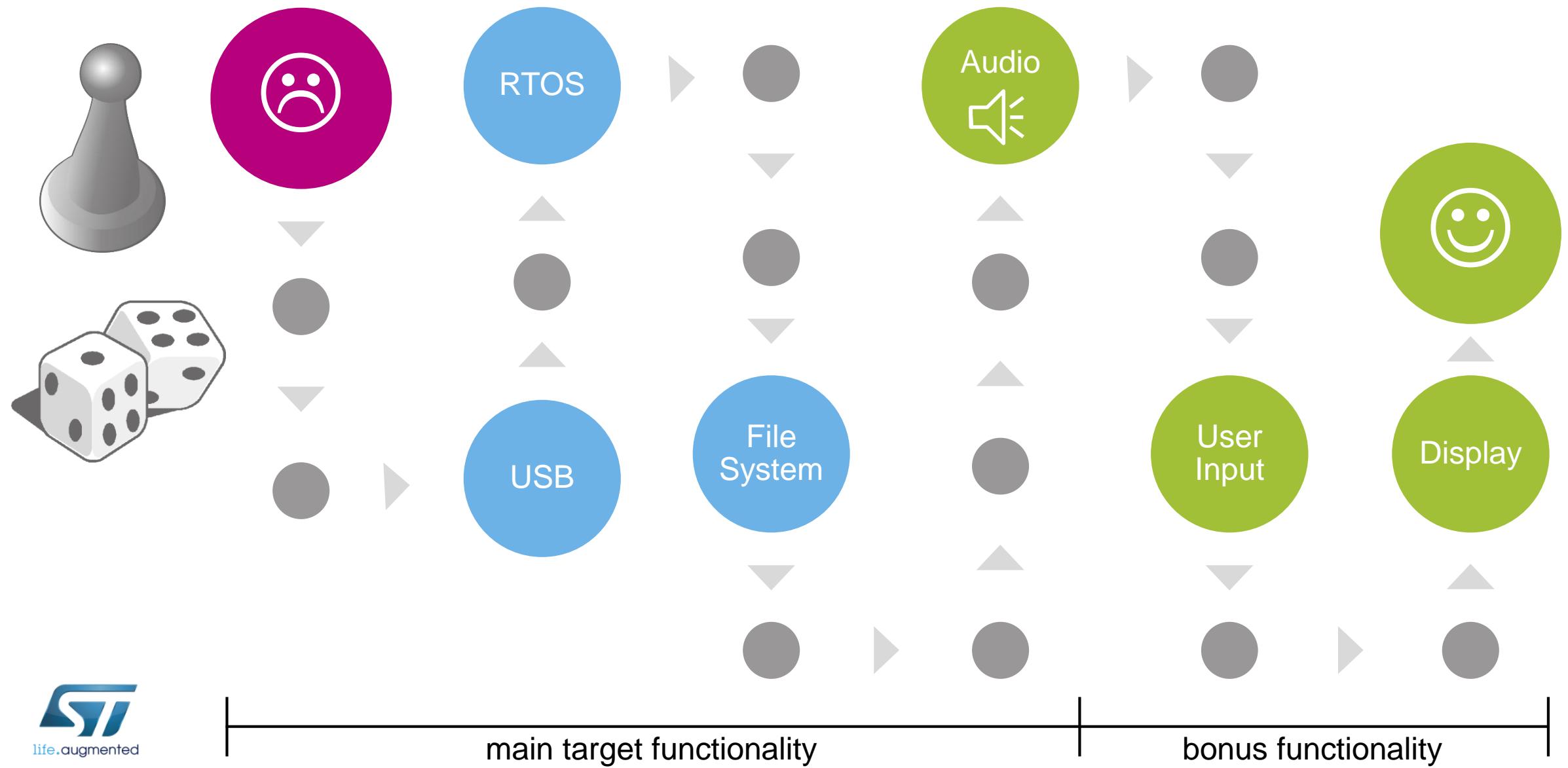
Simplified target block diagram

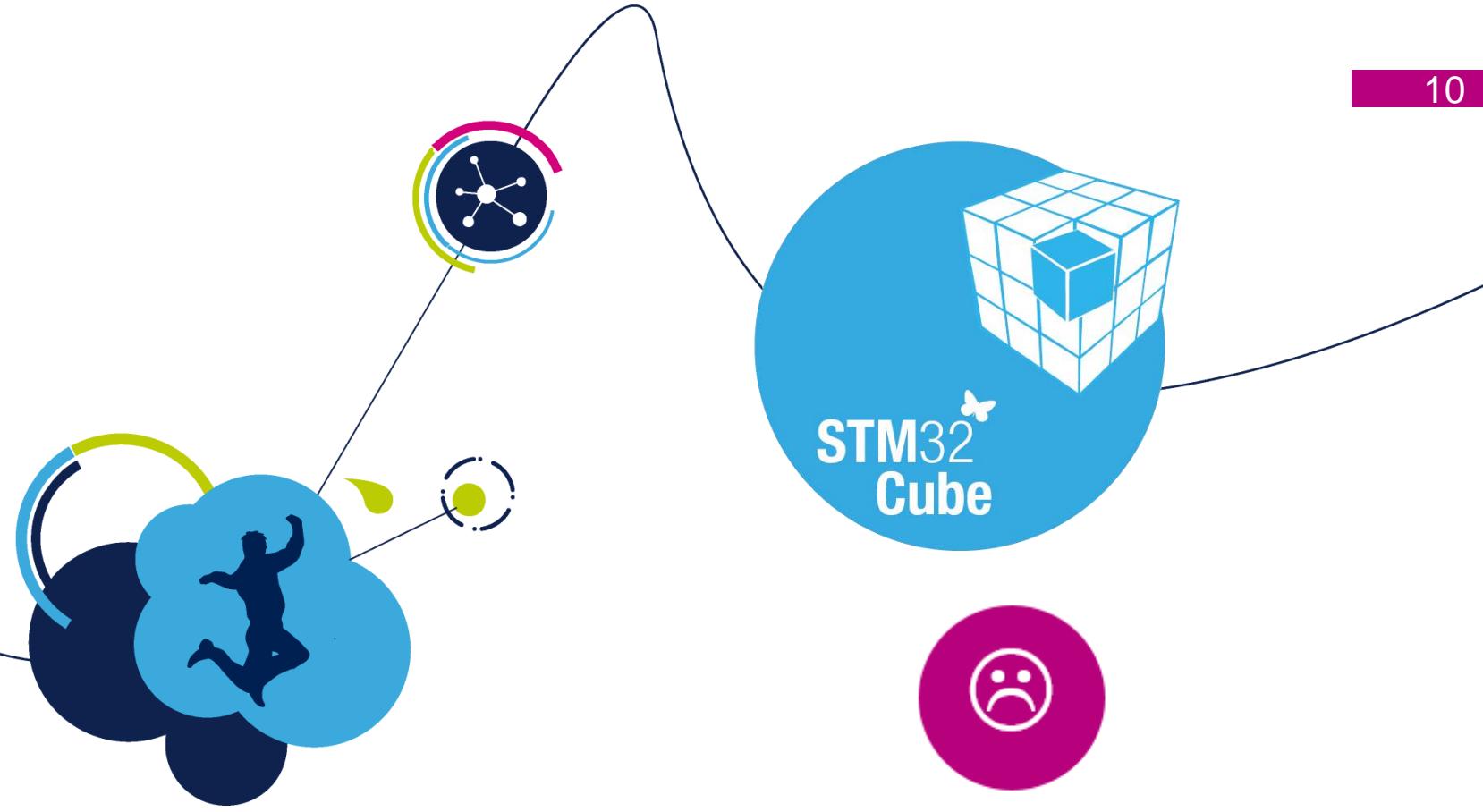
8



„Game“ flow

9



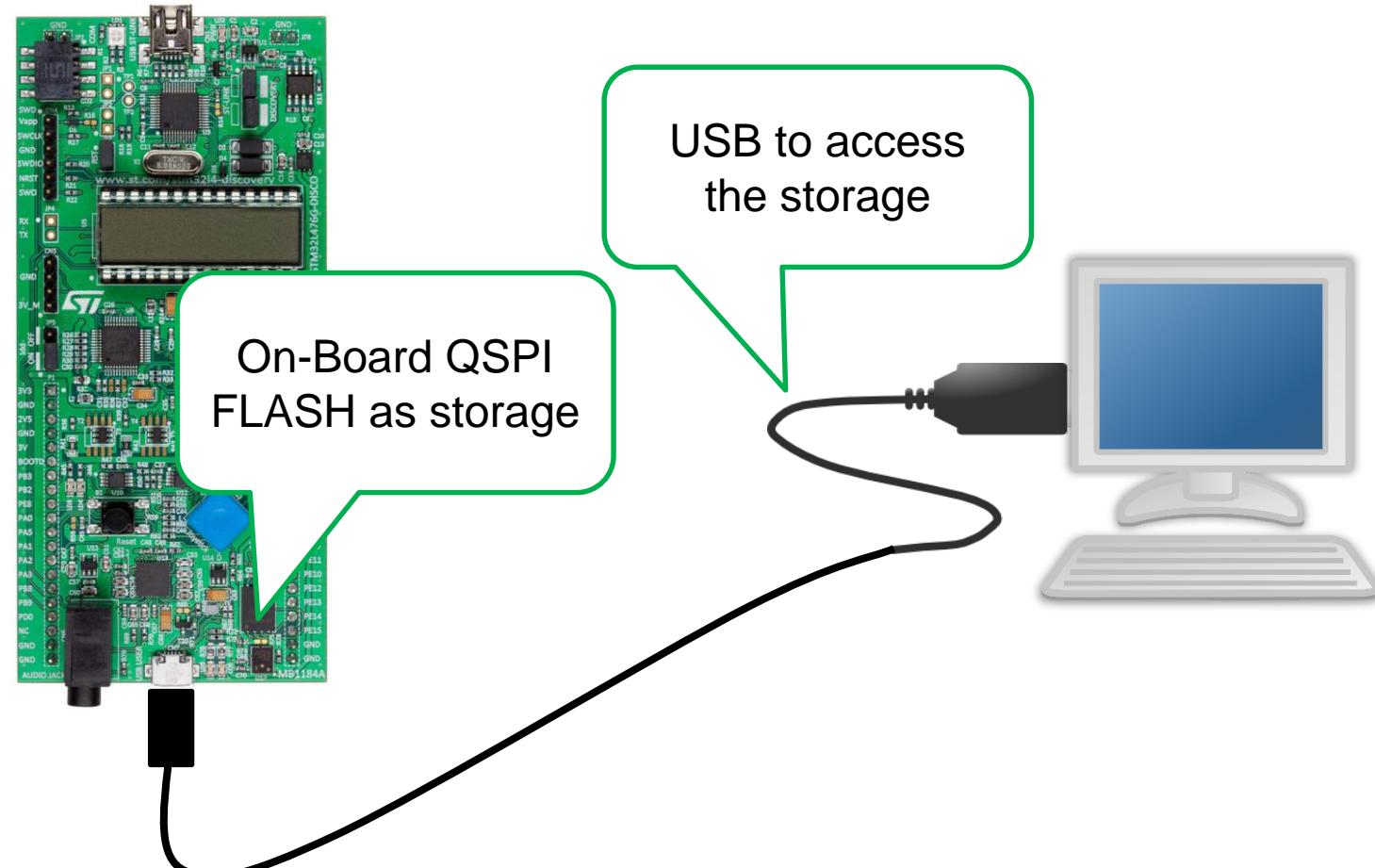


STEP1 – USB MSC Device

1 USB MSC Device with QUADSPI

11

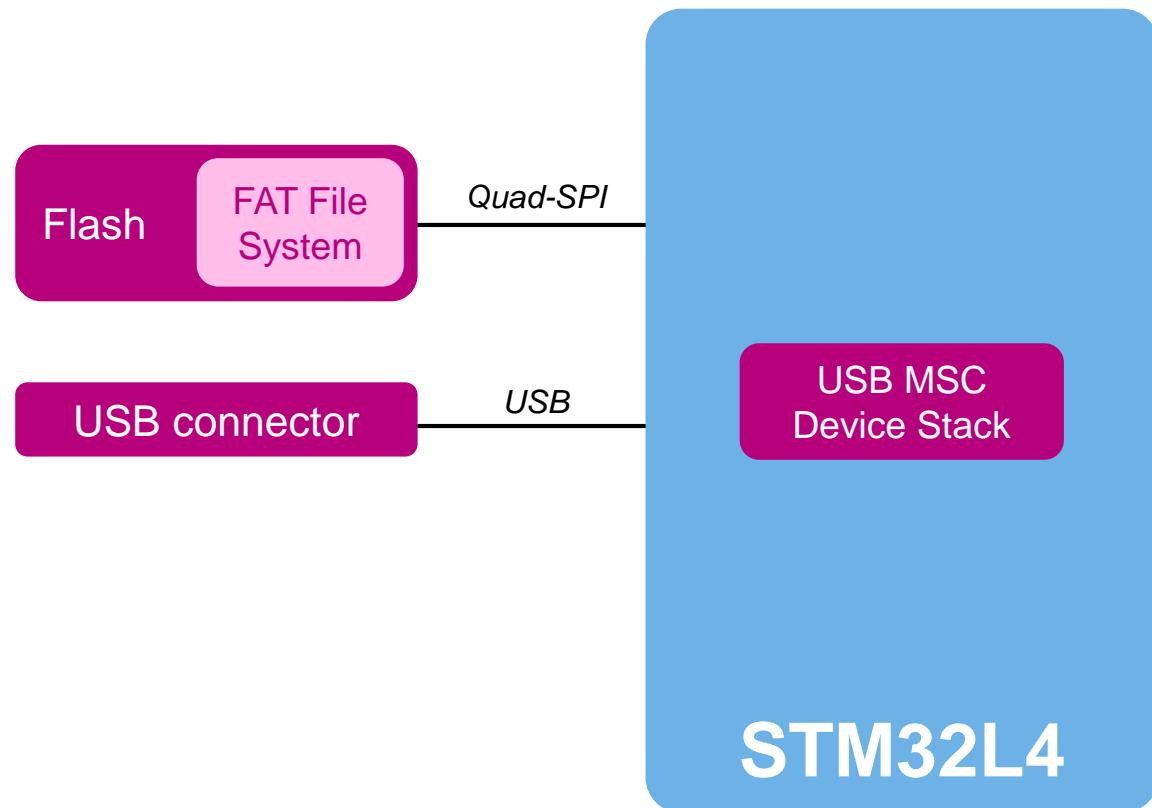
- We need storage for our audio files!



1 Adding USB MSC Device functionality

12

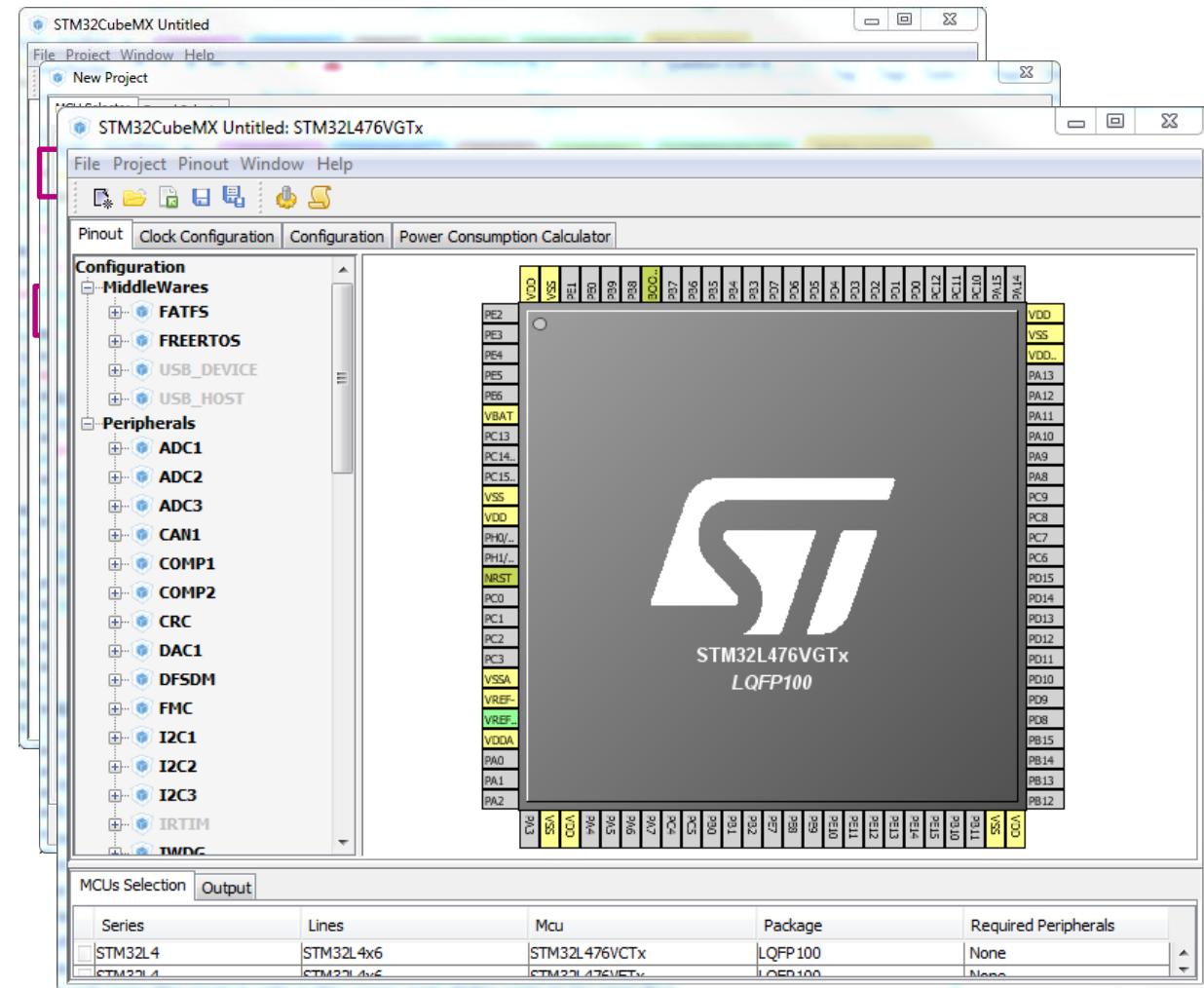
- We need storage for our audio files!

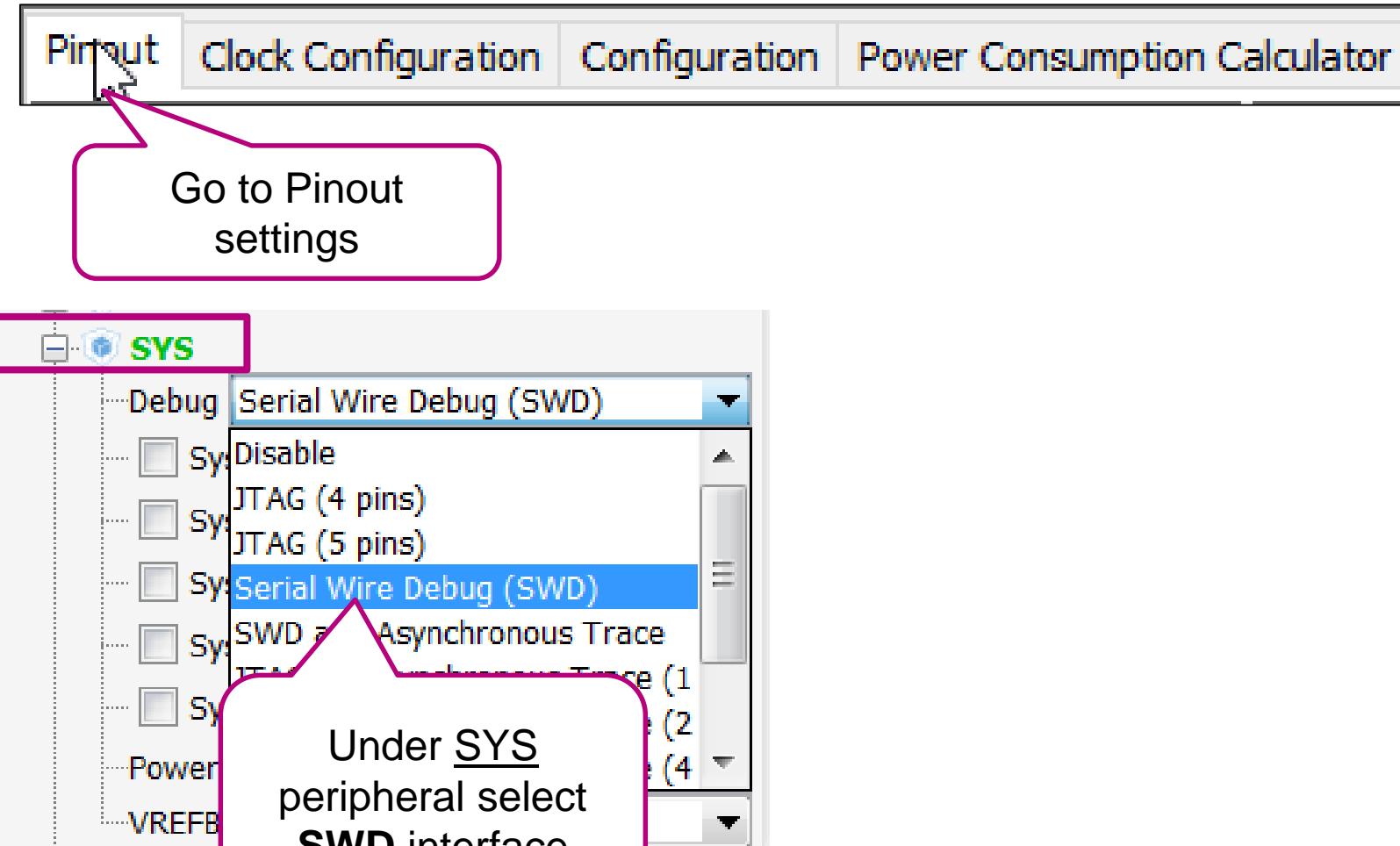


1 Create new project in CubeMX

13

- 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



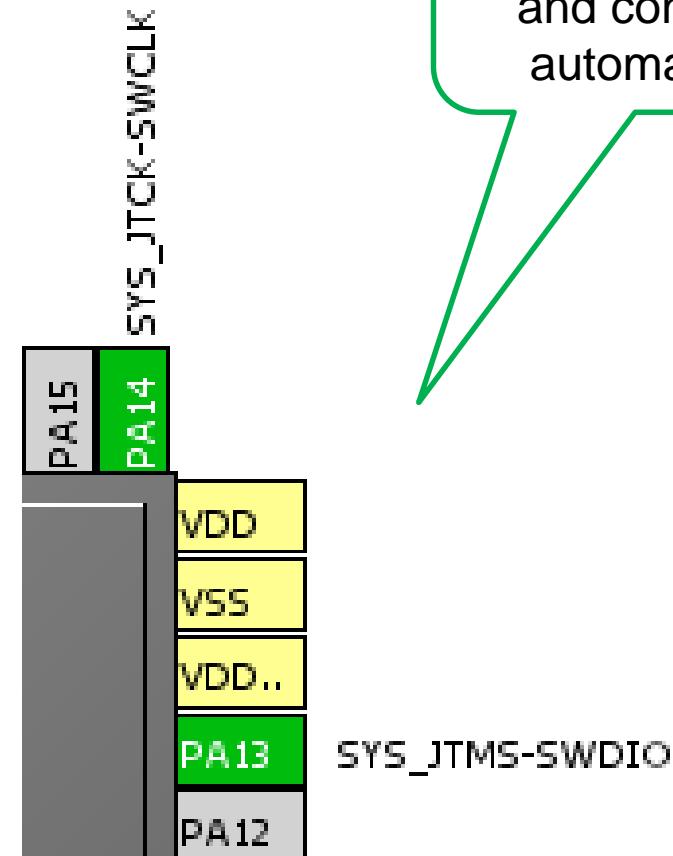


1 Configure debug interface

15

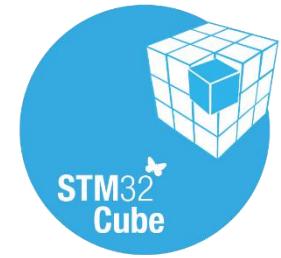
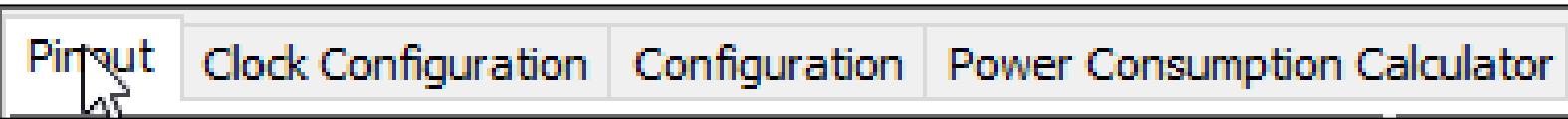
The screenshot shows the STM32CubeMX software interface. The top navigation bar includes tabs for Pinout, Clock Configuration, Configuration, and Power Consumption Calculator. The Pinout tab is currently active, indicated by a mouse cursor icon. In the main window, the **SYS** group is expanded, showing the **Debug** section. The **Serial Wire Debug (SWD)** option is selected, highlighted with a blue background. A dropdown menu below it lists other options: Disable, JTAG (4 pins), JTAG (5 pins), Serial Wire Debug (SWD), SWD and Asynchronous Trace, JTAG and Synchronous Trace (1), JTAG and Synchronous Trace (2), and JTAG and Synchronous Trace (4). Below the Debug section, there is a VREFBUF Mode setting set to **Disable**. A large green arrow points from the selected **Serial Wire Debug (SWD)** option to the pinout diagram on the right.

The corresponding pins are assigned and configured automatically!

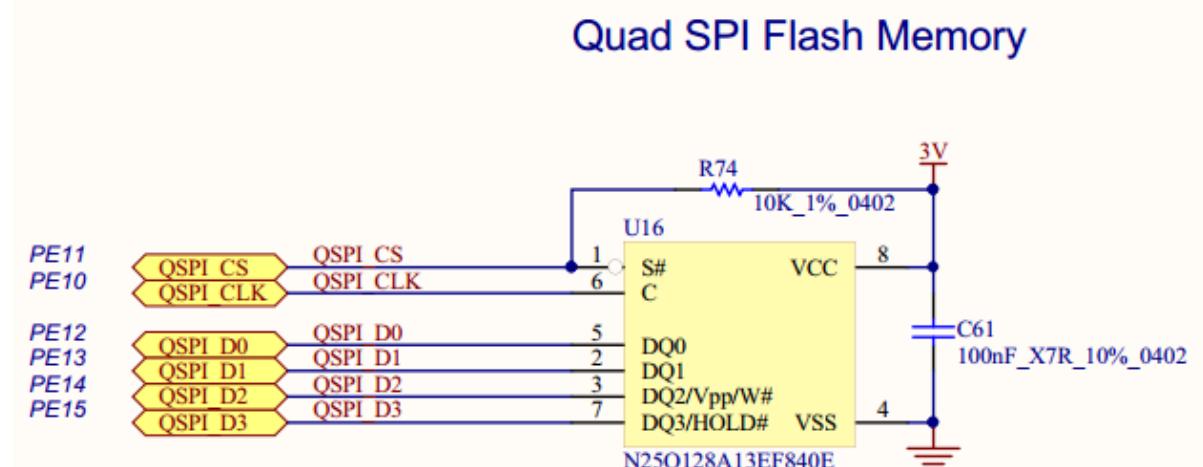


1 Configure QUADSPI interface

16

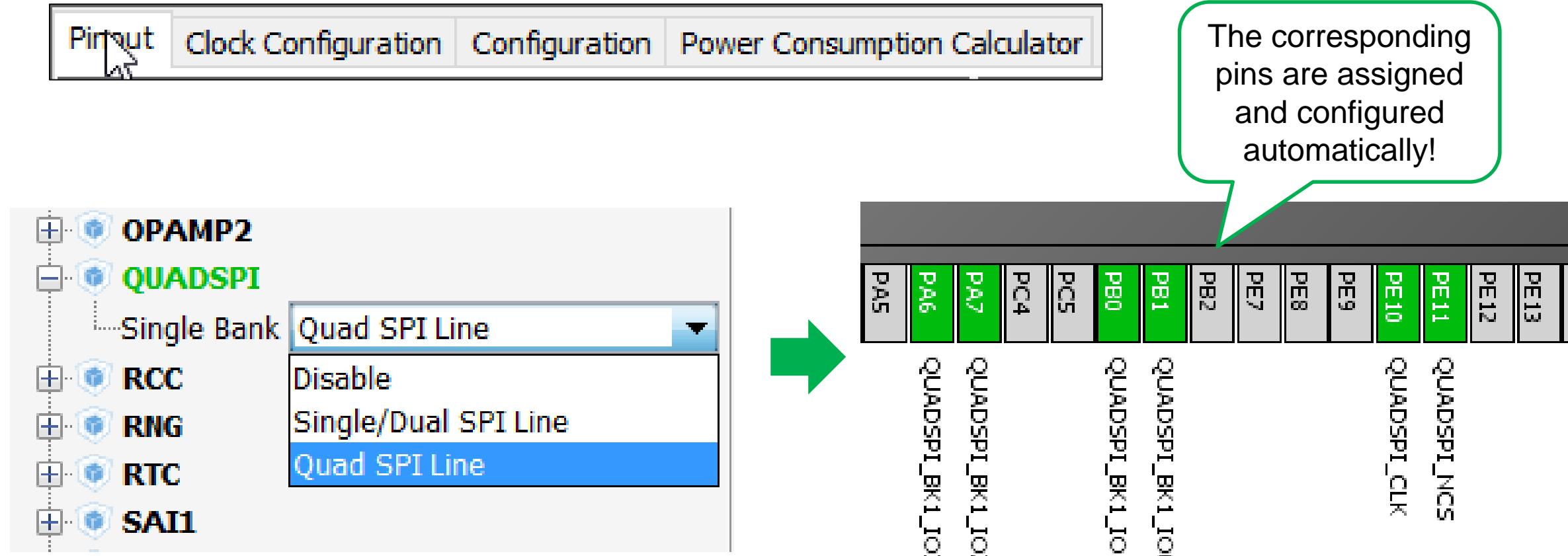


Under QUADSPI
peripheral select
Quad SPI Line
configuration



1 Configure QUADSPI interface

17



1 QUADSPI interface pins remapping

18

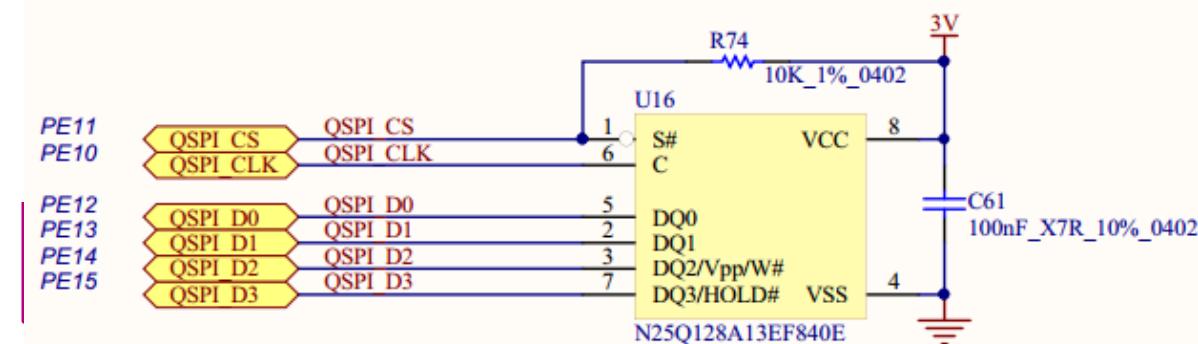


PA5	PA6	PA7	PC4	PC5	PB0	PB1	PB2	PE2	PE7	PE8	PE9	PE10	PE11	PE12	PE13	PE14	PE15
QUADSPI_BK1_IO2	QUADSPI_BK1_IO1																

PE11
PE10

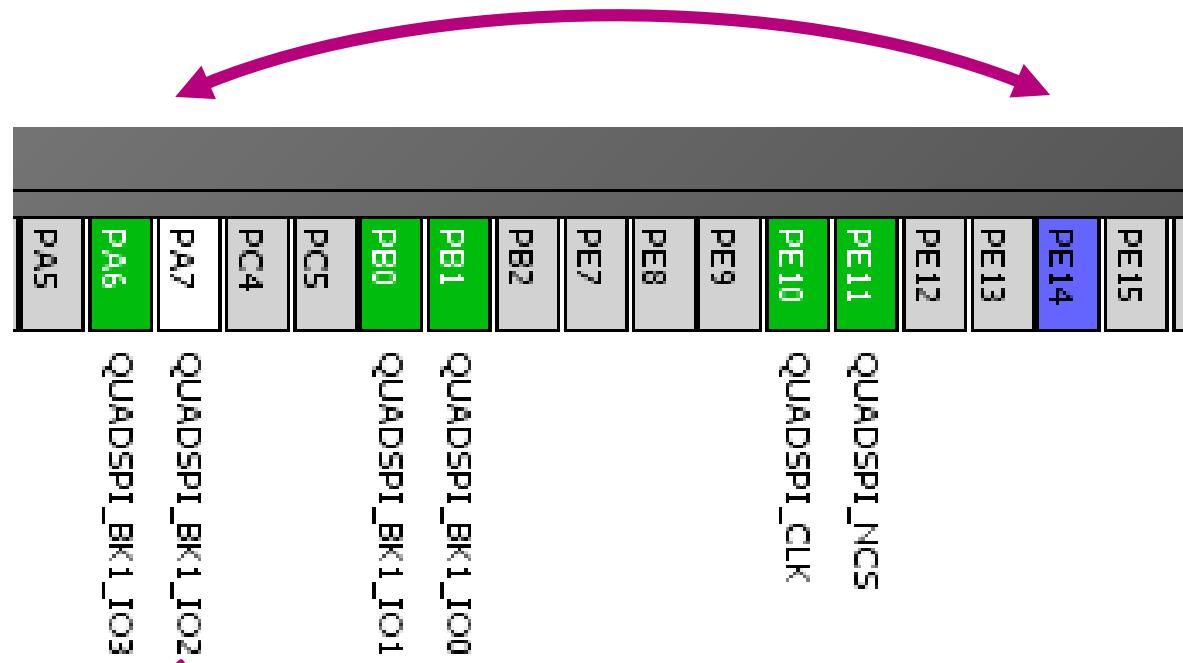
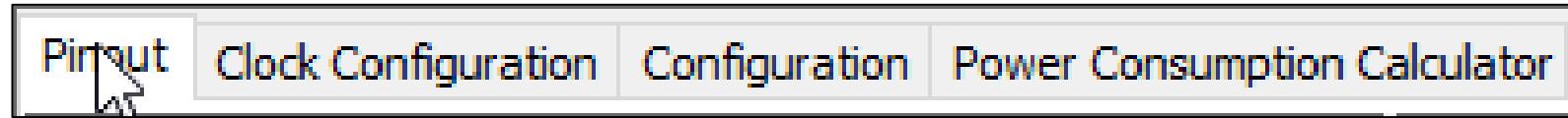
PE12
PE13

Quad SPI Flash Memory



1 QUADSPI interface pins remapping

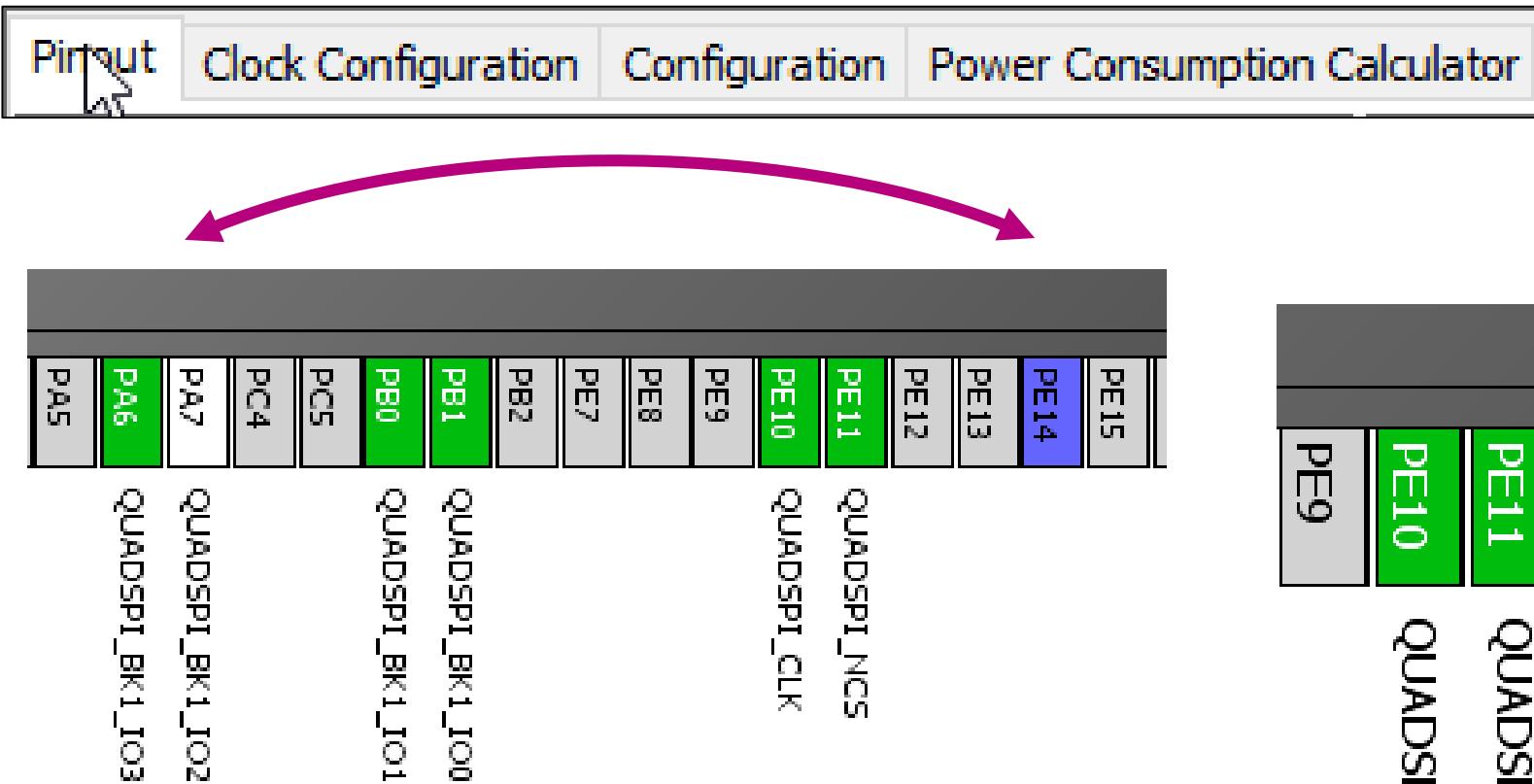
19



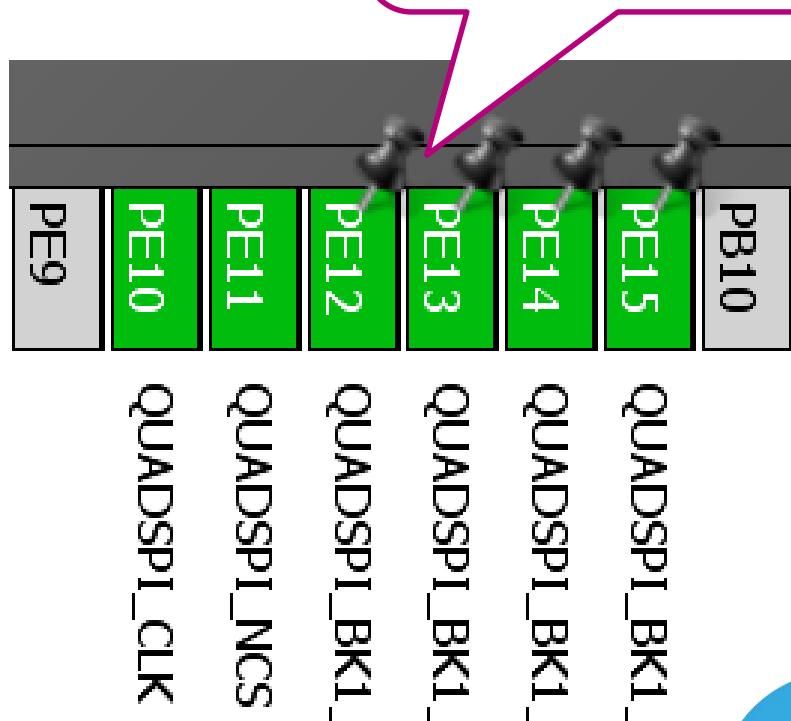
PA7 → PE14
PA6 → PE15
PB0 → PE13
PB1 → PE12

1 QUADSPI interface pins remapping

20

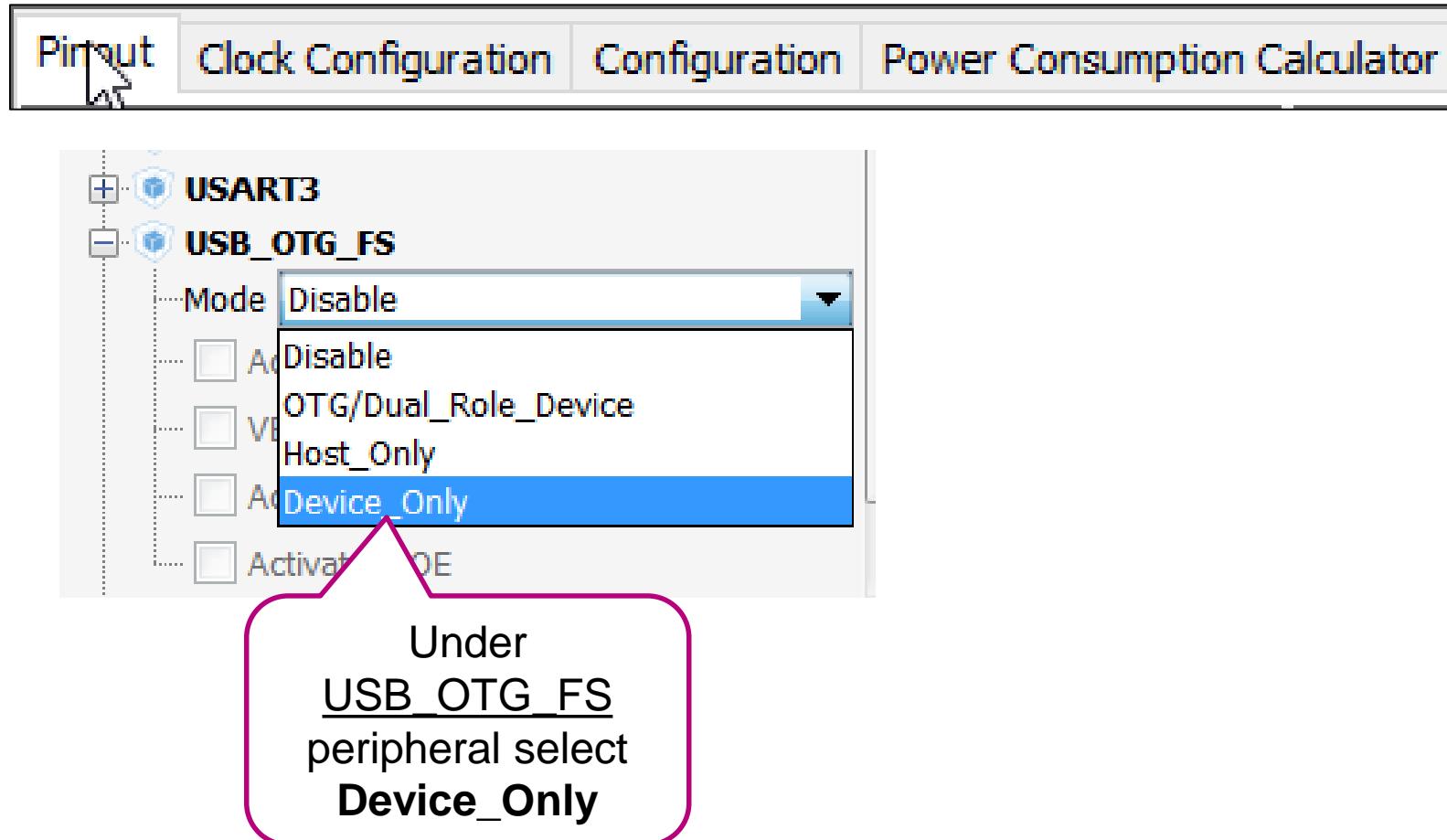


Desired configuration



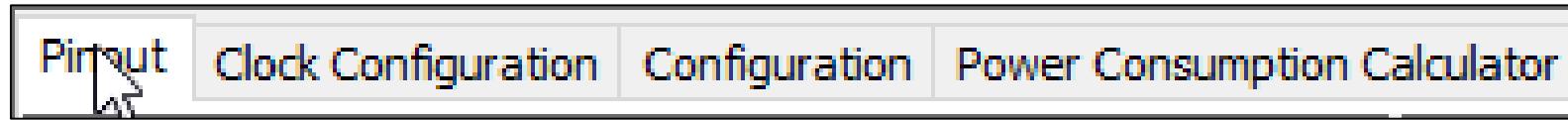
1 Configure USB_OTG_FS interface

21

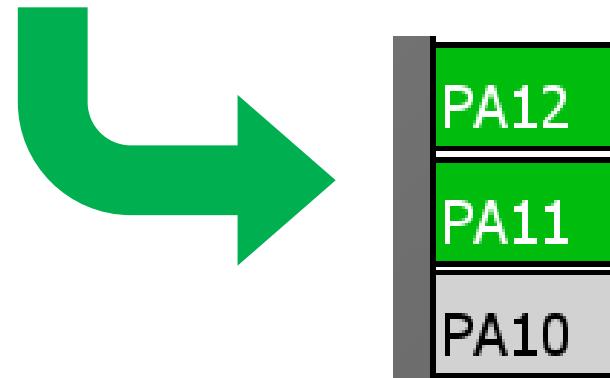


1 Configure USB_OTG_FS interface

22

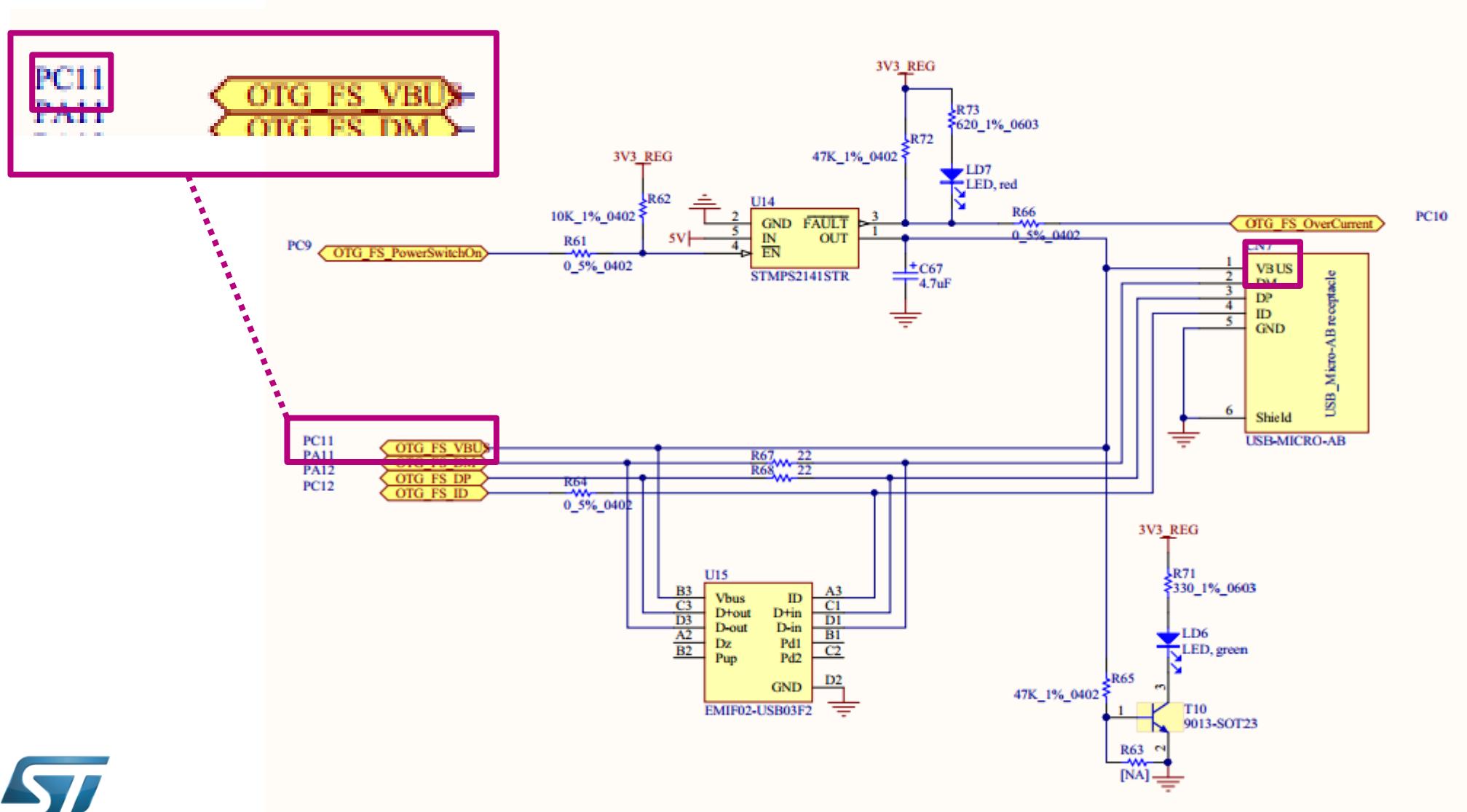


The corresponding pins are assigned and configured automatically!



1 Configure USB_OTG_FS interface

23



1 Configure USB_OTG_FS interface

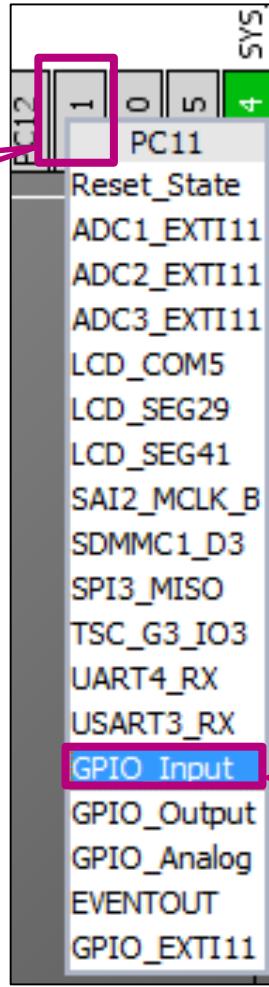
24



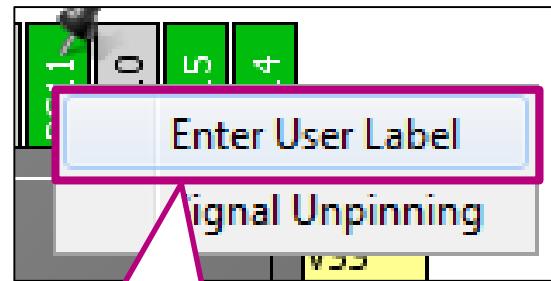
PC11



1

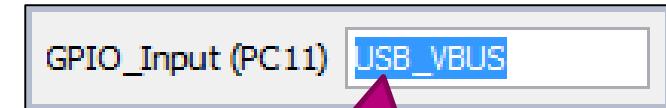


2

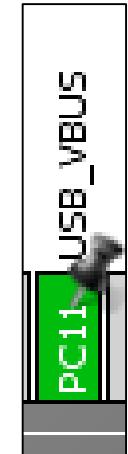
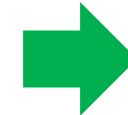


Select Enter User Label

3



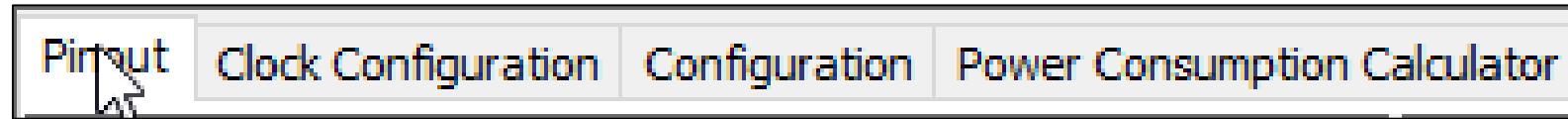
Enter USB_VBUS



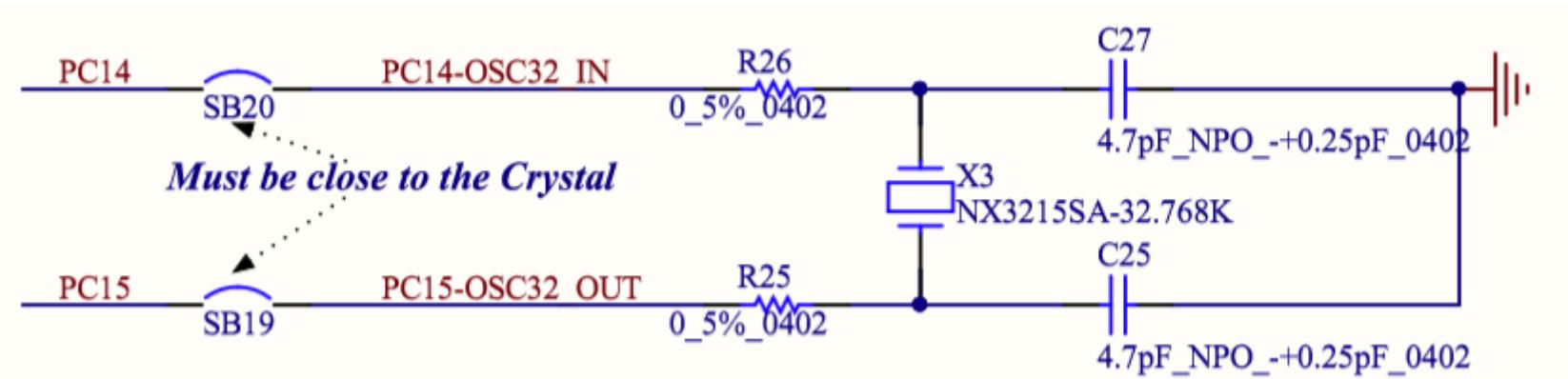
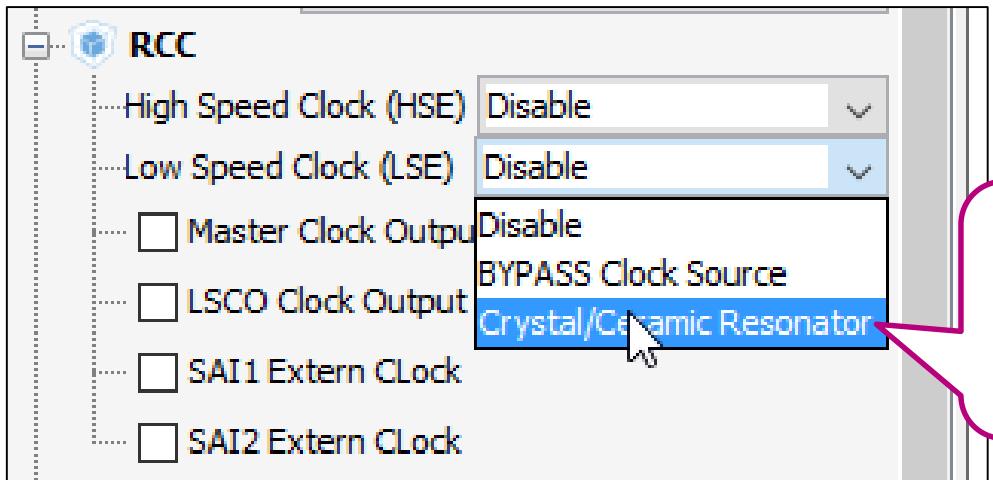
GPIO_Input

Make sure „Show user Label“ is checked





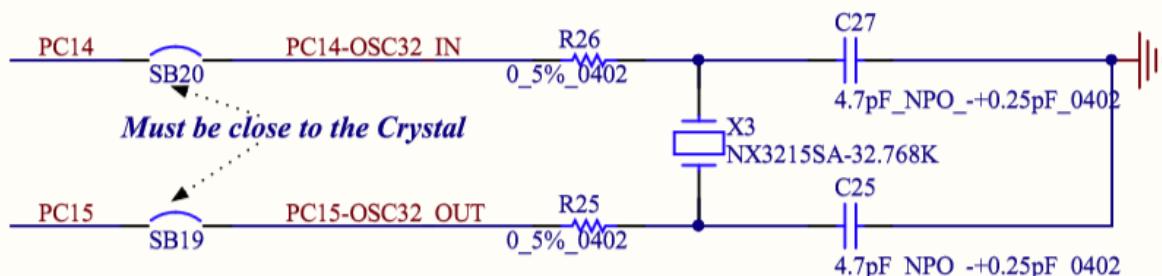
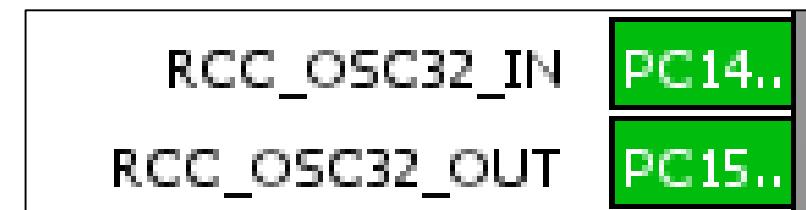
Under RCC Low Speed Clock (LSE)
select **Crystal/Ceramic Resonator**



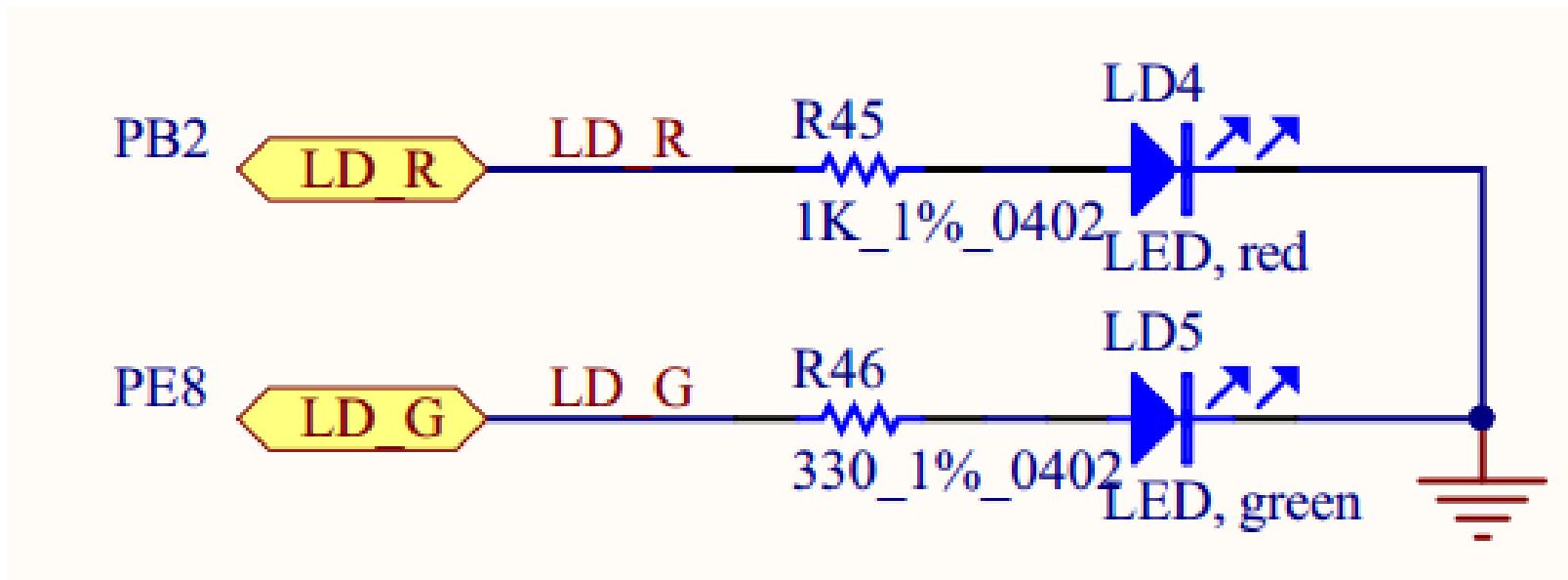
Configure LSE oscillator

Pinsout Clock Configuration Configuration Power Consumption Calculator

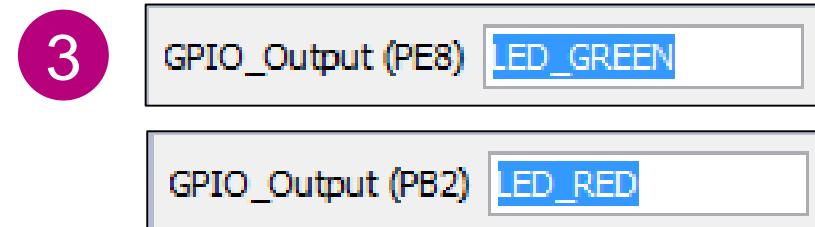
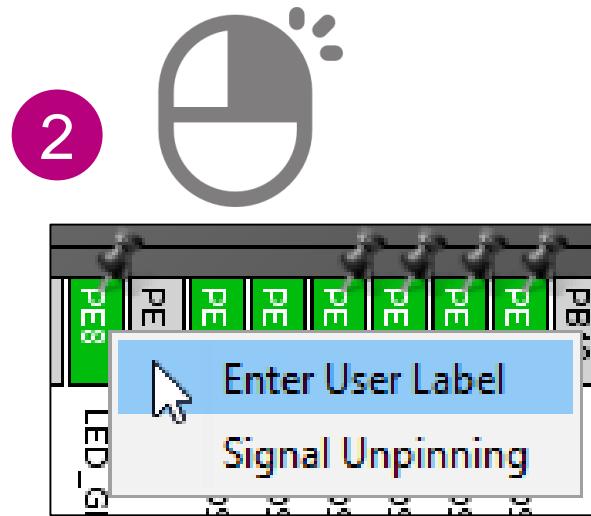
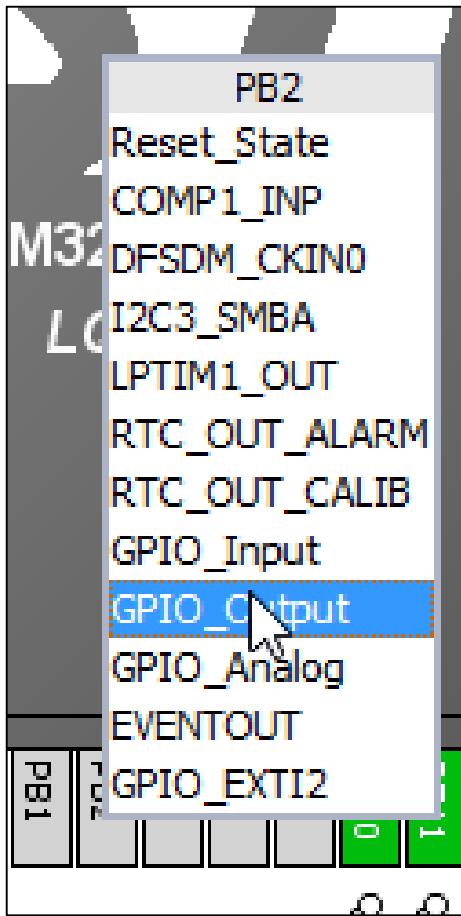
The screenshot shows the STM32Cube software interface. In the top navigation bar, 'Clock Configuration' is selected. On the left, under the 'RCC' tree, the 'Low Speed Clock (LSE)' section is expanded. A dropdown menu for 'Source' is open, with 'Crystal/Ceramic Resonator' highlighted. Other options in the menu include 'Disable' and 'BYPASS Clock Source'. Below the dropdown, there are checkboxes for 'Master Clock Output', 'LSCO Clock Output', 'SAI1 Extern Clock', and 'SAI2 Extern Clock', all of which are unchecked.



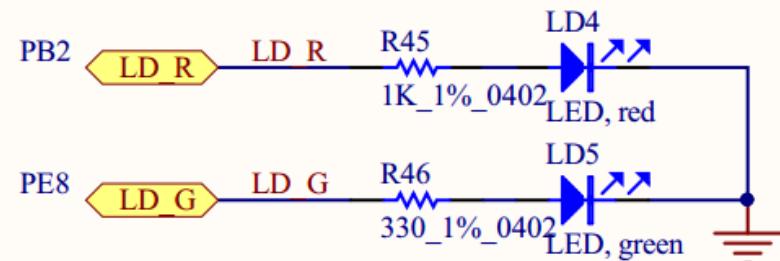
The corresponding pins are assigned and configured automatically!



Configure LED pins



LED_GREEN for PE8
LED_RED for PB2



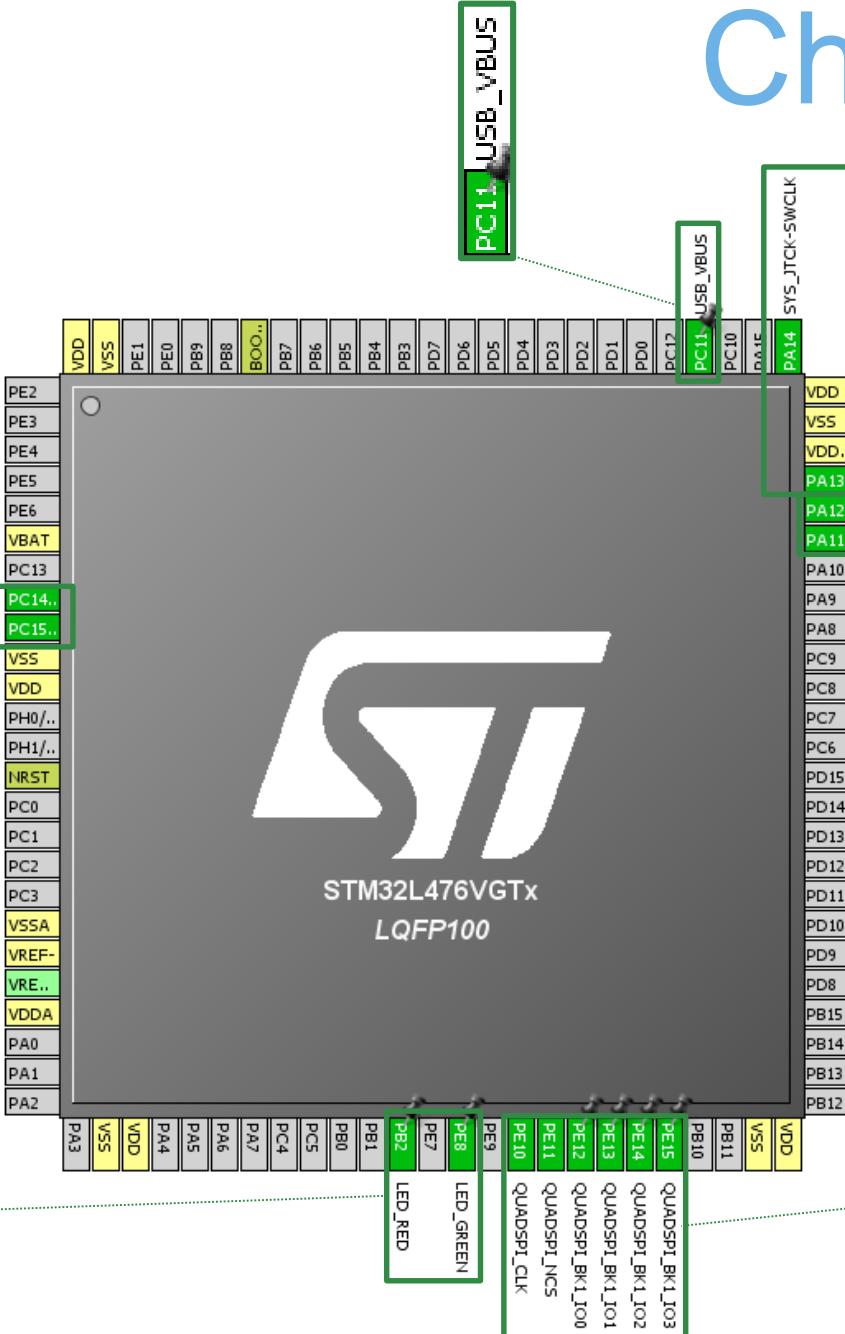
1



RCC_OSC32_IN PC1..
RCC_OSC32_OUT PC1..

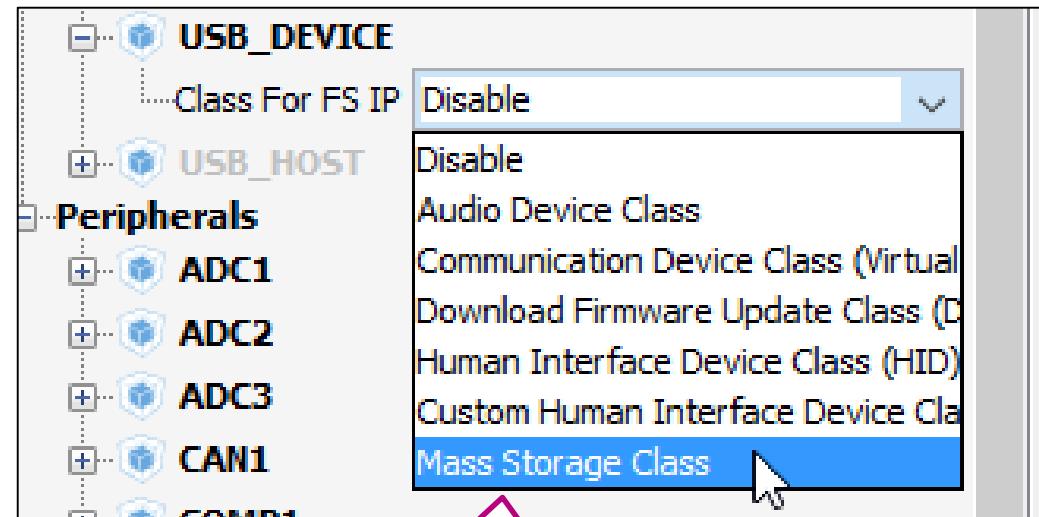
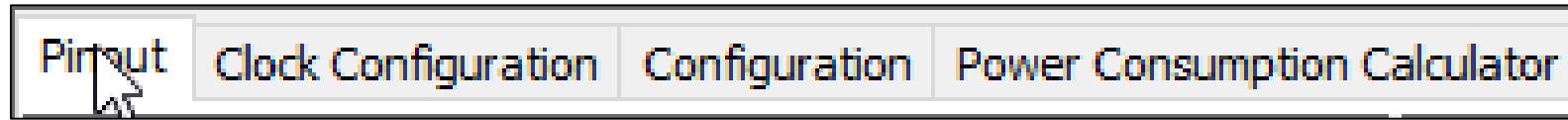
RCC_OSC32_IN PC14..
RCC_OSC32_OUT PC15..

PB8 LED_GREEN
PE7
PB2 LED_RED



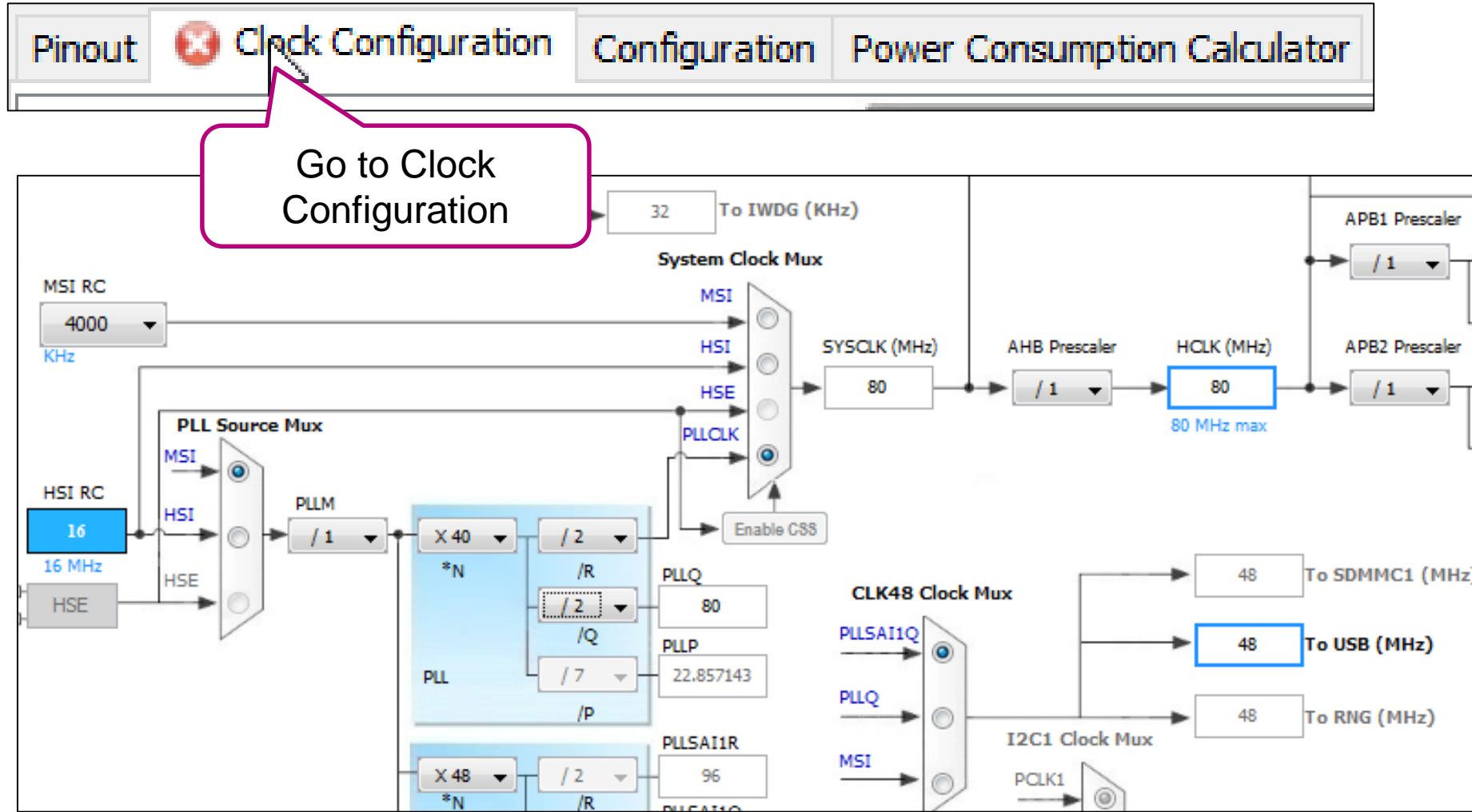
Checkpoint #1

29

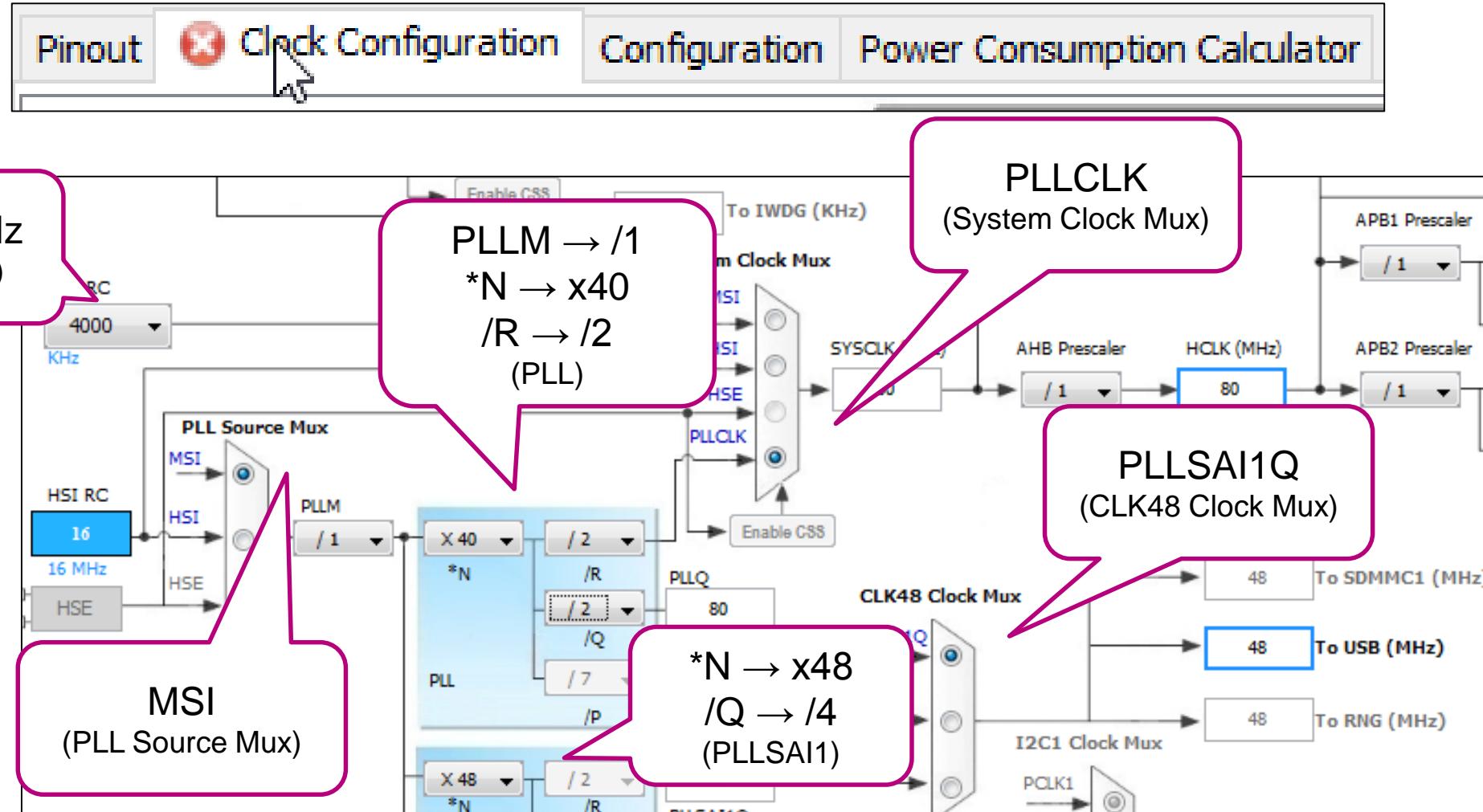


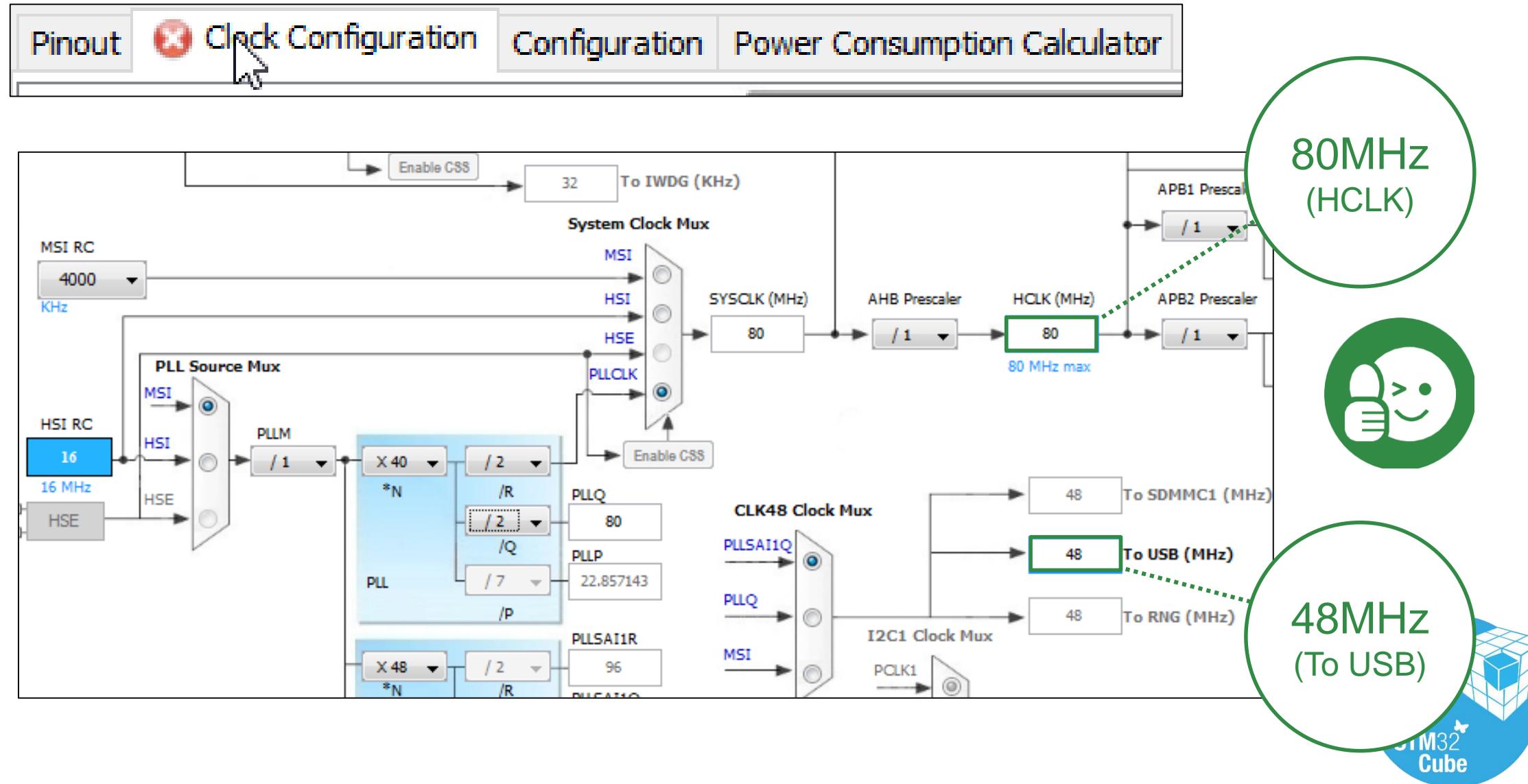
Under USB_DEVICE
MiddleWare select **Mass
Storage Class**

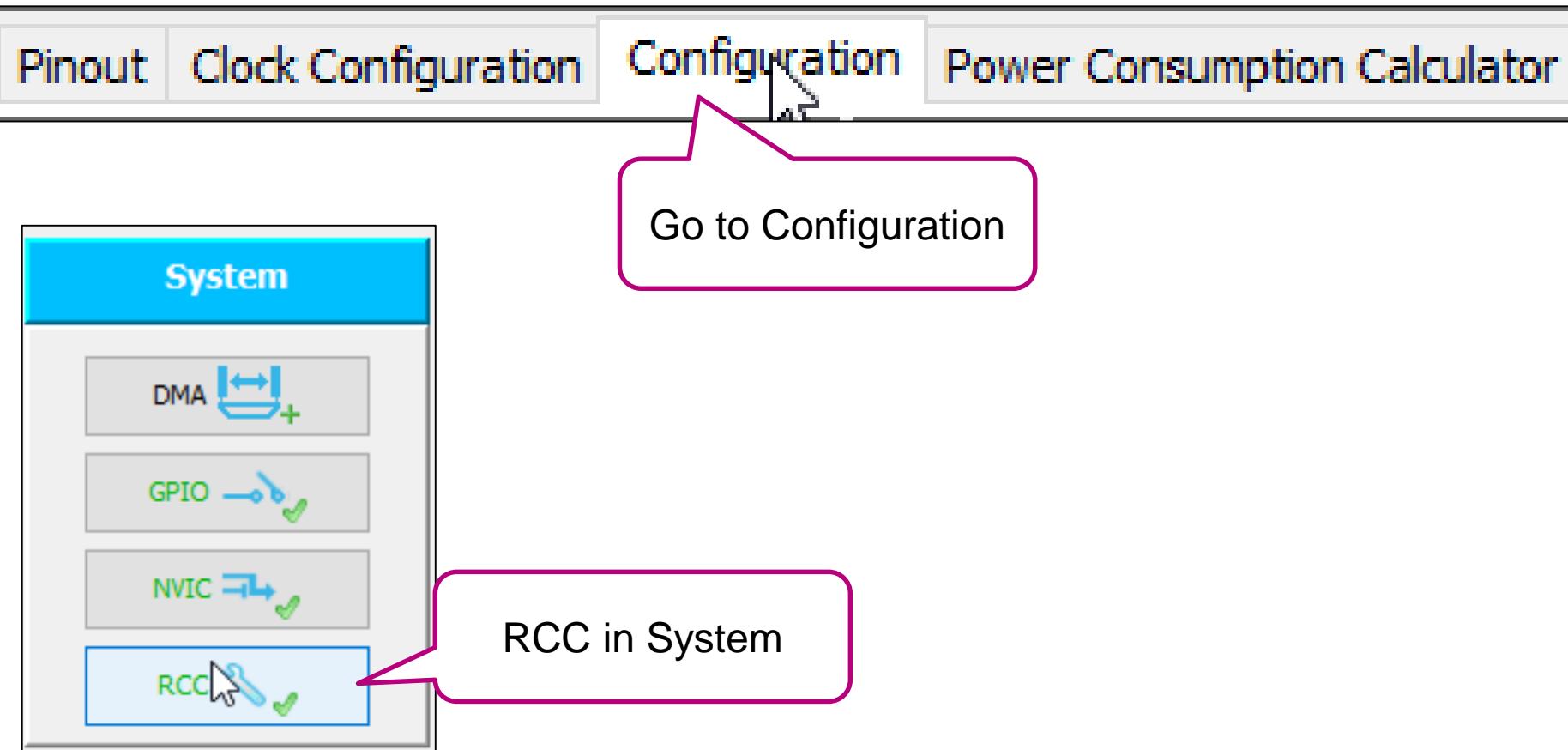
System Clock Configuration



System Clock Configuration



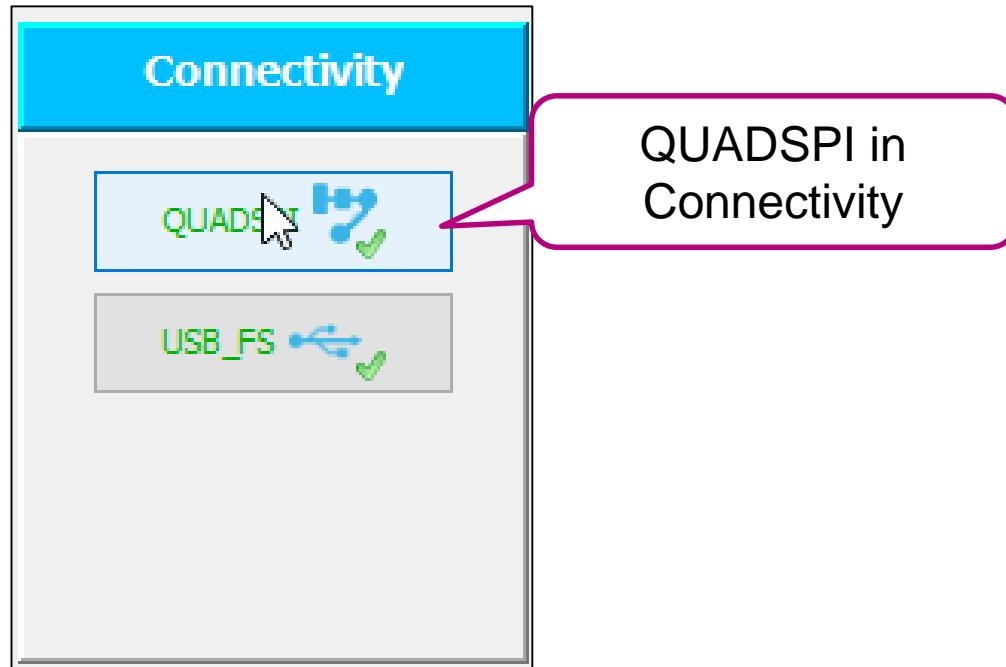




The screenshot illustrates the RCC Configuration check process within the STM32CubeMX software. It shows a navigation bar at the top with tabs: Pinout, Clock Configuration, Configuration (which is selected), and Power Consumption Calculator. Below the navigation bar is a system configuration panel titled "System". This panel contains four main sections: DMA, GPIO, NVIC, and RCC. The RCC section is highlighted with a cursor icon. A large green arrow points from the System panel to the detailed RCC Configuration dialog box. The dialog box is titled "RCC Configuration" and contains tabs for Parameter Settings, User Constants, NVIC Settings, and GPIO Settings, all of which are checked. The main area of the dialog box is titled "Configure the below parameters:" and lists several parameters under three categories: System Parameters, RCC Parameters, and Power Parameters.

Category	Parameter	Value
System Parameters	VDD voltage (V)	3.3 V
	Instruction Cache	Enabled
	Prefetch Buffer	Disabled
	Data Cache	Enabled
RCC Parameters	Flash Latency(WS)	4 WS (5 CPU cycle)
	HSI Calibration Value	16
	MSI Calibration Value	0
Power Parameters	MSI Auto Calibration	Enabled
	Power Regulator Voltage Scale	Power Regulator Voltage Scale 1

A callout bubble with a pink border points to the "MSI Auto Calibration" field, which is set to "Enabled".



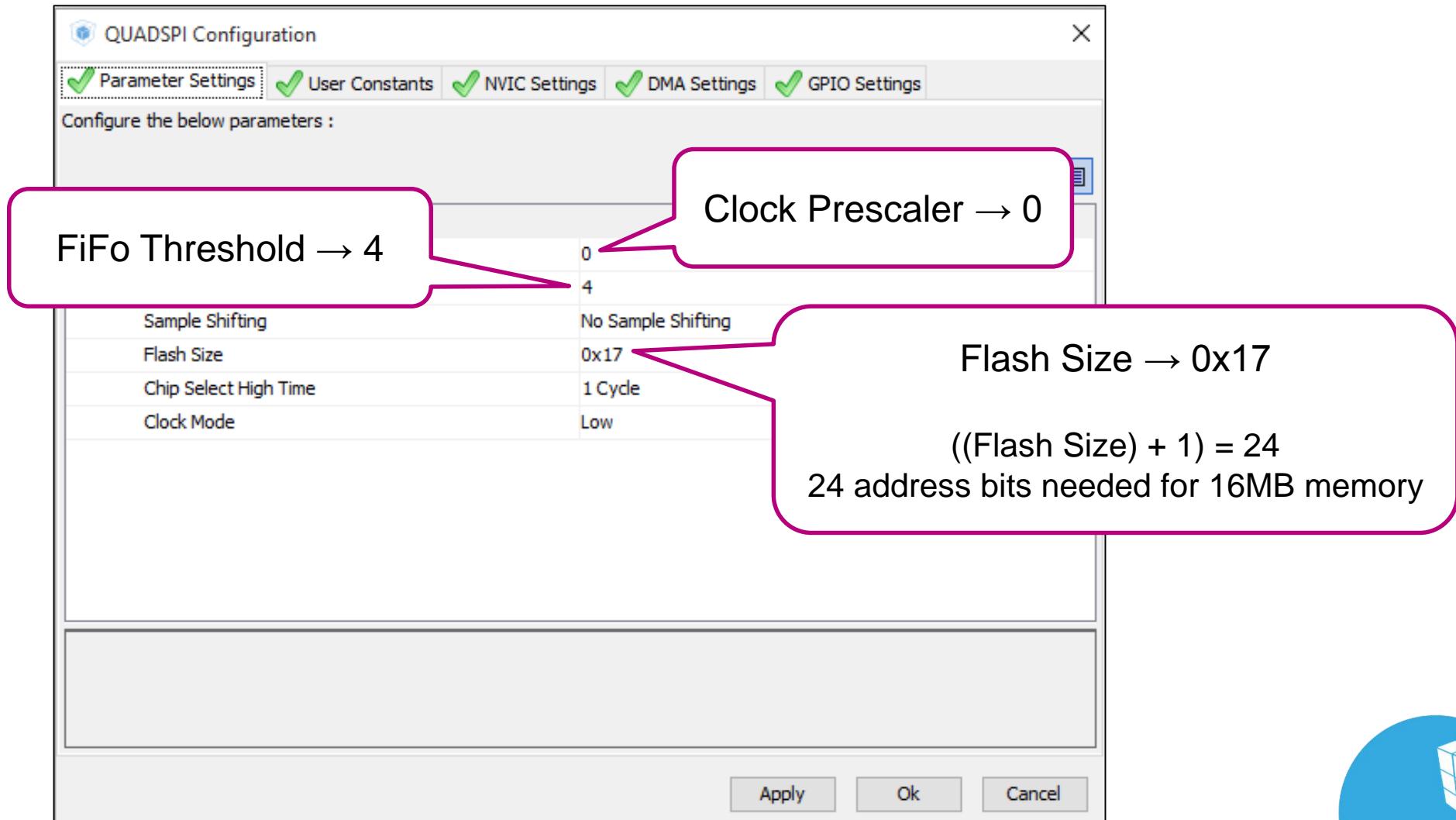
The screenshot illustrates the configuration flow for the QUADSPI interface in STM32CubeMX. It starts with a main menu bar at the top featuring tabs for Pinout, Clock Configuration, Configuration, and Power Consumption Calculator. A cursor is hovering over the Configuration tab. Below the menu is a central workspace divided into two main sections: Connectivity and Configuration.

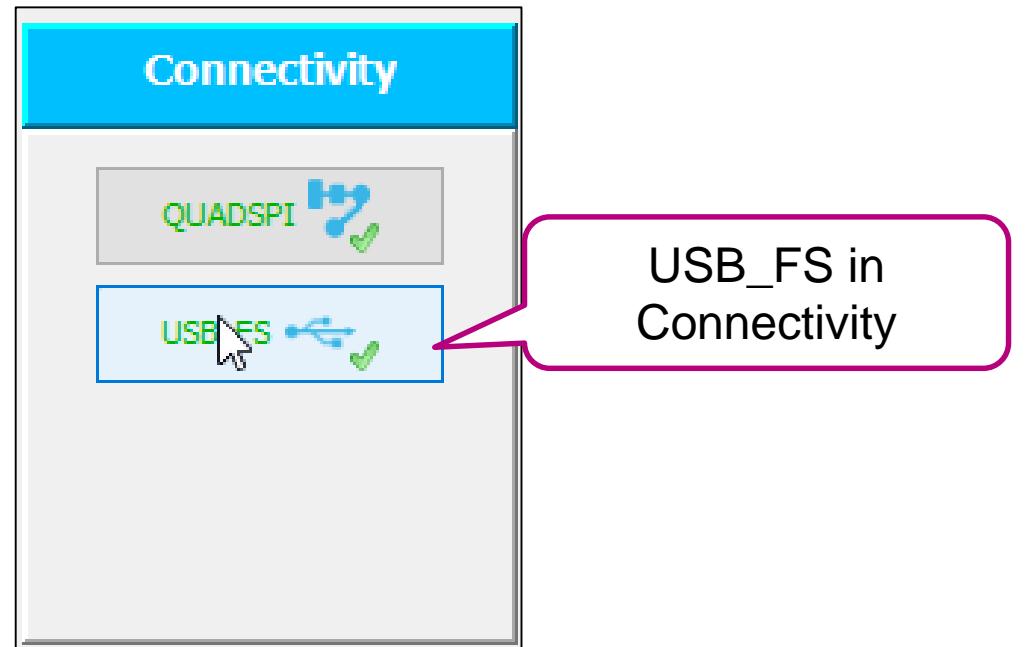
Connectivity: This section contains two items: QUADSPI (represented by a blue icon with a green checkmark) and USB_FS (represented by a grey icon with a green checkmark). A large green arrow points from the Connectivity section towards the Configuration window.

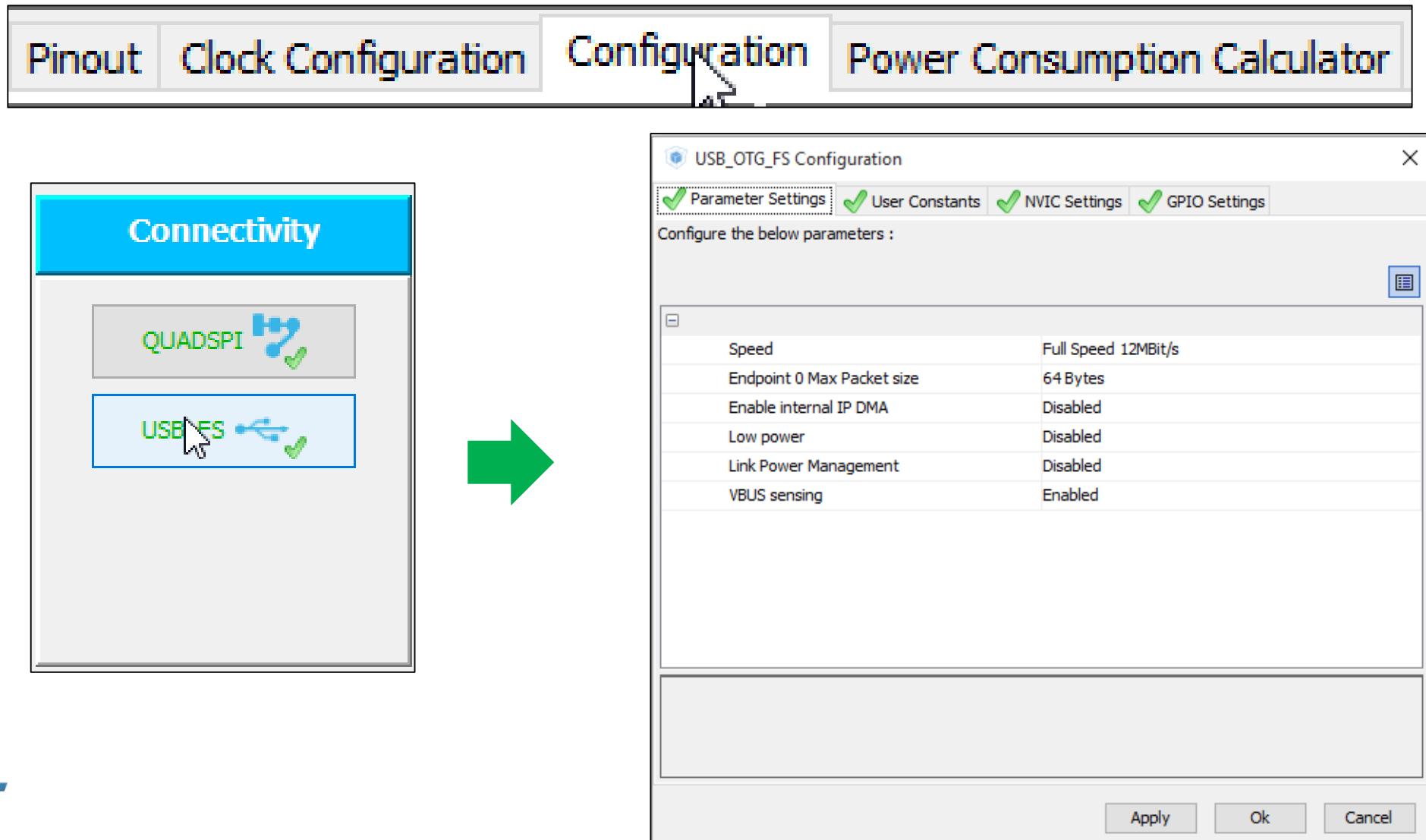
Configuration: This window is titled "QUADSPI Configuration". It includes a header with five checked status indicators: Parameter Settings, User Constants, NVIC Settings, DMA Settings, and GPIO Settings. Below the header is a sub-header: "Configure the below parameters :". The main content area displays a table of general parameters:

General Parameters	
Clock Prescaler	255
Fifo Threshold	1
Sample Shifting	No Sample Shifting
Flash Size	1
Chip Select High Time	1 Cycle
Clock Mode	Low

At the bottom of the Configuration window are three buttons: Apply, Ok, and Cancel.

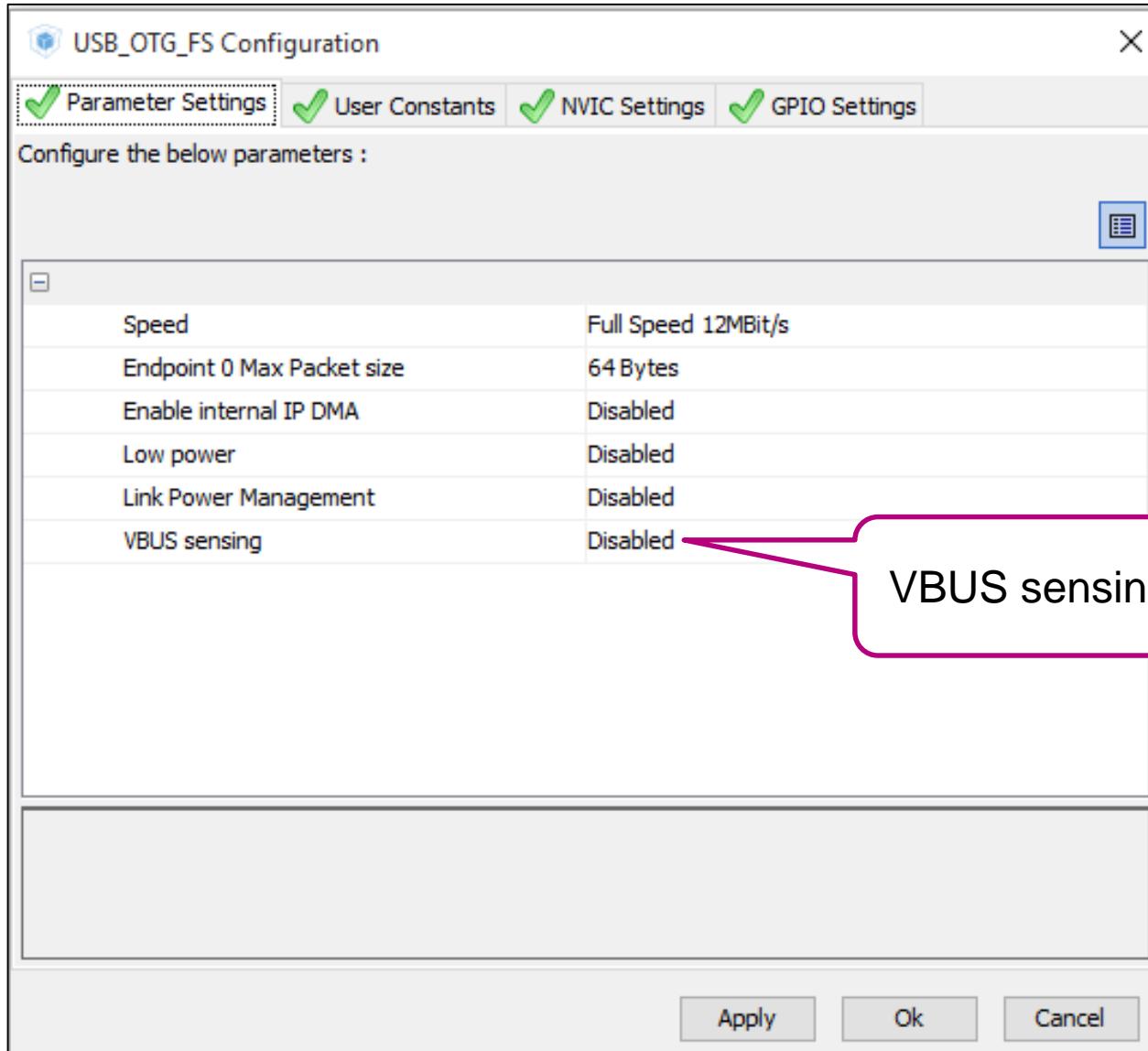


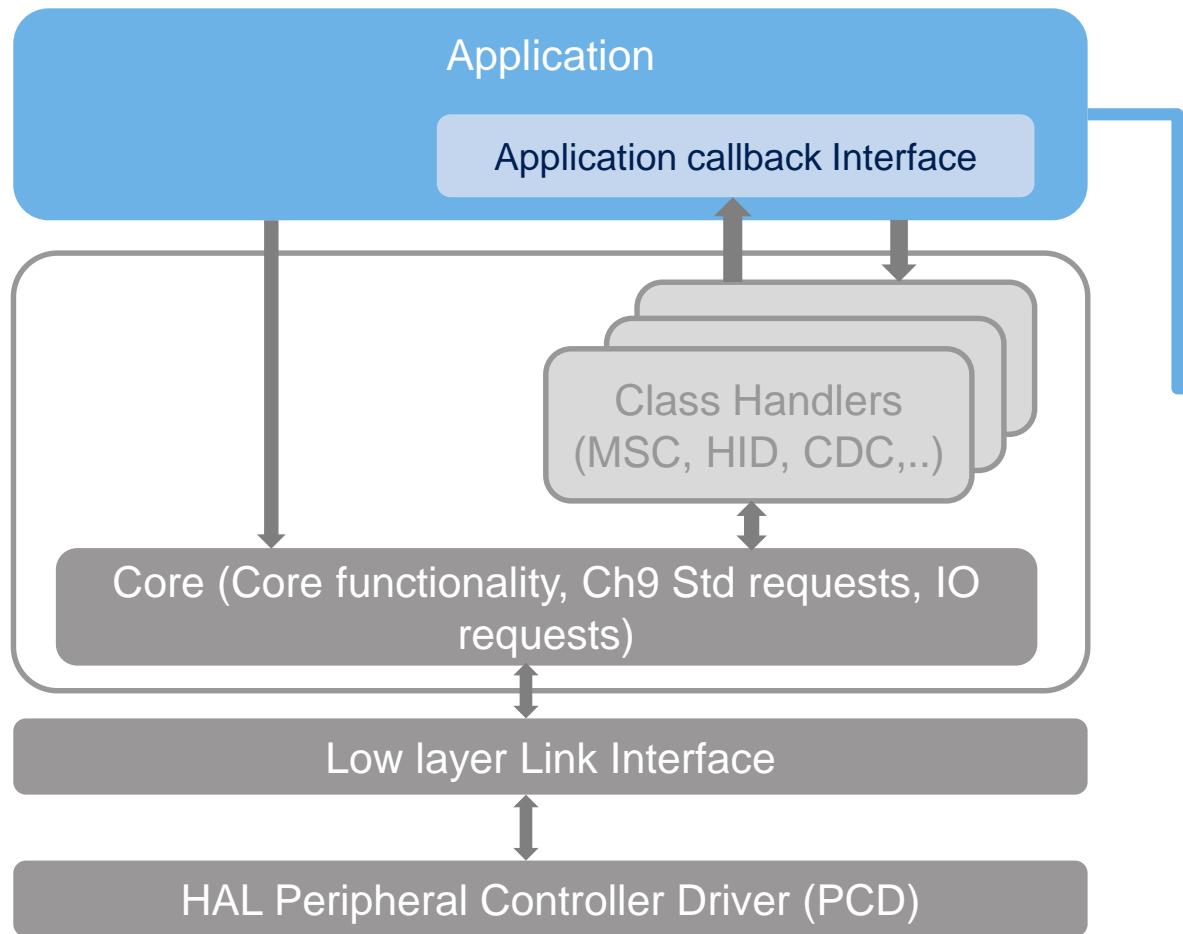




1 USB_OTG_FS configuration

41





- **USB wrapper for SCSI commands**



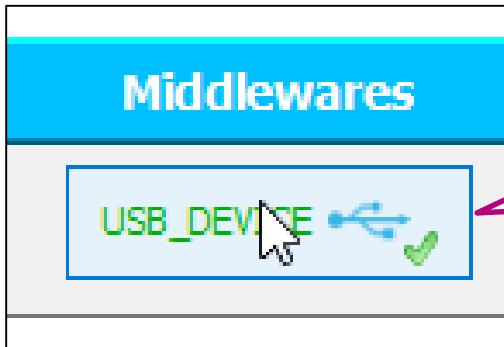
SCSI = Small Computer System Interface

SCSI command subset for USB

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

1 USB_DEVICE MiddleWare configuration

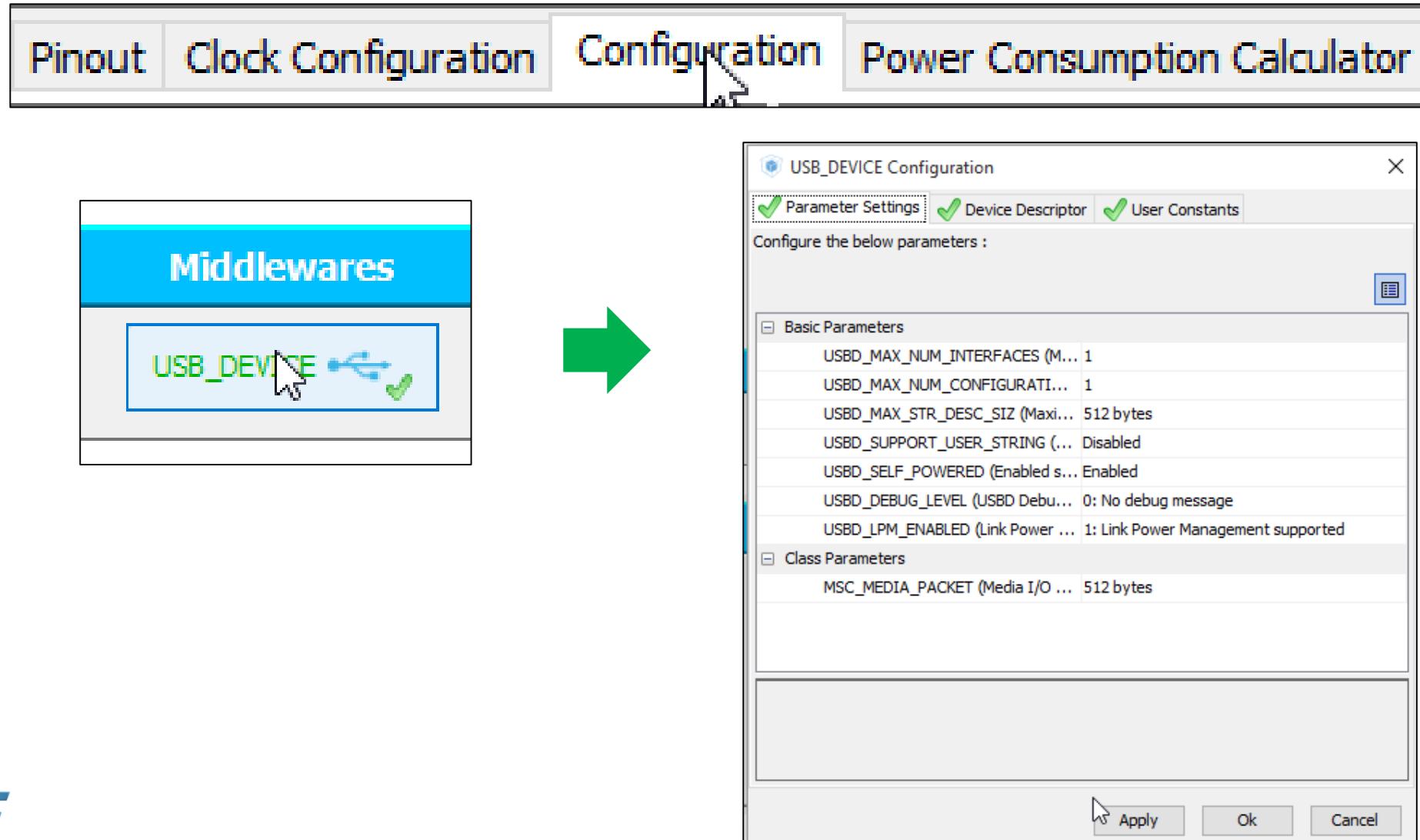
43



USB_DEVICE in
Middlewares

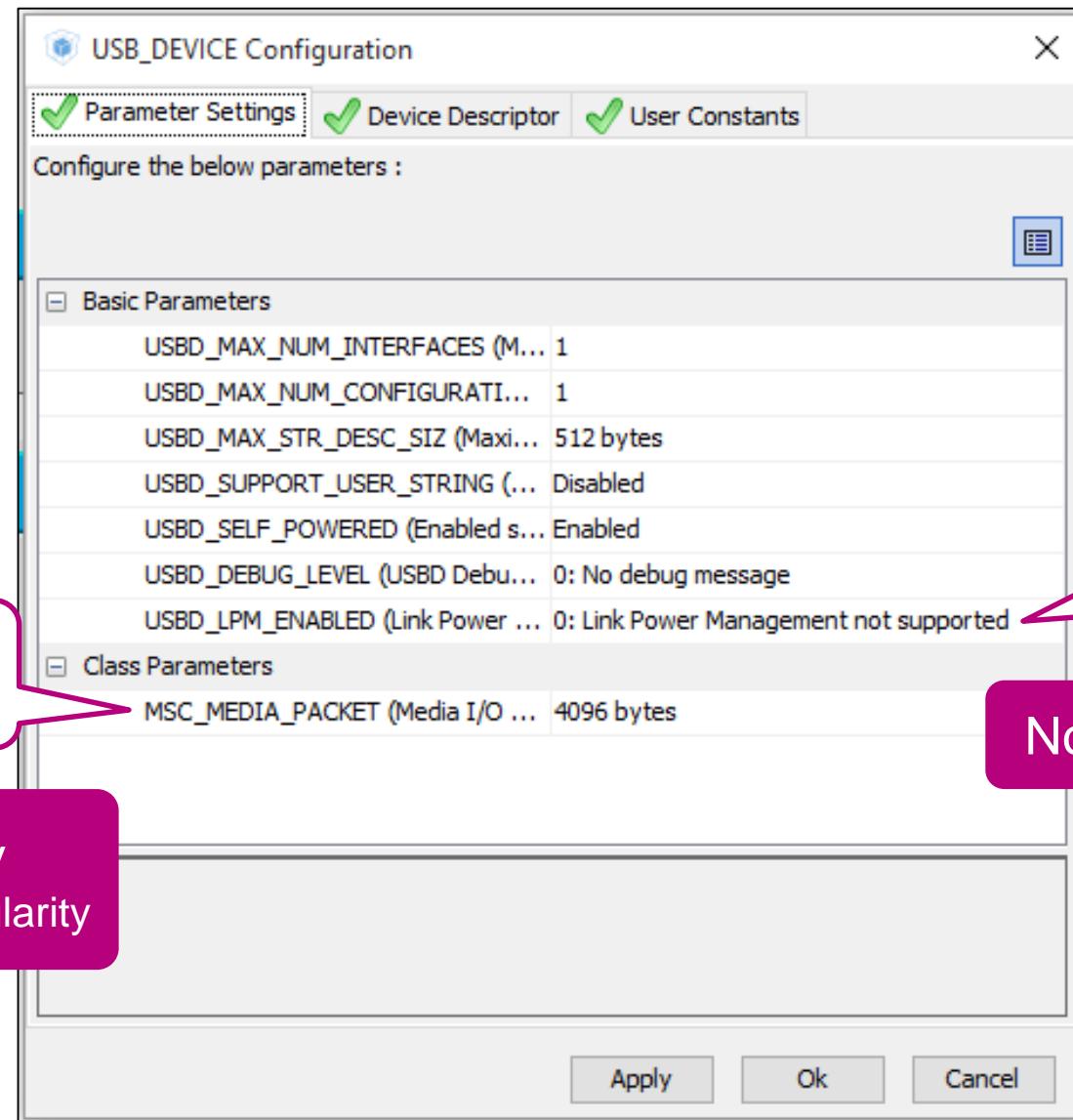
1 USB_DEVICE MiddleWare configuration

44



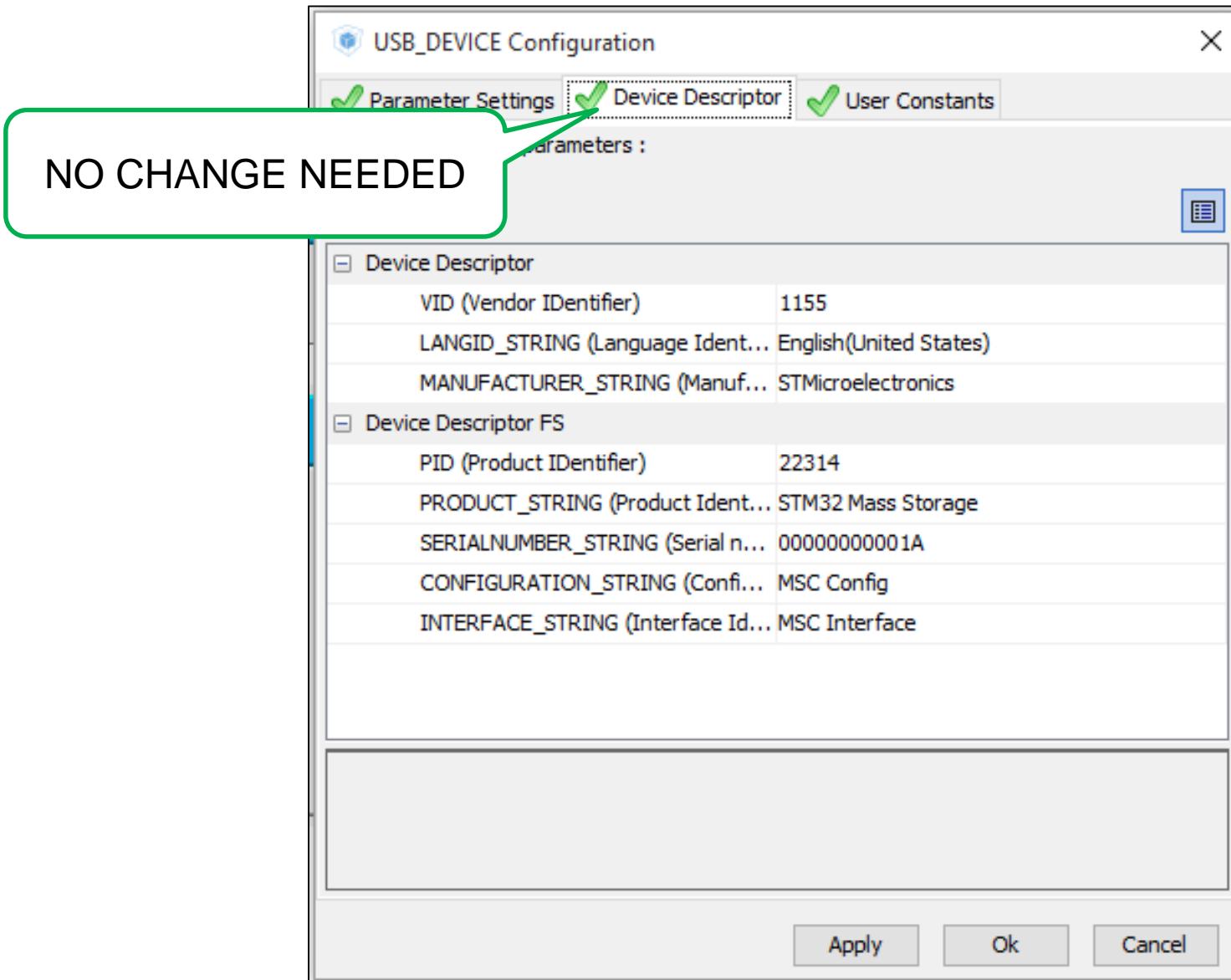
1 USB_DEVICE MiddleWare configuration

45

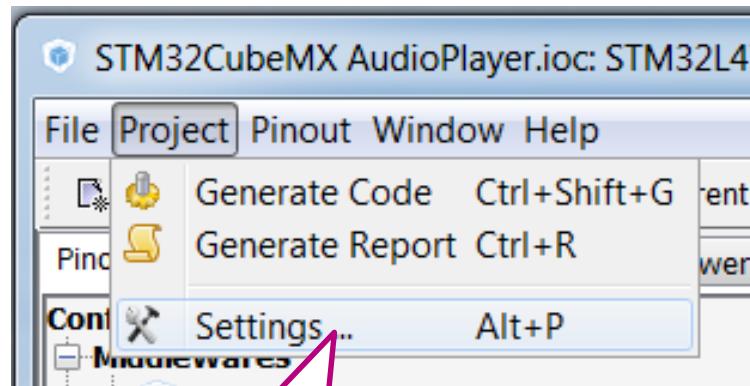


1 USB_DEVICE MiddleWare configuration

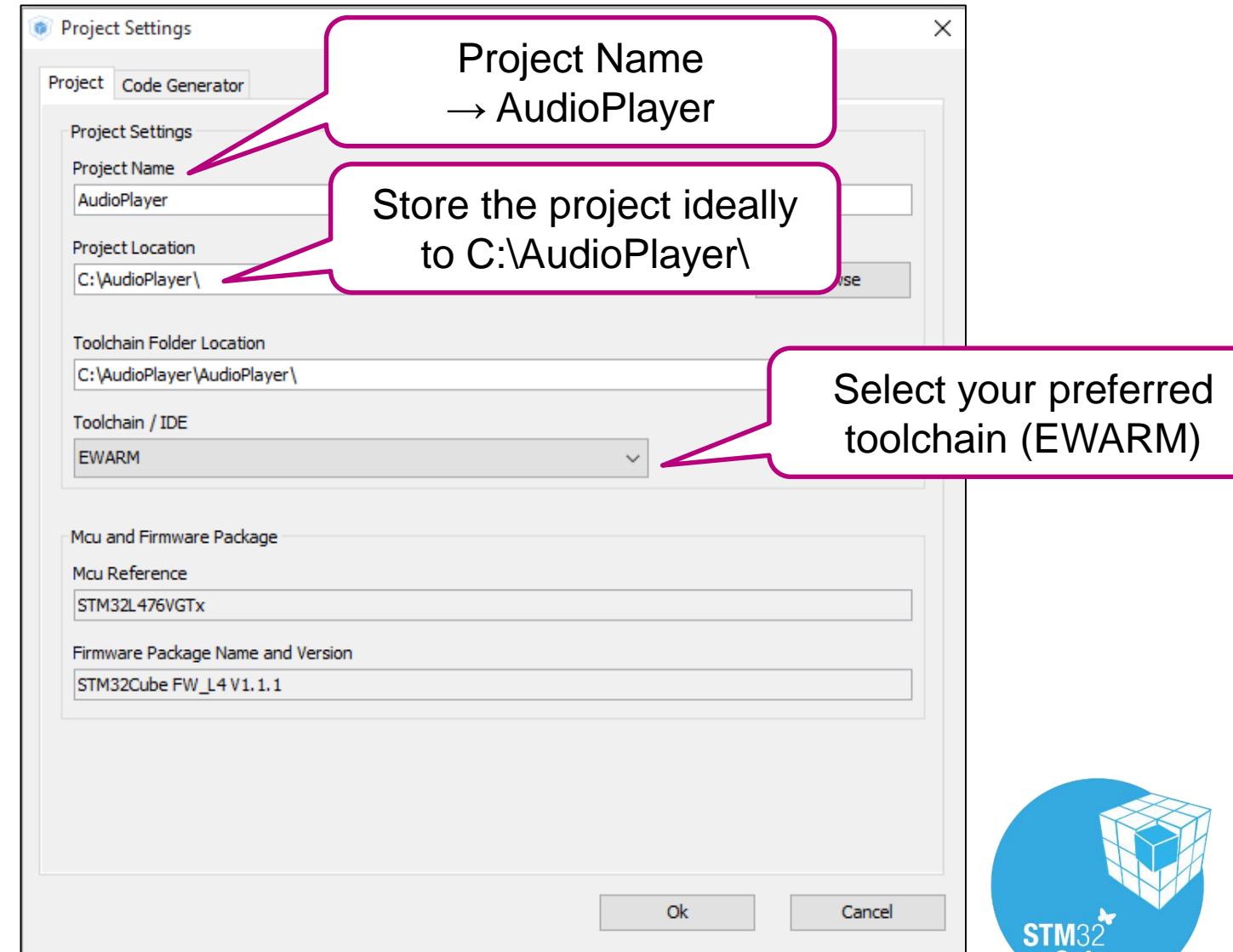
46



Project Settings



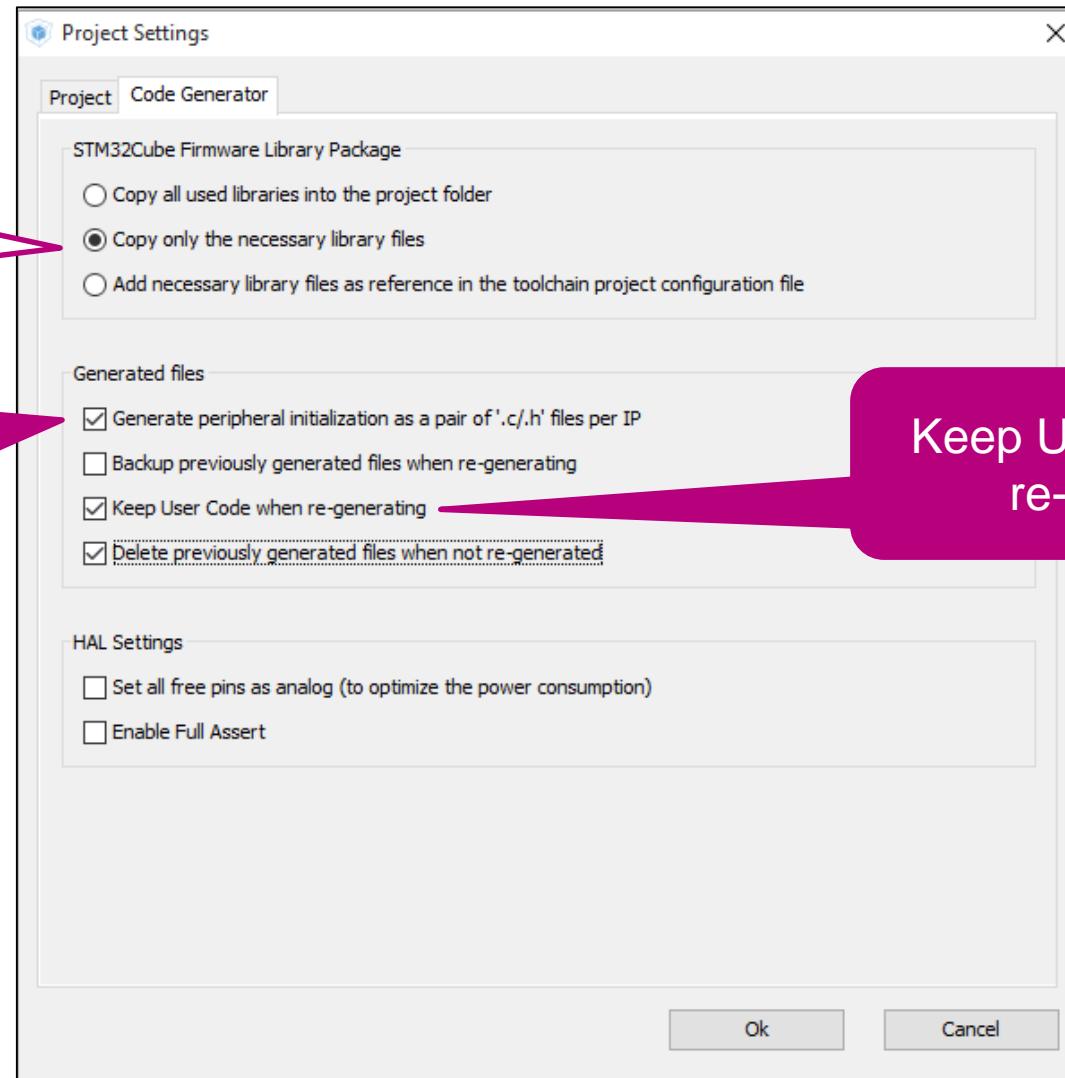
Configure the project
(Alt+P)



Project Name
→ AudioPlayer

Store the project ideally
to C:\AudioPlayer\

Select your preferred
toolchain (EWARM)



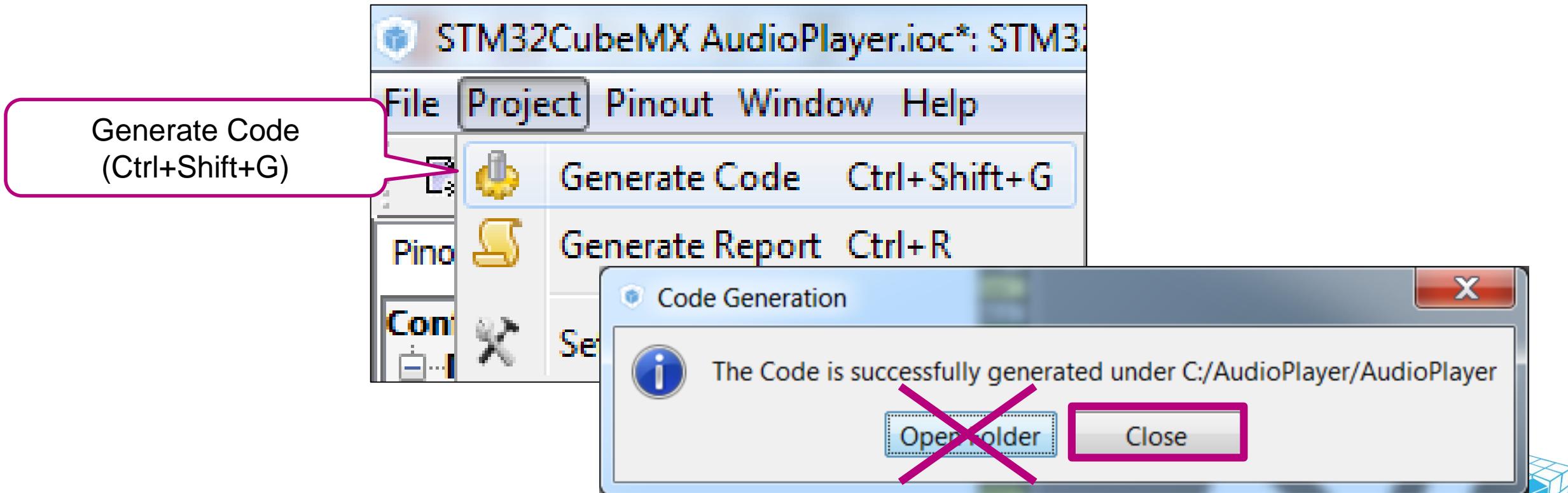
Copy only the necessary
library files

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

Keep User Code when
re-generating

Save and generate the project

Do not open the project yet



- 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 contain the main needed source code the user has to add to have the USB MSC device running together with QSPI FLASH storage



- quadspi.c

QSPI_Init(void)
QSPI_DelInit(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

- 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

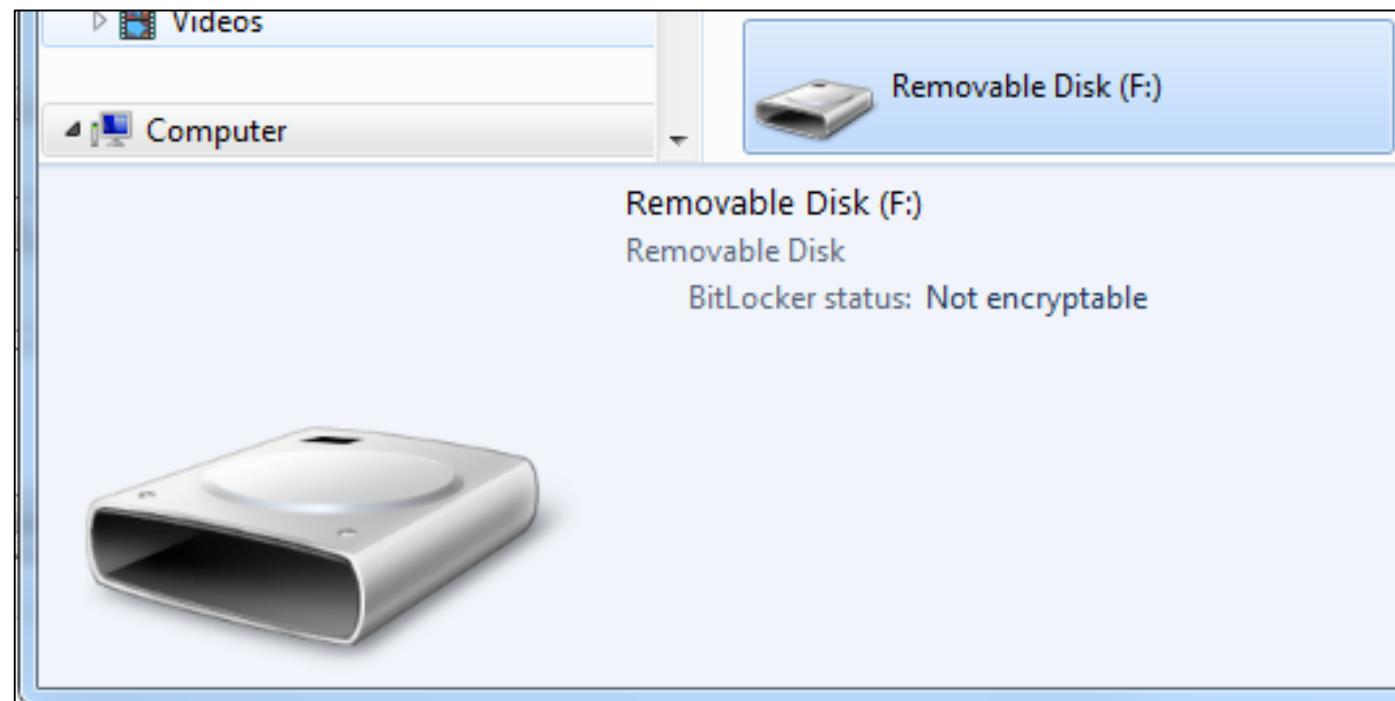
Templates generated by STM32CubeMX, just the bodies
of the functions were implemented.

1. 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

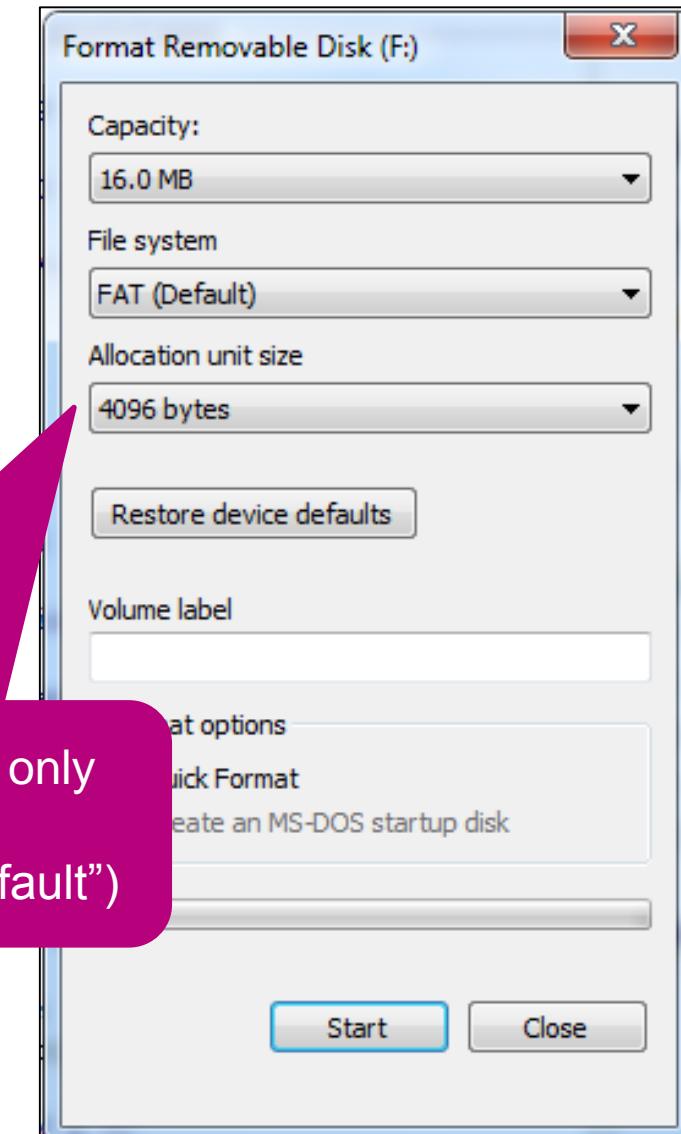
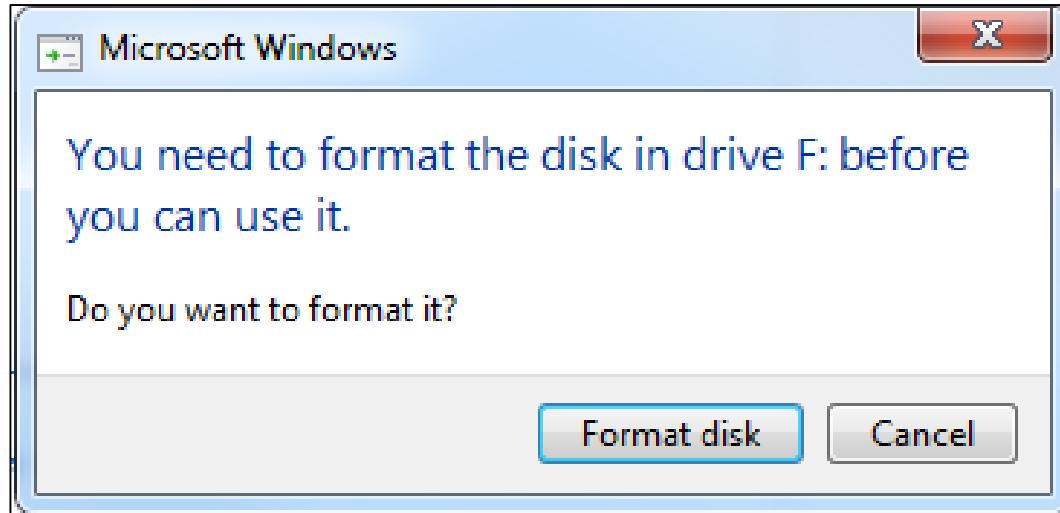
1 Connect the new “FLASH stick”

54

- 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”)

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

RED LED
(LD4)



GREEN LED
(LD5)



ERASING

RED LED
(LD4)



GREEN LED
(LD5)

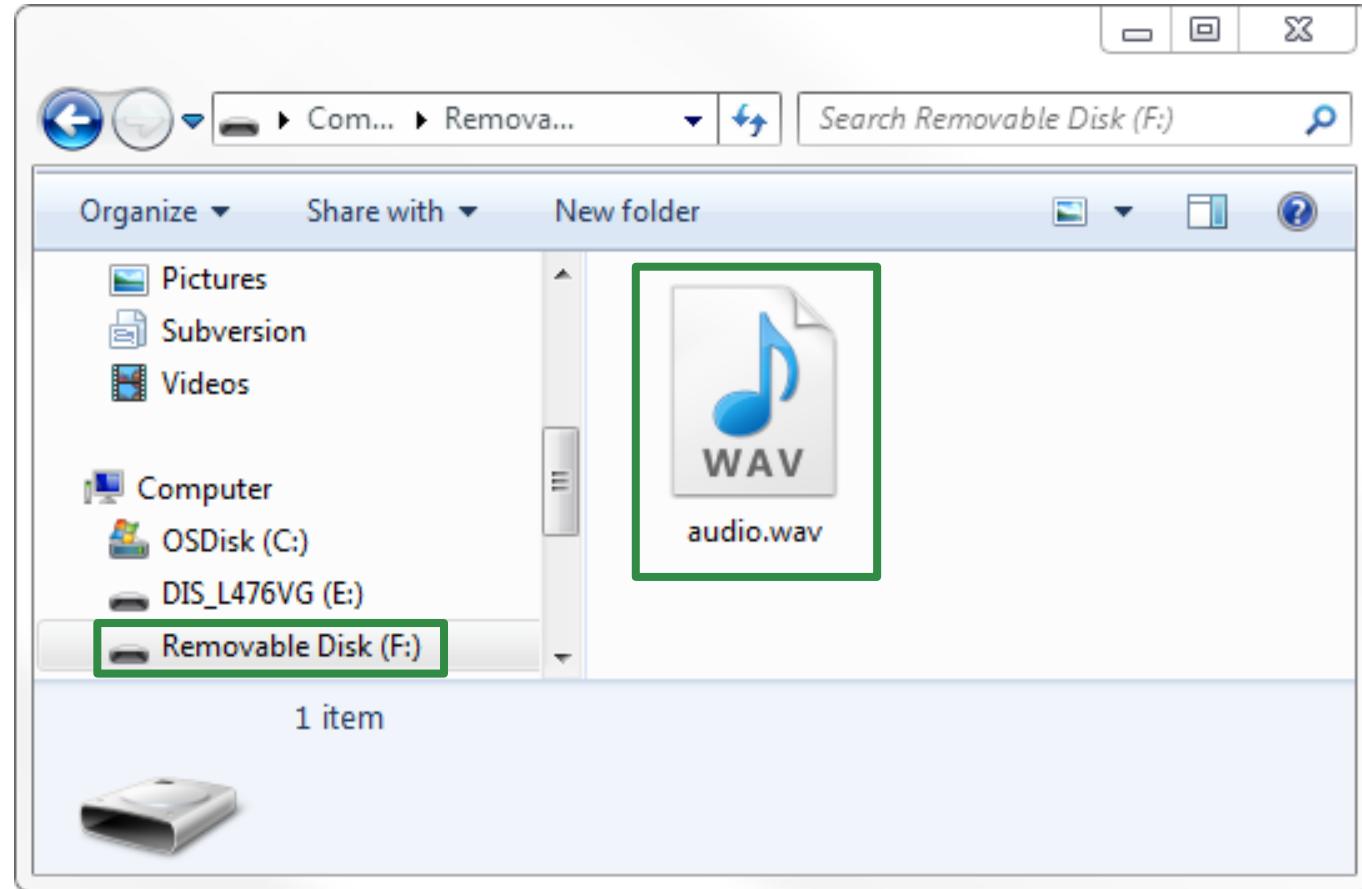


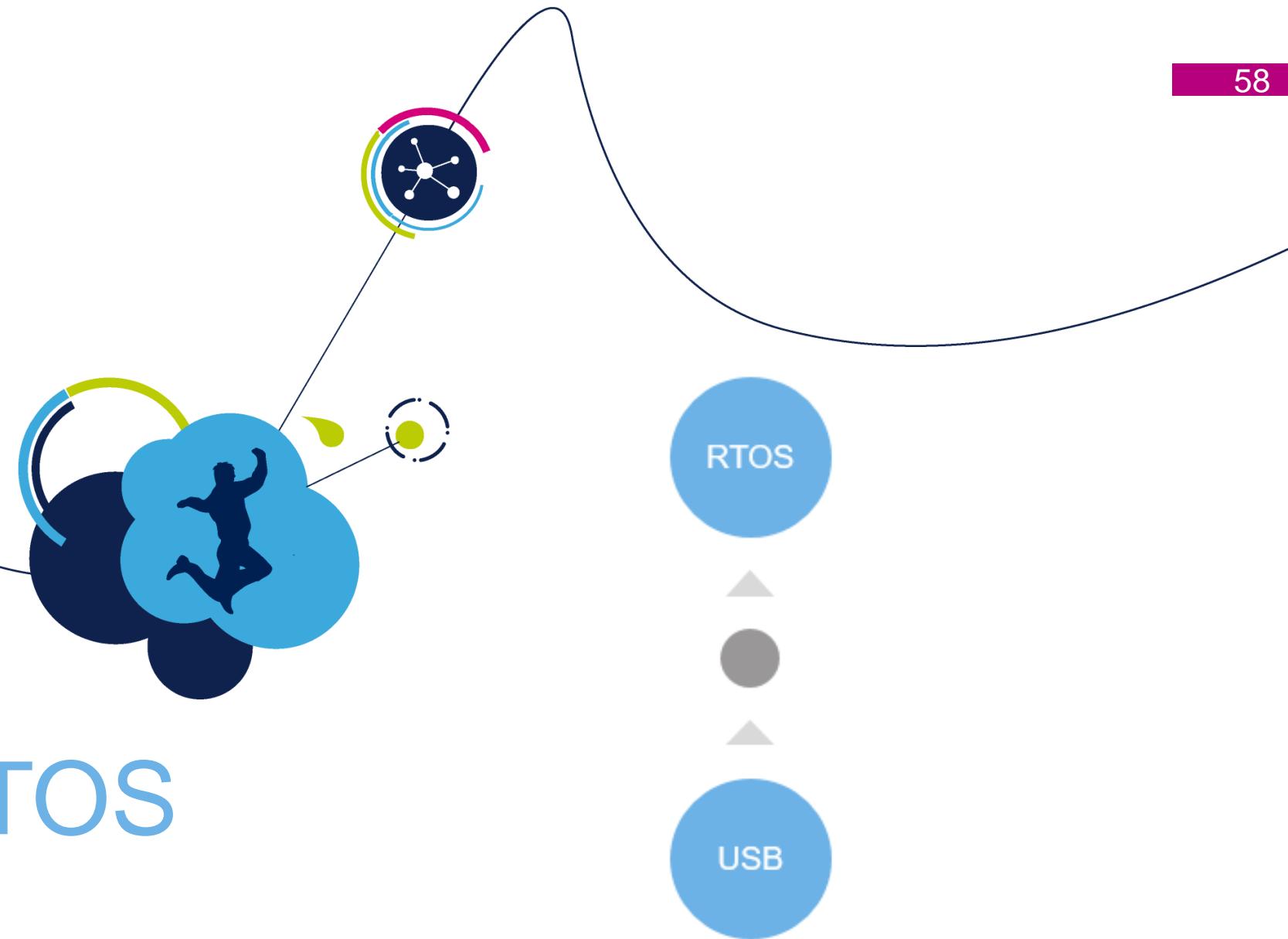
WRITING

1

Checkpoint #3

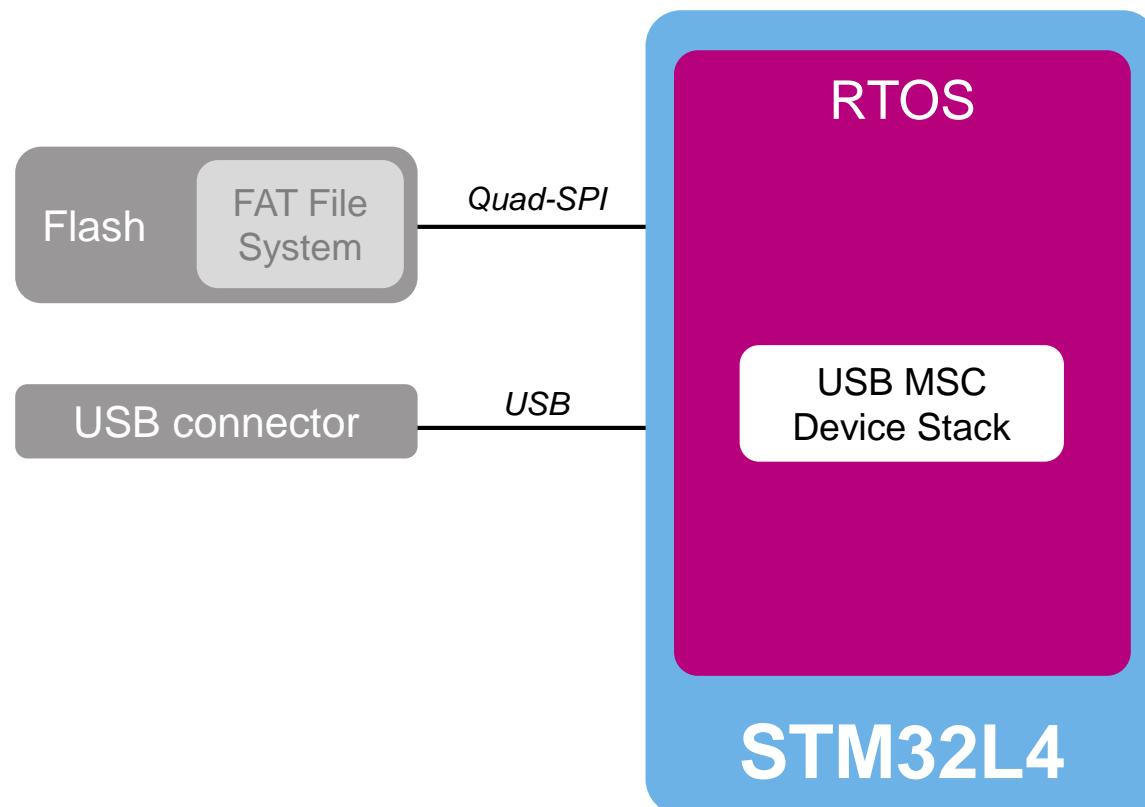
57





STEP2 – RTOS

- We want to have the application running on 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

Conventional app

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

RTOS app

Task 1 (Thread)

```
void main1(void)
{
    while(1)
    {
        ...
    }
}
```

Stack 1

Task 2 (Thread)

```
void main2(void)
{
    while(1)
    {
        ...
    }
}
```

Stack 2

Task 3 (Thread)

```
void main3(void)
{
    while(1)
    {
        ...
    }
}
```

Stack 3

Scheduler (RTOS kernel)
switching based on event, time, priority...

task A

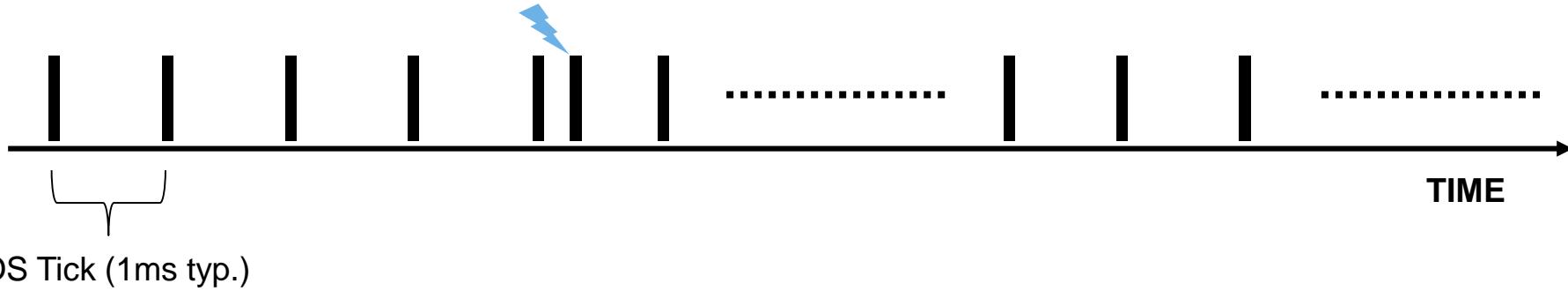
task B

CPU time and scheduling

Thread mode



Handler mode



- 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

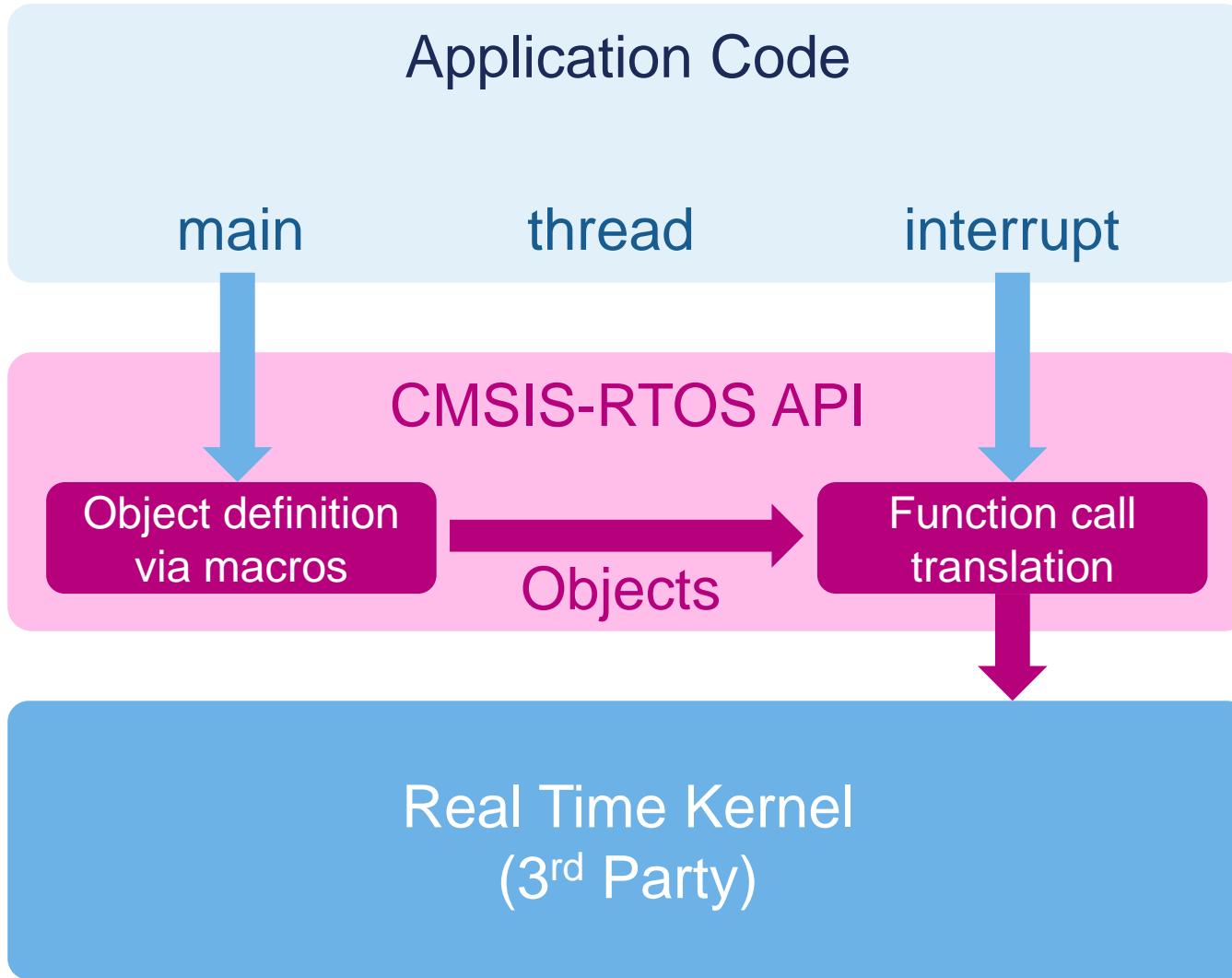
- flag that may be used to signal specific condition to a thread
- can be modified in an ISR or set from other threads.

- 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.



FreeRTOS



- UM1722 – Developing Applications on STM32Cube with RTOS

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

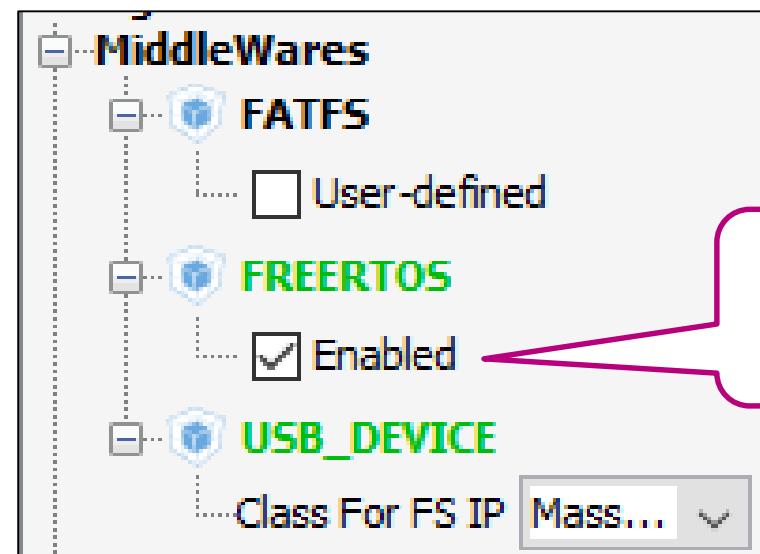
- <http://www.freertos.org>

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

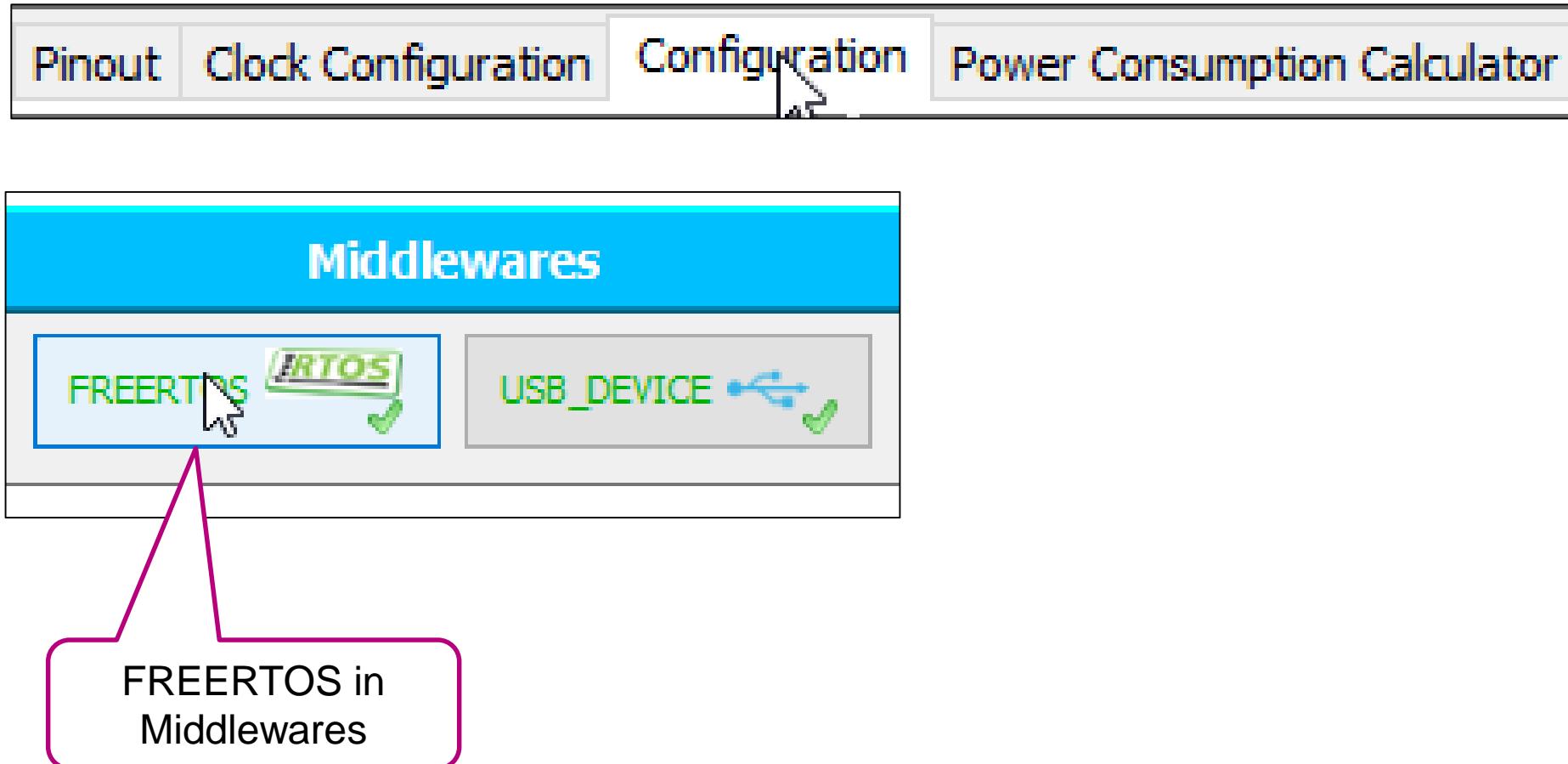
http://shop.freertos.org/FreeRTOS_Tutorial_Book_Generic_Cortex_M3_Edition_p/pdf_cortex-m3_tutorial_book.htm

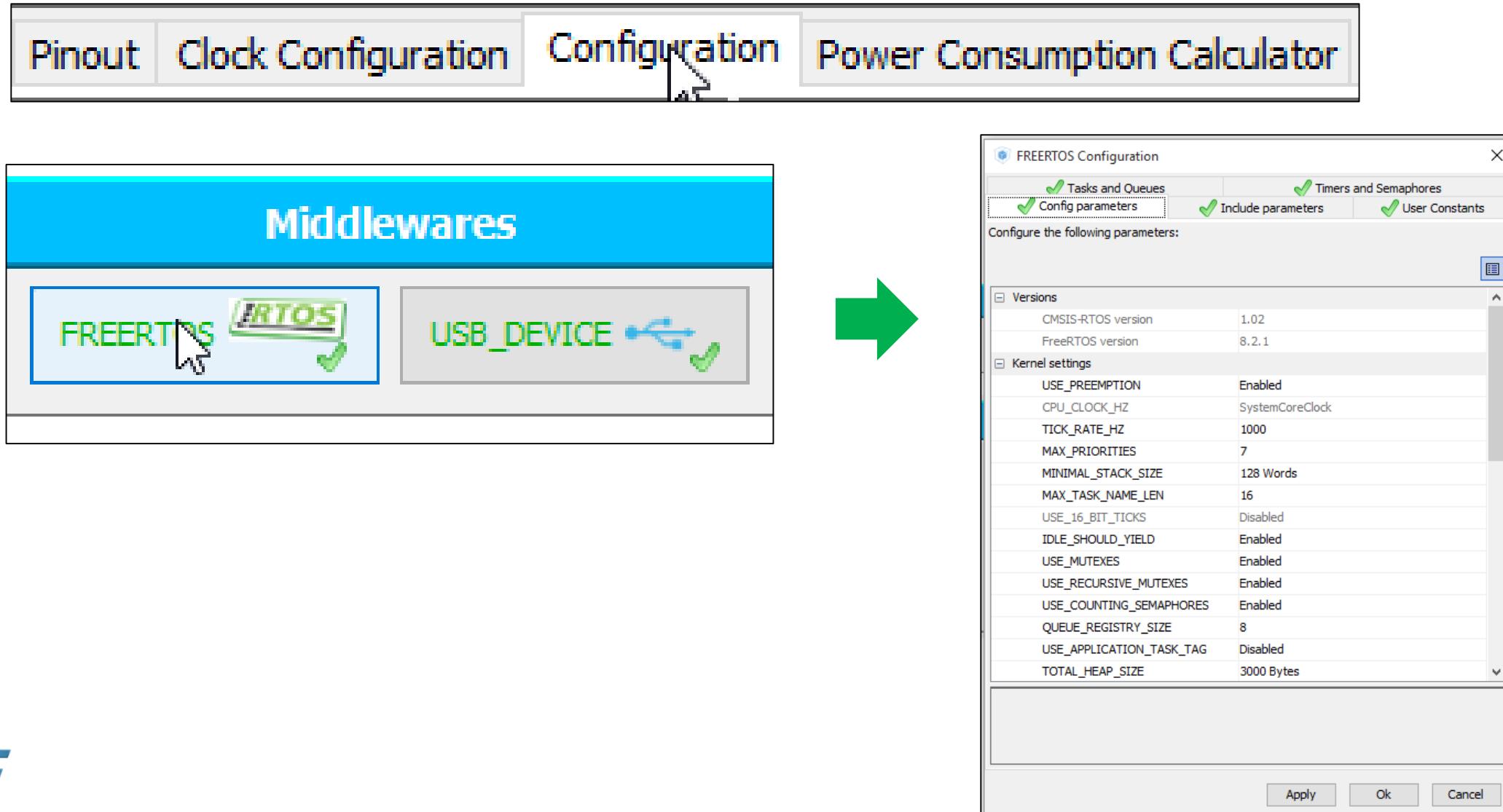


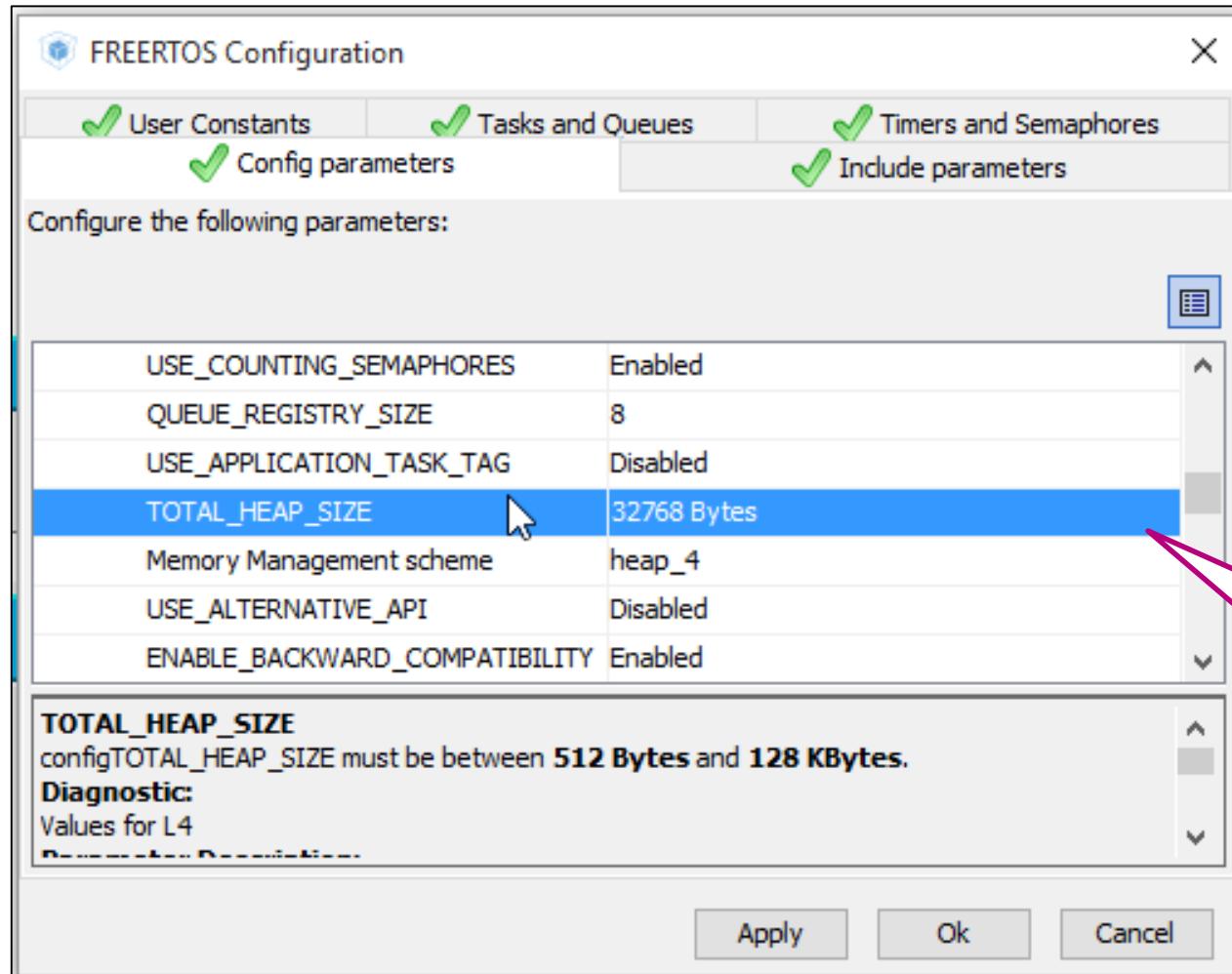
Go to Configuration



Enable FreeRTOS







TOTAL_HEAP_SIZE
→ 32768 bytes
(to have some reserve)

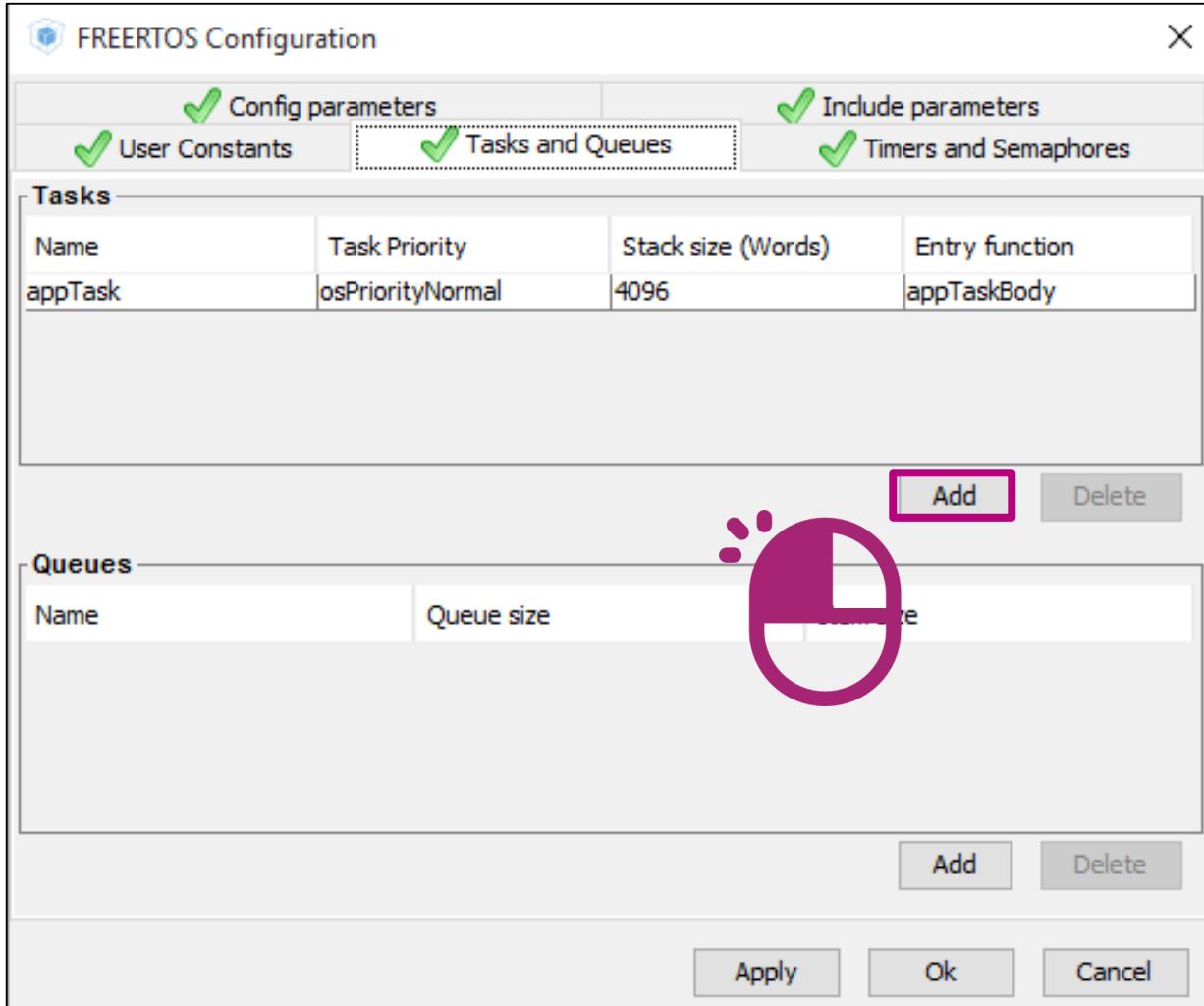
The screenshot shows the FreeRTOS Configuration interface. On the left, there's a sidebar with icons for Config parameters, User Constants, and Tasks and Queues. Below this, under 'Tasks', there's a table with columns: Name, Task Priority, Stack size (Words), and Entry function. A row for 'defaultTask' is selected, showing 'osPriorityNormal' as the priority and '128' as the stack size. Under 'Queues', there's a table with a single column 'Name', which is currently empty. At the bottom are buttons for 'Add', 'Delete', 'Apply', 'Ok', and 'Cancel'. A large pink callout bubble points to the 'Tasks and Queues' tab in the sidebar with the text 'Go to Tasks and Queues configuration'. Another pink callout bubble points to the 'defaultTask' row with the text 'Modify the default task'. A '2x' magnification icon is located on the far left.

appTask

A modal dialog box titled 'Edit Task' is shown. It has fields for 'Name' (containing 'appTask'), 'osPriorityNormal' (selected from a dropdown menu), 'Stack size (Words)' (set to '4096'), and 'Entry function' (set to 'appTaskBody'). Buttons at the bottom include 'OK' and 'Cancel'. A pink callout bubble points to the 'Name' field with 'appTask'. Another bubble points to the 'osPriorityNormal' dropdown with 'osPriorityNormal'. A third bubble points to the 'Stack size (Words)' field with '4096'. A fourth bubble points to the 'Entry function' field with 'appTaskBody'.

appTask

appTaskBody



usbTask

usbTask

New Task

Name

osPriorityHigh

usbTask

osPriorityHigh

256

Stack size (Words)

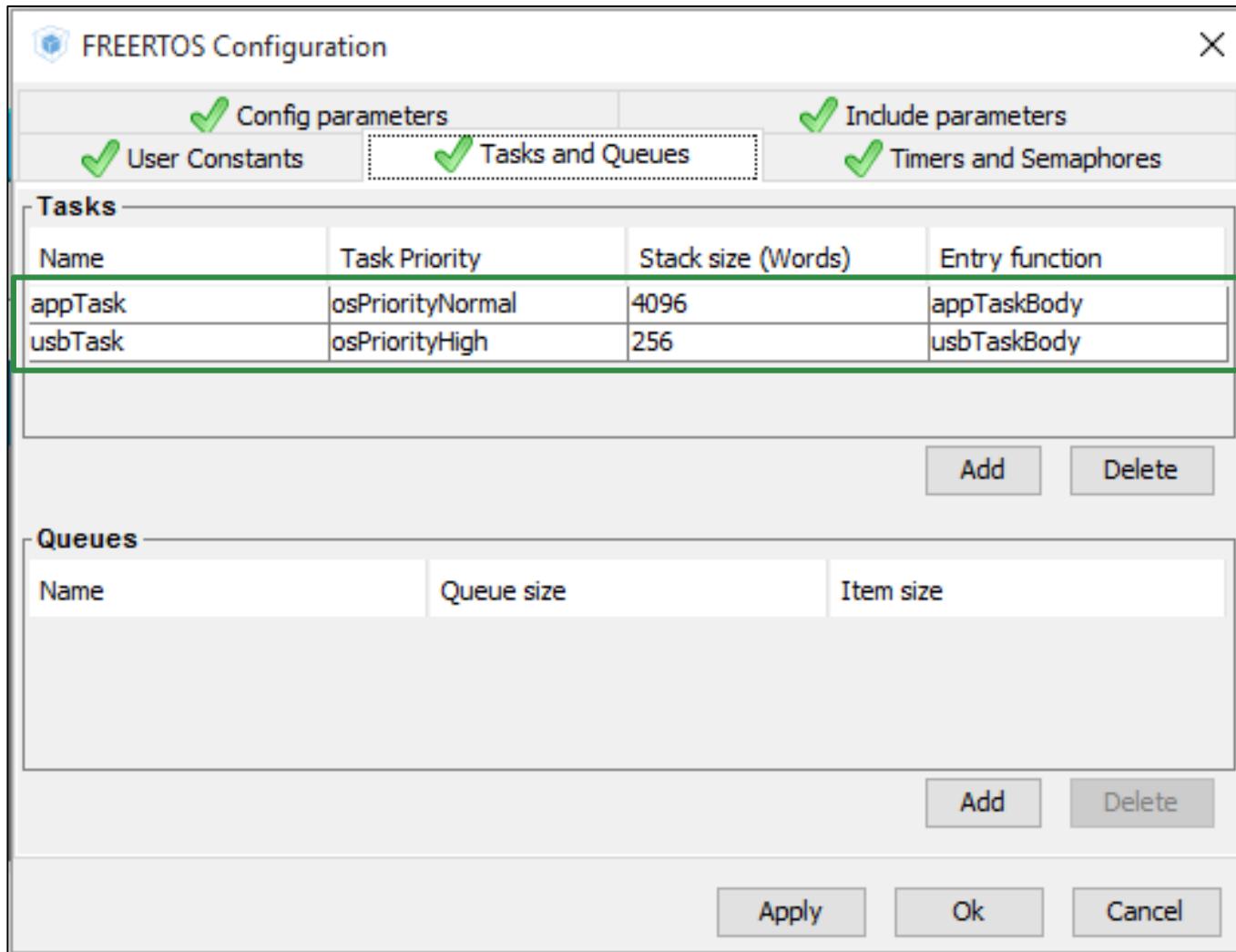
Entry function

usbTaskBody

OK

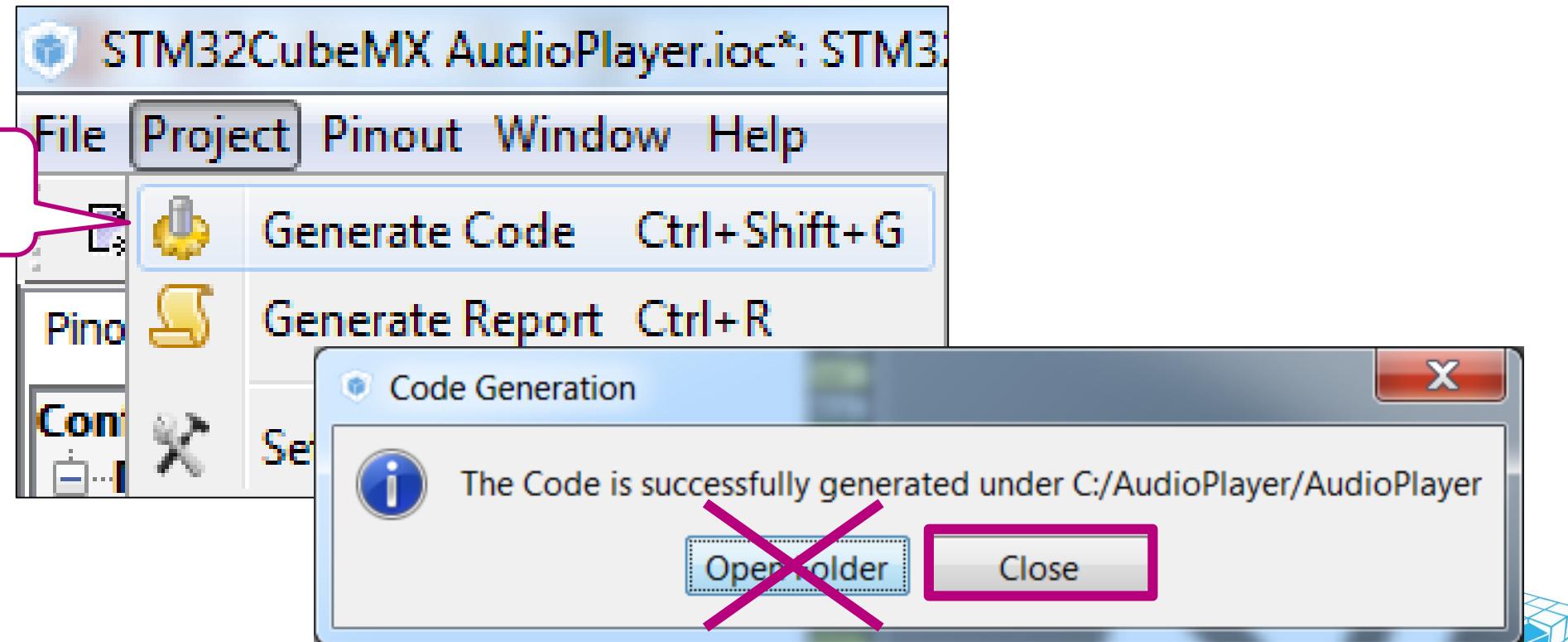
Cancel

usbTaskBody



Save and generate the project

Do not open the project yet



- 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

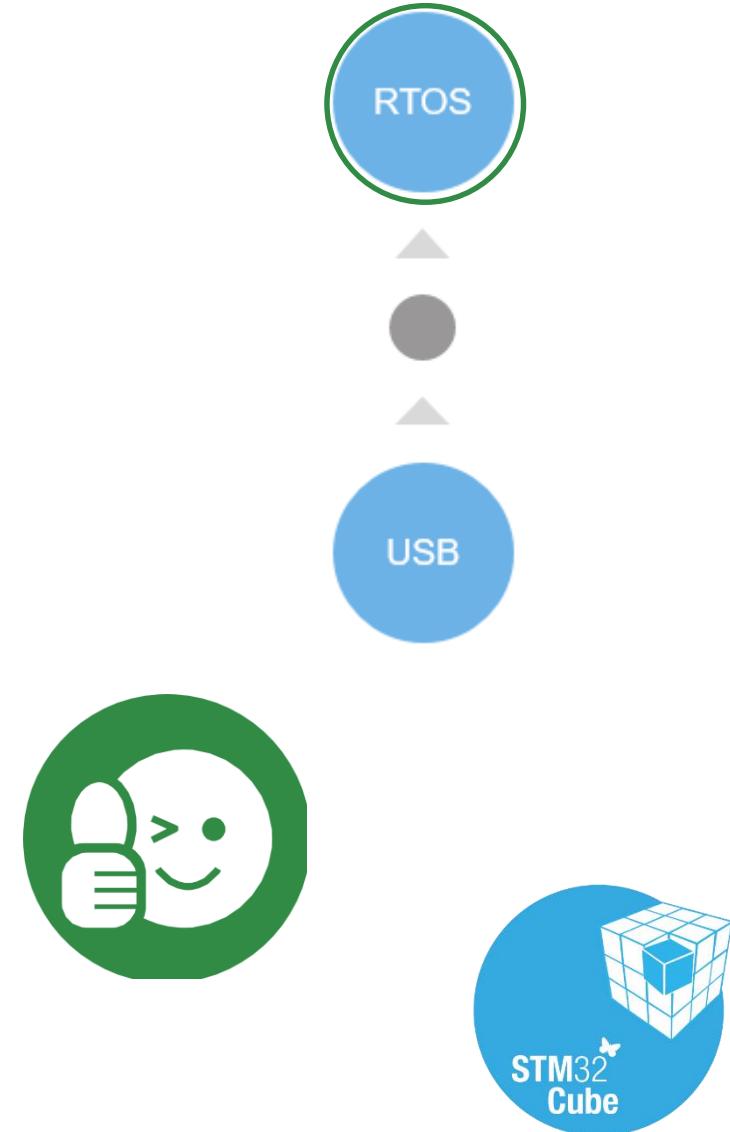
stm32l4xx_it.c

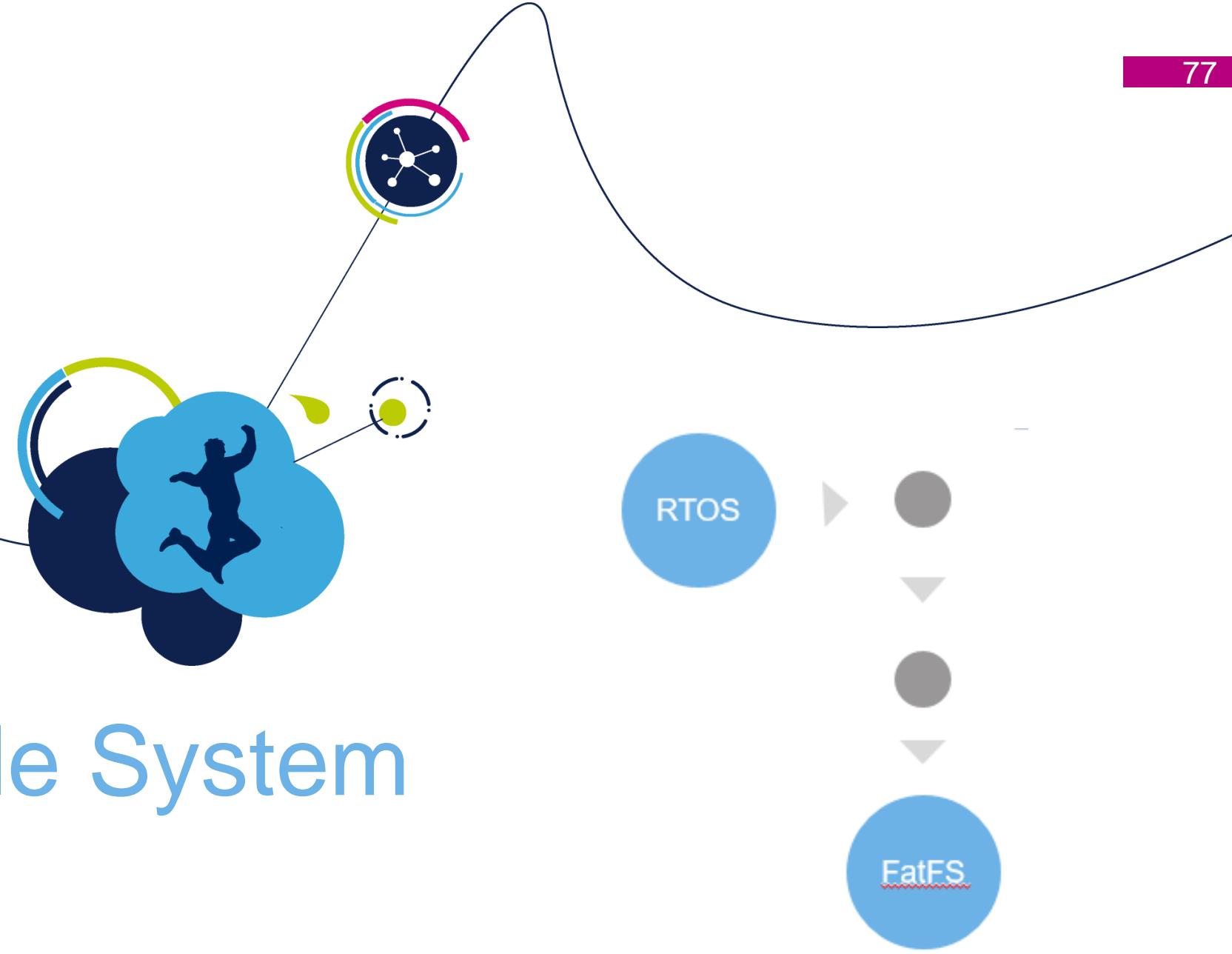
This operation is STM32CubeMX non-intrusive. You can re-generate the project later and the modifications will be retained.

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

1. Open the project
2. Compile and download
3. Press the reset button on discovery once download has finished

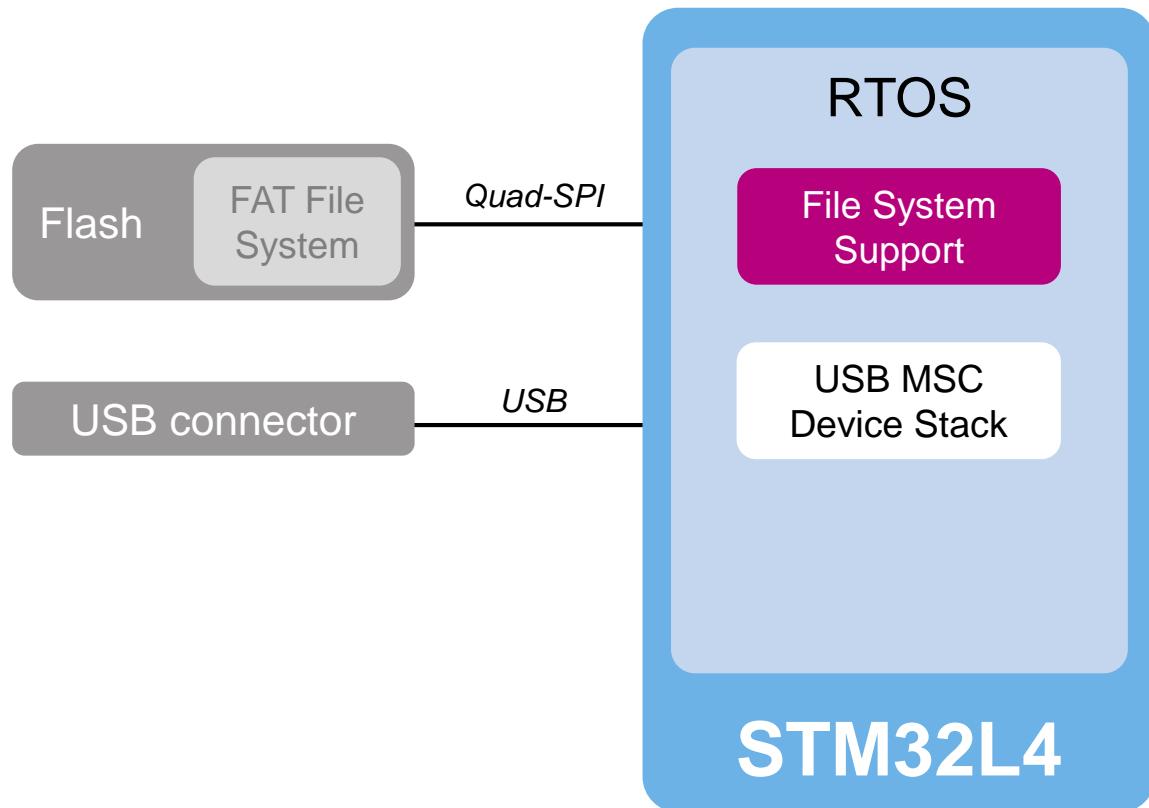
- appTask – GREEN LED blinking
- usbTask – USB connected → RED LED ON
- usbTask – USB connected → RED LED OFF





STEP3 – File System

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



PC 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

MCU application in C language

C standard libraries (stdio.h)

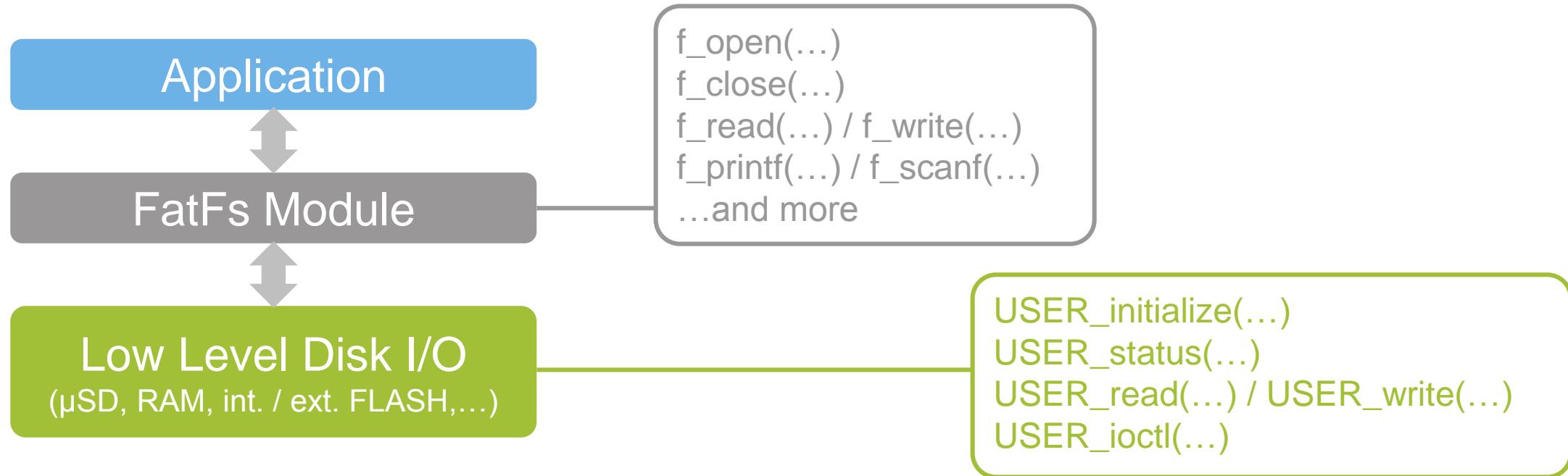
~~`fopen(...)` – opens a file~~
~~`fclose(...)` – closes a file~~
~~`fread(...)` – reads from a file~~
~~`fwrite(...)` – writes to a file~~
~~`fprintf(...)` – formatted output to a file~~
~~`fscanf(...)` – formatted input from a file~~

No OS + No standard File System =

No native support by
C standard IO library

Let's add “special” library to support that

- Windows compatible FAT file system support



UM1721 – Developing Applications on STM32Cube with FatFs

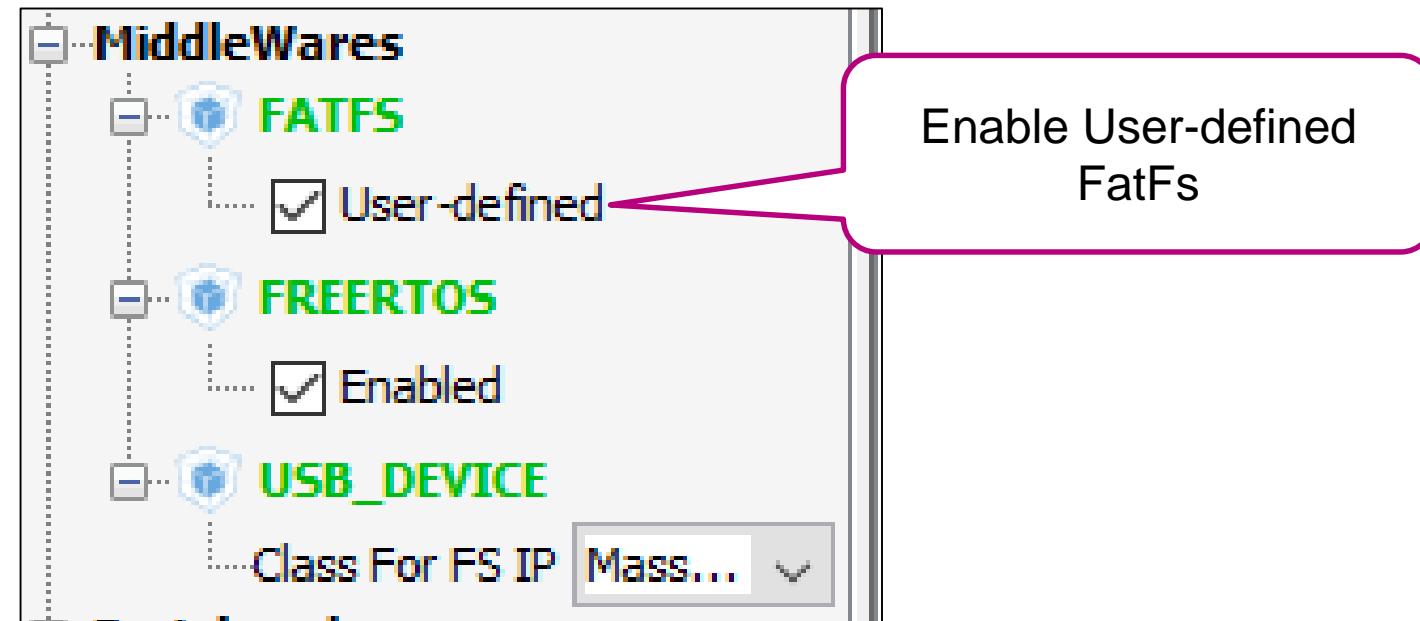
http://www.st.com/st-web-ui/static/active/en/resource/technical/document/user_manual/DM00105259.pdf

FatFs project home page

http://elm-chan.org/fsw/ff/00index_e.html



Go to Configuration





FATFS in
Middlewares

FATFS Configuration

Configure the below parameters :

<input checked="" type="checkbox"/> Set Defines	<input checked="" type="checkbox"/> User Constants
Configure the below parameters :	
Version FATFS version R.0.11	
Function Parameters <ul style="list-style-type: none"> FS_TINY (Tiny mode) Enabled FS_READONLY (Read-only mode) Disabled FS_MINIMIZE (Minimization level) Disabled USE_STRFUNC (String functions) Enabled with LF -> CRLF conversion USE_FIND (Find functions) Enabled USE_MKFS (Make filesystem function) Enabled USE_FORWARD (Forward function) Disabled USE_LABEL (Volume label functions) Enabled USE_FASTSEEK (Fast seek function) Enabled 	
Locale and Namespace Parameters <ul style="list-style-type: none"> CODE_PAGE (Code page on target) Latin 1 (Windows) USE_LFN (Use Long Filename) Disabled MAX_LFN (Max Long filename) 255 LFN_UNICODE (Enable Unicode) ANSI/OEM STRF_ENCODE (Character encoding) UTF-8 FS_RPATH (Relative Path) Disabled 	

Physical Drive Parameters <ul style="list-style-type: none"> VOLUMES (Logical drives) 1 MAX_SS (Maximum Sector Size) 4096 MIN_SS (Minimum Sector Size) 4096 MULTI_PARTITION (Volume partitio...) Disabled USE_TRIM (Erase feature) Disabled FS_NOFSINFO (Force full FAT scan) 0 	
System Parameters <ul style="list-style-type: none"> FS_NORTC (Timestamp feature) Dynamic timestamp NORTC_YEAR (Year for timestamp) 2015 NORTC_MON (Month for timestamp) 6 NORTC_MDAY (Day for timestamp) 4 WORD_ACCESS (Platform depende...) Byte access FS_REENTRANT (Re-Entrancy) Enabled FS_TIMEOUT (Timeout ticks) 1000 SYNC_t (O/S sync object) osSemaphoreId FS_LOCK (Number of files opened si...) 2 	

MAX_SS → 4096
MIN_SS → 4096

CODE_PAGE
→ Latin 1 (Windows)

USE_LFN
→ Disabled

LFN_UNICODE
→ ANSI/OEM

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

Mutex

appTask
FatFs

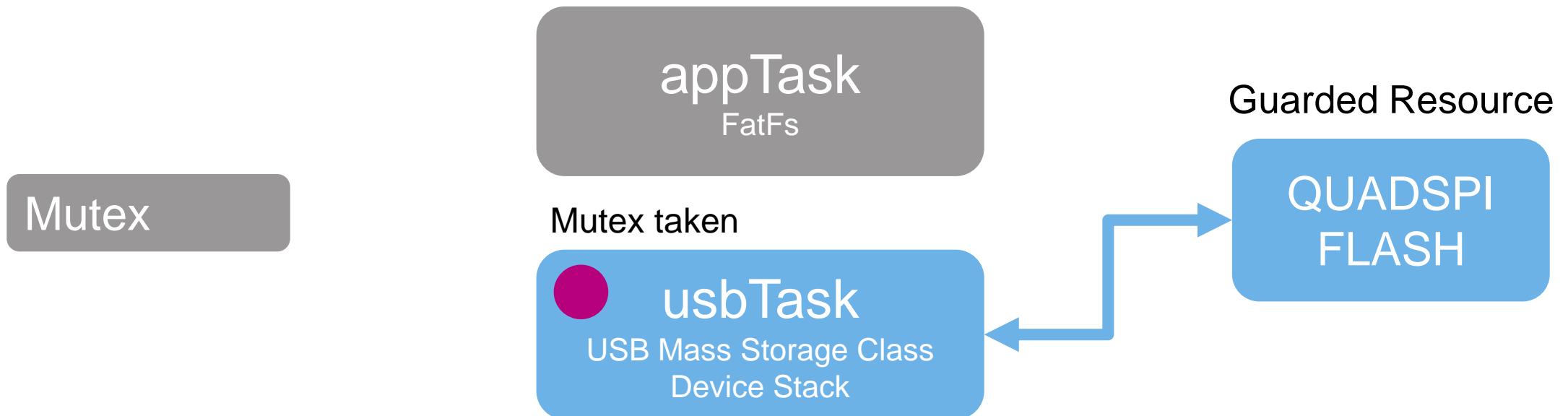
osMutexWait(...)

usbTask
USB Mass Storage Class
Device Stack

Guarded Resource

QUADSPI
FLASH

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



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

Mutex

osMutexWait(...)

appTask
FatFs

usbTask
USB Mass Storage Class
Device Stack

Guarded Resource

QUADSPI
FLASH

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

Mutex

Waiting for Mutex

appTask
FatFs

osMutexRelease(...)

usbTask

USB Mass Storage Class
Device Stack

Guarded Resource

QUADSPI
FLASH

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

Mutex 

Waiting for Mutex

appTask
FatFs

Mutex released

usbTask
USB Mass Storage Class
Device Stack

Guarded Resource

QUADSPI
FLASH

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

Mutex

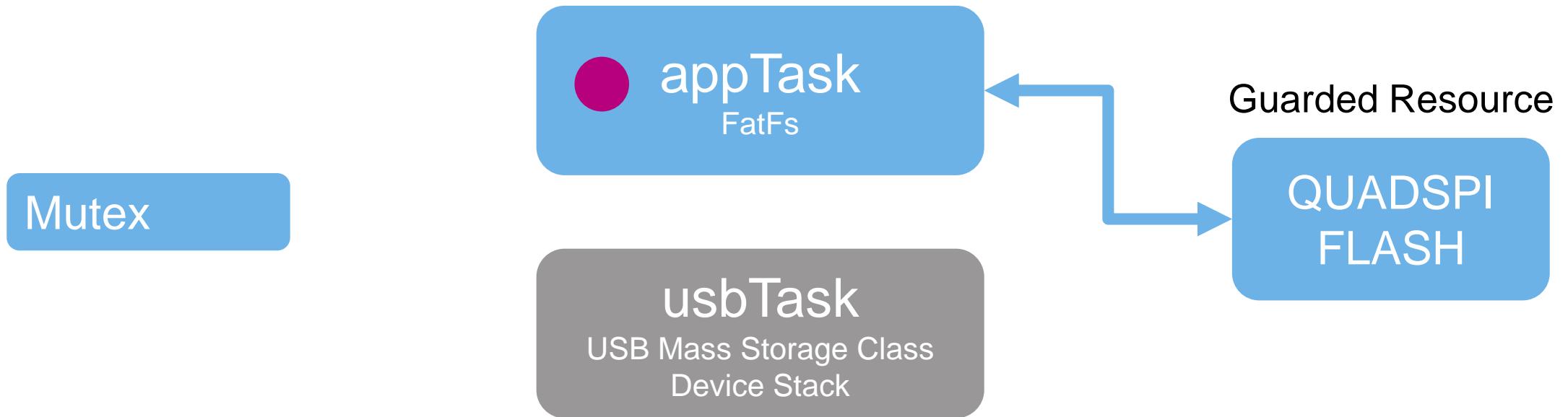
appTask
FatFs

usbTask
USB Mass Storage Class
Device Stack

Guarded Resource

QUADSPI
FLASH

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



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

Mutex

osMutexRelease(...)

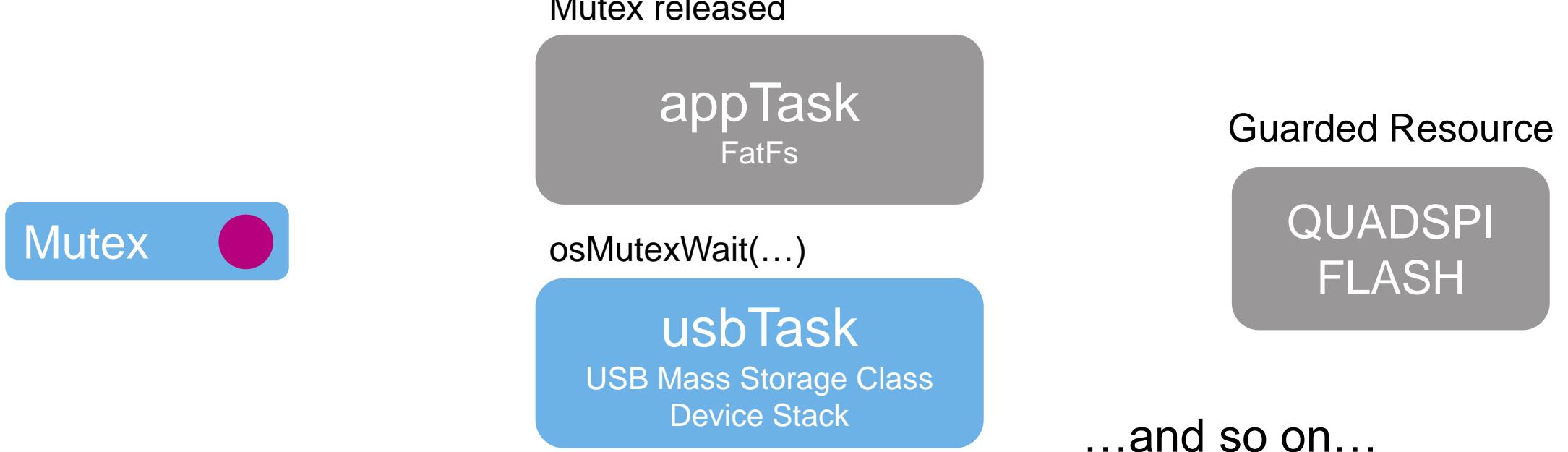
appTask
FatFs

usbTask
USB Mass Storage Class
Device Stack

Guarded Resource

QUADSPI
FLASH

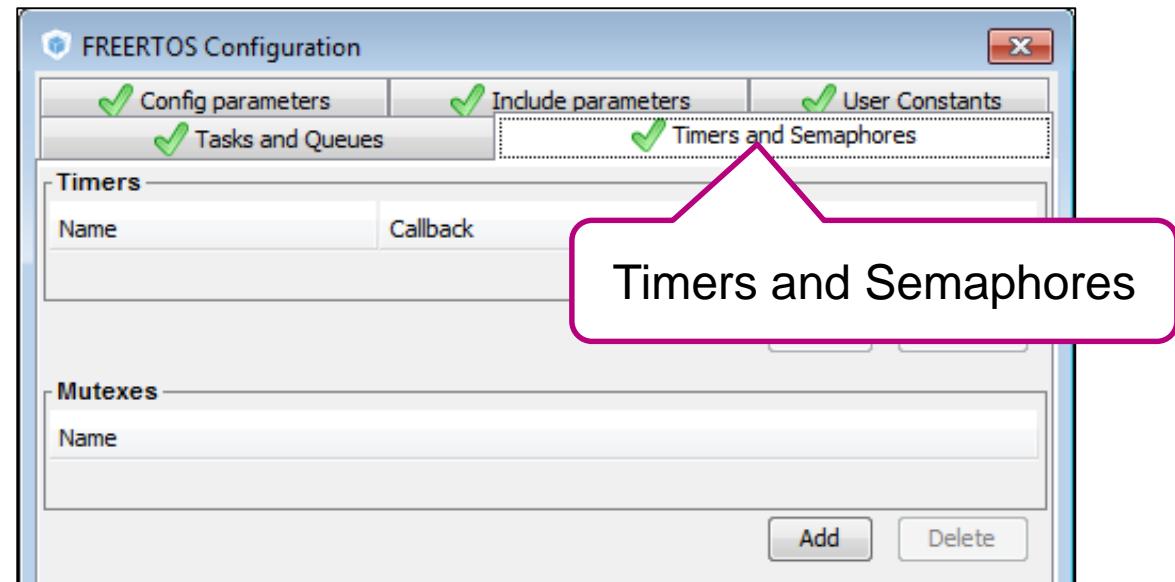
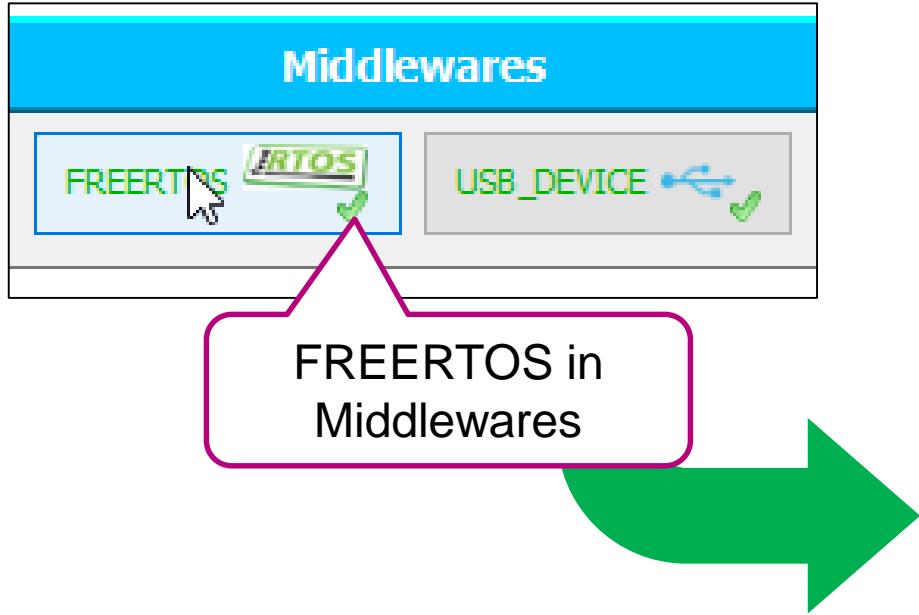
- Mutex needed as we can't have concurrent access to the QSPI FLASH

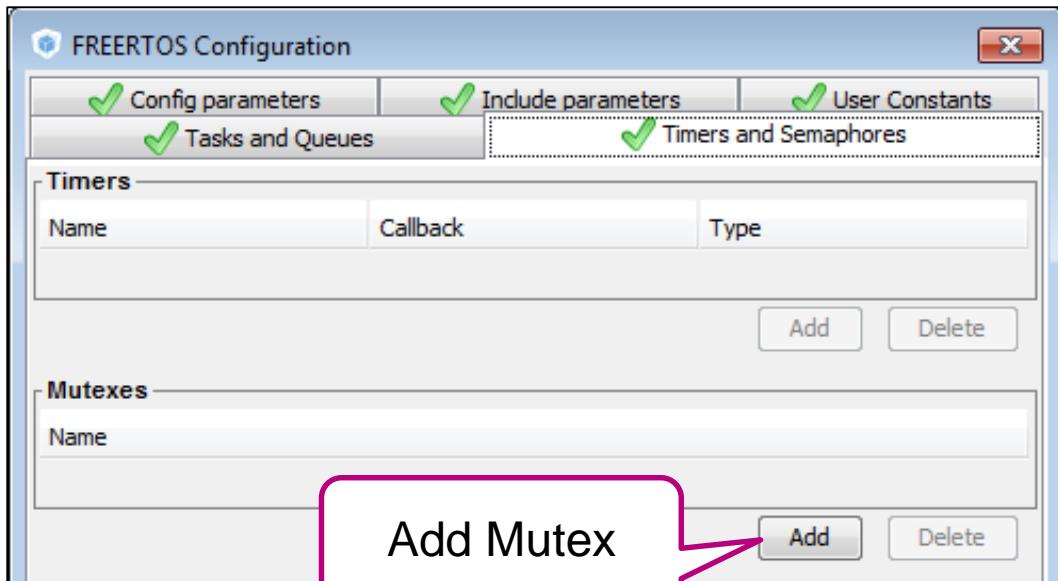
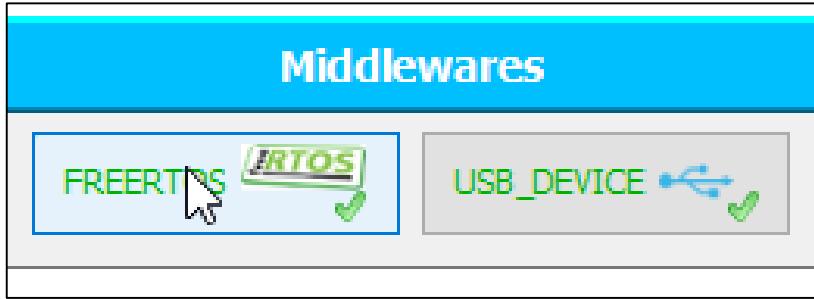


Single memory
Single QUADSPI interface

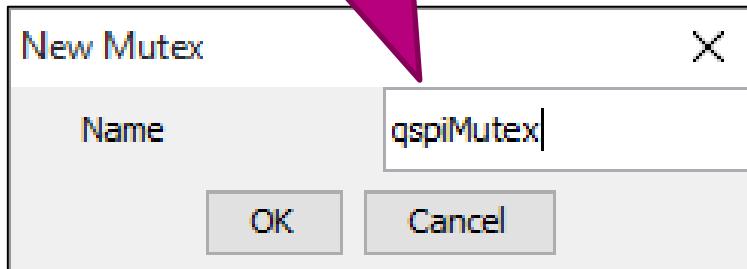


Concurrent access not
allowed





Name → qspiMutex



3

Checkpoint #6

95



FREERTOS Configuration

Config parameters

User Constants Tasks and Queues Timers and Semaphores

Timers

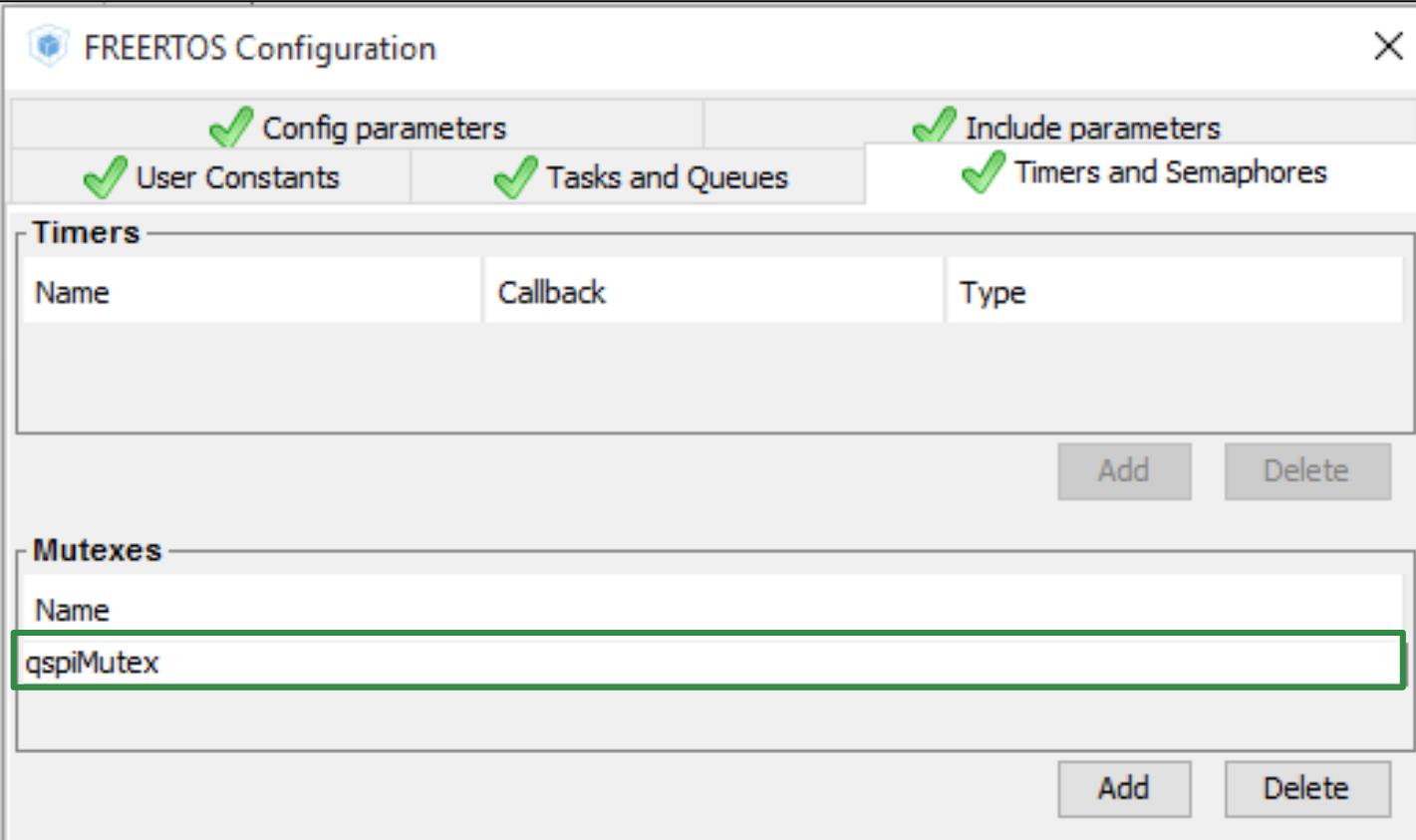
Name	Callback	Type

Add Delete

Mutexes

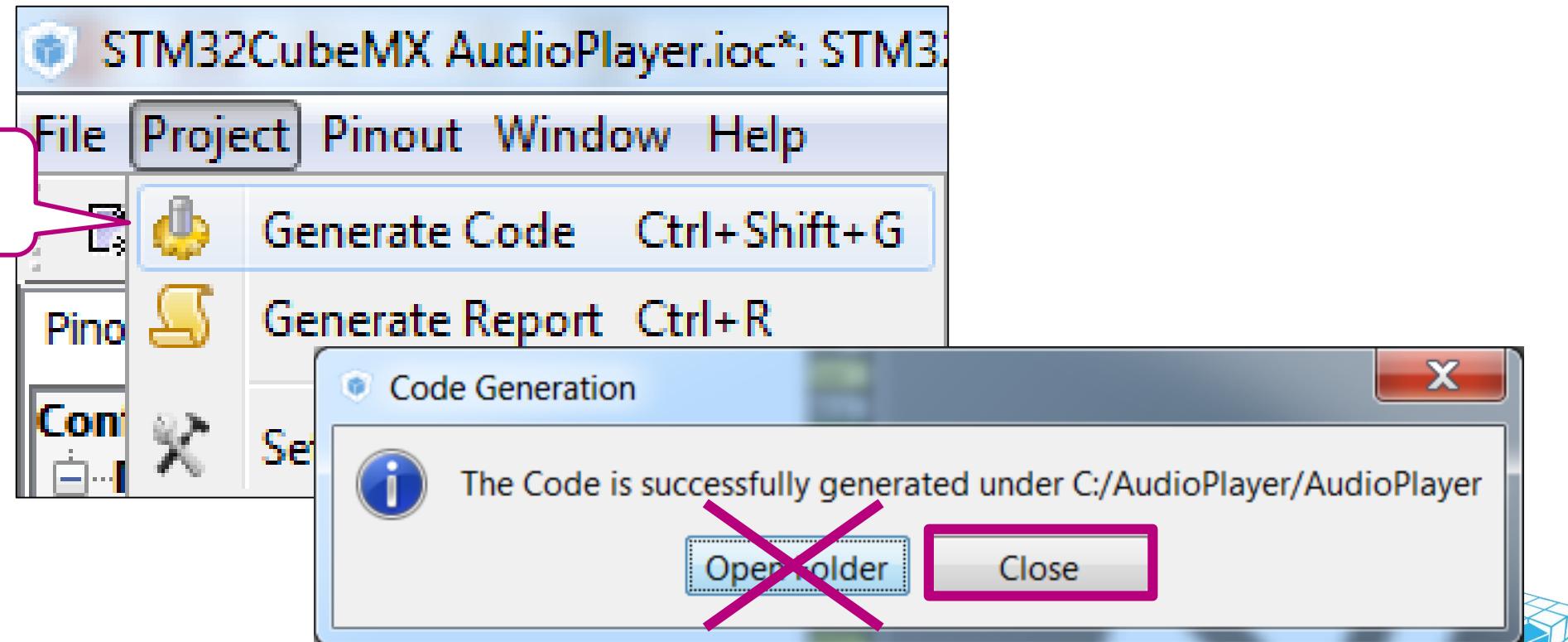
Name
qspiMutex

Add Delete



Save and generate the project

Do not open the project yet



- 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

- user_diskio.c

`USER_initialize (...)`

`USER_status (...)`



Storage initialization and Status info

`USER_read (...)`

`USER_write (...)`



Read / Write operations

`USER_ioctl (...)`



IO control

- `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

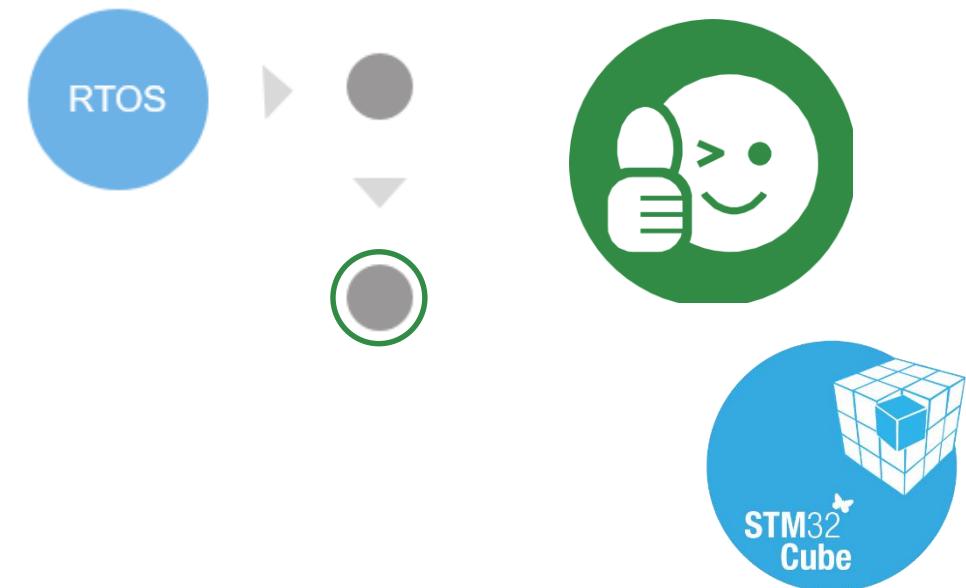
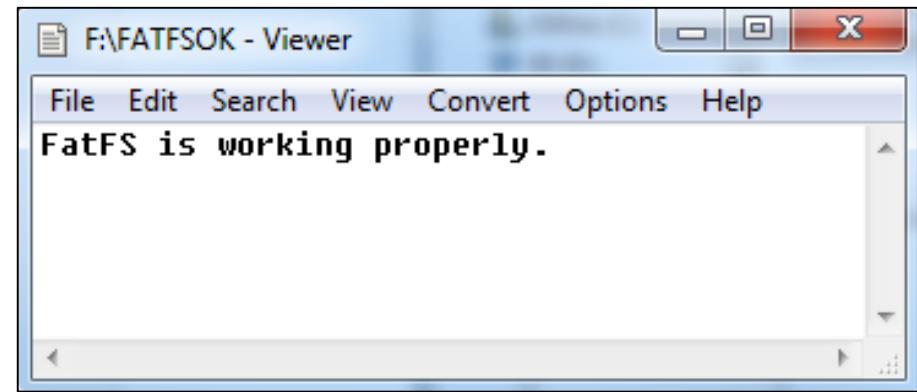
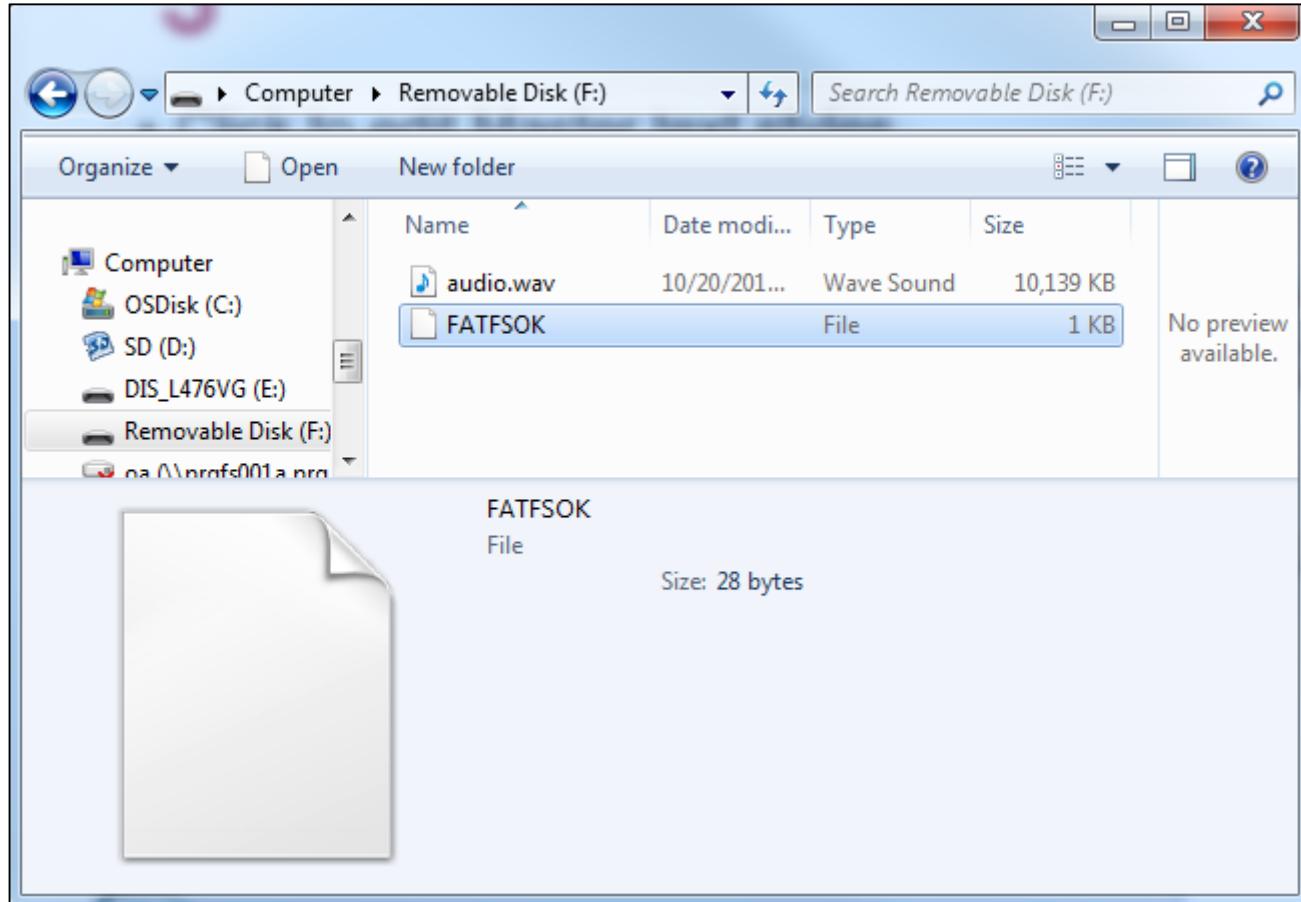
FATFSOK file in the root of the connected storage
Content should be (FatFS is working properly.)

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

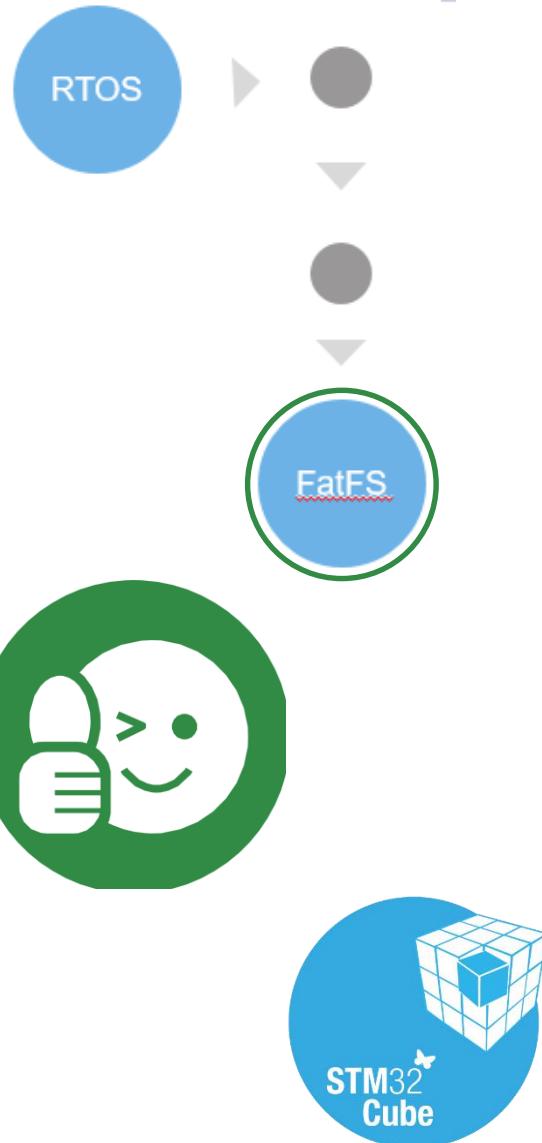
3

Checkpoint #7

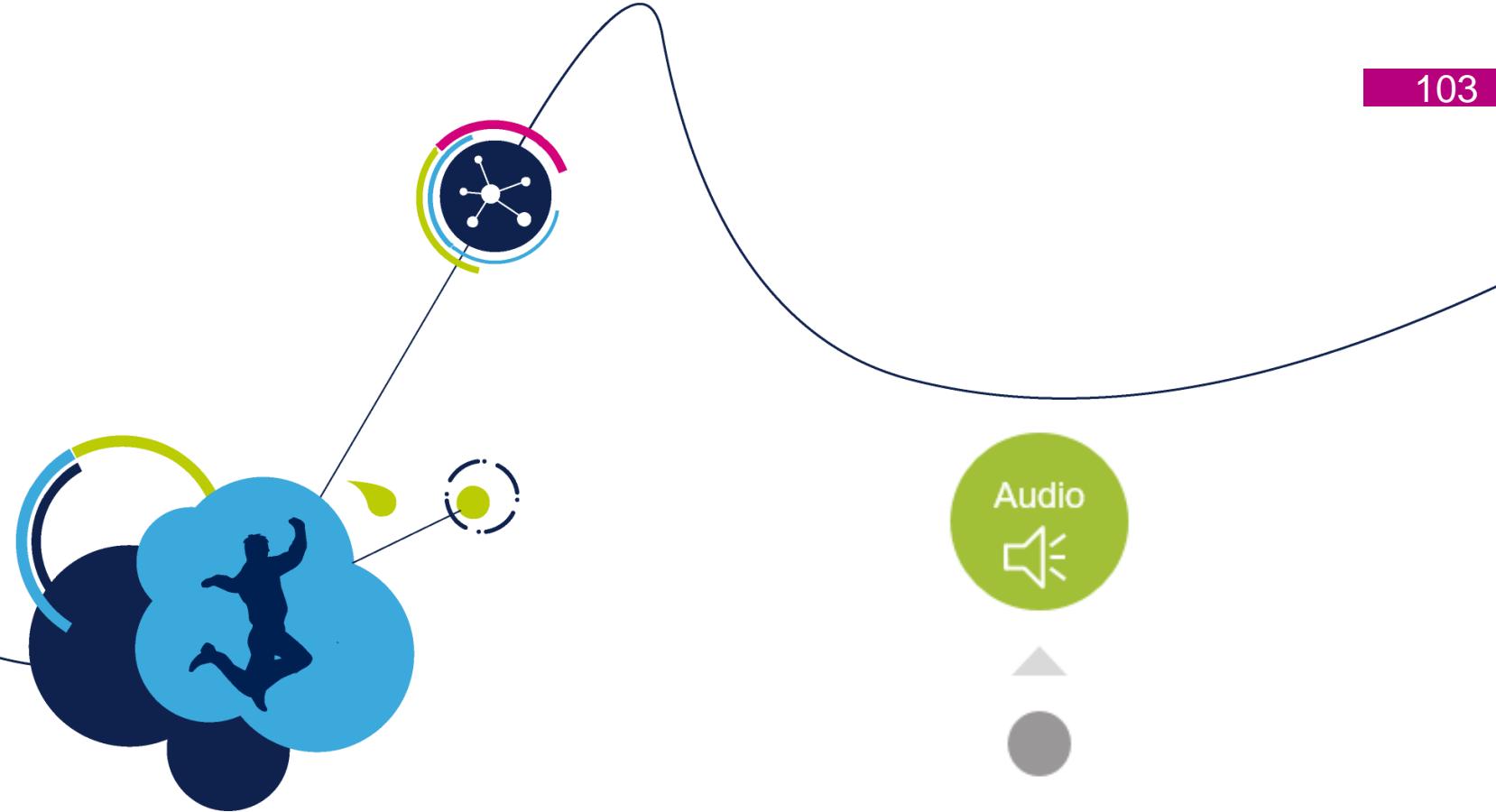
101



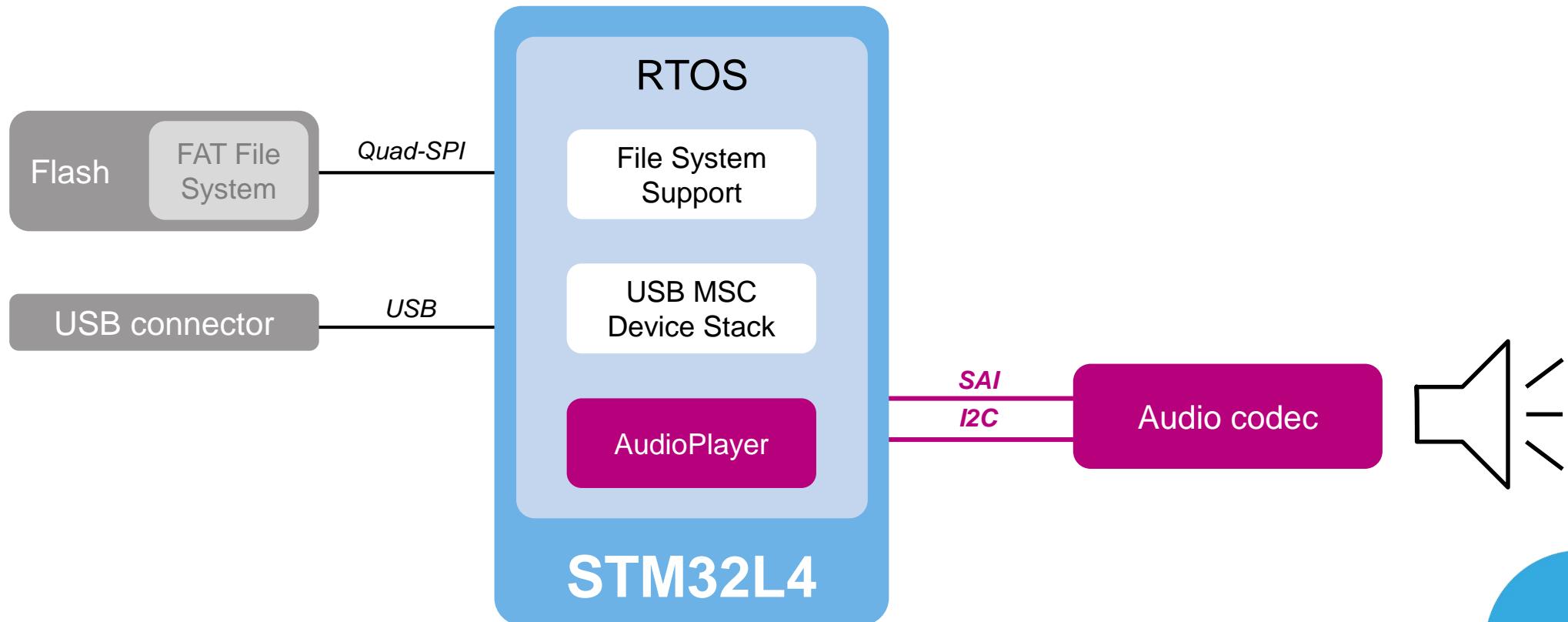
- LED_GREEN ON before qspiMutex take in appTask
- LED_GREEN OFF after qspiMutex release in appTask
- LED_RED ON on qspiMutex take in usbTask
- LED_RED OFF on qspiMutex release in usbTask

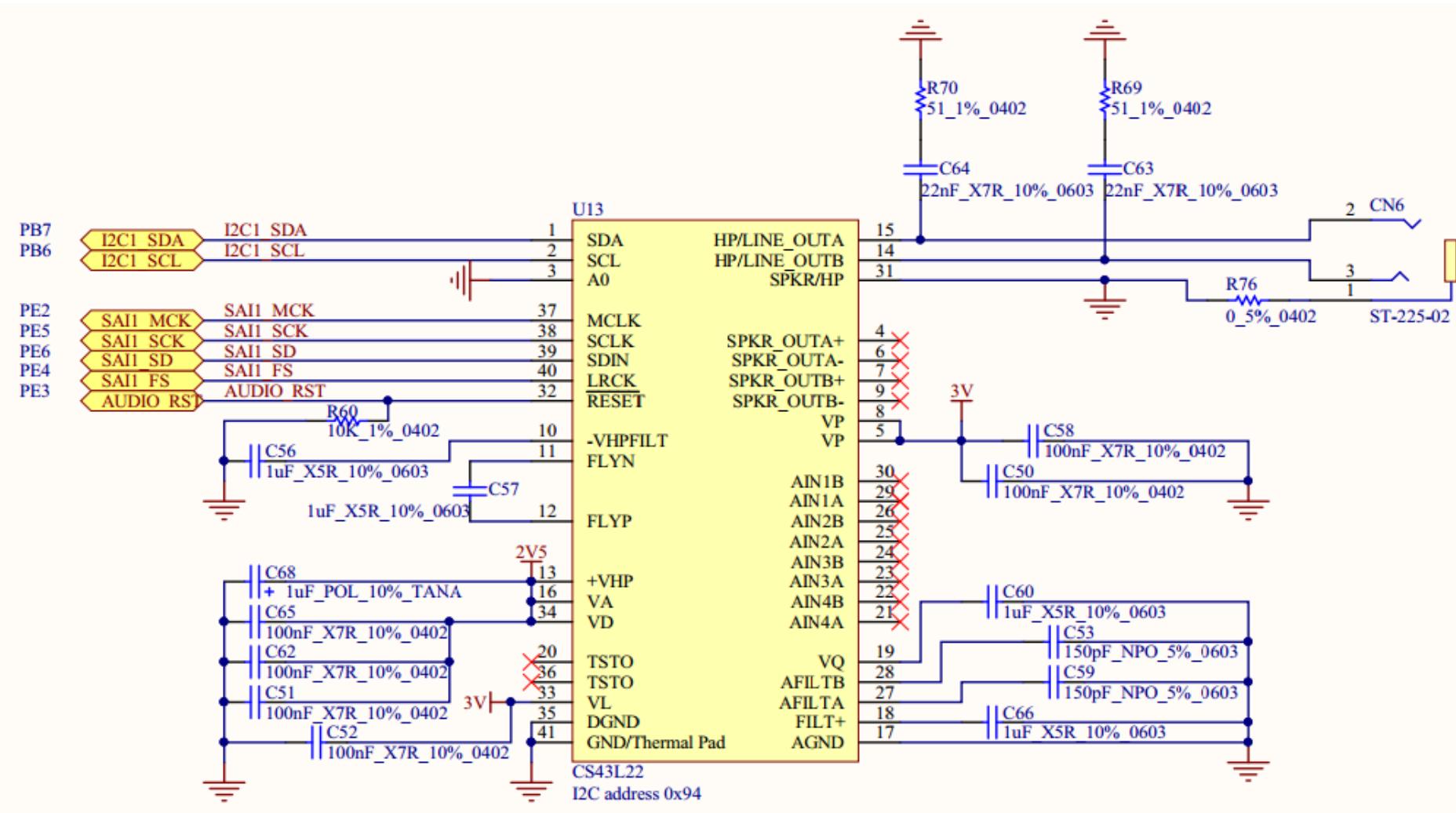


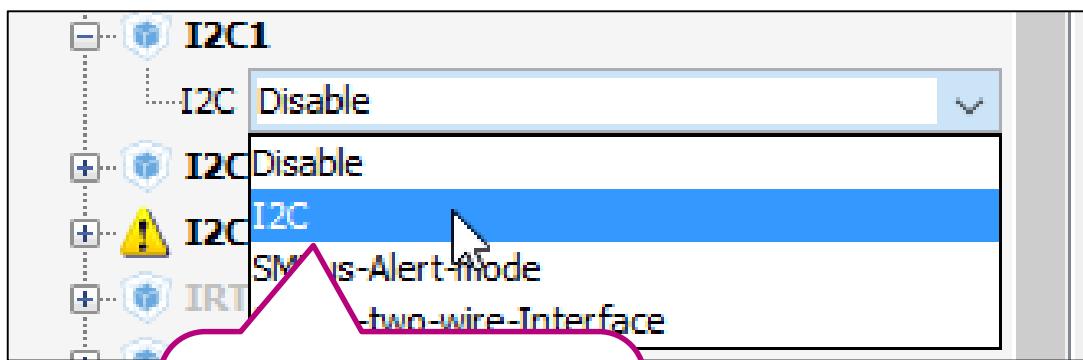
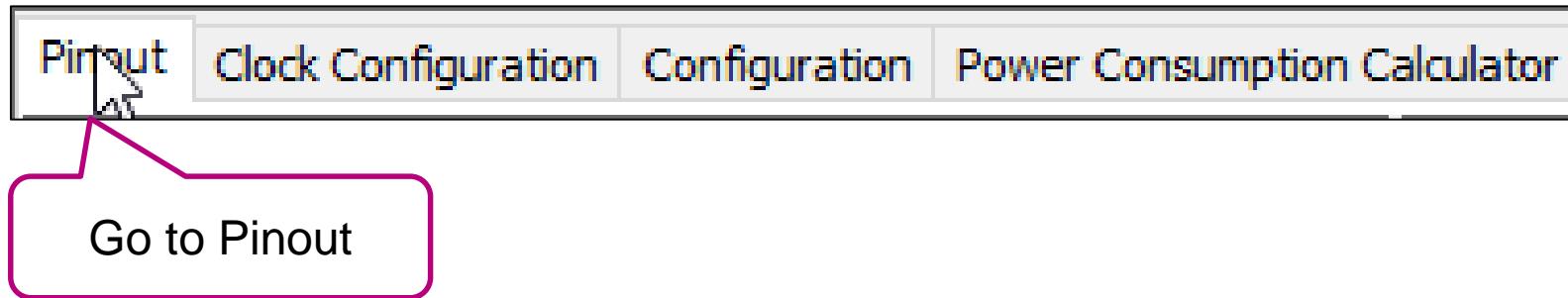
STEP4 – Audio



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





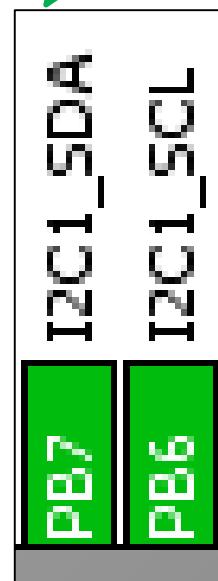
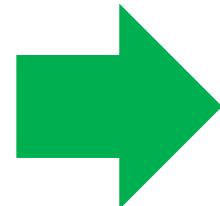
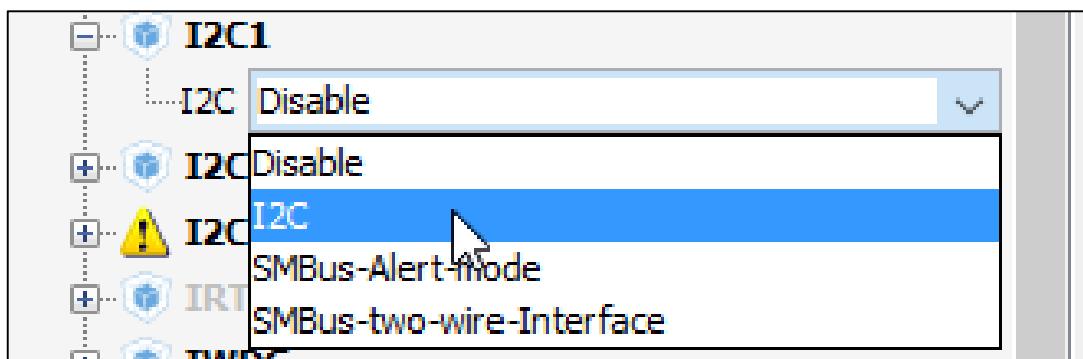


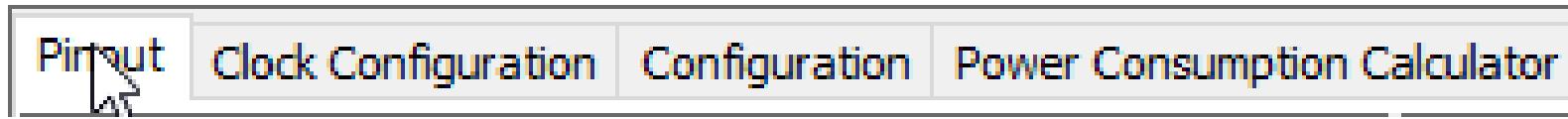
Under I2C1
peripheral select
I2C interface



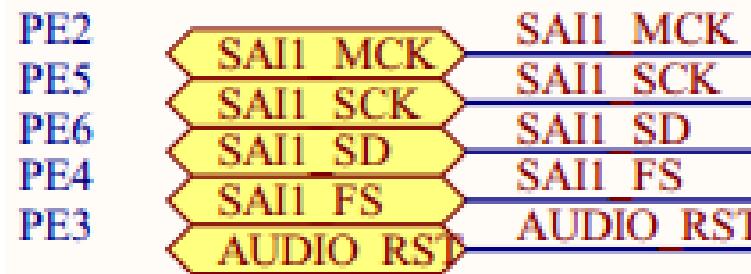


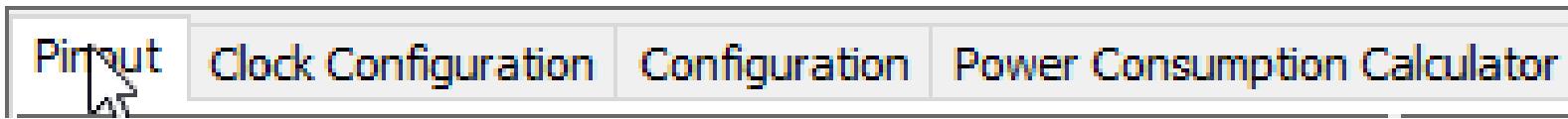
The corresponding pins are assigned and configured automatically!



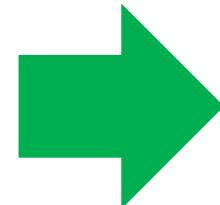
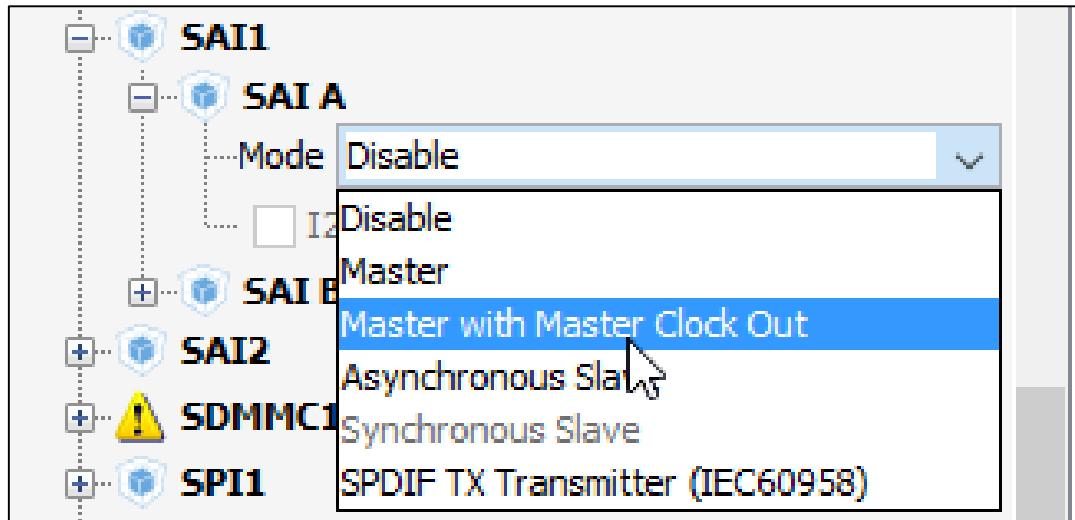


Under SAI1 peripheral select
SAI A interface to **Master**
with Master Clock Out

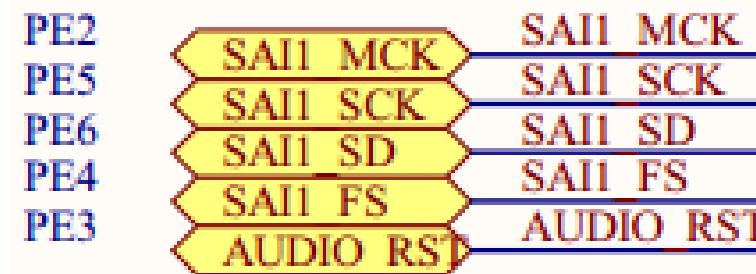


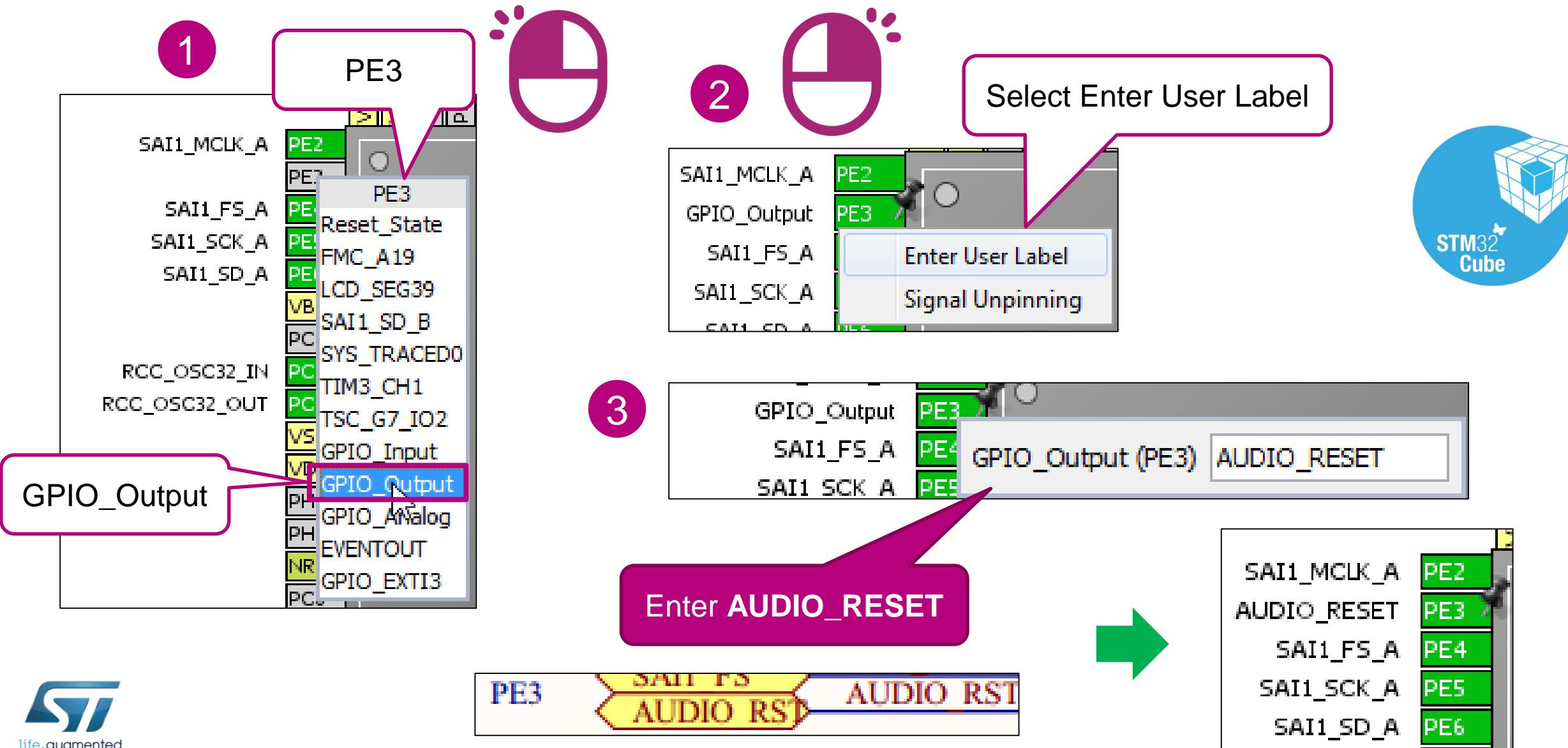


The corresponding pins are assigned and configured automatically!

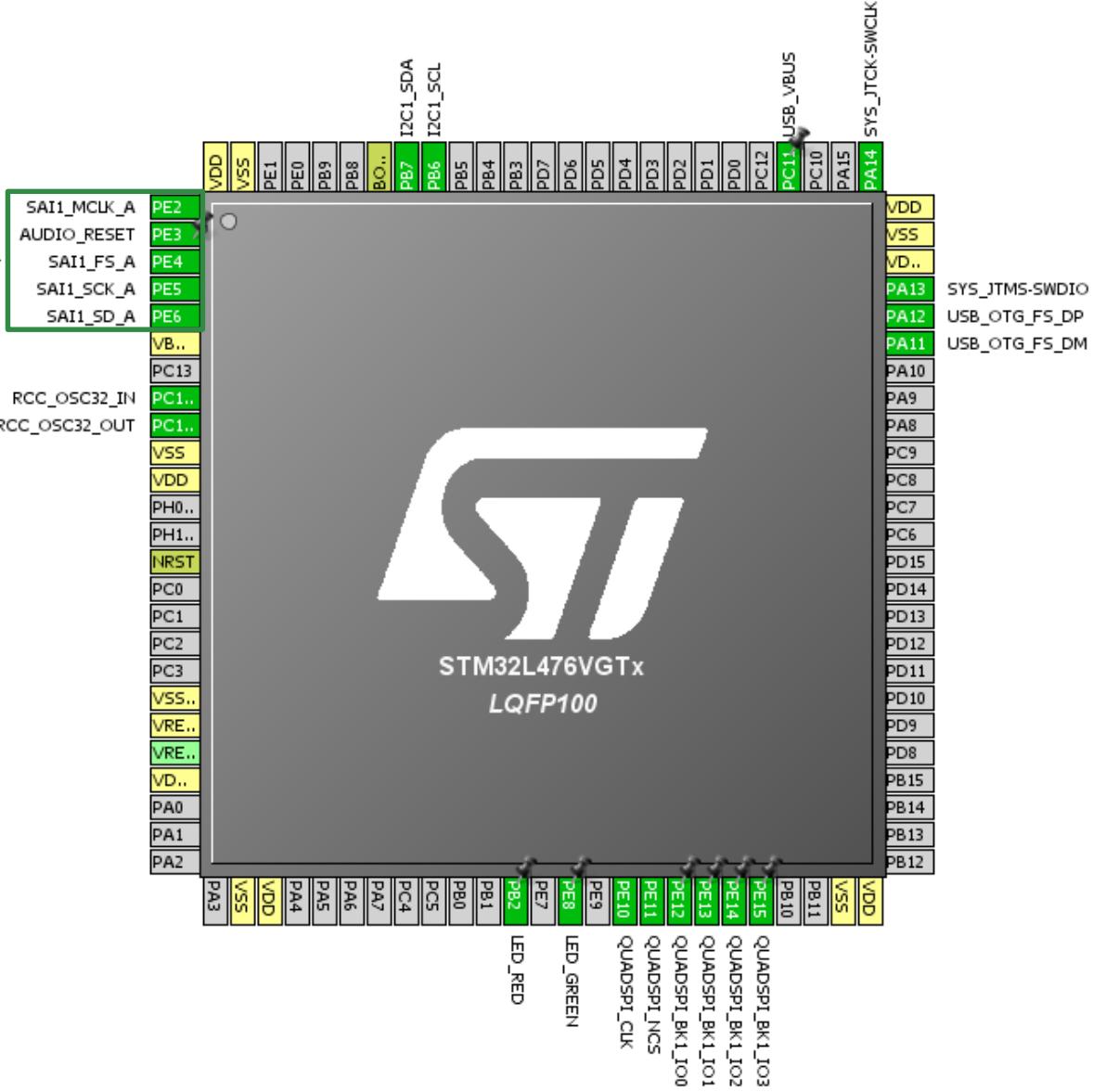


SAI1_MCLK_A	PE2
SAI1_FS_A	PE3
SAI1_SCK_A	PE4
SAI1_SD_A	PE5
	PE6





SAI1_MCLK_A	PE2
AUDIO_RESET	PE3
SAI1_FS_A	PE4
SAI1_SCK_A	PE5
SAI1_SD_A	PE6



SAI MCLK configuration

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



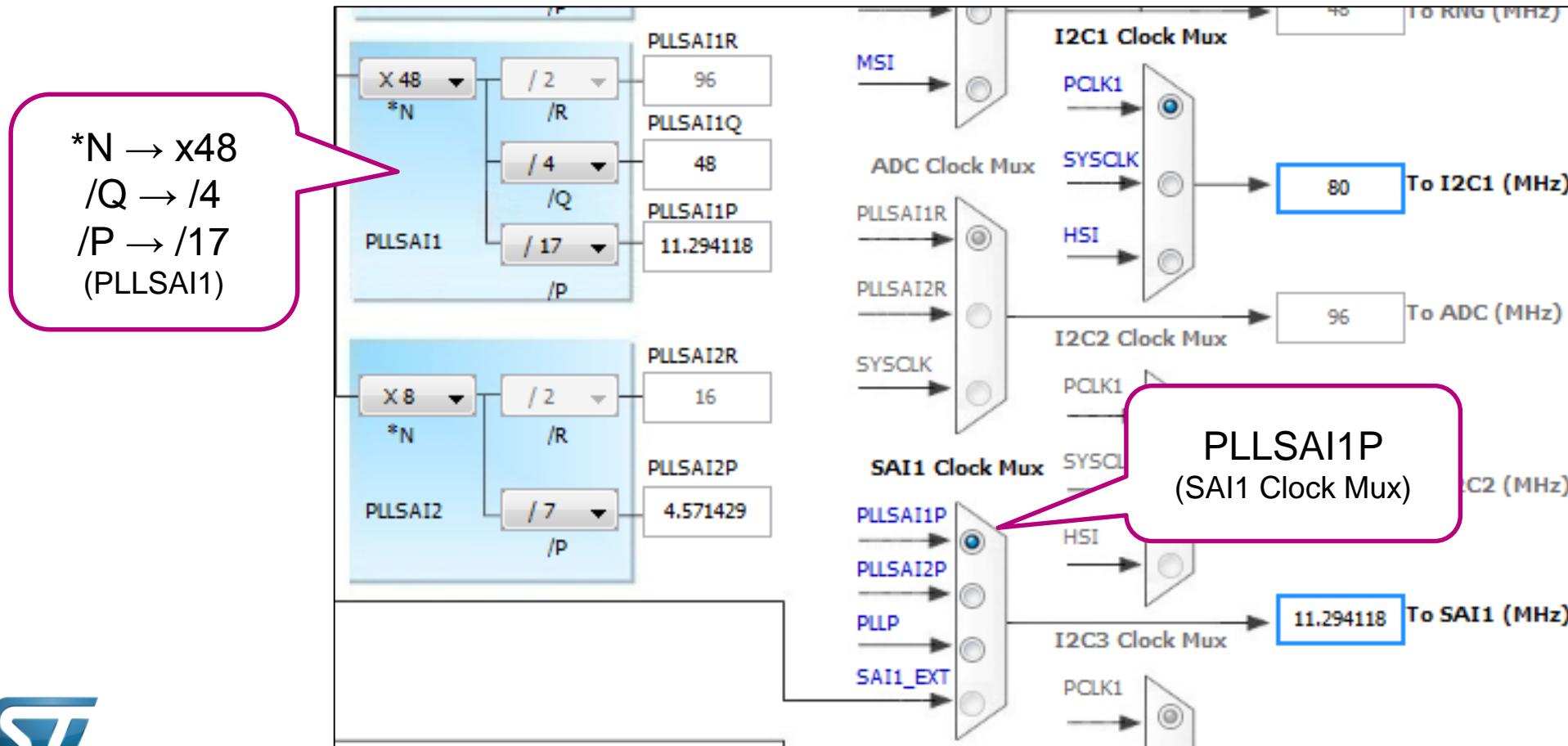
CS43L22

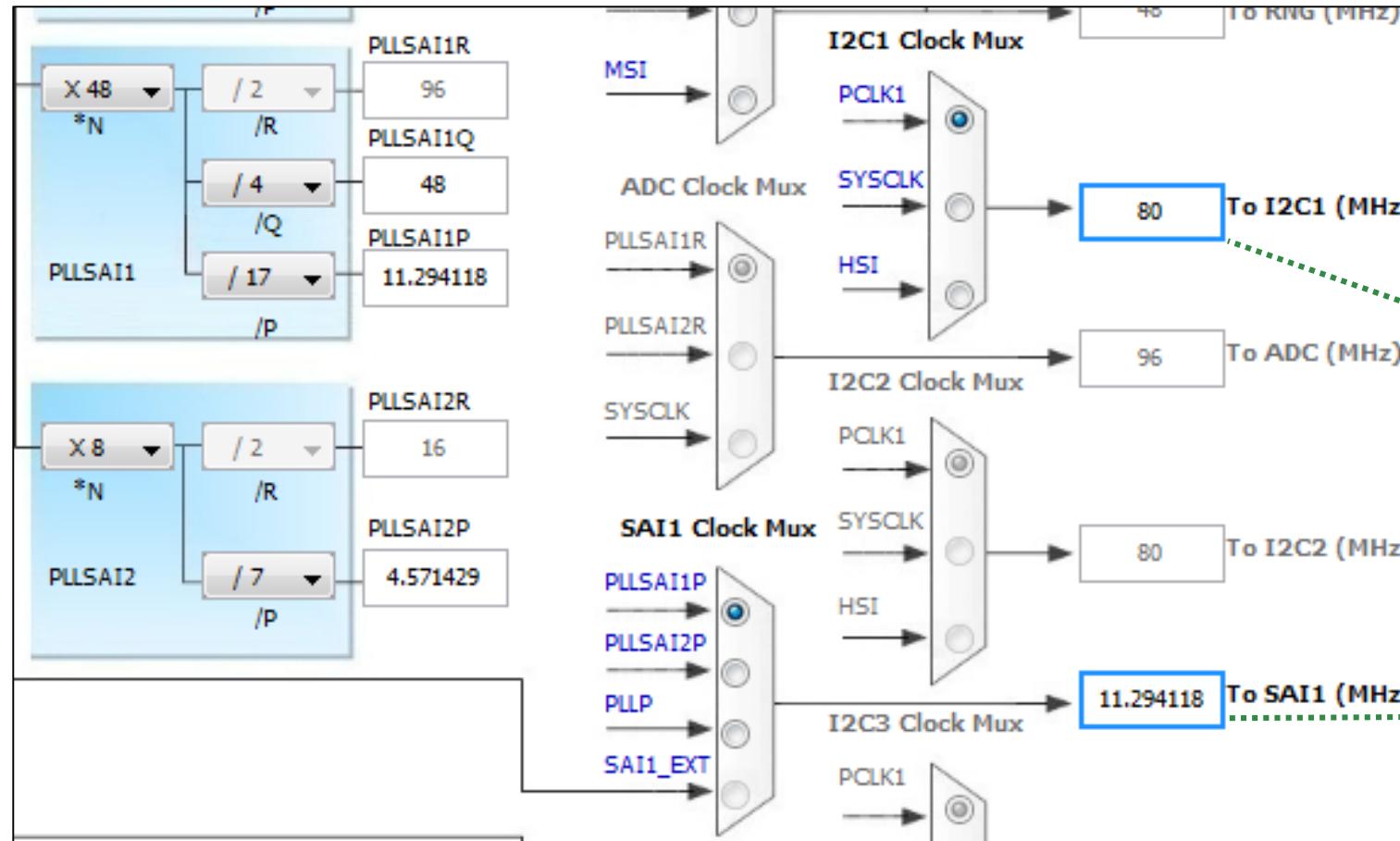
4.6 Serial Port Clocking

The CS43L22 serial audio interface port operates either as a slave or master, determined by the M/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/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

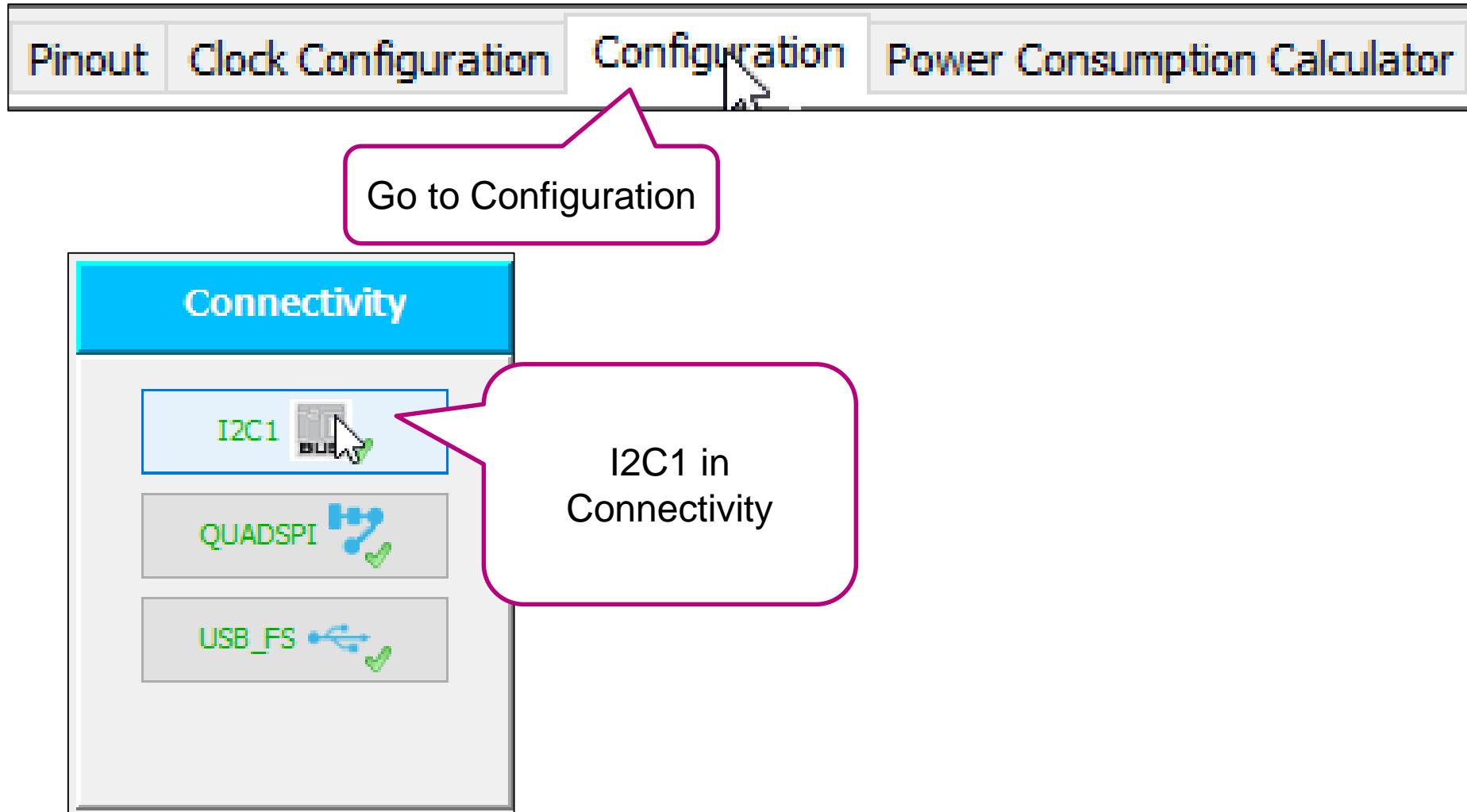
MCLK (MHz)	Sample Rate, Fs (kHz)	SPEED[1:0] (AUTO='0'b)	32kGROUP	VIDEOCLK	RATIO[1:0]	MCLKDIV2
12.2880	8.0000	11	1	0	00	0
	12.0000	11	0	0	00	0
	16.0000	10	1	0	00	0
	24.0000	10	0	0	00	0
	32.0000	01	1	0	00	0
	48.0000	01	0	0	00	0
	96.0000	00	0	0	00	0
11.2896	11.0250	11	0	0	00	0
	22.0500	10	0	0	00	0
	44.1000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





80MHz
(To I2C1)

11.294118
MHz
(To SAI1)



The image shows the STM32CubeMX software interface. On the left, there is a 'Connectivity' configuration page with three main sections: I2C1, QUADSPI, and USB_FS. A large green arrow points from this page to a detailed configuration dialog box on the right.

I2C1 Configuration Dialog Box:

Timing configuration	
I2C Speed Mode	Standard Mode
I2C Speed Frequency (KHz)	100
Rise Time (ns)	0
Fall Time (ns)	0
Coefficient of Digital Filter	0
Analog Filter	Enabled
Timing	0x10909CEC

Slave Features	
Clock No Stretch Mode	Disabled
General Call Address Detection	Disabled
Primary Address Length selection	7-bit
Dual Address Acknowledged	Disabled
Primary slave address	0

At the bottom of the dialog box are 'Apply', 'Ok', and 'Cancel' buttons.



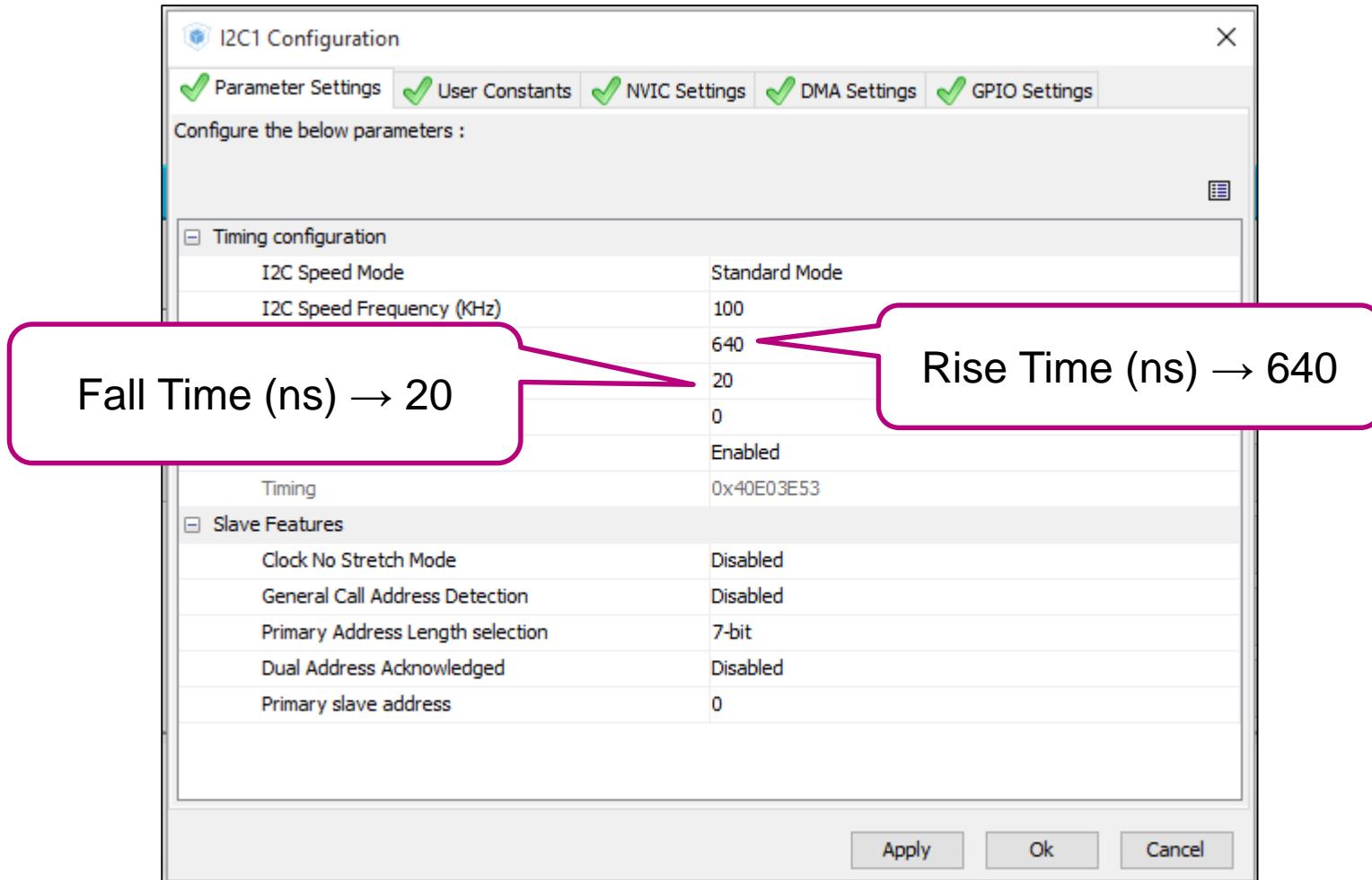
CIRRUS LOGIC®

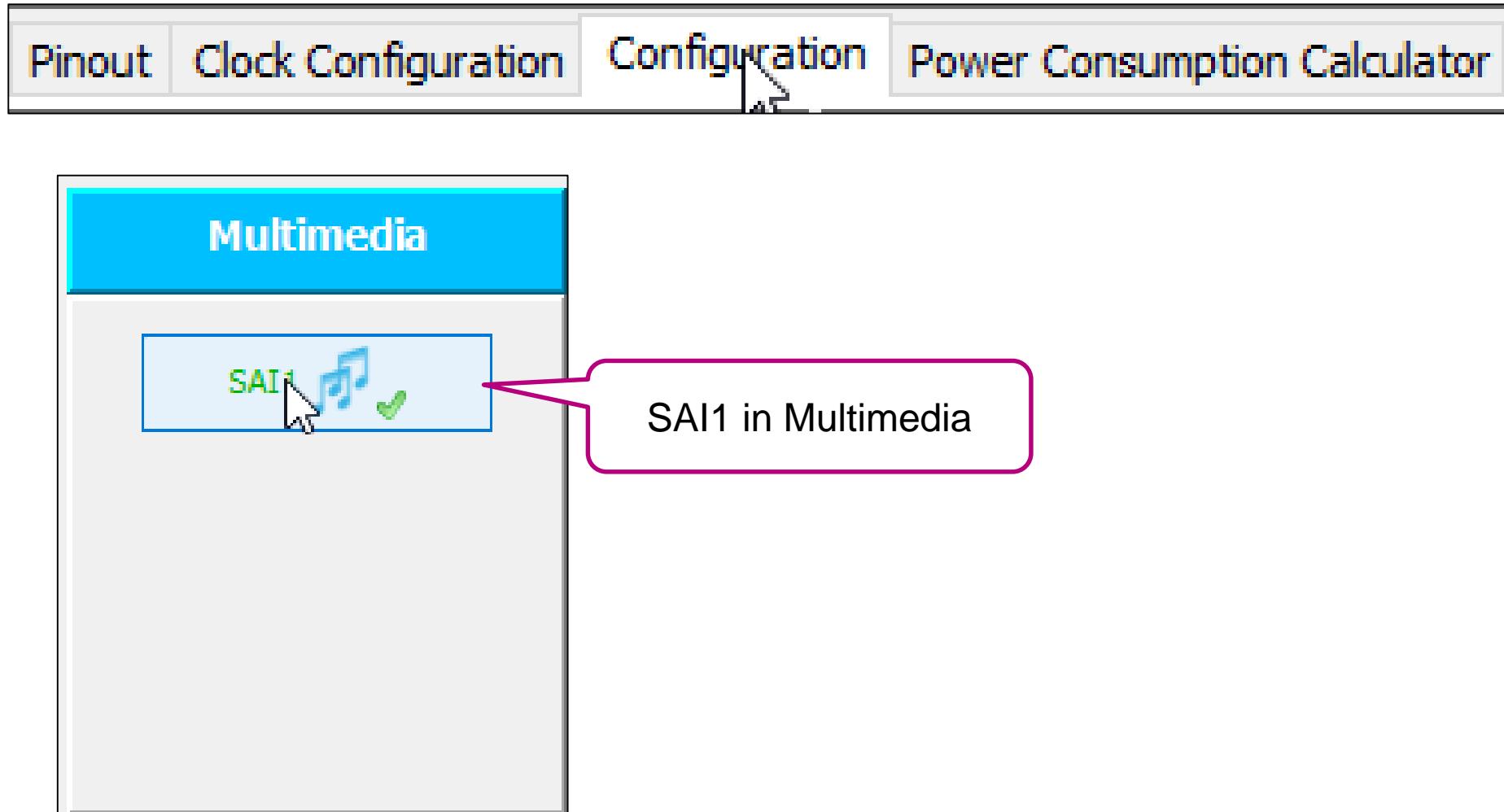
CS43L22

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





The screenshot shows the STM32CubeMX software interface. The top navigation bar includes tabs for Pinout, Clock Configuration, Configuration (which is currently selected), and Power Consumption Calculator. Below the navigation bar, a 'Multimedia' window is open, displaying a SAI1 configuration icon with musical notes. A large green arrow points from this window to a detailed configuration dialog box titled 'SAI1 Configuration'. This dialog box contains several tabs: Parameter Settings, User Constants, NVIC Settings, DMA Settings, and GPIO Settings. The 'Parameter Settings' tab is active, showing configuration parameters for SAI A, such as:

SAI A	
Parameter	Value
Protocol	Free
Audio Mode	Master Transmit
Frame Length	8 bits
Data Size	24 Bits
Slot Size	DataService
Output Mode	Stereo
Companding Mode	No companding mode
SAI SD Line Output Mode	Driven
Frame Parameters	
First Bit	MSB First
Frame Synchro Active Level Length	1
Frame Synchro Definition	Start Frame
Frame Synchro Polarity	Active Low
Frame Synchro Offset	First Bit
Slot Parameters	
First Bit Offset	0
Number of Slots	1
Slot Active Final Value	0x00000000
Slot Active	Neither
Clock Parameters	
Master Clock Divider	Enabled
Audio Frequency	192 KHz
Real Audio Frequency	107.142 KHz
Error between Selected	-44.19 %
Clock Strobing	Falling Edge
Advanced Parameters	
Fifo Threshold	Empty
Output Drive	Disabled
Synchronization External	Disabled

At the bottom of the dialog box are buttons for Apply, Ok, and Cancel.

SAI1 Configuration

Configure the below parameters :

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	<input checked="" type="checkbox"/>
Slot 1 Active	<input checked="" type="checkbox"/>
Audio Frequency	44.1 KHz
Clock Strobing	Rising Edge
Fifo Threshold	One Quarter Full
Output Drive	Enabled

Slot 0 Slot 1

16b sample 16b sample

Left channel + Right channel = STEREO

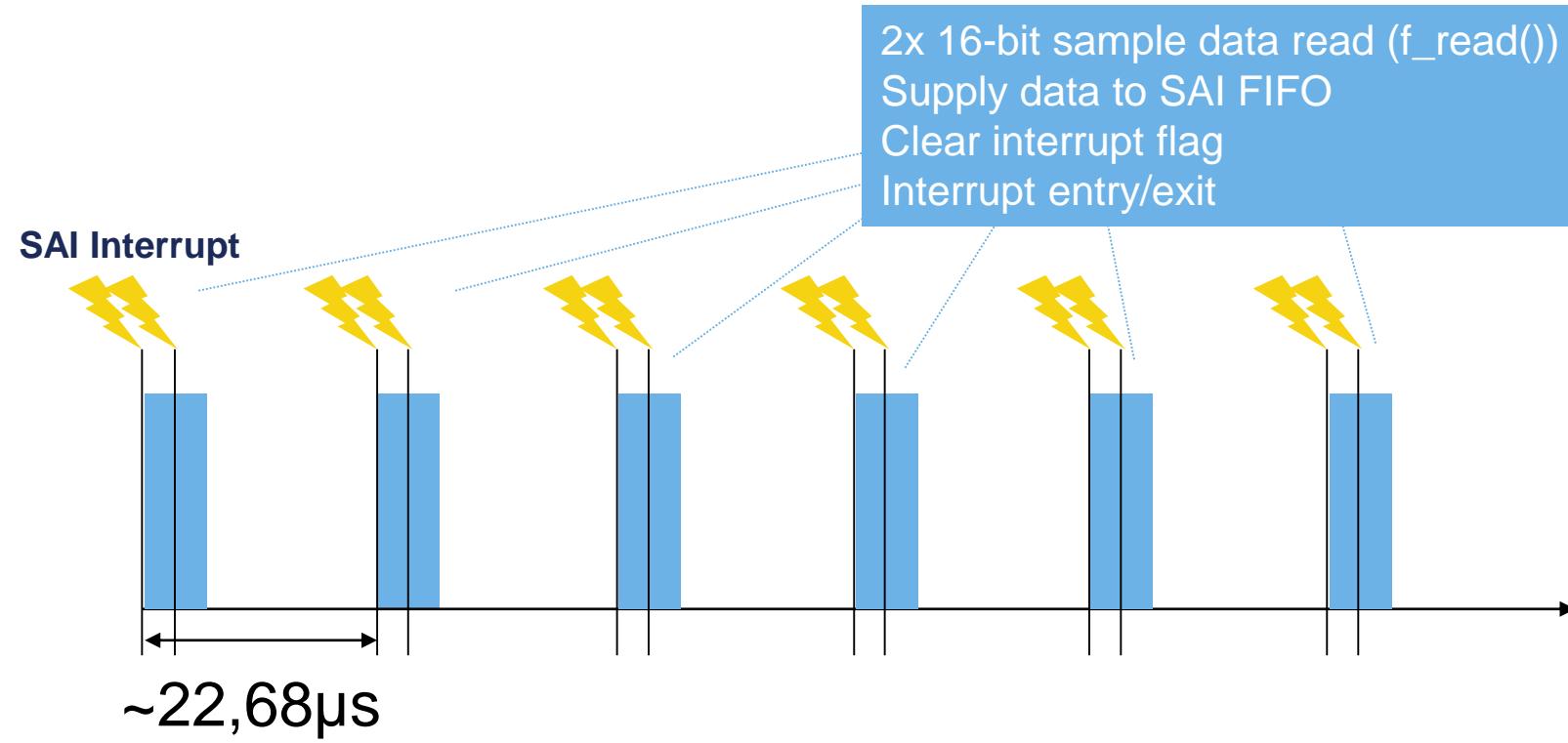
32b frame

audio.wav

WAVE Information

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

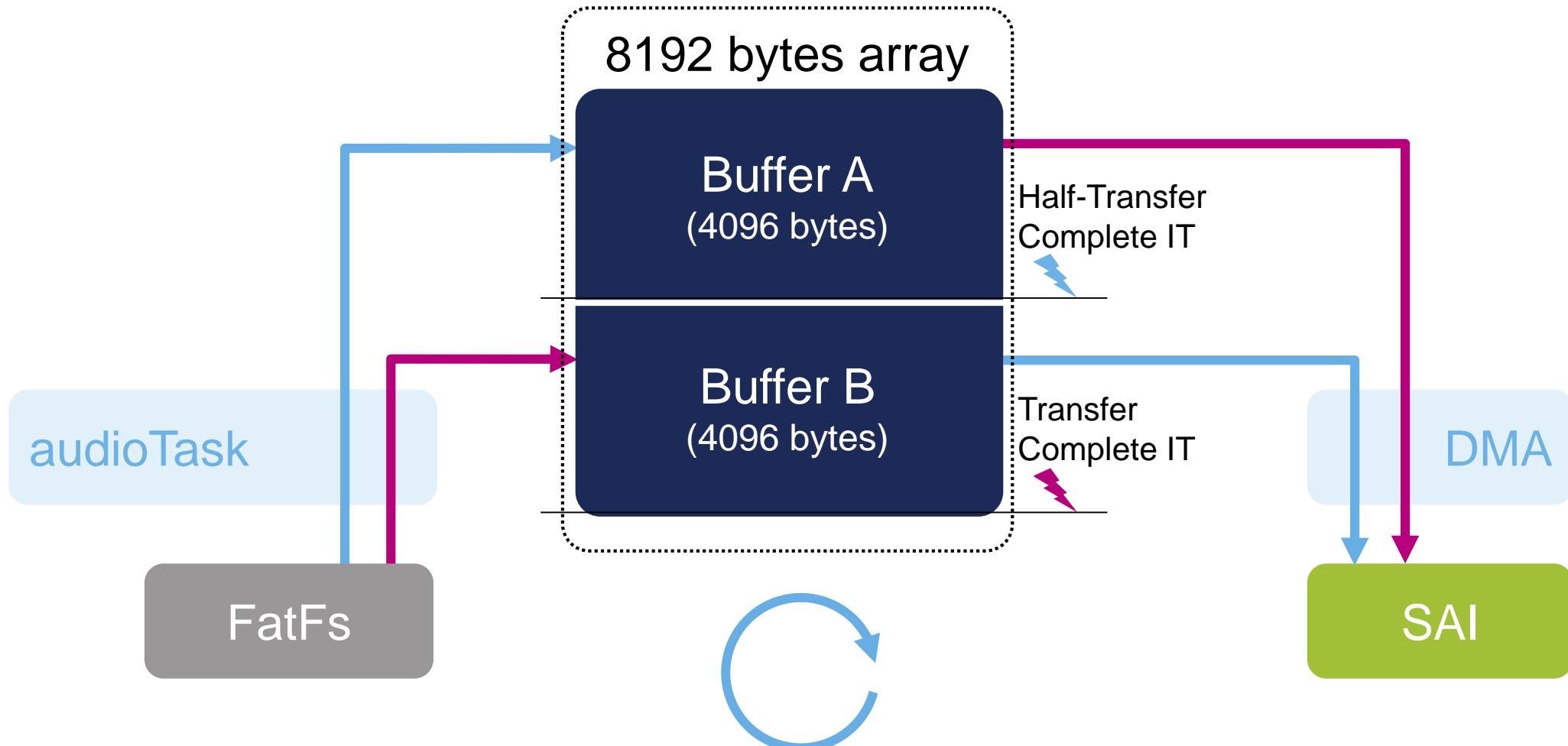
audio.wav	
WAVE Information	
Length	00:58
Compression	PCM
Bitrate (bps)	16
Frequency (Hz)	44 100
Mode	Stereo



~ 1814 clk cycles @80 MHz \rightarrow far enough for the task

But, is it “best practices” implementation of such task?

NO!



2  **Switch to SAI1 DMA Settings**

DMA Request → SAI1_A

Channel → DMA2 Channel 1

Direction → Memory To Peripheral

Priority → High

3  **Mode → Circular**

4  **Increment Address → Memory (only)**

5  **Data Width → Half Word (both)**

DMA Request	Channel	Direction	Priority
SAI1_A	DMA2 Channel 1	Memory To Peripheral	High

DMA Request Settings

Mode: Circular

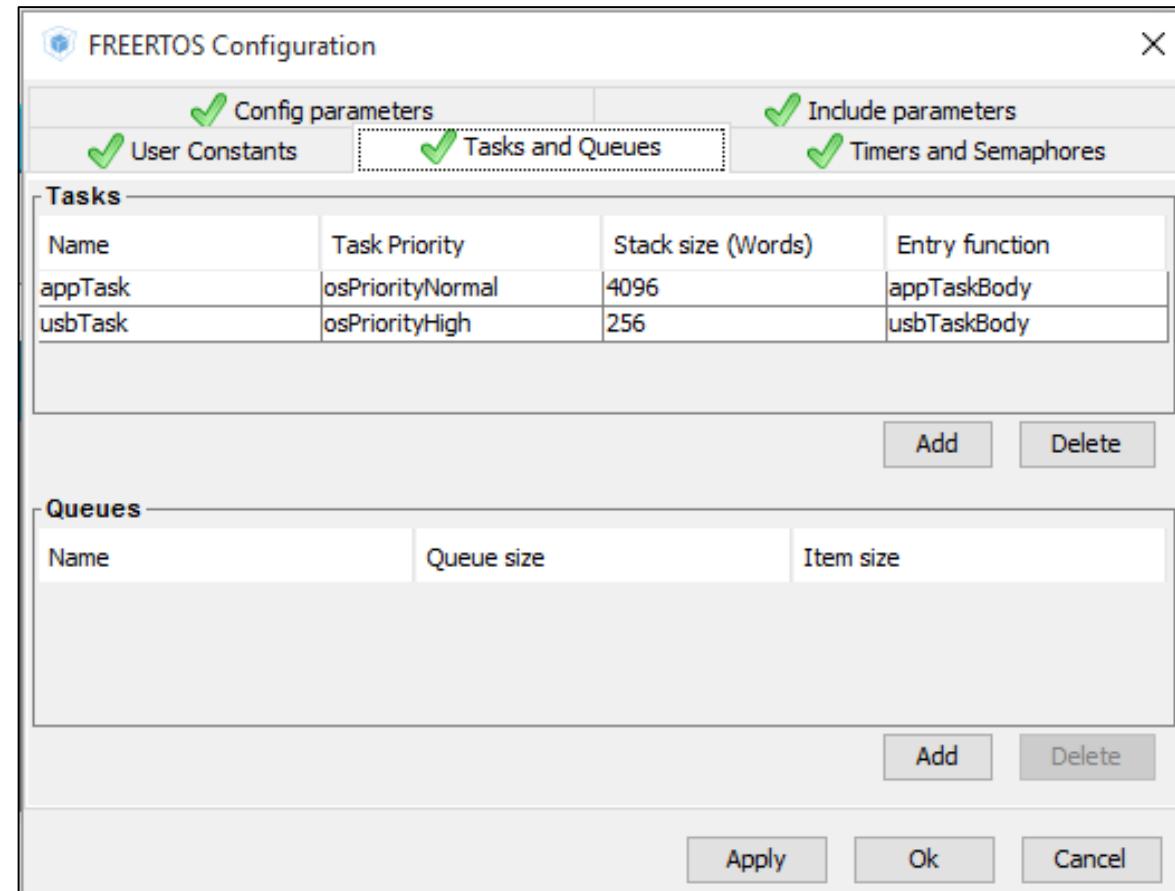
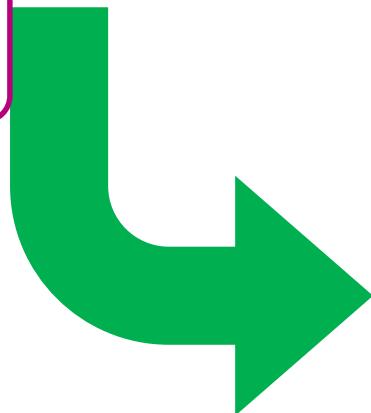
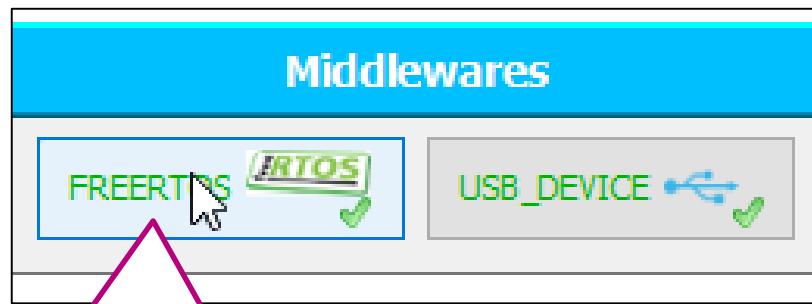
Increment Address:

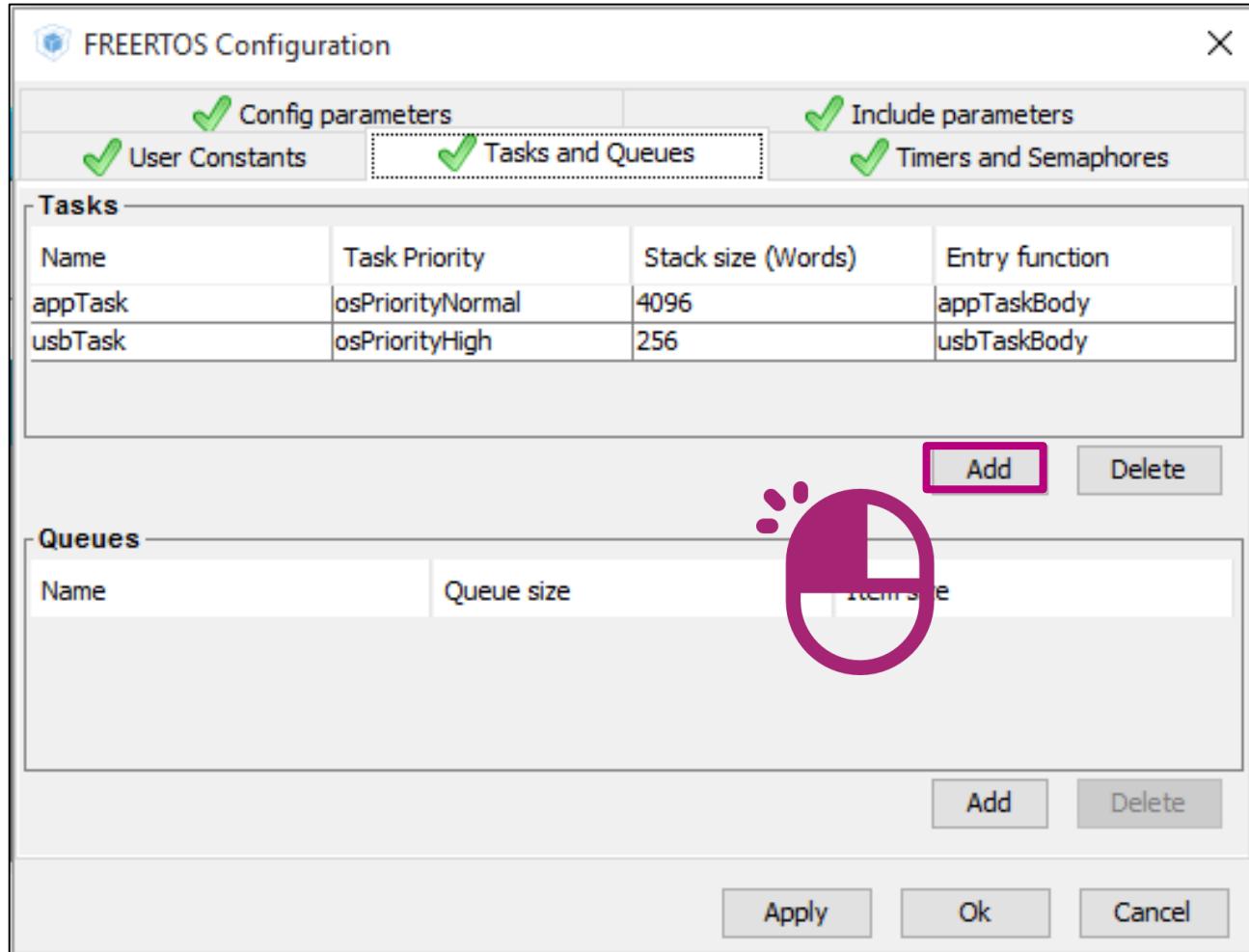
Memory:

Data Width: Half Word

4 FreeRTOS Configuration Change

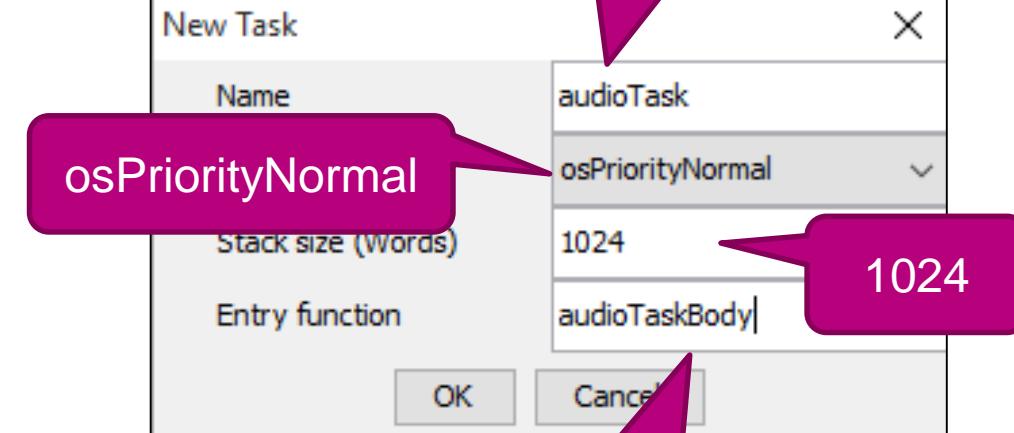
125





audioTask

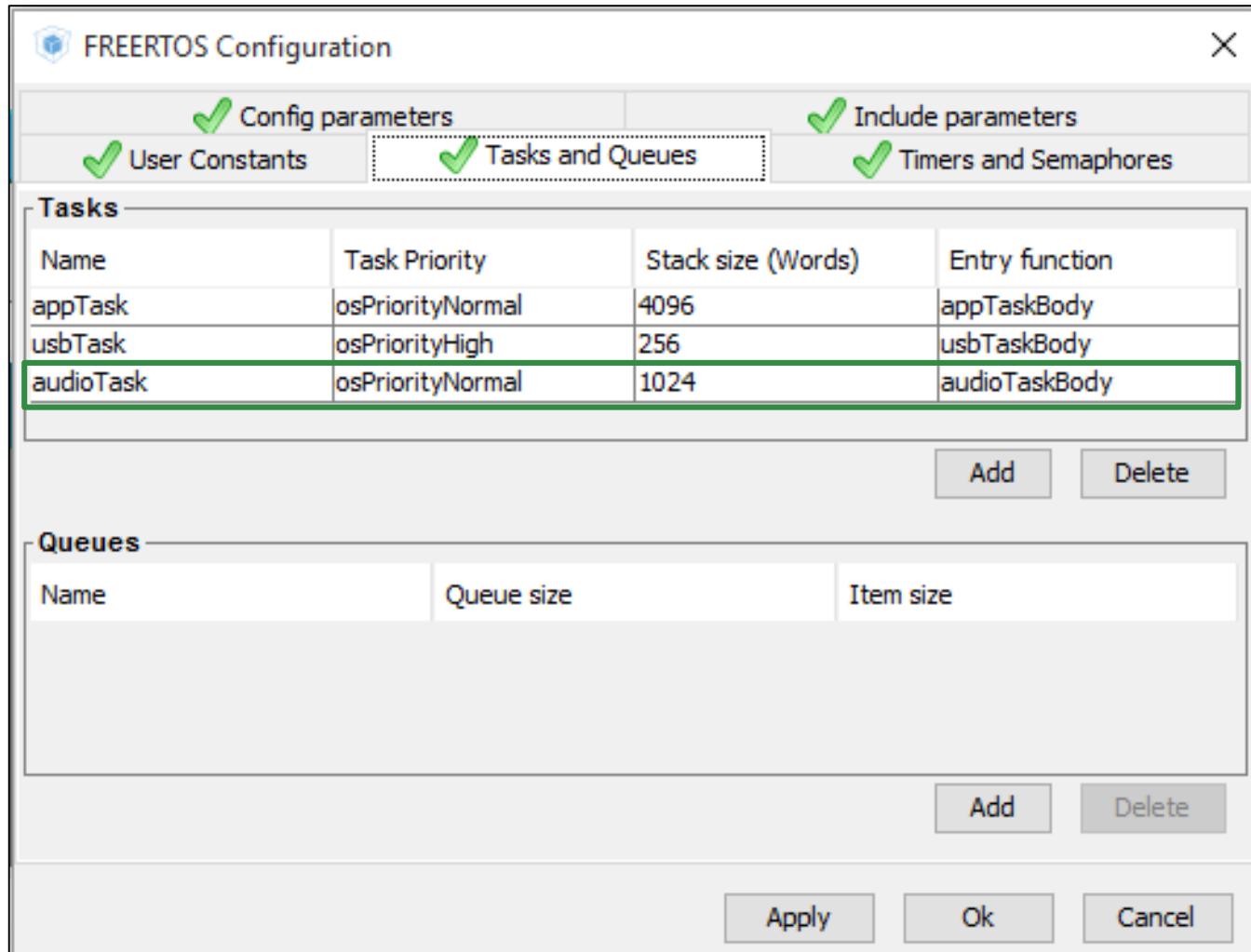
audioTask

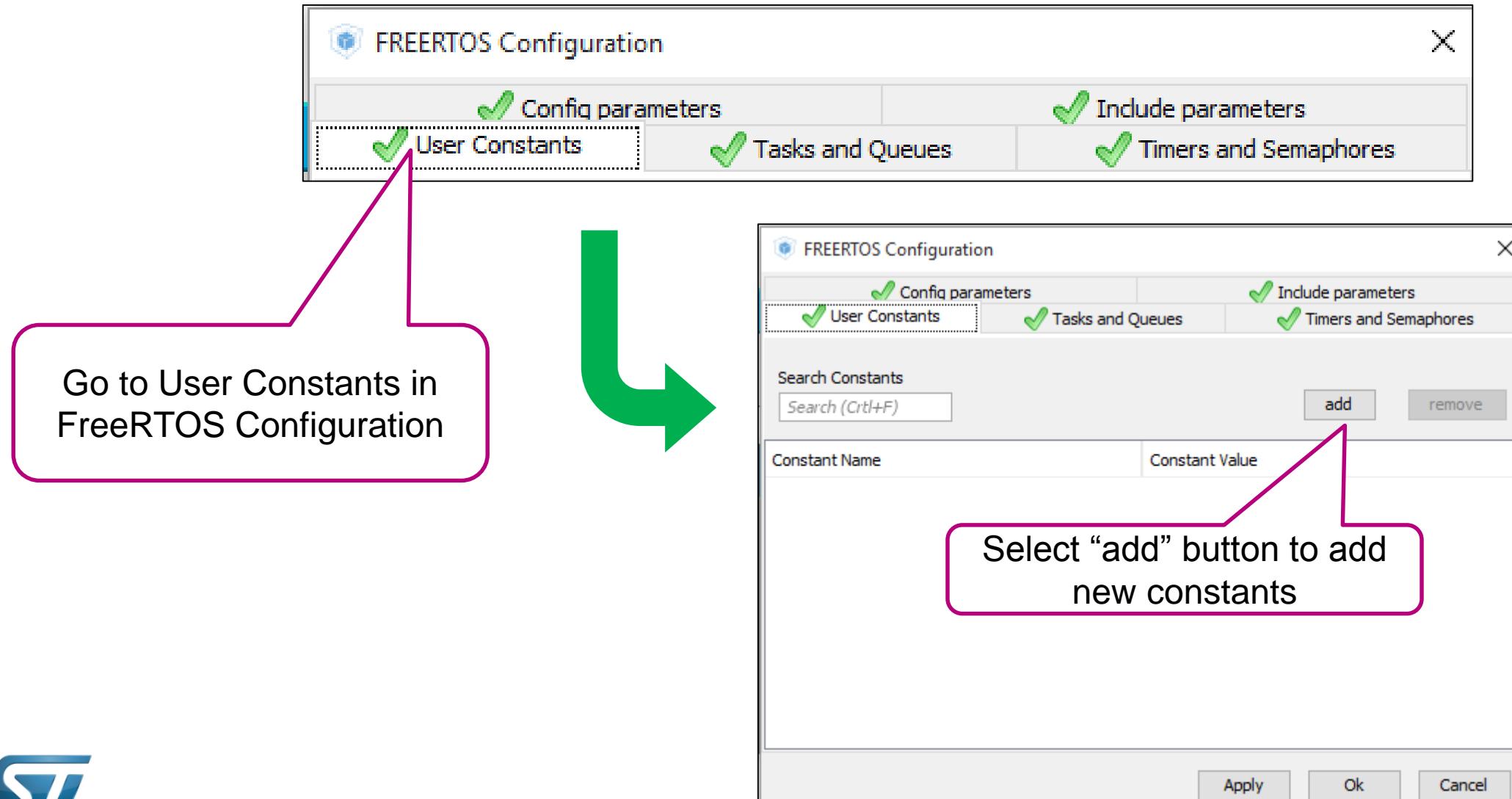


osPriorityNormal

1024

audioTaskBody





A

User Constants	
constant Name	A
constant Value	0x1

0x1

OK Cancel

B

User Constants	
constant Name	B
constant Value	0x2

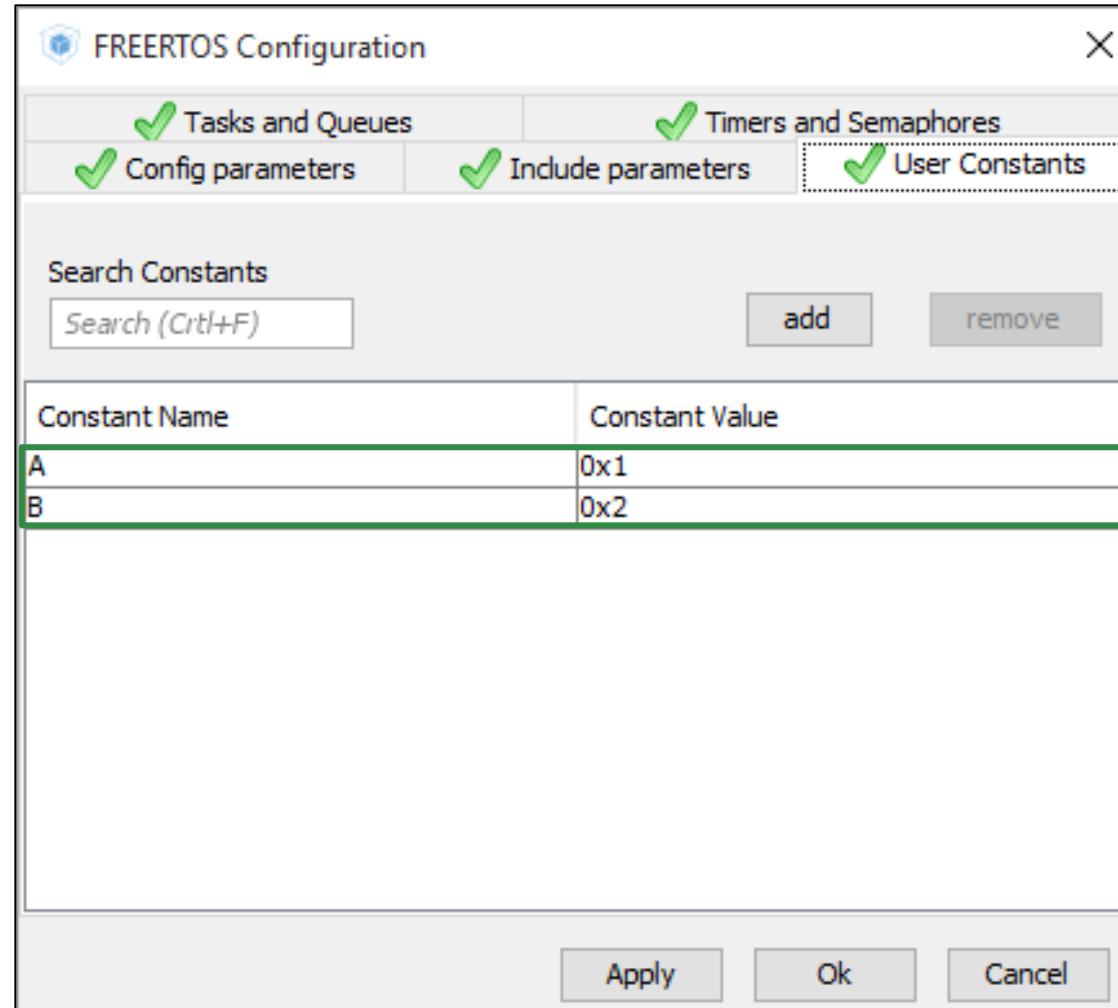
0x2

OK Cancel

4

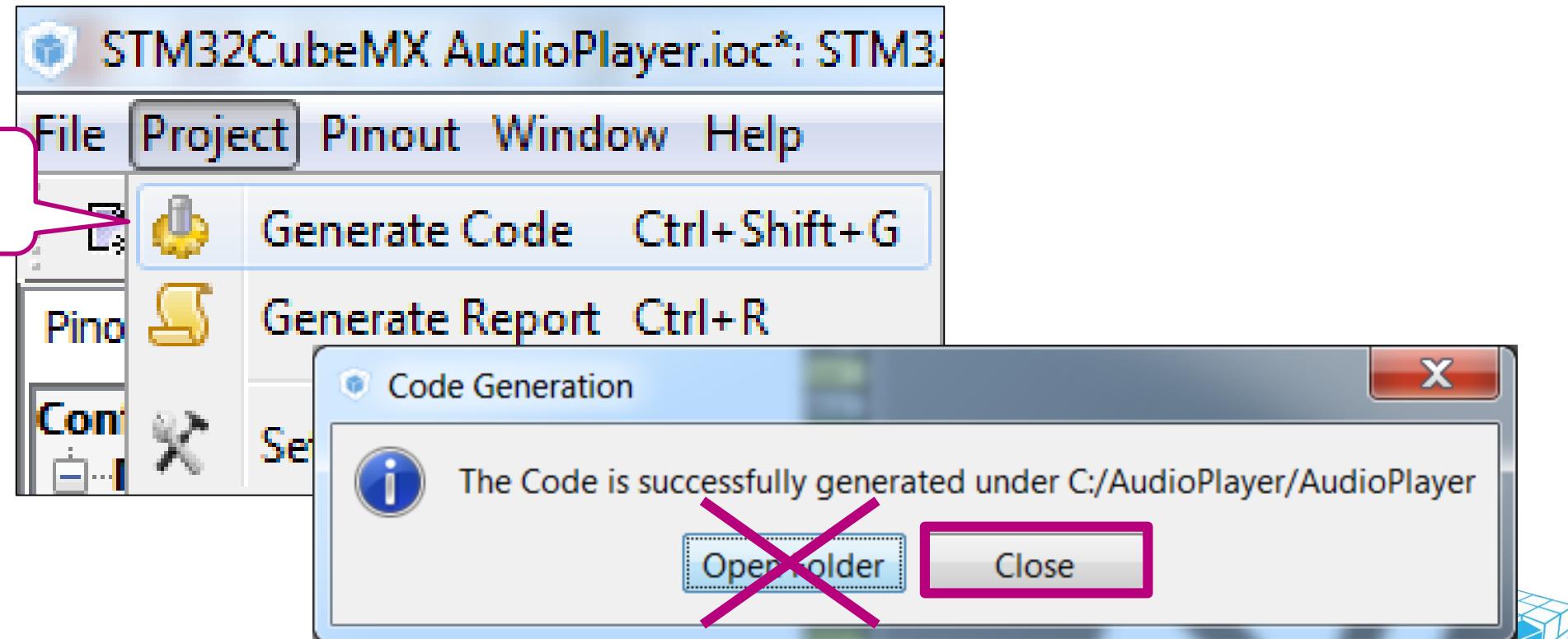
Checkpoint #12

130



Save and generate the project

Do not open the project yet



- 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)

- i2c.c

AUDIO_Init(...)

AUDIO_DelInit(void)

AUDIO_ReadID(...)

AUDIO_Play(...)

AUDIO_Pause(...)

AUDIO_Resume(...)

AUDIO_Stop(...)

AUDIO_SetVolume(...)

AUDIO_SetFrequency(...)

AUDIO_SetMute(...)

AUDIO_SetOutputMode(...)

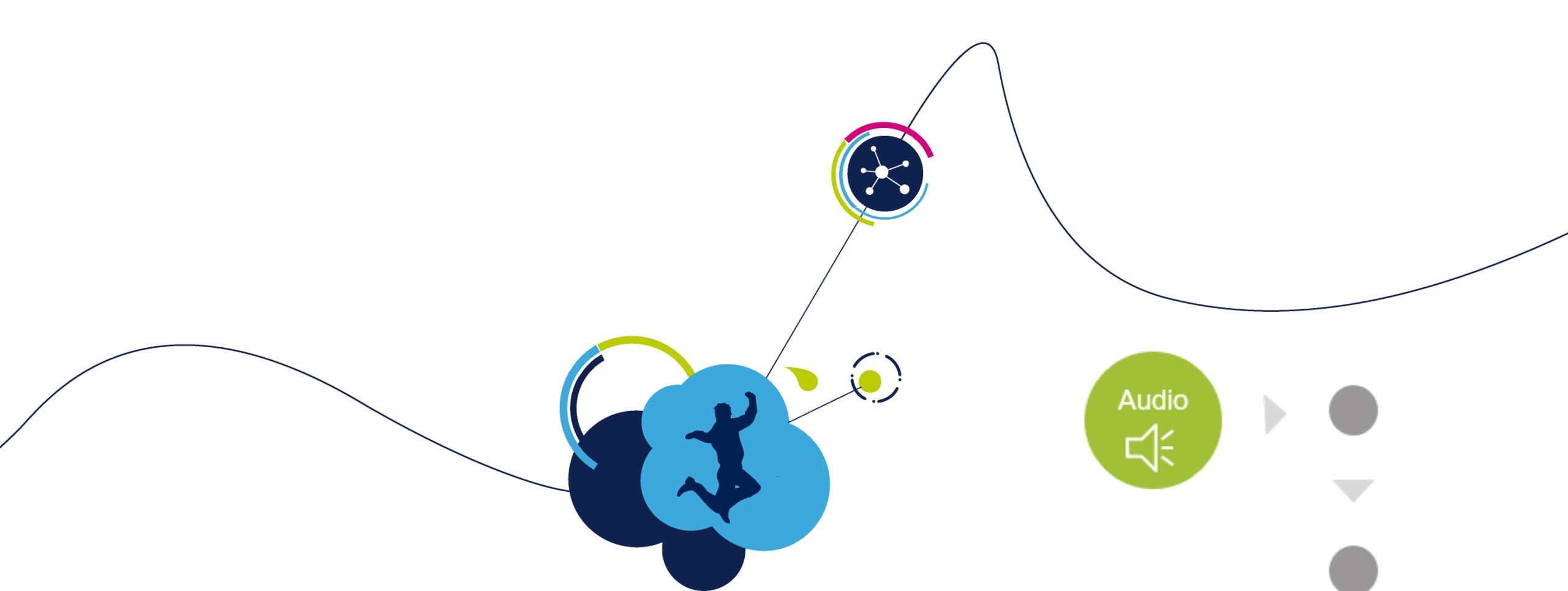
1. Open the project
2. Compile and download
3. Press the reset button to restart the application

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



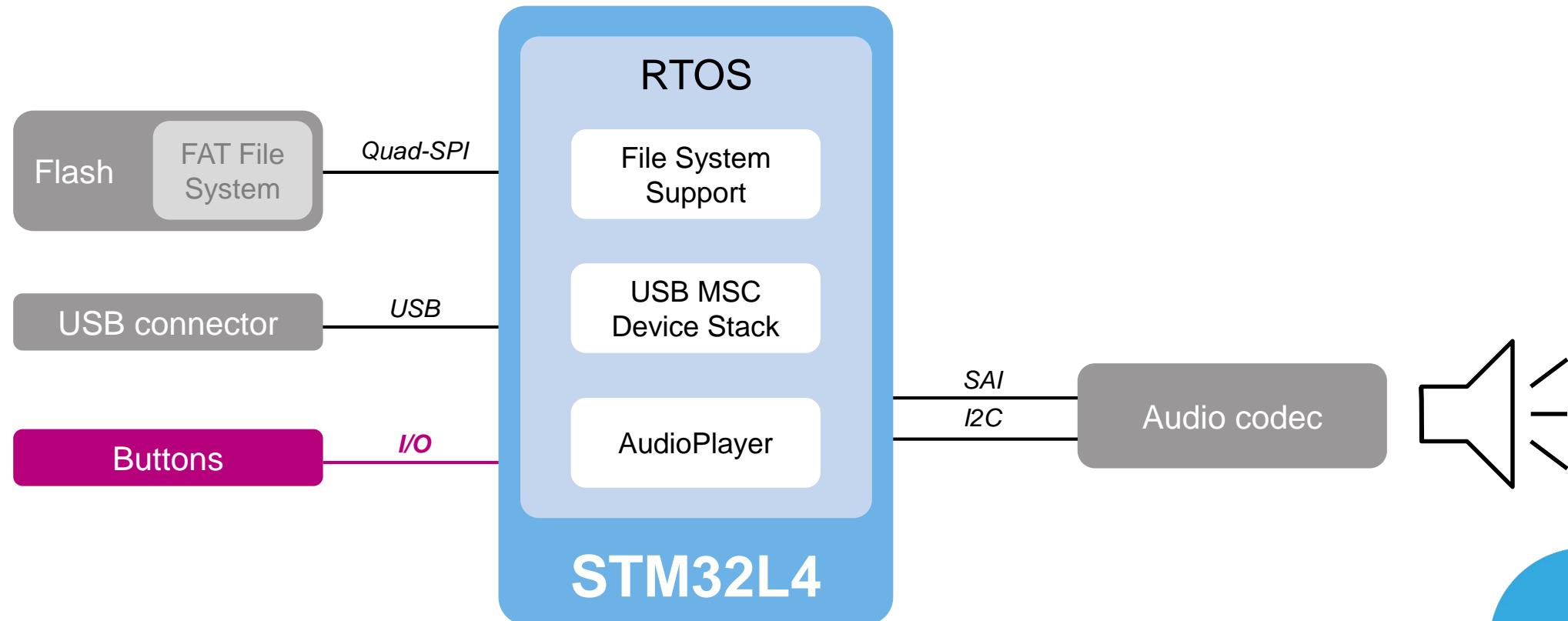
Synchronization of the dual buffer filling mechanism:

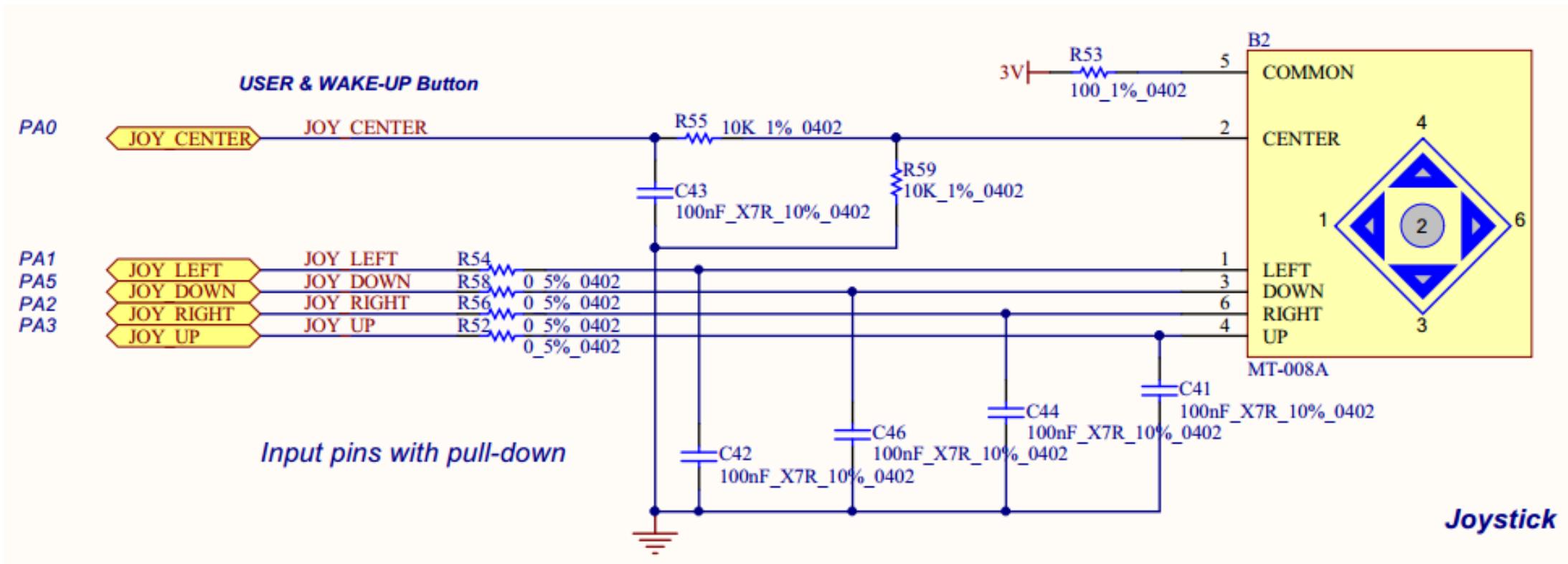
- if both at the same time → error, we are not fast enough maybe
- “Low cost” => fast management by Scheduler, 0 memory usage



STEP5 – Joystick Input

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

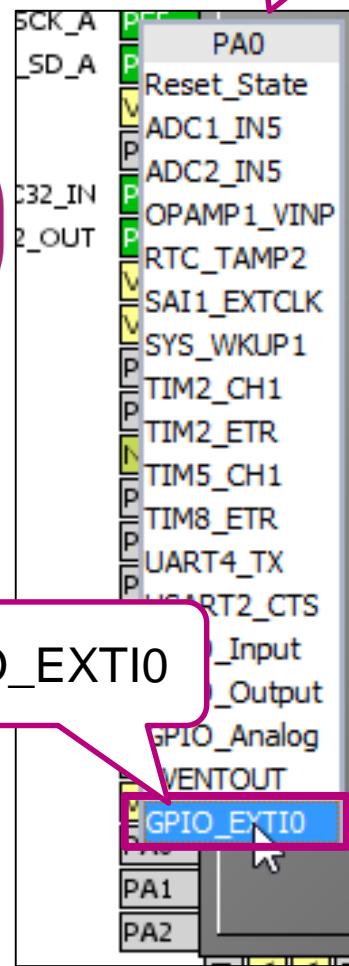




5

PA0

1



2

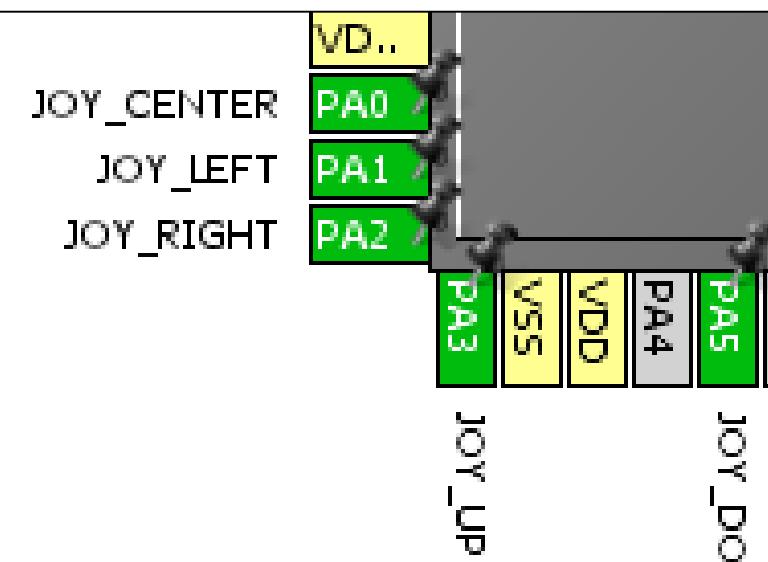


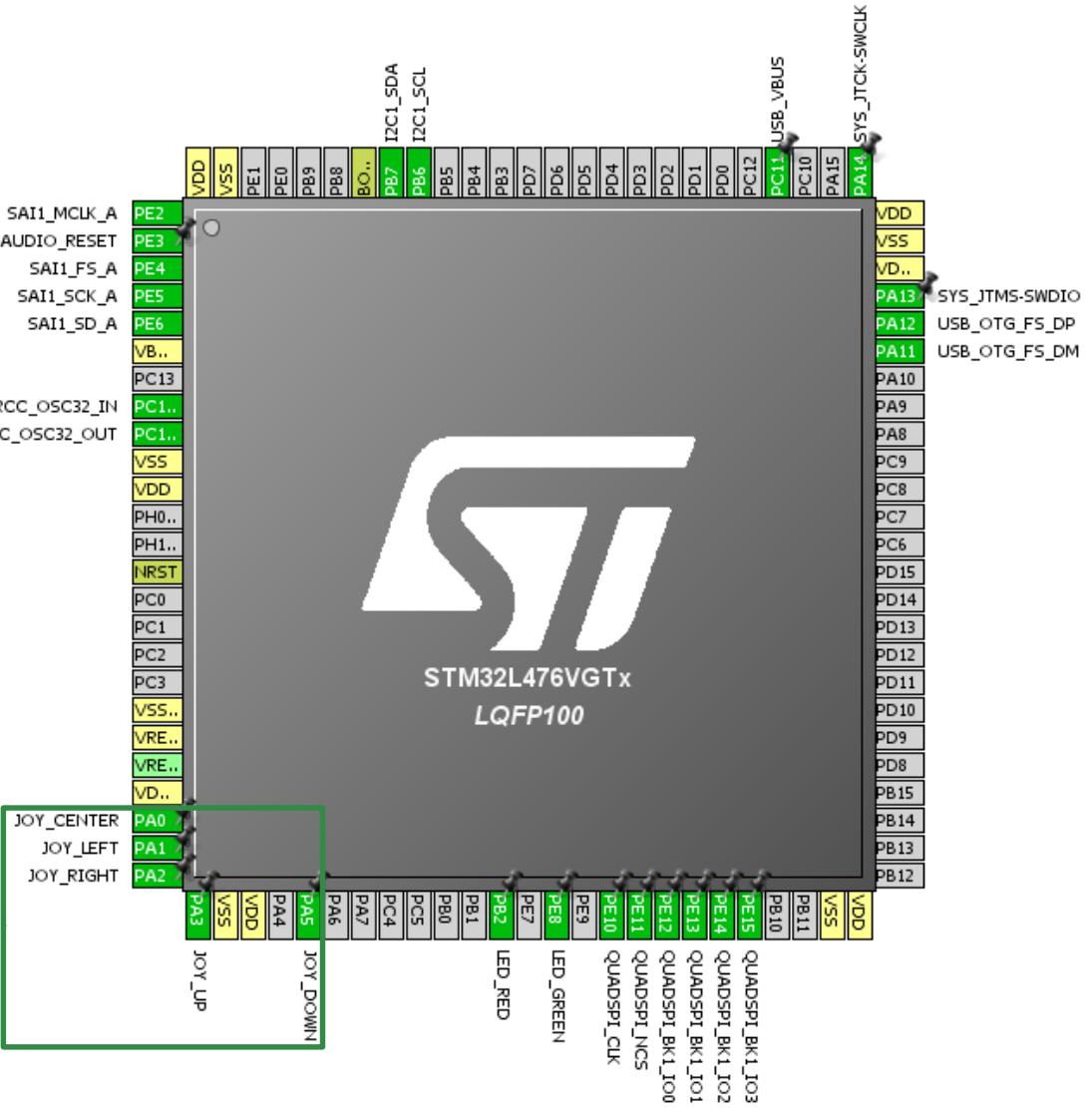
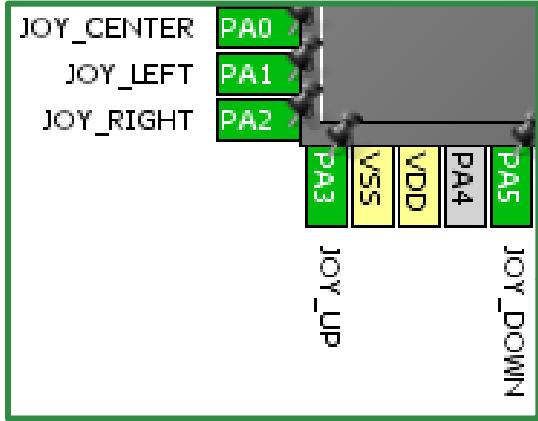
PA0 → JOY_CENTER
 PA1 → JOY_LEFT
 PA2 → JOY_RIGHT
 PA3 → JOY_UP
 PA5 → JOY_DOWN

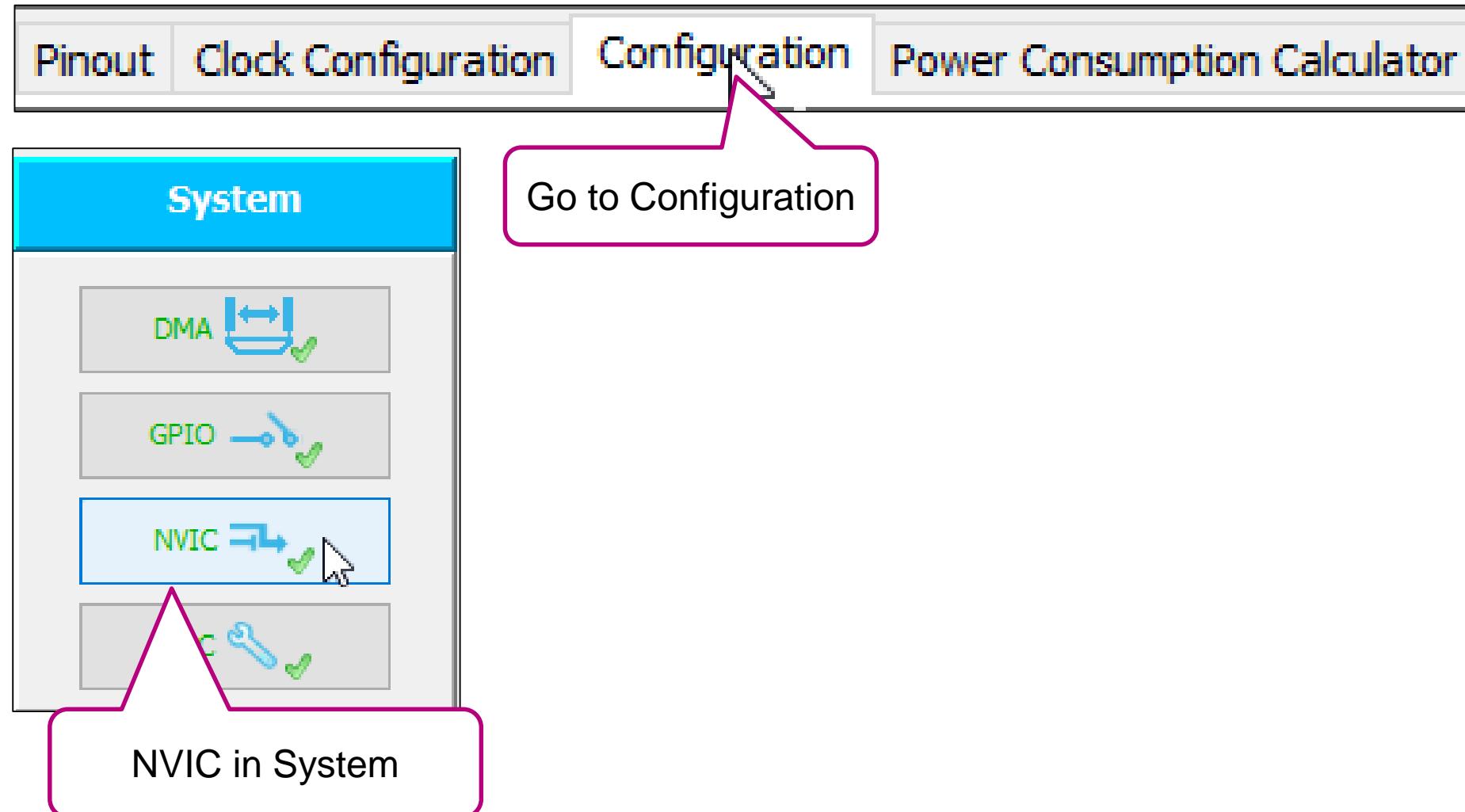
3



Enter JOY_CENTER







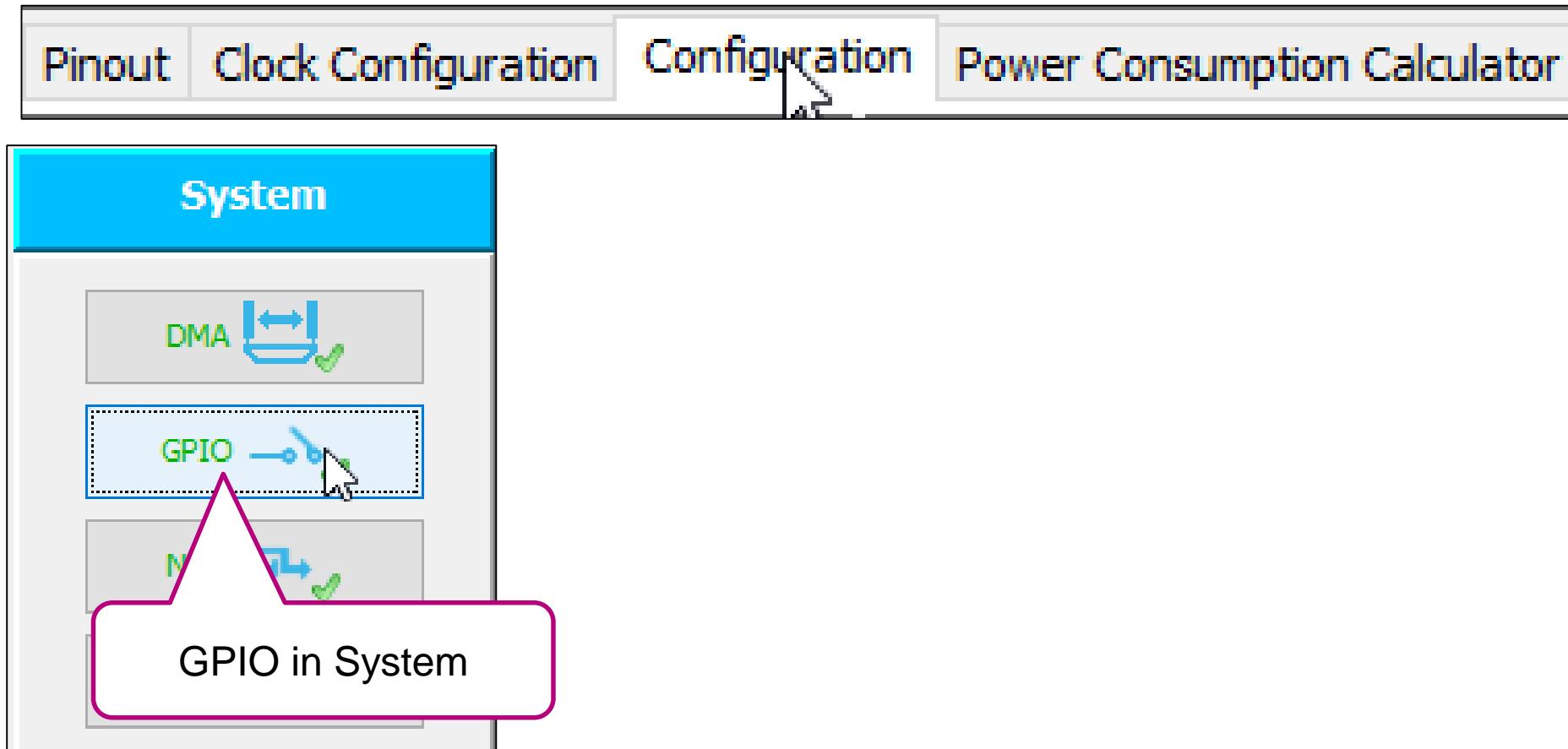
The image shows the STM32CubeMX software interface. At the top, there are tabs: Pinout, Clock Configuration, Configuration (which is selected), Power Consumption Calculator, and a fifth tab that is partially visible. Below the tabs, the main window has a blue header bar labeled "System". Inside the "System" window, there are four icons: DMA (with a double-headed arrow icon), GPIO (with a pin icon), NVIC (with a gear and checkmark icon), and RCC (with a wrench icon). A green arrow points from the NVIC icon in the System window to a detailed configuration dialog box titled "NVIC Configuration". This dialog box contains a table with columns for Interrupt Table, Enabled, Preemption Priority, and Sub Priority. The table lists various interrupt types, each with a checkbox for enabling it and numerical values for priority levels.

Interrupt Table	Enabled	Preemption Priority	Sub Priority
Non maskable interrupt	<input type="checkbox"/>	0	0
Memory management fault	<input type="checkbox"/>	0	0
Prefetch fault, memory access fault	<input type="checkbox"/>	0	0
Undefined instruction or illegal state	<input type="checkbox"/>	0	0
Debug monitor	<input type="checkbox"/>	0	0
System tick timer	<input checked="" type="checkbox"/>	0	0
PVD/PVM1/PVM2/PVM3/PVM4 interrupts through EXTI lines 16/35/36/3...	<input type="checkbox"/>	5	0
Flash global interrupt	<input type="checkbox"/>	5	0
RCC global interrupt	<input type="checkbox"/>	5	0
EXTI line0 interrupt	<input type="checkbox"/>	5	0
EXTI line1 interrupt	<input type="checkbox"/>	5	0
EXTI line2 interrupt	<input type="checkbox"/>	5	0
EXTI line3 interrupt	<input type="checkbox"/>	5	0
EXTI line[9:5] interrupts	<input type="checkbox"/>	5	0
I2C1 event interrupt	<input checked="" type="checkbox"/>	5	0
I2C1 error interrupt	<input checked="" type="checkbox"/>	5	0
DMA2 channel1 global interrupt	<input checked="" type="checkbox"/>	5	0
USB OTG FS global interrupt	<input checked="" type="checkbox"/>	5	0
QUADSPI global interrupt	<input type="checkbox"/>	5	0
SAI1 global interrupt	<input type="checkbox"/>	5	0

The screenshot shows the NVIC Configuration dialog box from STM32CubeMX. It displays a table of interrupt priorities. A pink box highlights the row for 'EXTI line[9:5] interrupts', which includes five entries: EXTI line0, EXTI line1, EXTI line2, EXTI line3, and EXTI line[9:5]. All these entries have their 'Enabled' checkboxes checked and a Preemption Priority of 5. A callout bubble points to this row with the text: 'EXTI line0 interrupt → Enabled', 'EXTI line1 interrupt → Enabled', 'EXTI line2 interrupt → Enabled', 'EXTI line3 interrupt → Enabled', and 'EXTI line[9:5] interrupts → Enabled'.

Interrupt Table	Enabled	Preemption Priority	Sub Priority
Non maskable interrupt	<input type="checkbox"/>	0	0
Memory management fault	<input type="checkbox"/>	0	0
Prefetch fault, memory access fault	<input type="checkbox"/>	0	0
Undefined instruction or illegal state	<input type="checkbox"/>	0	0
Debug monitor	<input type="checkbox"/>	0	0
System tick timer	<input checked="" type="checkbox"/>	0	0
PVD/PVM1/PVM2/PVM3/PVM4 interrupts through EXTI lines 16/35/36/3...	<input type="checkbox"/>	5	0
Flash global interrupt	<input type="checkbox"/>	5	0
RCC global interrupt	<input type="checkbox"/>	5	0
EXTI line0 interrupt	<input checked="" type="checkbox"/>	5	0
EXTI line1 interrupt	<input checked="" type="checkbox"/>	5	0
EXTI line2 interrupt	<input checked="" type="checkbox"/>	5	0
EXTI line3 interrupt	<input checked="" type="checkbox"/>	5	0
EXTI line[9:5] interrupts	<input checked="" type="checkbox"/>	5	0
I2C1 event interrupt	<input checked="" type="checkbox"/>	5	0
...	<input checked="" type="checkbox"/>	5	0
	<input checked="" type="checkbox"/>	5	0
	<input checked="" type="checkbox"/>	5	0
	<input type="checkbox"/>	5	0
	<input type="checkbox"/>	5	0

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



The screenshot illustrates the STM32CubeMX software interface. On the left, the 'System' tab is selected, displaying icons for DMA, GPIO, NVIC, and RCC. A green arrow points from the GPIO icon towards the right side of the screen. On the right, a 'Pin Configuration' dialog box is open, showing a table of pins and their configurations. The table includes columns for Pin Name, Signal on Pin, GPIO mode, GPIO Pull-up..., Maximum out..., Fast Mode, User Label, and Modified. The 'GPIO' tab is selected in the dialog header.

Pin Name	Signal on Pin	GPIO mode	GPIO Pull-up...	Maximum out...	Fast Mode	User Label	Modified
PA0	n/a	External Interr...	No pull-up and...	n/a	n/a	JOY_CENTER.	<input checked="" type="checkbox"/>
PA1	n/a	External Interr...	No pull-up and...	n/a	n/a	JOY_LEFT	<input checked="" type="checkbox"/>
PA2	n/a	External Interr...	No pull-up and...	n/a	n/a	JOY_RIGHT	<input checked="" type="checkbox"/>
PA3	n/a	External Interr...	No pull-up and...	n/a	n/a	JOY_UP	<input checked="" type="checkbox"/>
PA5	n/a	External Interr...	No pull-up and...	n/a	n/a	JOY_DOWN	<input checked="" type="checkbox"/>
PB2	n/a	Output Push Pull	No pull-up and...	Low	n/a	LED_RED	<input checked="" type="checkbox"/>
PC11	n/a	Input mode	No pull-up and...	n/a	n/a	USB_VBUS	<input checked="" type="checkbox"/>
PE3	n/a	Output Push Pull	No pull-up and...	Low	n/a	AUDIO_RESET	<input checked="" type="checkbox"/>
PE8	n/a	Output Push Pull	No pull-up and...	Low	n/a	LED_GREEN	<input checked="" type="checkbox"/>

! Select Pins from table to configure them. **Multiple selection is Allowed.**

Group By IP

Apply Ok Cancel

Pin Configuration

GPIO I2C1 QUADSPI RCC SAI1 SYS USB_OTG_FS

Search Signals Search (Ctrl+F) Show only Modified Pins

Pin Name	Signal on Pin	GPIO mode	GPIO Pull-u...	Maximum o...	Fast Mode	User Label	Modified
PA0	n/a	External Int...	Pull-down	n/a	n/a	JOY_CENTER	<input checked="" type="checkbox"/>
PA1	n/a	External Int...	Pull-down	n/a	n/a	JOY_LEFT	<input checked="" type="checkbox"/>
PA2	n/a	External Int...	Pull-down	n/a	n/a	JOY_RIGHT	<input checked="" type="checkbox"/>
PA3	n/a	External Int...	Pull-down	n/a	n/a	JOY_UP	<input checked="" type="checkbox"/>
PA5	n/a	External Int...	Pull-down	n/a	n/a	JOY_DOWN	<input checked="" type="checkbox"/>
PB2	n/a	Output Push...	No pull-up a...	Low	n/a	LED_RED	<input checked="" type="checkbox"/>
			No pull-up a...	n/a	n/a	USB_VBUS	<input checked="" type="checkbox"/>
			No pull-up a...	Low	n/a	AUDIO_RESET	<input checked="" type="checkbox"/>
			No pull-up a...	Low	n/a	LED_GREEN	<input checked="" type="checkbox"/>

GPIO Pull-Up/Pull-Down

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

User Label: JOY_CENTER

Group By IP Apply Ok Cancel

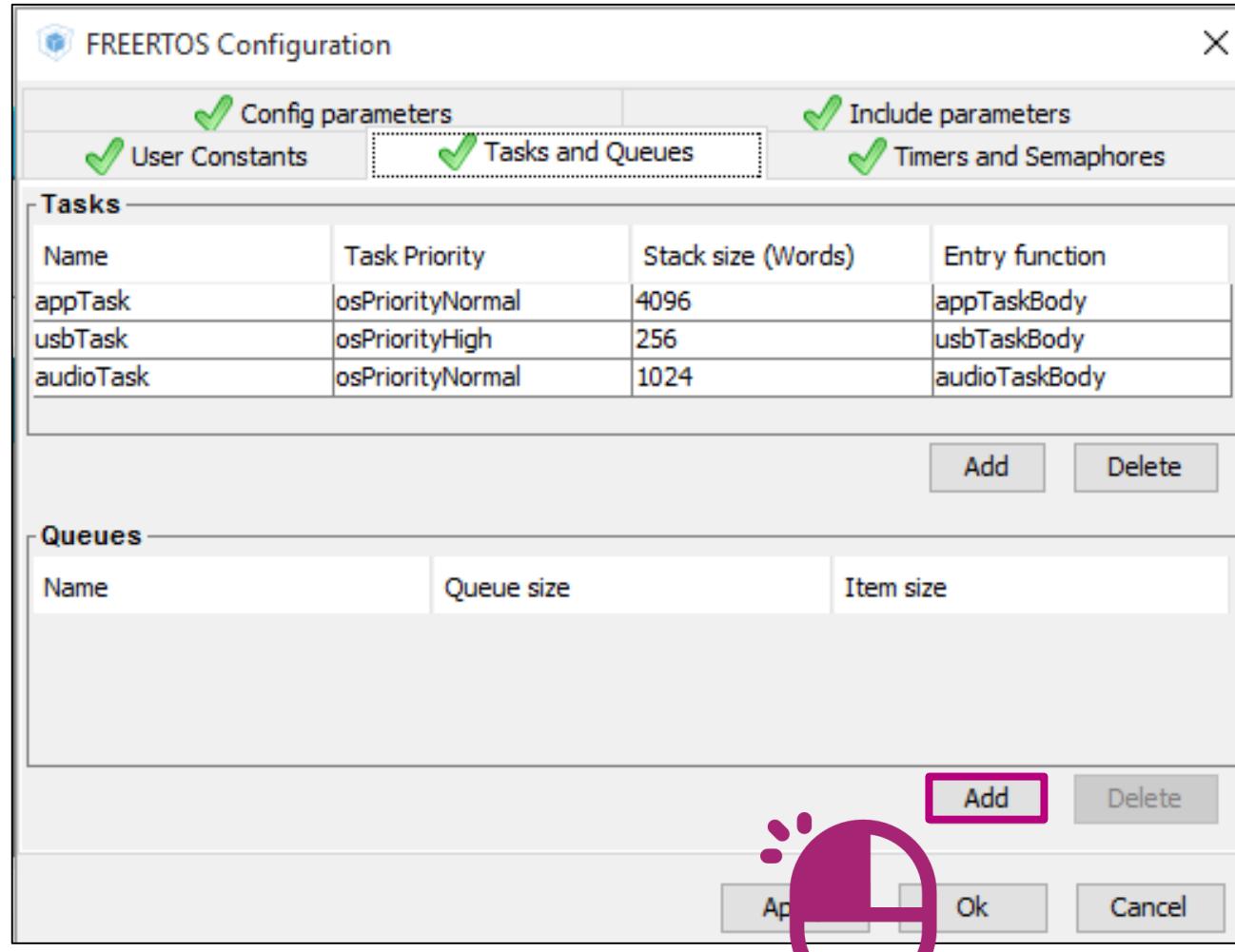
The screenshot shows the STM32CubeMX software interface. At the top, there is a navigation bar with five tabs: Pinout, Clock Configuration, Configuration, Power Consumption Calculator, and another Configuration tab which is currently active. Below the navigation bar is a large window titled "Middlewares". Inside this window, there are two items: "FREERTOS" and "USB_DEVICE", each accompanied by a small icon and a green checkmark. A pink callout arrow points from a box labeled "FREERTOS in Middlewares" towards the "FREERTOS" item.

FREERTOS in Middlewares

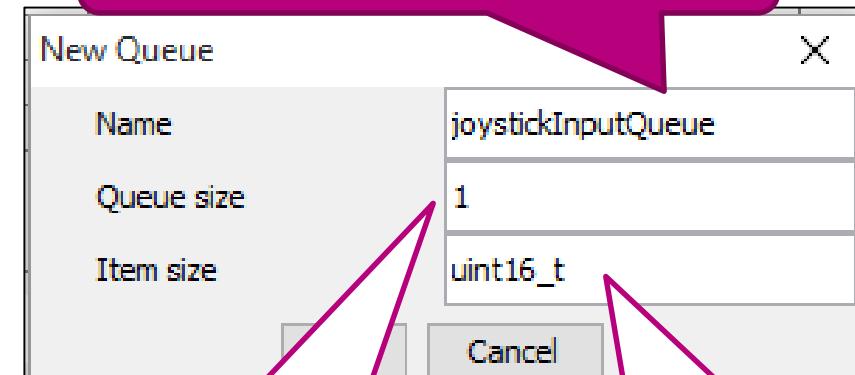
The screenshot shows the STM32CubeMX software interface. At the top, there are tabs: Pinout, Clock Configuration, Configuration (which is selected), Power Consumption Calculator, and others. Below the tabs, there's a 'Middlewares' section with two items: 'FREERTOS' and 'USB_DEVICE'. A large green arrow points from the 'FREERTOS' item towards a detailed configuration dialog box. This dialog box is titled 'FREERTOS Configuration' and contains sections for 'Config parameters', 'User Constants', 'Tasks', and 'Queues'. Under 'Tasks', there are three entries:

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

At the bottom of the dialog box are 'Apply', 'Ok', and 'Cancel' buttons.



Name → joystickInputQueue



Queue size → 1

Item size
→ uint16_t



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

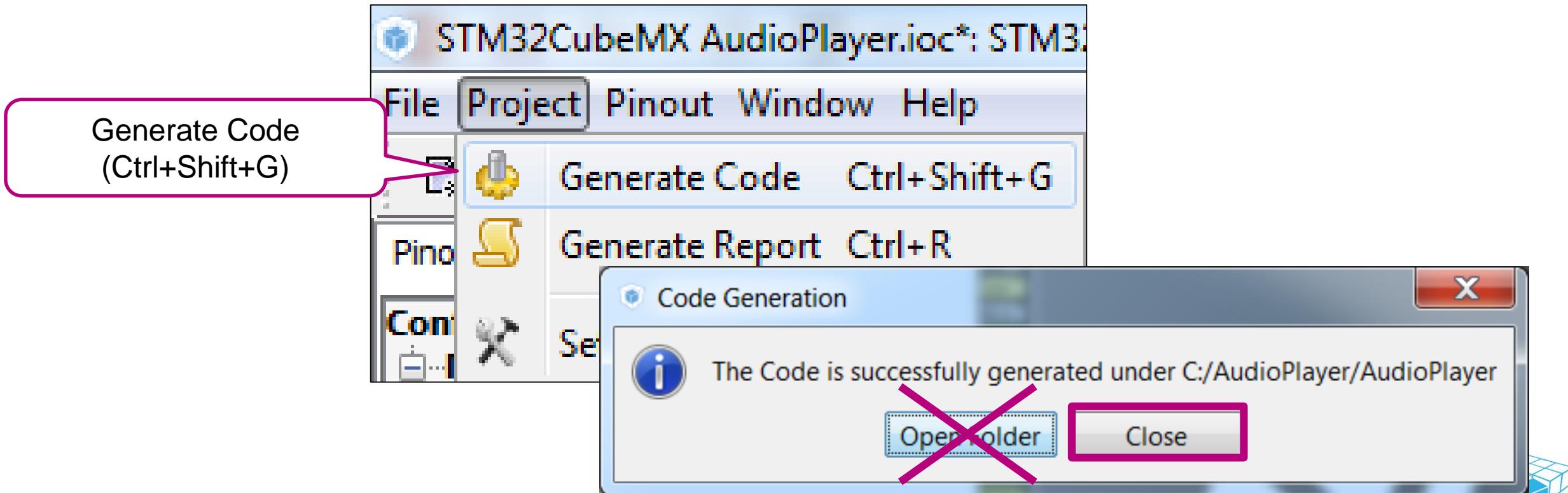
Apply Ok Cancel

This screenshot shows the 'Tasks and Queues' tab of the FreeRTOS Configuration tool. It lists three tasks: appTask, usbTask, and audioTask, each with its priority, stack size, and entry function specified. Below the tasks is a queue named joystickInputQueue with a queue size of 1 and item size of uint16_t. Buttons for adding, deleting, applying, and canceling changes are visible at the bottom.



Save and generate the project

Do not open the project yet



- 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.

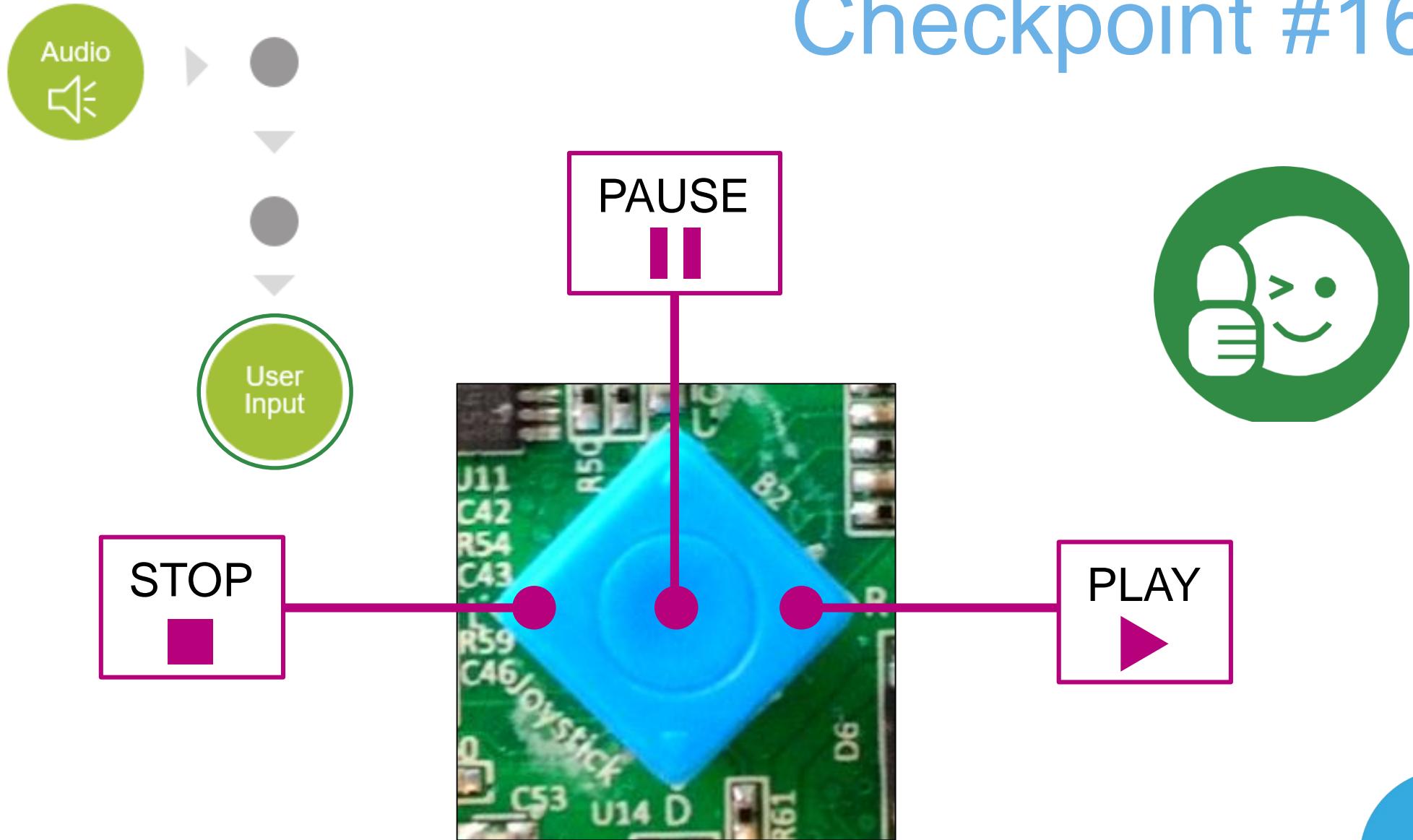
- This operation is STM32CubeMX non-intrusive. You can re-generate the project later and the modifications will be retained.

1. Open the project
2. Compile and download
3. Press the reset button on discovery once download has finished

5

Checkpoint #16

155

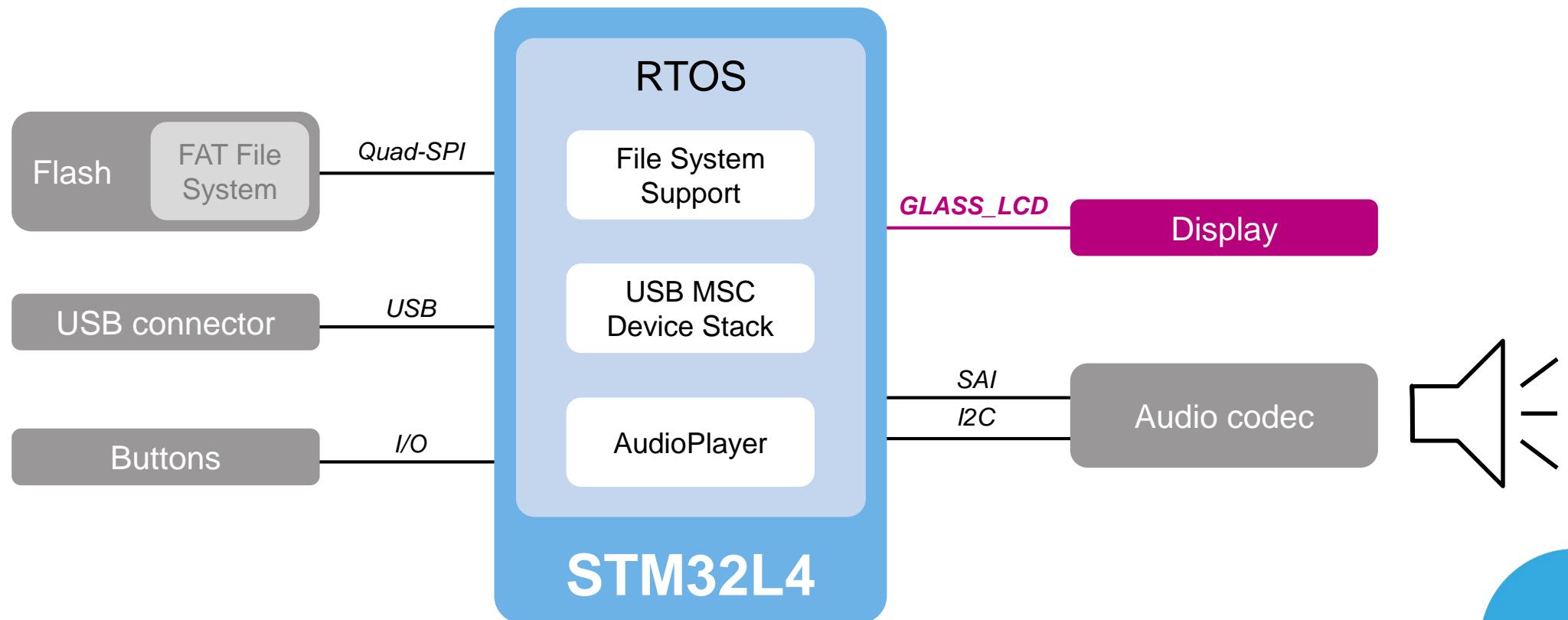


- Because we want to process just one input event at a time
- We are passing the command (queue item) from the EXTI ISR
- Why not signal?
 - Multiple signals can be set at a time

STEP6 – LCD



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

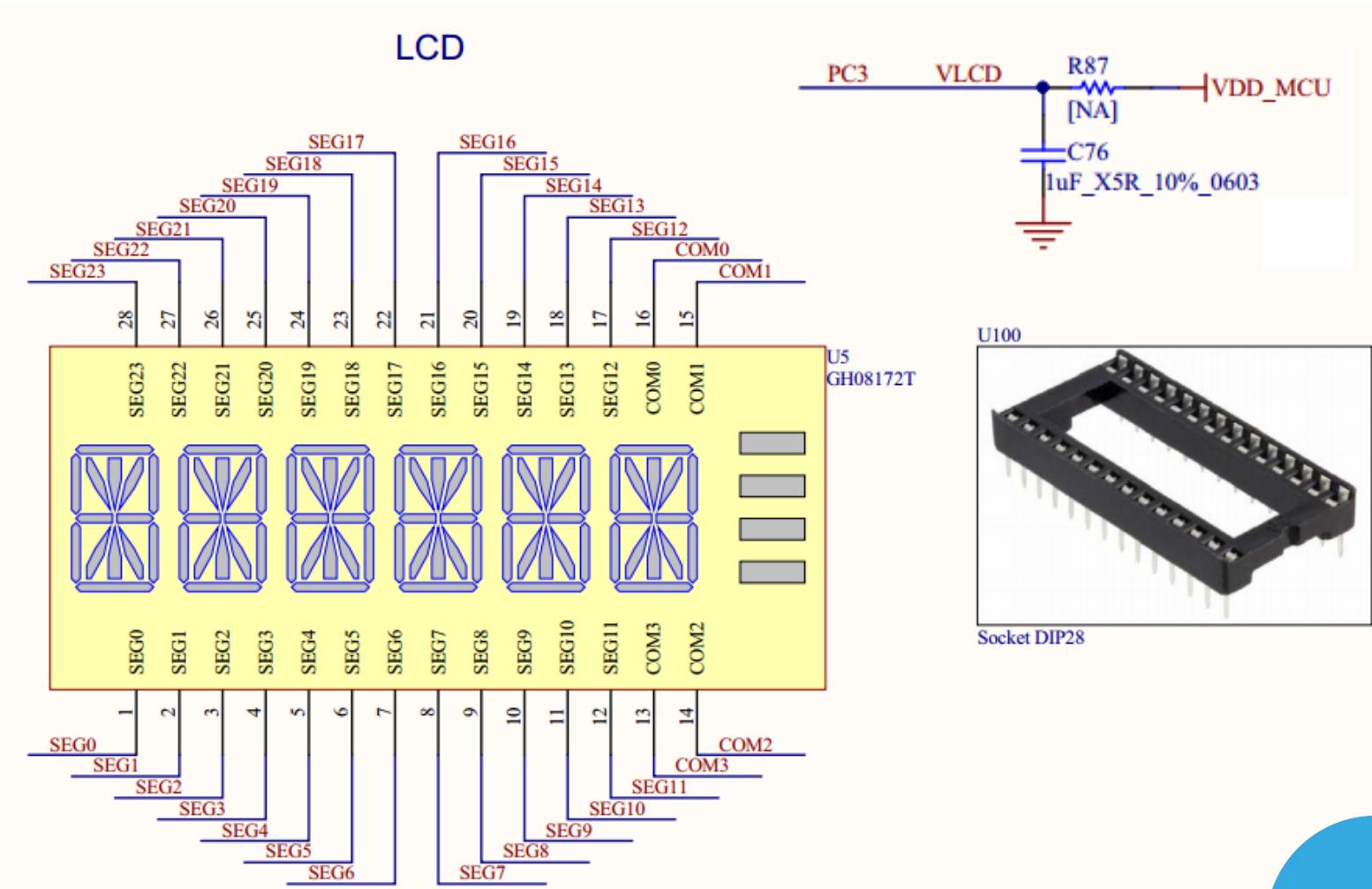


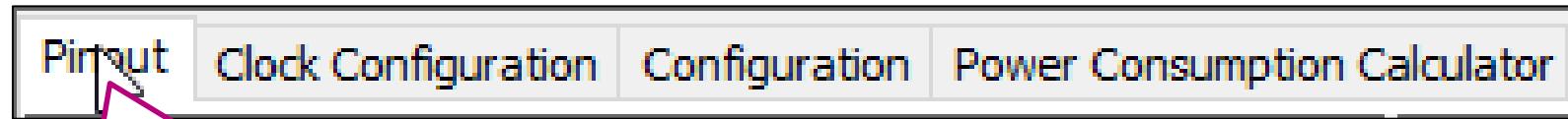


PA8	COM0
PA9	COM1
PA10	COM2
PB9	COM3

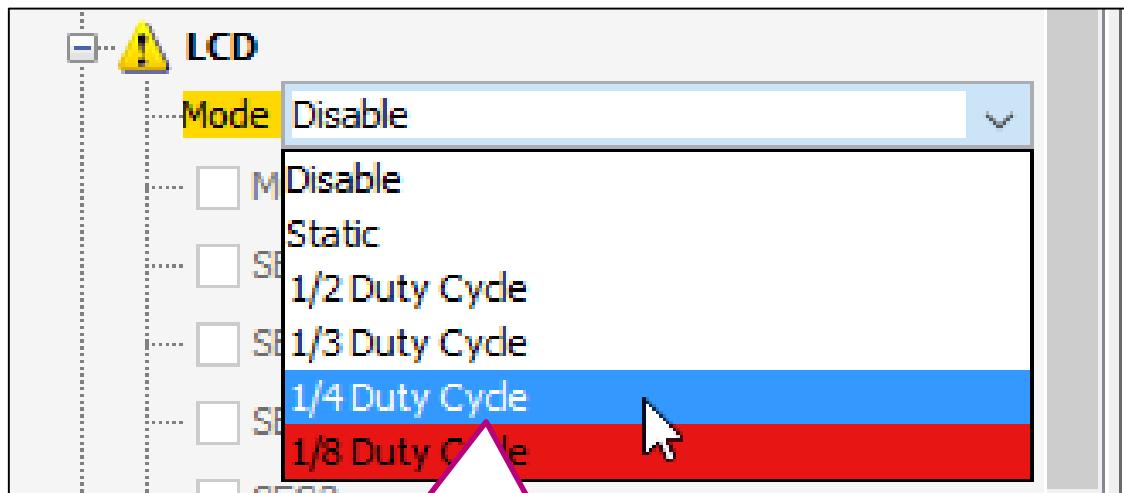
PA7	SEG0
PC5	SEG1
PB1	SEG2
PB13	SEG3
PB15	SEG4
PD9	SEG5
PD11	SEG6
PD13	SEG7
PD15	SEG8
PC7	SEG9
PA15	SEG10
PB4	SEG11
PB5	SEG12

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

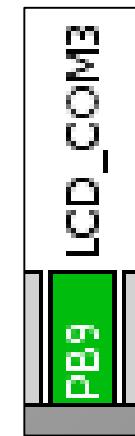
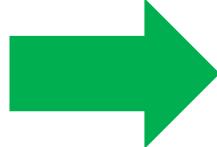
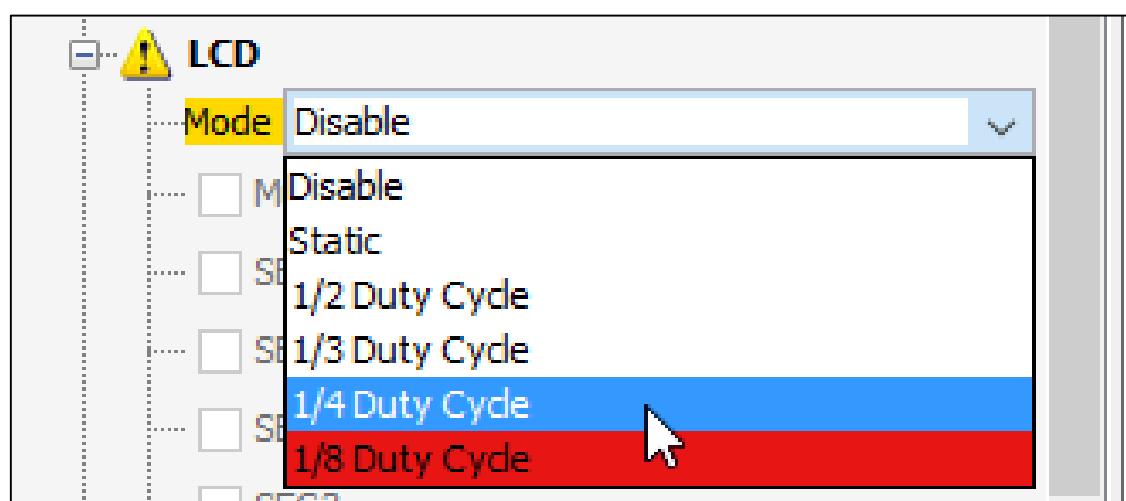
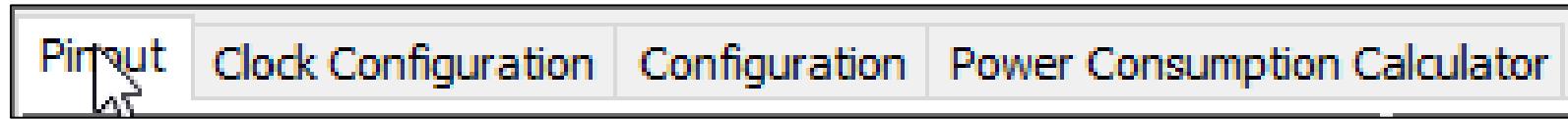




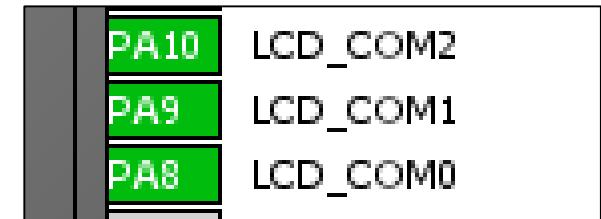
Go to Pinout

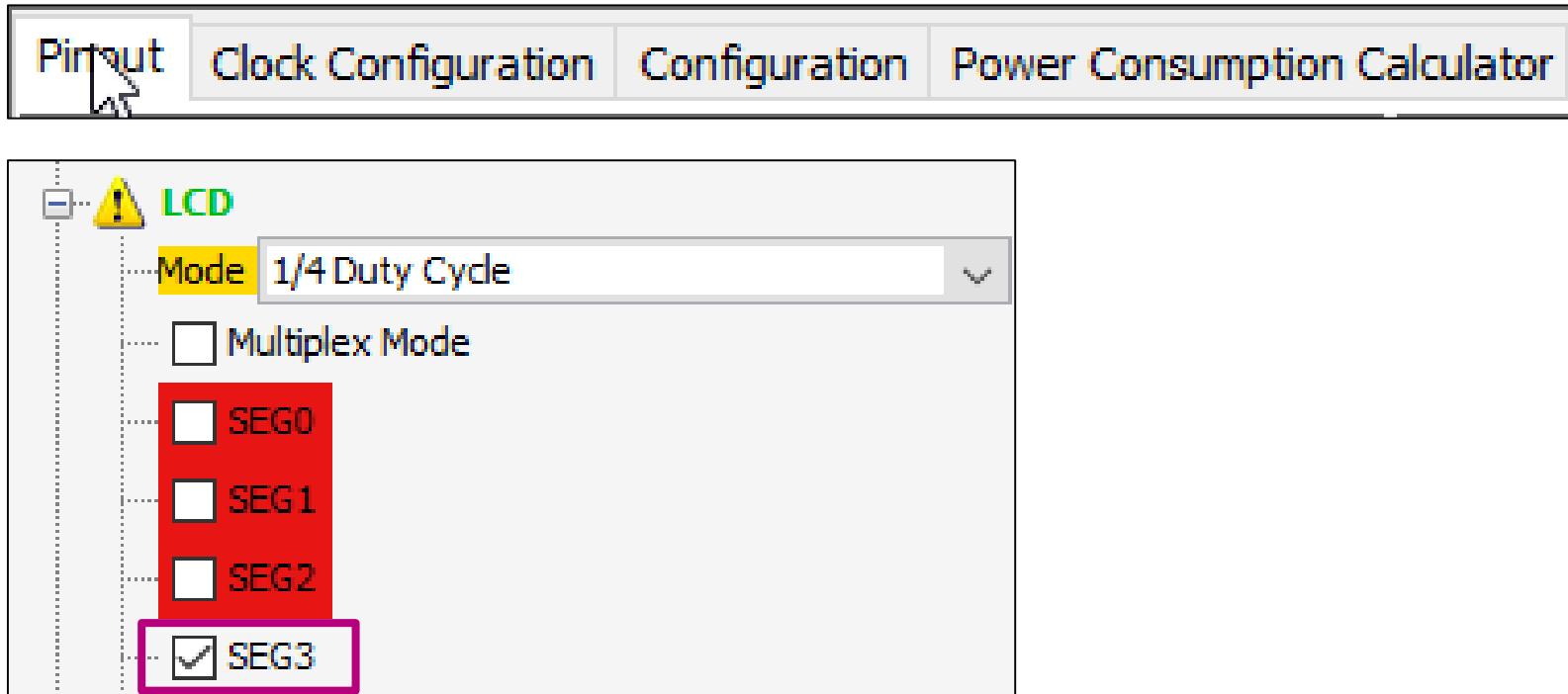


Under LCD peripheral select
1/4 Duty Cycle mode (as we
are using 4 commons)



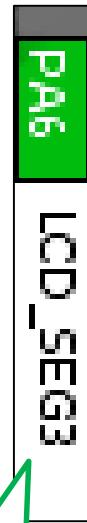
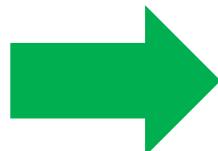
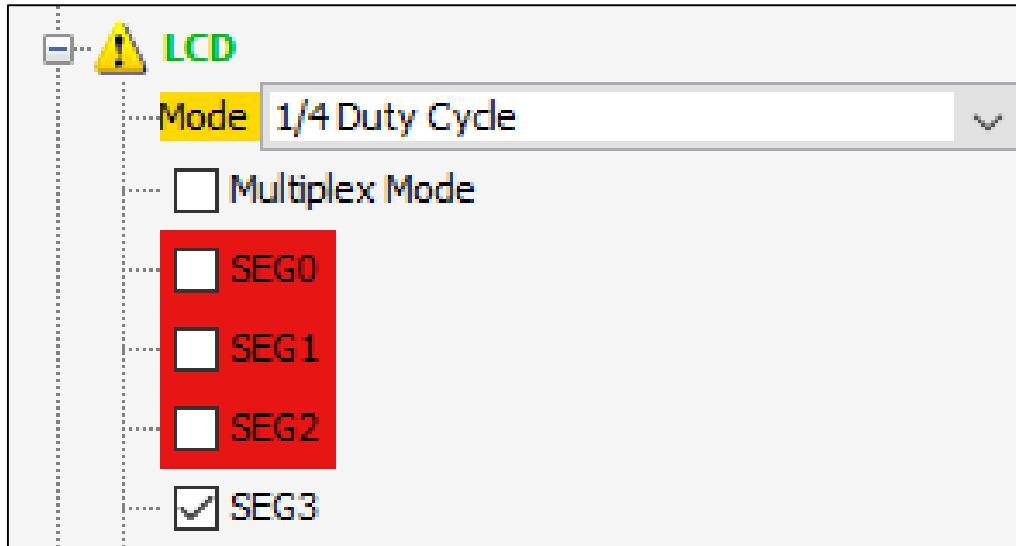
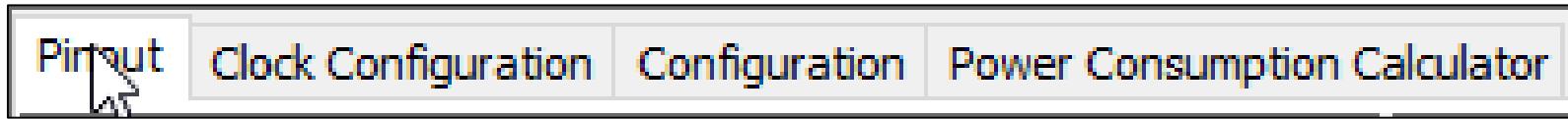
The corresponding pins are assigned and configured automatically!





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**



The corresponding pins are assigned and configured automatically!

6



User
Input



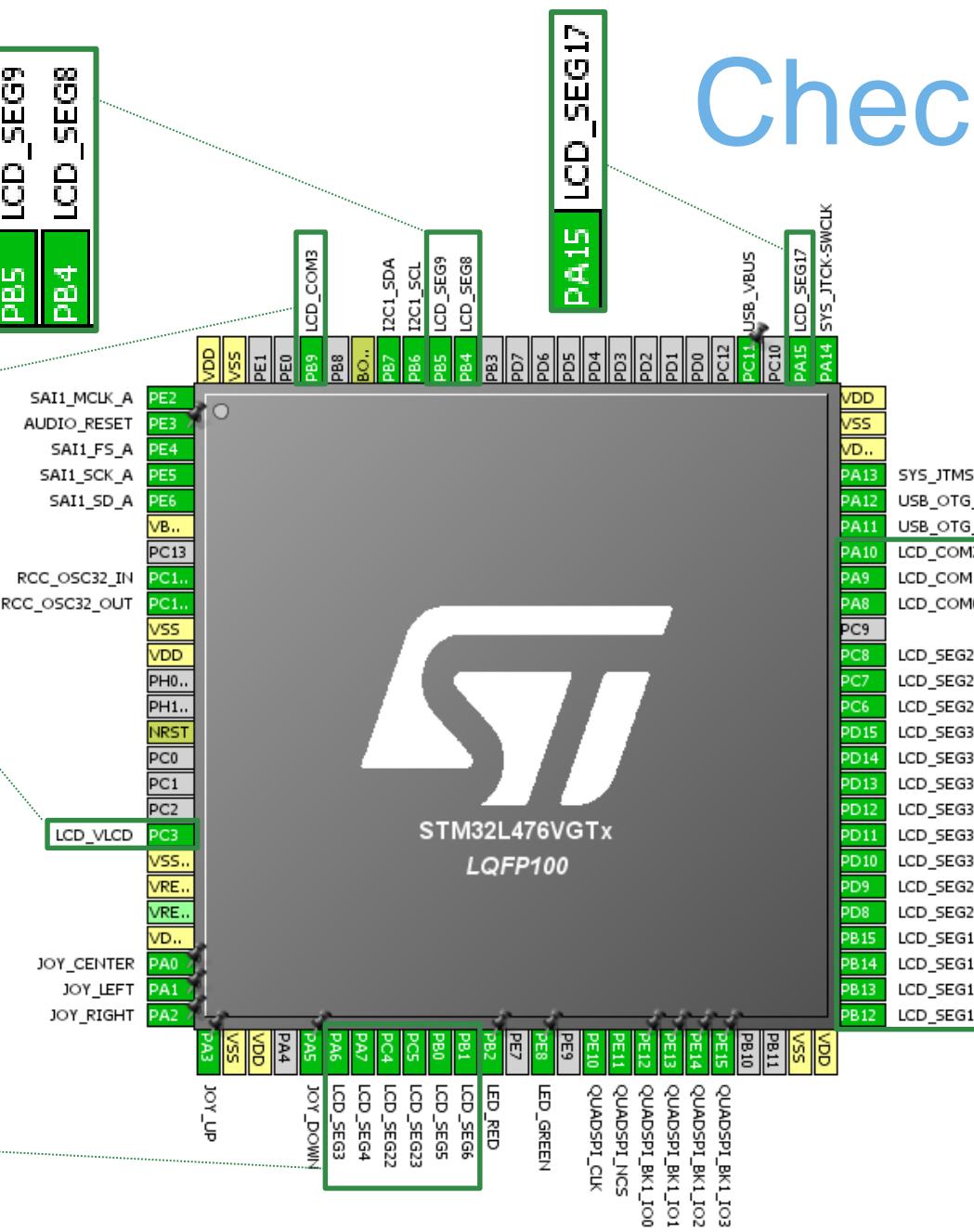
PA6	LCD_SEG3
PA7	LCD_SEG4
PA8	LCD_SEG5
PC4	LCD_SEG22
PC5	LCD_SEG23
PC6	LCD_SEG24
PC7	LCD_SEG25
PC8	LCD_SEG26
PB0	LCD_SEG55
PB1	LCD_SEG66

LCD_VLCD PC3

PB9 LCD_COM3

PB5 LCD_SEG9
PB4 LCD_SEG8

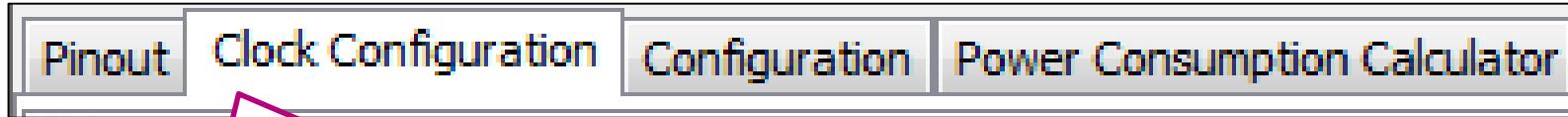
PA15 LCD_SEG17



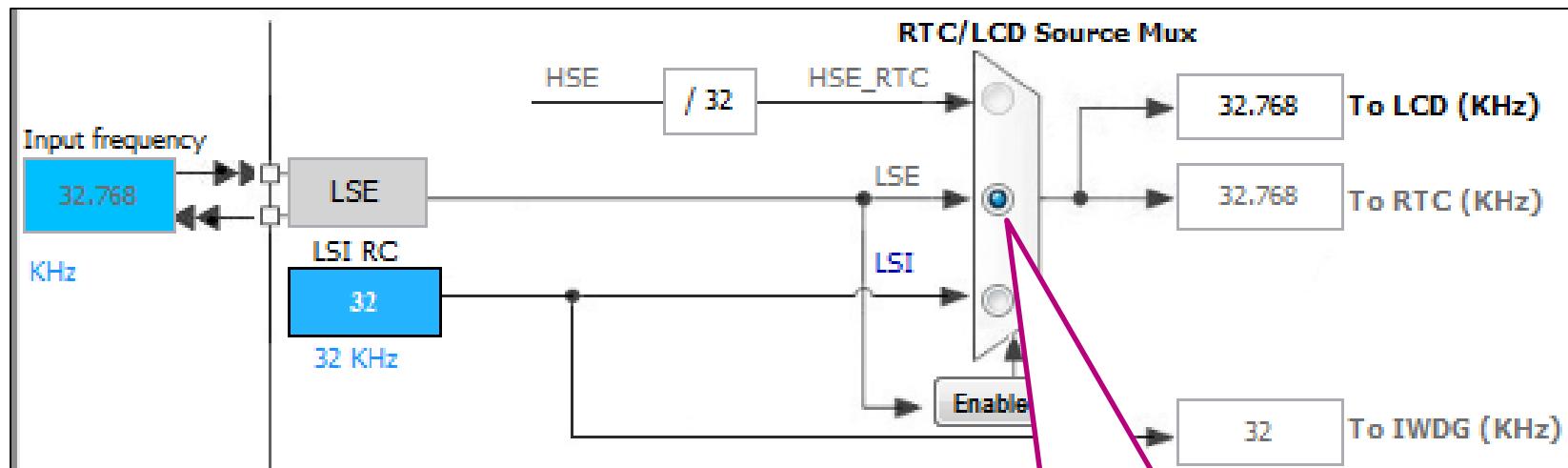
Checkpoint #17

164

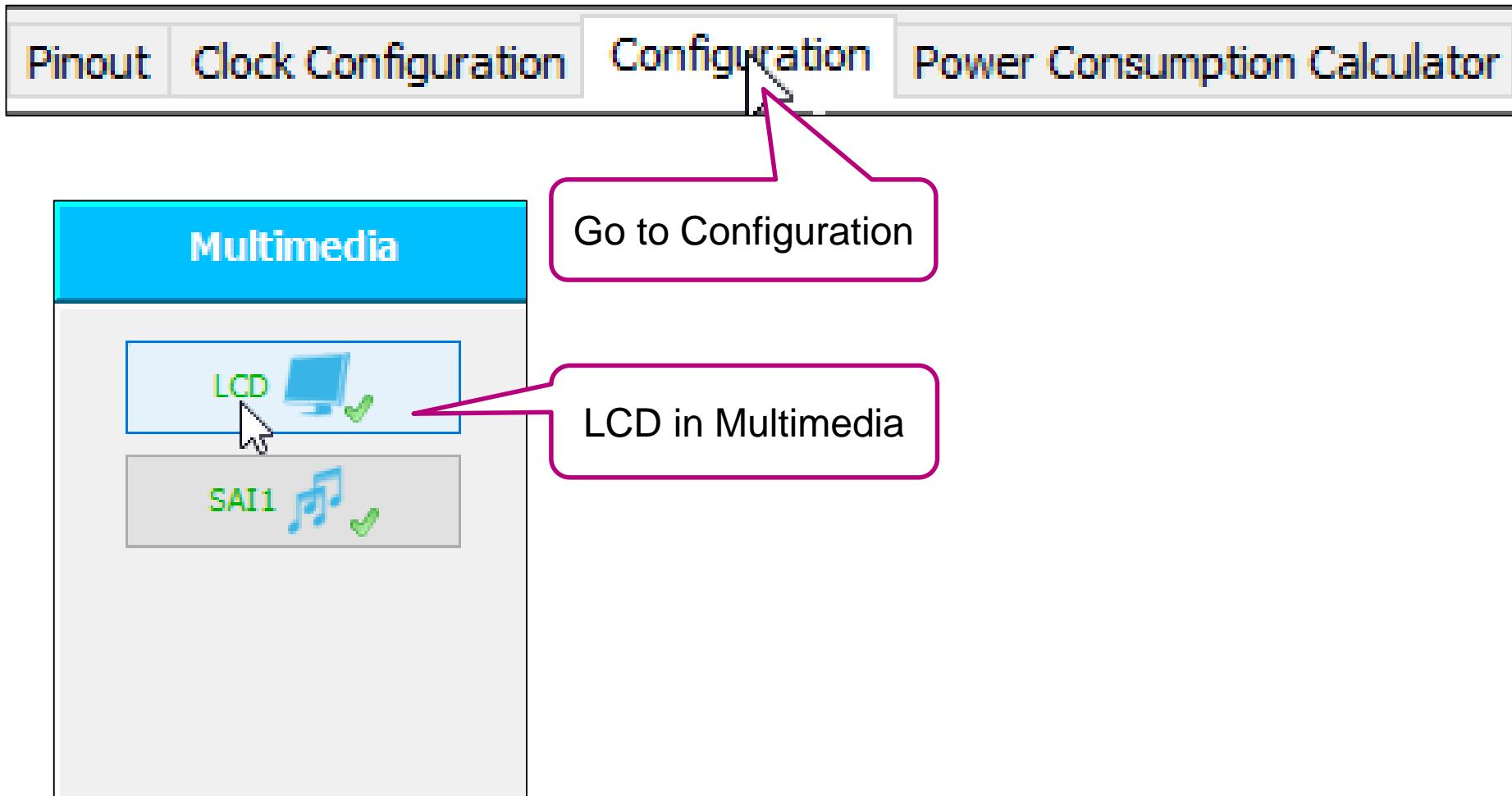
PA10	LCD_COM2
PA9	LCD_COM1
PA8	LCD_COM0
PC9	LCD_SEG26
PC8	LCD_SEG25
PC7	LCD_SEG24
PC6	LCD_SEG35
PD15	LCD_SEG34
PD14	LCD_SEG33
PD13	LCD_SEG32
PD12	LCD_SEG31
PD11	LCD_SEG30
PD10	LCD_SEG29
PD9	LCD_SEG28
PD8	LCD_SEG27
PB15	LCD_SEG15
PB14	LCD_SEG14
PB13	LCD_SEG13
PB12	LCD_SEG12

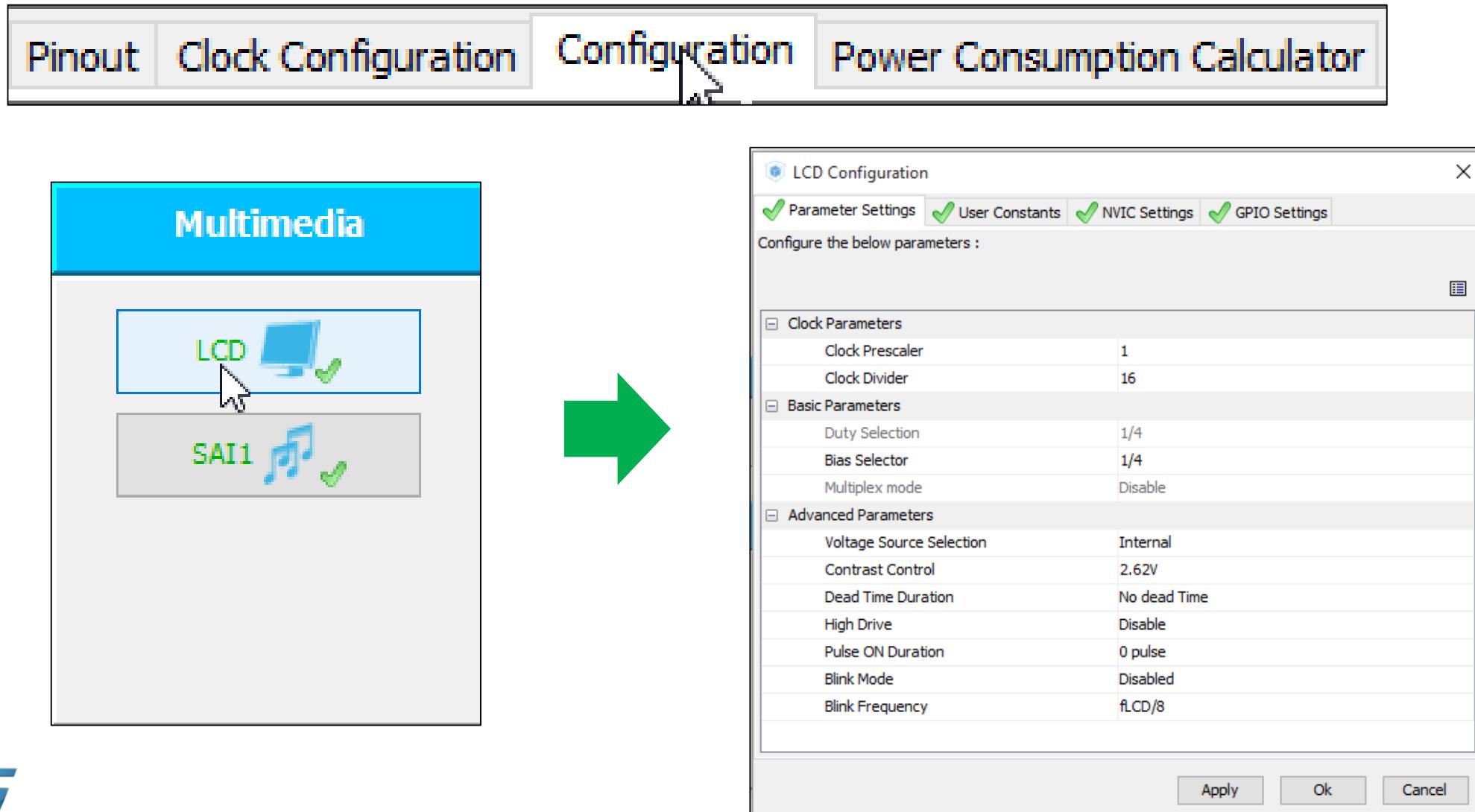


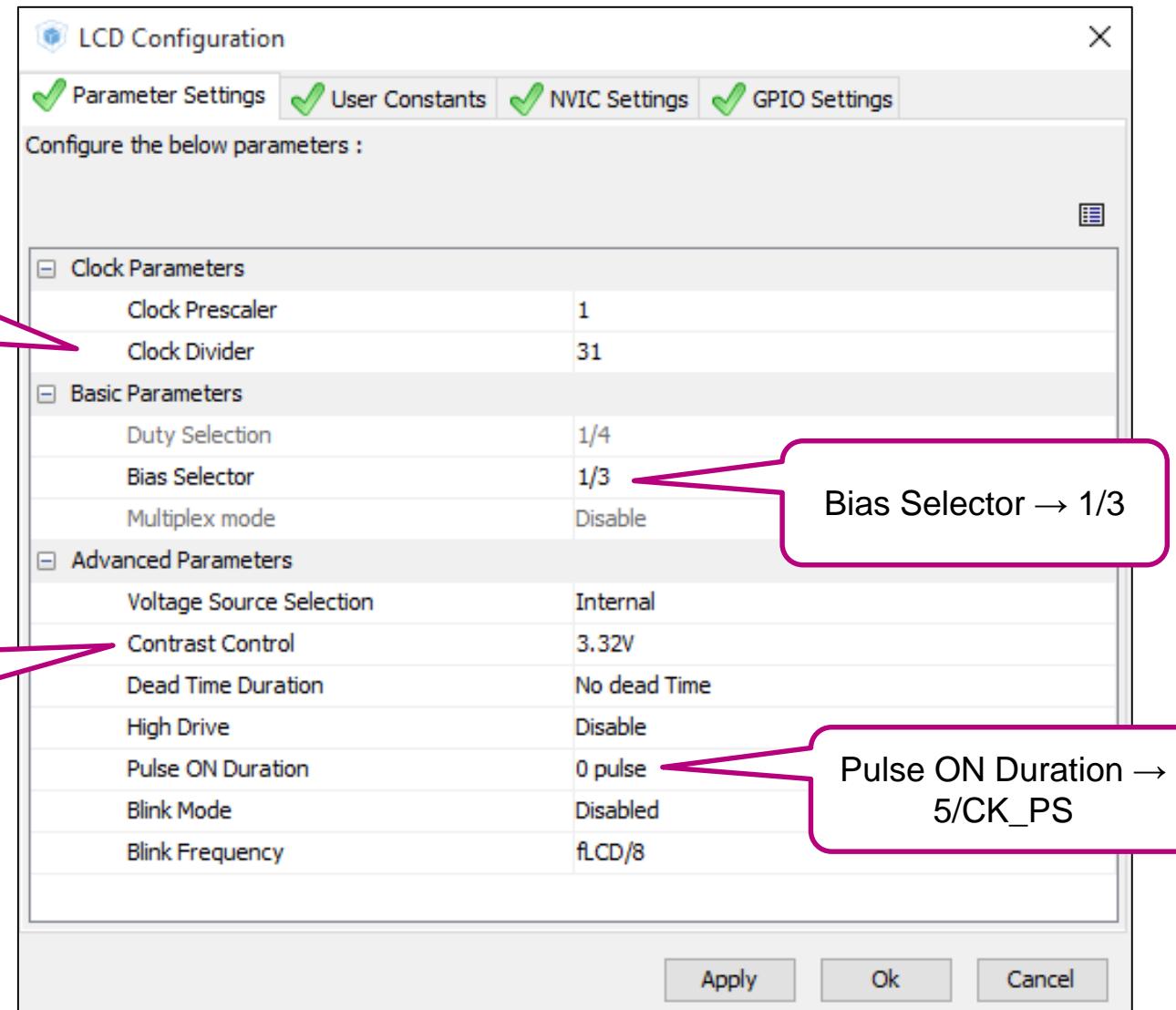
Go to Clock Configuration

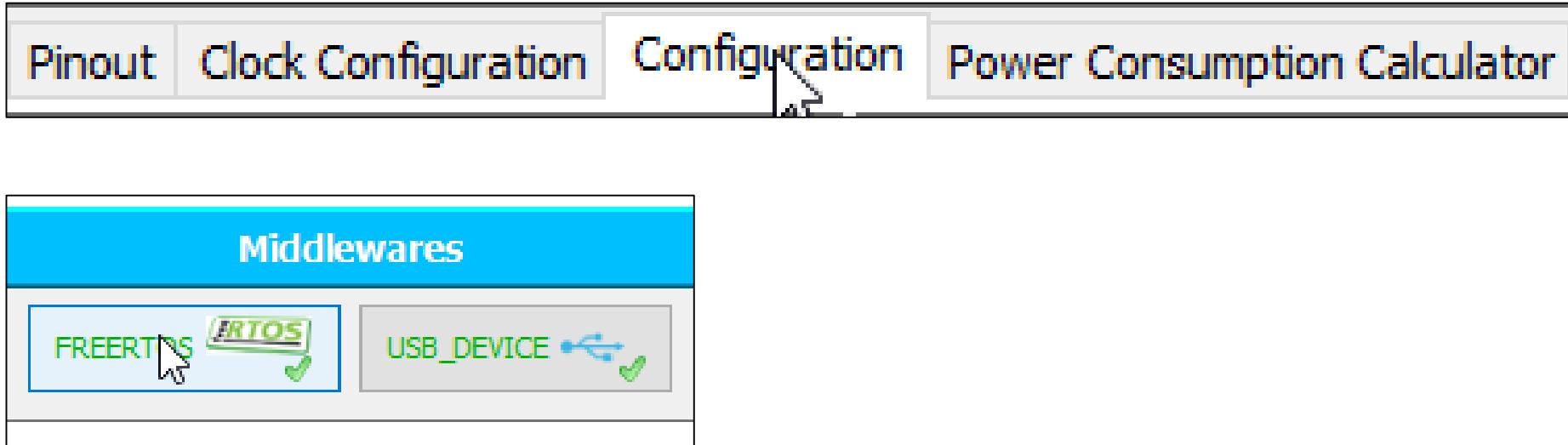


LSE
(RTC/LCD Source Mux)









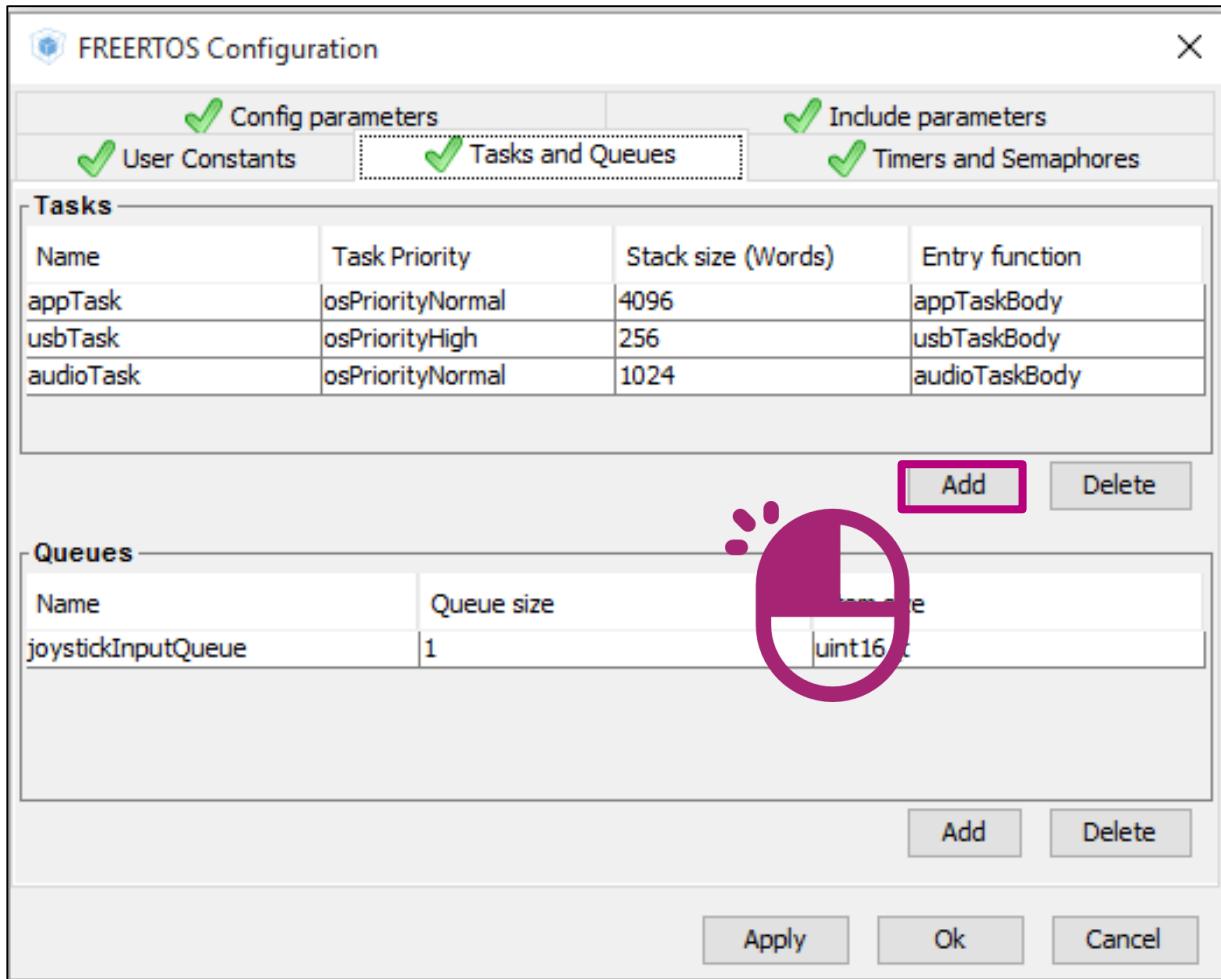
The screenshot shows the STM32CubeMX software interface. At the top, there are tabs: Pinout, Clock Configuration, Configuration (which is selected), Power Consumption Calculator, and others. Below the tabs, the main area has a 'Middlewares' section with two items: 'FREERTOS' and 'USB_DEVICE'. A large green arrow points from the 'FREERTOS' item towards a detailed configuration dialog box. This dialog box is titled 'FREERTOS Configuration' and contains sections for 'Config parameters', 'User Constants', 'Tasks and Queues', and 'Include parameters'. It also includes sections for 'Tasks' and 'Queues', each with a table. The 'Tasks' table lists three tasks:

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

The 'Queues' table lists one queue:

Name	Queue size	Item size
joystickInputQueue	1	uint16_t

At the bottom of the dialog box are 'Apply', 'Ok', and 'Cancel' buttons.



displayTask

displayTask

New Task

Name

displayTask

osPriorityNormal

Stack size (Words)

128

Entry function

displayTaskBody

OK

Cancel

displayTaskBody





FREERTOS Configuration

Config parameters

User Constants Tasks and Queues Include parameters 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

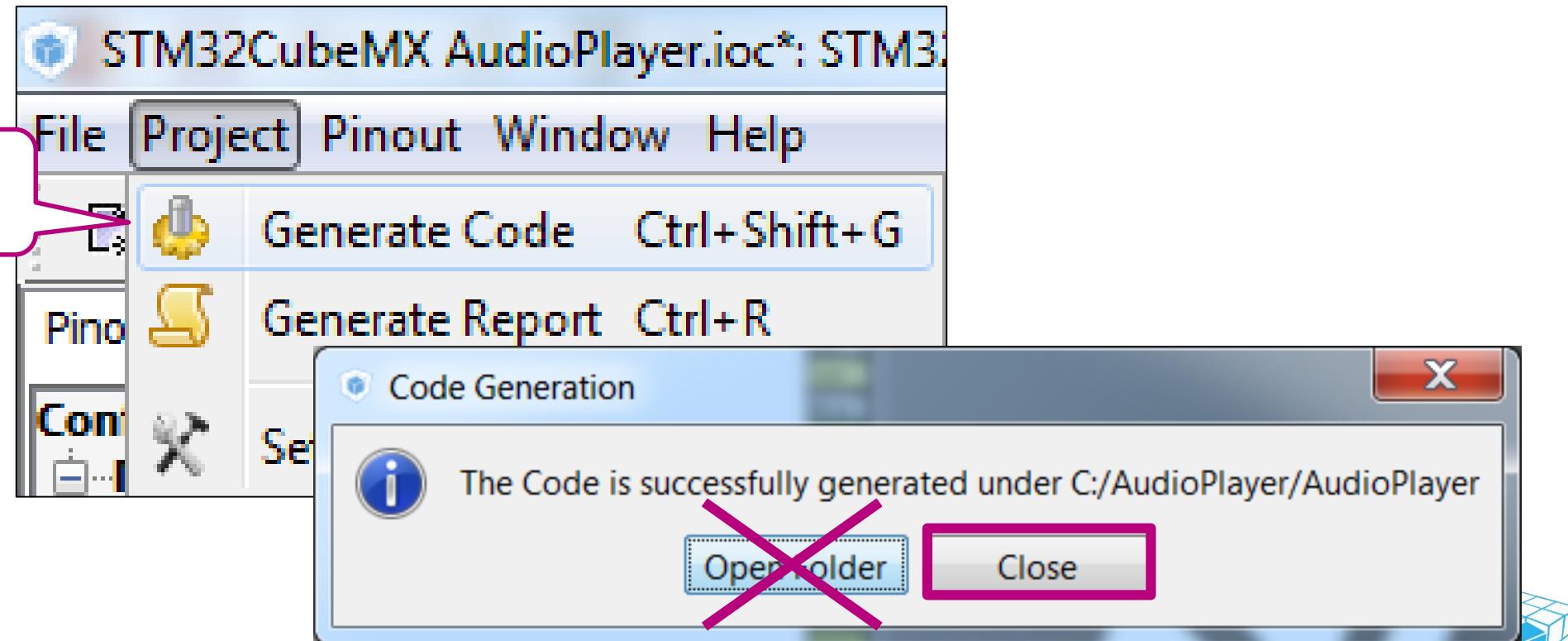
Apply Ok Cancel

This screenshot shows the FreeRTOS Configuration tool interface. It displays configuration parameters for tasks and queues. The 'Tasks' section lists four tasks: appTask, usbTask, audioTask, and displayTask, each with specific priority, stack size, and entry function. The 'Queues' section lists one queue, joystickInputQueue, with a queue size of 1 and item size of uint16_t. The 'Tasks and Queues' tab is selected. The 'displayTask' row is highlighted with a green border.



Save and generate the project

Do not open the project yet



6

- 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

Apply small patch

174

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

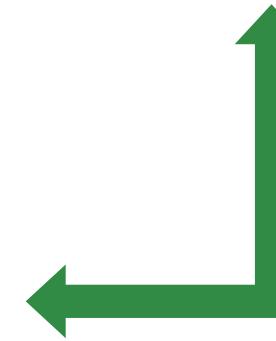
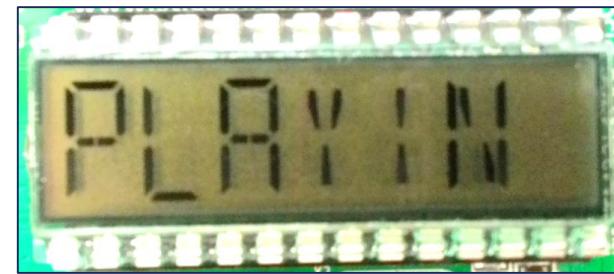
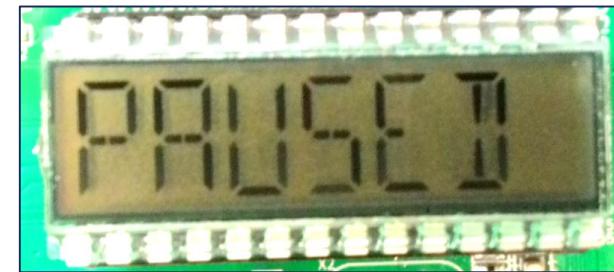
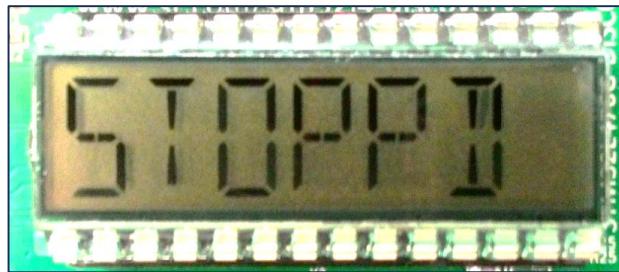
```
LCD_GLASS_Clear(void);  
LCD_GLASS_DisplayString(...);
```

1. Open the project
2. Compile and download
3. Press the reset button on discovery once download has finished

6

Checkpoint #19

177



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

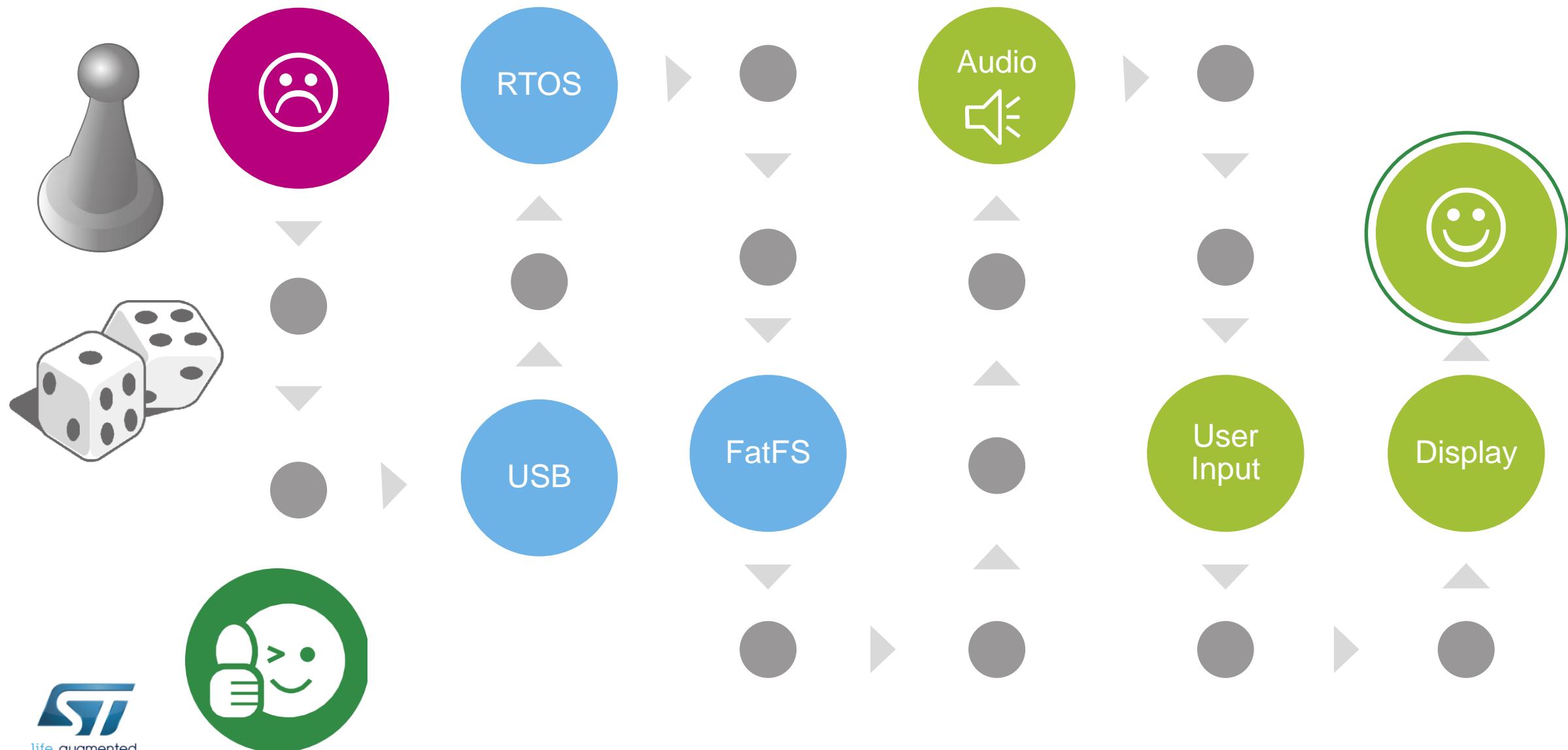
- You can implement previous state change checking mechanism

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

179



Enjoy!

7



The image features a man in a dynamic breakdancing pose, captured mid-motion against a plain white background. He is wearing a purple tank top, red pants, and yellow gloves.

STM32 L4

The STM32 L4 logo is displayed within a green circle. To the right of the logo, a blue circular area contains several icons: a smartphone, a person running, a microchip, and a small green circle with a pattern.

[/STM32](#)

[@ST_World](#)

[st.com/e2e](#)

www.st.com/stm32l4