



life.augmented

# STM32 AZURE RTOS workshop

USBX



# Introduction

- Rich class offer
- Native support for chaining multiple classes in one device – either host or device
  - Composite device and universal host capable communicate with composite or various classes
- Robustness, Certification
- ThreadX like memory allocation options
- Native connection to ThreadX and other Azure RTOS components
  - Consistent API



# USBX device classes

USB device classes	
AUDIO	Version 2.0 supported
CDC/ACM	Communication Data Class – Abstract Control Model (virtual COM port)
CDC/ECM	Communication Data Class – Ethernet Control Module
DFU	Device Firmware Update
HID	Human Interface Device
PIMA (PTP/MTP)	Picture (media) Transfer Protocol ★
Storage	Mass Storage class
RNDIS	Remote Network Driver Interface Specification



# USBX host classes

USBX host classes	
ASIX	USB/ethernet adaptor with proprietary vendor commands★
Audio	Version 2.0 supported
CDC/ACM	Communication Data Class – Abstract Control Model (virtual COM port)
CDC/ECM	Communication Data Class – Ethernet Control Module ★
GSER	Gadget serial ★
HID	Human Interface Device
HUB	USB hub class ★
PIMA (PTP/MTP)	Picture (media) Transfer Protocol
Printer	USB printer class ★
Prolific	USB/serial convertors ★
Storage	Mass Storage class

# HandsOn: Create USB composite device – HID + CDC ACM (VCP)



# Example description

- Create USB composite device configuration with HID and CDC-ACM interface
  - HID periodically moves mouse cursor
  - CDC virtual COM port echoes incoming data packets
- STM32CubeMX creates project with basic configuration
- Incomplete application functionality is copied from prepared template
  - Templates are based on repository example [Ux\\_Device\\_HID\\_CDC\\_ACM](#)
  - Easy migration of USB functionality between different STM32 MCUs



# Start from STM32CubeMX – MCU Selector

New Project from a MCU/MPU

MCU/MPU Selector | Board Selector | Example Selector | Cross Selector

MCU/MPU Filters

Part Number: stm32h723

Core >

Series >

Line >

Package >

Other >

Peripheral >

Features | Block Diagram | Docs & Resources | Datasheet | Buy | **Start Project**

STM32H7 Series


**STM32H723ZG**

High-performance and DSP with DP-FPU, Arm Cortex-M7 MCU with 1 MByte Flash, 564 KB RAM, 550 MHz CPU, L1 cache, external memory interface, subset of peripherals

**ACTIVE** Active  
Product is in mass production

Unit Price for 10kU (US\$) : 5.125

Board: [NUCLEO-H723ZG](#)

 LQFP144

STM32H723xE/G devices are based on the high-performance Arm® Cortex®-M7 32-bit RISC core operating at up to 550 MHz. The Cortex®-M7 core features a floating point unit (FPU) which supports Arm® double-precision (IEEE 754 compliant) and single-precision data-processing instructions and data types. The Cortex-M7 core includes 32 Kbytes of instruction cache and 32 Kbytes of data cache. STM32H723xE/G devices support a full set of DSP instructions and a memory protection unit (MPU) to enhance application security. STM32H723xE/G devices incorporate high-speed embedded memories with up to 1 Mbyte of Flash memory, up to 564 Kbytes of RAM (including 102 Kbytes that can be shared).

MCUs/MPUs List: 8 items

[+ Display similar items](#)

[Export](#)

	Part No	Reference	Marketing Stat...	Unit Price for 10k...	Board	Package	Flash	RAM	IO	Freq.
☆	STM32H723VE	STM32H723VEHx	Active	4.487		TFBGA100	512 kBytes	564 kBytes	82	550 MHz
☆		STM32H723VETx	Active	4.487		LQFP100	512 kBytes	564 kBytes	82	550 MHz
☆	STM32H723VG	STM32H723VGHx	Active	4.806		TFBGA100	1024 kBytes	564 kBytes	82	550 MHz
☆		STM32H723VGTx	Active	4.806		LQFP100	1024 kBytes	564 kBytes	82	550 MHz
☆	STM32H723ZE	STM32H723ZElx	Active	4.806		UFBGA144	512 kBytes	564 kBytes	114	550 MHz
☆		STM32H723ZETx	Active	4.806		LQFP144	512 kBytes	564 kBytes	114	550 MHz
☆		STM32H723ZGlx	Active	5.125		UFBGA144	1024 kBytes	564 kBytes	114	550 MHz
☆	STM32H723ZG	STM32H723ZGTx	Active	5.125	NUCLEO-H723ZG	LQFP144	1024 kBytes	564 kBytes	114	550 MHz

2

1



# Select USB device

Home > STM32H723ZGTx > Untitled - Pinout & Configuration

Pinout & Configuration | Clock Configuration | Software Packs | Pin

Search: [ ] Categories: A-Z

System Core > Analog > Timers > **Connectivity**

- ETH
- FDCAN1
- FDCAN2
- FDCAN3
- FMC
- I2C1
- I2C2
- I2C3
- I2C4
- I2C5
- LPUART1
- MDIOS
- OCTOSPI1
- OCTOSPI2
- SDMMC1
- SDMMC2
- SPI1
- SPI2
- SPI3
- SPI4
- SPI5
- SPI6
- SWPMI1
- UART4
- UART5
- UART7
- UART8
- UART9
- USART1
- USART2
- USART3
- USART6
- USART10
- USB\_OTG\_HS**

USB\_OTG\_HS Mode and Configuration

Mode

External Phy: Disable

**Internal FS Phy: Device\_Only** (1)

☐ Activate\_SOF

Activate\_VBUS: Disable

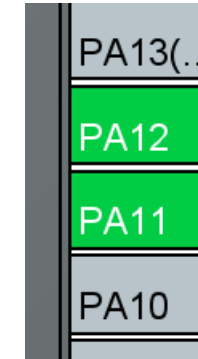
Configuration

Reset Configuration

Parameter Settings | User Constants | **NVIC Settings** | GPIO Settings

NVIC Interrupt Table	Enabled	Preemption Priority	Sub Priority
USB On The Go HS End Point 1 Out global interrupt	<input type="checkbox"/>	0	0
USB On The Go HS End Point 1 In global interrupt	<input type="checkbox"/>	0	0
<b>USB On The Go HS global interrupt</b>	<input checked="" type="checkbox"/>	0	0

(2) Do not forget to enable USB interrupt!



USB\_OTG\_HS\_DP

USB\_OTG\_HS\_DM

PA11 and PA12 are assigned to USB

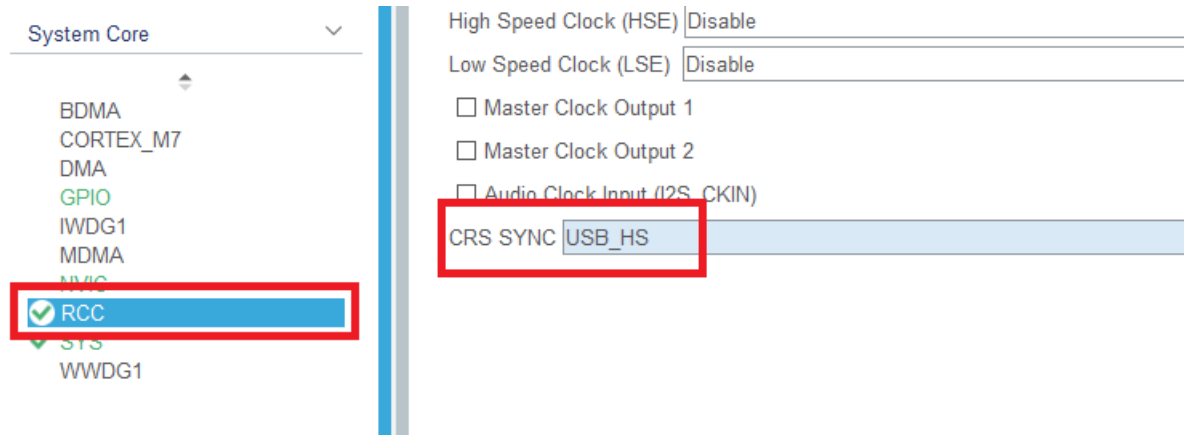




# USB clock source

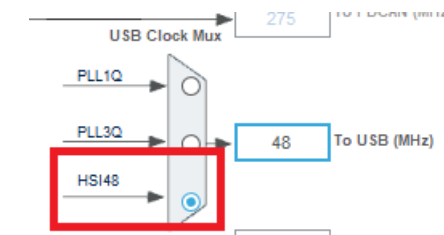
- Precise clock are mandatory for USB applications
  - STM32 USB host needs HSE
  - STM32 USB device can use HSE or advanced MCU features of some STM32 families
    - HSI48 + LSE (STM32L4)
    - HSI48 + CRS (multiple STM32 families including STM32H723), will be used in this example

1



2

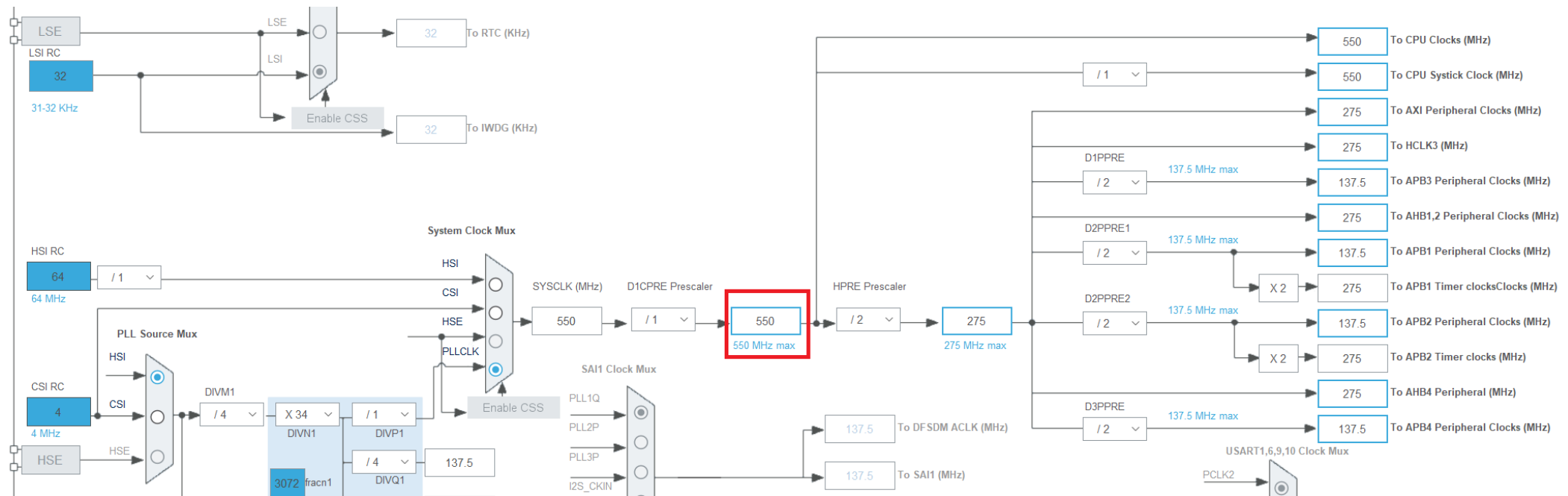
## Clock Configuration panel





# Setting system clock

- Using Clock configuration wizard set full MCU clock frequency – 550 MHz
  - Such high frequency is not mandatory, minimum HCLK frequency with OTG\_xS used is 14.2 MHz





# Add ThreadX

Pinout & Configuration | Clock Configuration | Pinout

1

Software Packs

Select Components

Manage Software Packs

Add pack software component to the project

RCC Mode and Configuration

Mode

High Speed Clock (HSE) Disable

STMicroelectronics.X-CUBE-AZRTOS-H7	✓	2.0.0	
RTOS ThreadX	✓	6.1.7	
ThreadX / Core	✓		<input checked="" type="checkbox"/>
ThreadX / PerformanceInfo	<input type="checkbox"/>		<input type="checkbox"/>
ThreadX / TraceX support	<input type="checkbox"/>		<input type="checkbox"/>
ThreadX / Low Power support	<input type="checkbox"/>		<input type="checkbox"/>

2



# Add USBX

▼ USB USBX	✓	6.1.7	
USBX / CoreSystem	✓		<input checked="" type="checkbox"/>
USBX / TraceX Support			<input type="checkbox"/>
USBX / UX Host CoreStack			<input type="checkbox"/>
USBX / UX Host Controllers			<input type="checkbox"/>
USBX / UX Device CoreStack	✓		<input checked="" type="checkbox"/>
USBX / UX Device Controllers	✓		<input checked="" type="checkbox"/>
USBX / UX Network Driver			<input type="checkbox"/>
USBX / UX Host Class HID Core			<input type="checkbox"/>
USBX / UX Host Class HID Keyboard			<input type="checkbox"/>
USBX / UX Host Class HID Mouse			<input type="checkbox"/>
USBX / UX Host Class HID RCU			<input type="checkbox"/>
USBX / UX Host Class STORAGE			<input type="checkbox"/>
USBX / UX Host Class CDC ACM			<input type="checkbox"/>
USBX / UX Host Class CDC ECM			<input type="checkbox"/>
USBX / UX Host Class AUDIO			<input type="checkbox"/>
USBX / UX Host Class VIDEO			<input type="checkbox"/>
USBX / UX Host Class PRINTER			<input type="checkbox"/>
USBX / UX Host Class GSER			<input type="checkbox"/>
USBX / UX Host Class ASIX			<input type="checkbox"/>
USBX / UX Host Class PIMA			<input type="checkbox"/>
USBX / UX Host Class PROLIFIC			<input type="checkbox"/>
USBX / UX Host Class SWAR			<input type="checkbox"/>
USBX / UX Device Class HID	✓		<input checked="" type="checkbox"/>
USBX / UX Device Class STORAGE			<input type="checkbox"/>
USBX / UX Device Class CDC ACM	✓		<input checked="" type="checkbox"/>
USBX / UX Device Class CDC ECM			<input type="checkbox"/>
USBX / UX Device Class DFU			<input type="checkbox"/>
USBX / UX Device Class AUDIO			<input type="checkbox"/>
USBX / UX Device Class PIMA			<input type="checkbox"/>
USBX / UX Device Class RNDIS			<input type="checkbox"/>



# Configure AzureRTOS memory

Pinout & Configuration

Search

Categories A->Z

- System Core
- Analog
- Timers
- Connectivity
- Multimedia
- Security
- Computing
- Middleware
- Trace and Debug
- Power and Thermal
- Software Packs

STMMicroelectronics.X-CUBE-AZRTOS-H7.2.0.0 Mode and Configuration

Mode

- ☒ RTOS ThreadX
- ☒ USB USBX

UXDevice\_memory pool size (18224)

Configuration

Reset Configuration

AzureRTOS Application ThreadX USBX User Constants

Configure the below parameters :

Search (Ctrl+F)

Memory Configuration

- Memory Allocation
- ThreadX memory pool size
- UXDevice memory pool size

Use Static MemPool Allocation

1024

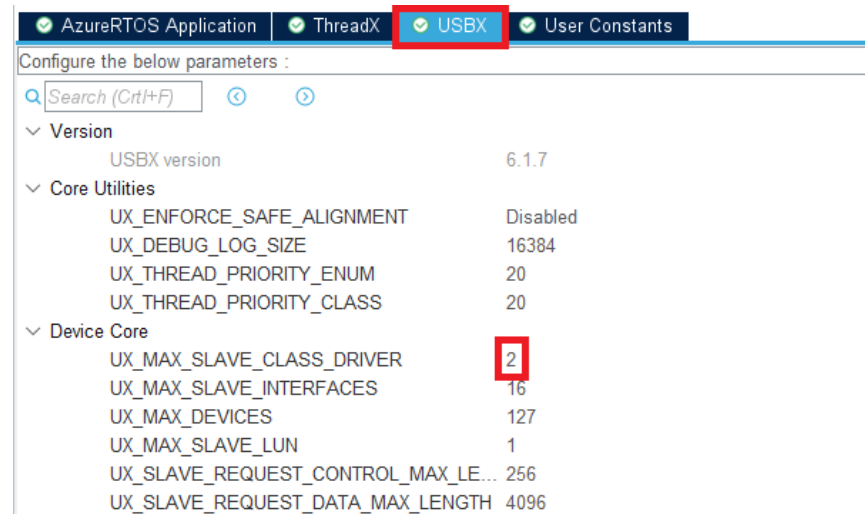
18224

STMMicroelectronics.X-CUBE-AZRTOS-H7.2.0.0



# USBX setting 1

- Set UX\_MAX\_SLAVE\_CLASS\_DRIVER to 2
  - Two classes are used in composite device





# USBX setting 2

- Chose endpoint addresses
  - Different address need to be used for each endpoint.

Reset Configuration

☒ AzureRTOS Application ☒ ThreadX ☒ USBX ☒ User Constants

Configure the below parameters :

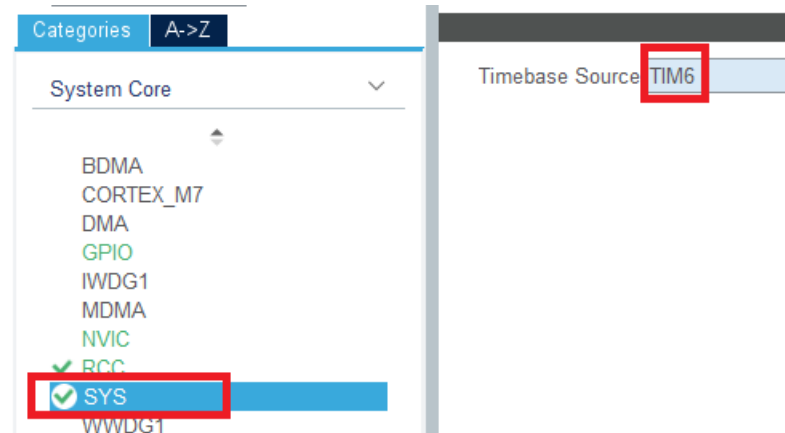
Search (Ctrl+F)

UX_MAX_DEVICE_INTERFACES	6
USBD_DEVICE_FRAMEWORK_BUILDER_ENABLED	Enabled
Device Class HID	
UX_DEVICE_CLASS_HID_EVENT_BUFFER_LENGTH	64
UX_DEVICE_CLASS_HID_MAX_EVENTS_QUEUE	8
USBD_HID_ENDPOINT_IN_ADDRESS	1
USBD_HID_ENDPOINT_IN_FS_MPS	4
USBD_HID_ENDPOINT_IN_HS_MPS	4
USBD_HID_ENDPOINT_IN_FS_BINTERVAL	5
USBD_HID_ENDPOINT_IN_HS_BINTERVAL	5
Device Class CDC ACM	
USBD_CDCACM_ENDPOINT_IN_CMD_ADDR	2
USBD_CDCACM_ENDPOINT_IN_CMD_FS_MPS	8
USBD_CDCACM_ENDPOINT_IN_CMD_HS_MPS	8
USBD_CDCACM_ENDPOINT_IN_CMD_FS_BINTERVAL	5
USBD_CDCACM_ENDPOINT_IN_CMD_HS_BINTERVAL	5
USBD_CDCACM_ENDPOINT_IN_ADDR	3
USBD_CDCACM_ENDPOINT_IN_FS_MPS	64
USBD_CDCACM_ENDPOINT_IN_HS_MPS	512
USBD_CDCACM_ENDPOINT_OUT_ADDR	4
USBD_CDCACM_ENDPOINT_OUT_FS_MPS	64
USBD_CDCACM_ENDPOINT_OUT_HS_MPS	512



# System Timebase Source

- SysTick used by Azure, use different source for HAL timing







# Project setting

- Set project name
- Used IDE – STM32CubeIDE

	Pinout & Configuration	Clock Configuration	Project Manager
Project	<div>Project Settings</div> <div>Project Name USBX_HID_CDC_composite_device</div> <div>Project Location C:\Azure_RTOS</div> <div>Application Structure Advanced <input type="checkbox"/> Do not generate the main()</div>		
Code Generator	<div>Toolchain Folder Location C:\Azure_RTOS\USBX_HID_CDC_composite_device\</div> <div><div>Toolchain / IDE STM32CubeIDE</div><div><input checked="" type="checkbox"/> Generate Under Root</div></div> <div>Linker Settings</div> <div>Minimum Heap Size 0x200</div> <div>Minimum Stack Size 0x400</div>		



# Code Generator options

	Pinout & Configuration	Clock Configuration
Project	<p>STM32Cube MCU packages and embedded software packs</p> <ul style="list-style-type: none"><li><input type="radio"/> Copy all used libraries into the project folder</li><li><input checked="" type="radio"/> Copy only the necessary library files</li><li><input type="radio"/> Add necessary library files as reference in the toolchain project configuration file</li></ul>	
Code Generator	<p>Generated files</p> <ul style="list-style-type: none"><li><input checked="" type="checkbox"/> Generate peripheral initialization as a pair of '.c/.h' files per peripheral</li><li><input type="checkbox"/> Backup previously generated files when re-generating</li><li><input checked="" type="checkbox"/> Keep User Code when re-generating</li><li><input checked="" type="checkbox"/> Delete previously generated files when not re-generated</li></ul>	
	<p>HAL Settings</p> <ul style="list-style-type: none"><li><input type="checkbox"/> Set all free pins as analog (to optimize the power consumption)</li><li><input type="checkbox"/> Enable Full Assert</li></ul>	
Advanced Settings	<p>Template Settings</p> <p>Select a template to generate customized code</p> <p>Settings...</p>	



# Advanced settings

Pinout & Configuration

Clock Configuration

Project Manager

Project

Code Generator

Advanced Settings

Driver Selector

Generated Function Calls

Search (Ctrl+F)

GPIO

RCC

USB\_OTG\_HS

CORTEX\_M7

HAL

HAL

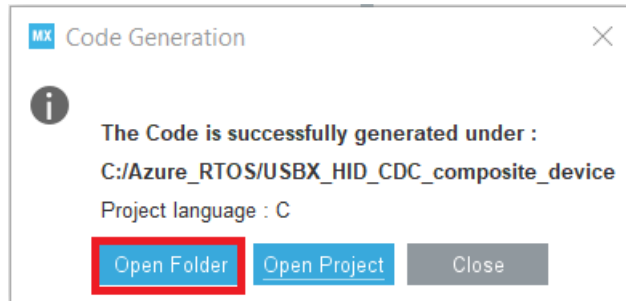
HAL

HAL

Generate Code	Rank	Function Name	Peripheral Instance Name	<input type="checkbox"/> Do Not Generate Function Call	<input type="checkbox"/> Visibility (Static)
<input checked="" type="checkbox"/>	1	MX_GPIO_Init	GPIO	<input type="checkbox"/>	<input checked="" type="checkbox"/>
<input checked="" type="checkbox"/>	2	SystemClock_Config	RCC	<input type="checkbox"/>	<input type="checkbox"/>
<input checked="" type="checkbox"/>	3	MX_USB_OTG_HS_PCD_Init	USB_OTG_HS	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>



# Code generation



Name	Date modified	Type	Size
AZURE_RTOS	15-Apr-21 15:27	File folder	
Core	15-Apr-21 15:27	File folder	
Drivers	15-Apr-21 15:27	File folder	
Middleware	15-Apr-21 15:27	File folder	
STM32CubeIDE	15-Apr-21 15:27	File folder	
USBX	15-Apr-21 15:27	File folder	
.mxproject	15-Apr-21 15:28	MXPROJECT File	30 KB
USBX_HID_CDC_composite_device.ioc	15-Apr-21 15:27	STM32CubeMX	10 KB

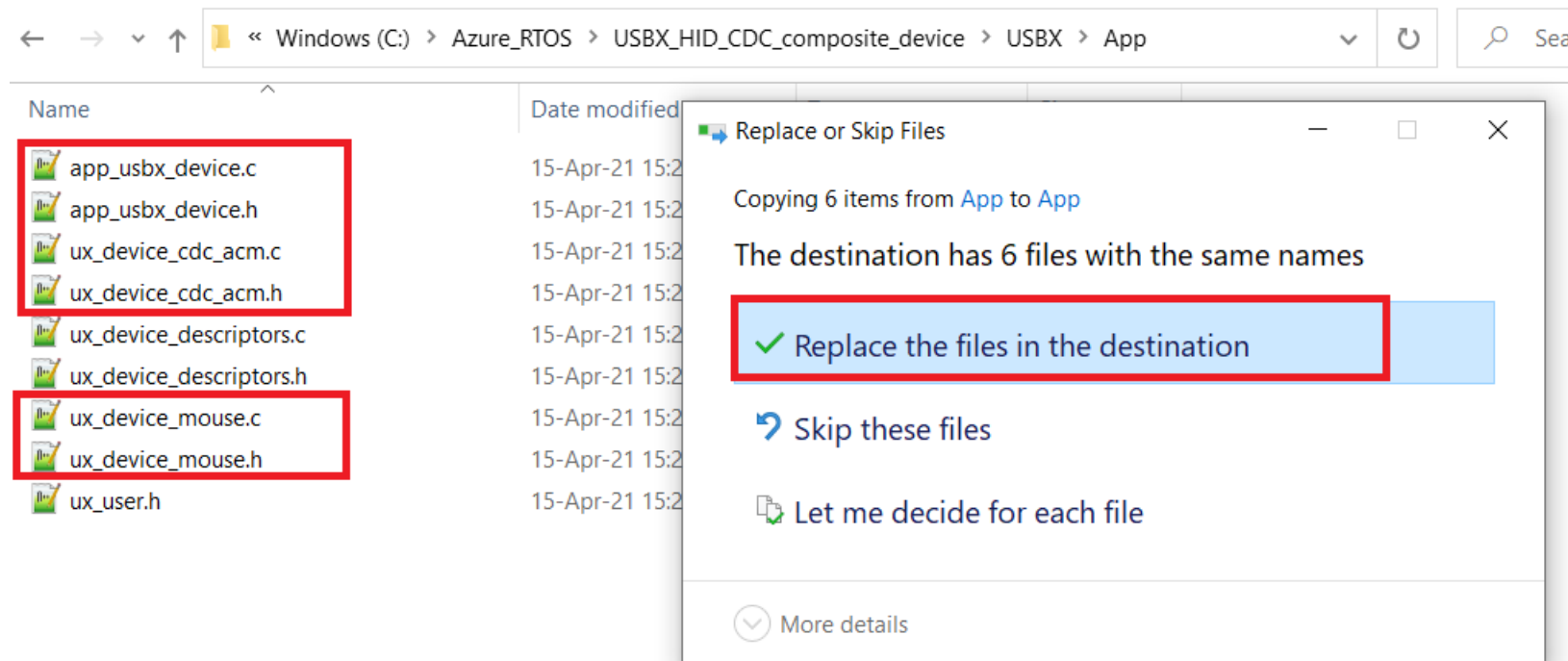


# Replace empty templates

- Default path to files for copy

C:\tmp\STM32-AZURE-RTOS-WS-Material\HandsOn\USBX\USBX\_copy\_files

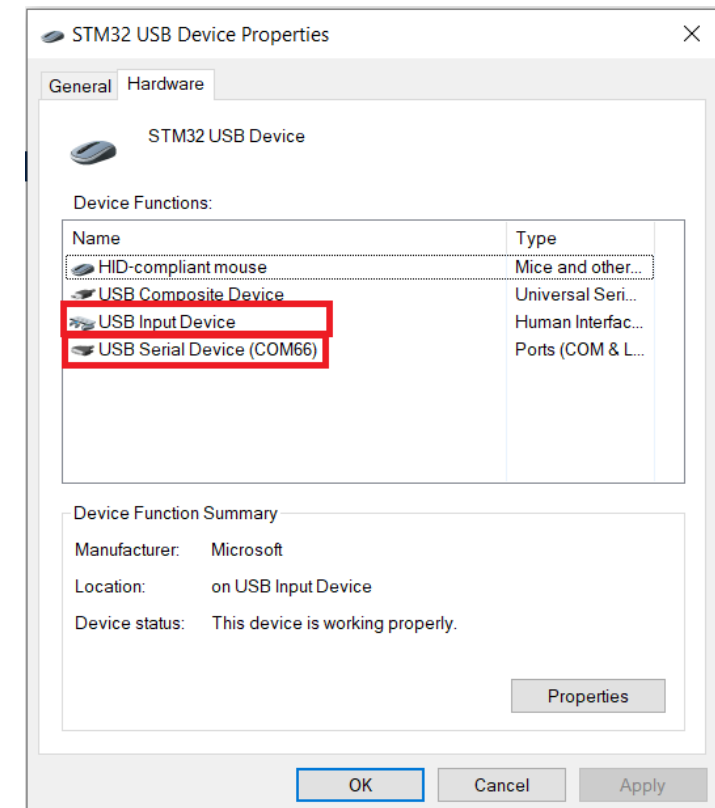
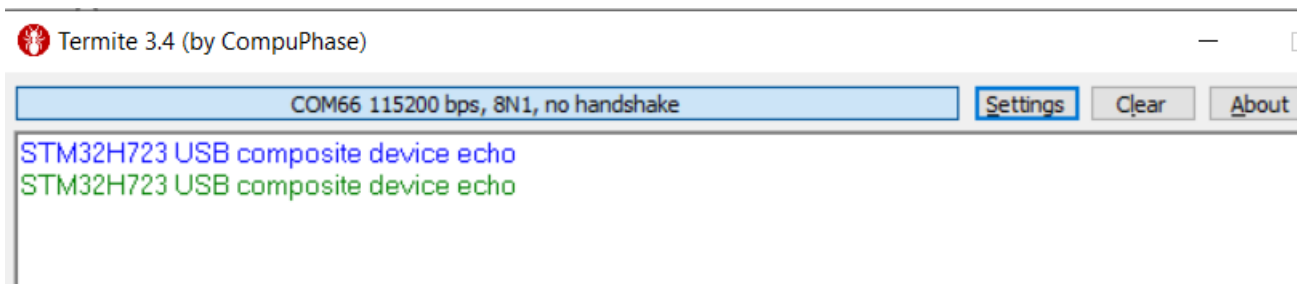
- Copy folder content into project folder opened in previous step





# Project testing

- All set – run project on your STM32!
- Plug USB cable between PC and User USB port of nucleo board (CN13)
- Both HID and CDC interfaces are visible in one device for USB host
- HID class moves mouse cursor
- Virtual COM port echoes incoming messages





# Template changes – app\_usbx\_device

- MX\_USBX\_Device\_Init

- 1) Get frameworks (descriptors) created in ux\_device\_descriptor.c

- Descriptor builder use parameters entered in STM32CubeMX

```
/* Get_Device_Framework_High_Speed and get the length */
device_framework_high_speed = USB_D_Get_Device_Framework_Speed(USB_D_HIGH_SPEED,
                                                                &device_framework_hs_length);

/* Get_Device_Framework_Full_Speed and get the length */
device_framework_full_speed = USB_D_Get_Device_Framework_Speed(USB_D_FULL_SPEED,
                                                                &device_framework_fs_length);

/* Get_String_Framework and get the length */
string_framework = USB_D_Get_String_Framework(&string_framework_length);

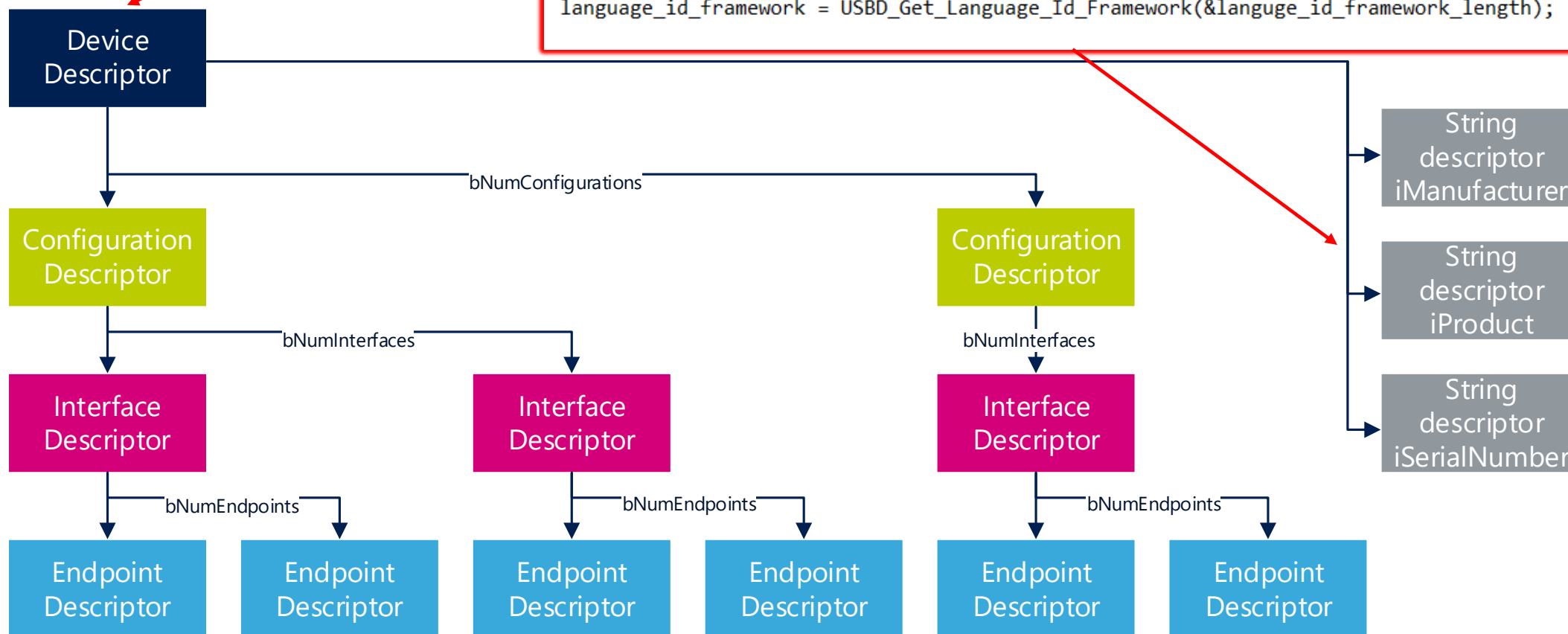
/* Get_Language_Id_Framework and get the length */
language_id_framework = USB_D_Get_Language_Id_Framework(&language_id_framework_length);
```



# Device framework

```
/* Get_Device_Framework_High_Speed and get the length */  
device_framework_high_speed = USBD_Get_Device_Framework_Speed(USBD_HIGH_SPEED,  
    &device_framework_hs_length);  
  
/* Get_Device_Framework_Full_Speed and get the length */  
device_framework_full_speed = USBD_Get_Device_Framework_Speed(USBD_FULL_SPEED,  
    &device_framework_fs_length);
```

```
/* Get_String_Framework and get the length */  
string_framework = USBD_Get_String_Framework(&string_framework_length);  
  
/* Get_Language_Id_Framework and get the length */  
language_id_framework = USBD_Get_Language_Id_Framework(&language_id_framework_length);
```







# Template changes – app\_usbx\_device

- MX\_USBX\_Device\_Init

## 2) Initialize USB device with obtained descriptors

- Does not contain any class dependent functionality yet

```
/* The code below is required for installing the device portion of USBX. In this application */  
ret = ux_device_stack_initialize(device_framework_high_speed,  
                                device_framework_hs_length,  
                                device_framework_full_speed,  
                                device_framework_fs_length,  
                                string_framework,  
                                string_framework_length,  
                                language_id_framework,  
                                language_id_framework_length, UX_NULL);
```



# Template changes – app\_usb\_device

- MX\_USBX\_Device\_Init

## 3) Link HID class to USB device as interface 0

```
hid_parameter.ux_device_class_hid_parameter_report_address = USBD_Get_Device_HID_MOUSE_ReportDesc();  
  
hid_parameter.ux_device_class_hid_parameter_report_length = USBD_HID_MOUSE_REPORT_DESC_SIZE;  
  
hid_parameter.ux_device_class_hid_parameter_report_id = UX_TRUE;  
hid_parameter.ux_device_class_hid_parameter_callback = app_usb_device_thread_hid_callback;  
  
/* Initialize the device hid class. The class is connected with interface 0 */  
ret = ux_device_stack_class_register(_ux_system_slave_class_hid_name,  
                                     ux_device_class_hid_entry, 1, 0, (VOID *)&hid_parameter);
```

Configuration  
number

Interface  
number



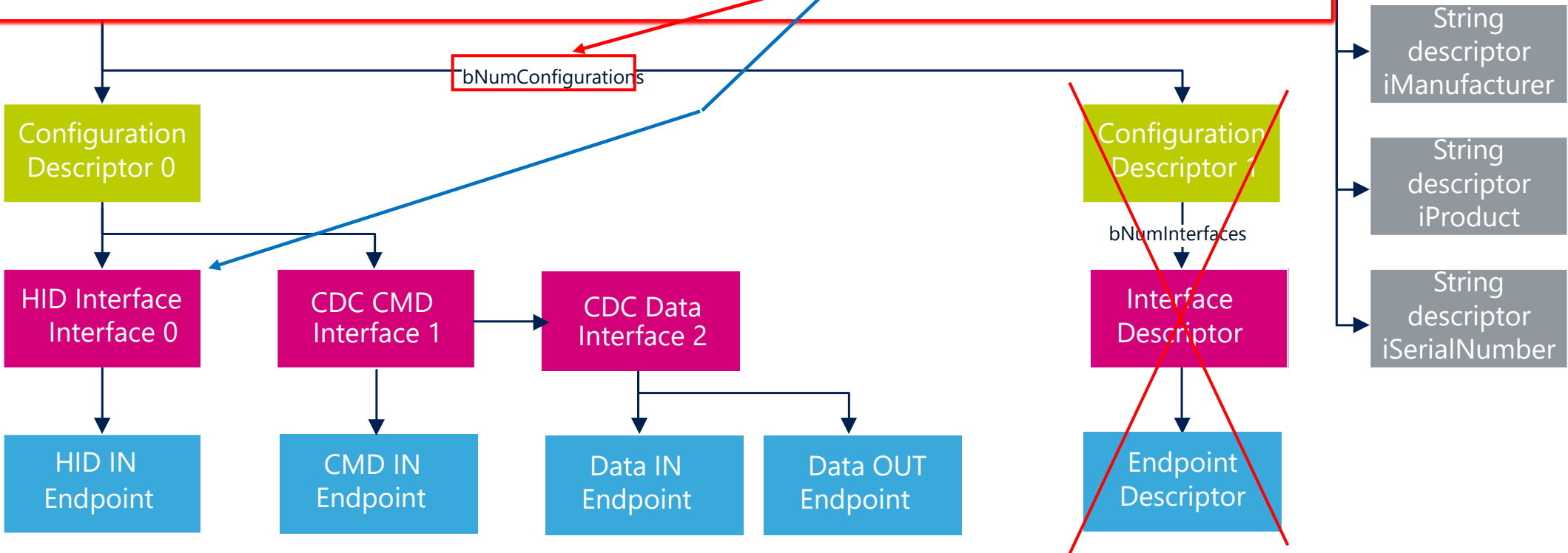
# Class initialization

```
/* Initialize the hid class parameters for the device. */
hid_parameter.ux_device_class_hid_parameter_report_address = USBD_Get_Device_HID_MOUSE_ReportDesc();

hid_parameter.ux_device_class_hid_parameter_report_length = USBD_HID_MOUSE_REPORT_DESC_SIZE;

hid_parameter.ux_device_class_hid_parameter_report_id = UX_TRUE;
hid_parameter.ux_device_class_hid_parameter_callback = app_usbx_device_thread_hid_callback;

/* Initialize the device hid class. The class is connected with interface 0 */
ret = ux_device_stack_class_register(_ux_system_slave_class_hid_name,
                                     ux_device_class_hid_entry, 1, 0, (VOID *)&hid_parameter);
```





# Template changes – app\_usb\_device

- MX\_USBX\_Device\_Init

4) Create one thread to maintain USB device and one thread for HID interface communication handling

```
ret = tx_byte_allocate(byte_pool, (VOID **) &pointer, USBX_APP_STACK_SIZE, TX_NO_WAIT);  
ret = tx_thread_create(&ux_app_thread, "main_usb_app_thread_entry", usb_app_thread_entry, 0,  
                        pointer, USBX_APP_STACK_SIZE, 20, 20, 1, TX_AUTO_START);  
  
ret = tx_byte_allocate(byte_pool, (VOID **) &pointer, USBX_APP_STACK_SIZE, TX_NO_WAIT);  
ret = tx_thread_create(&ux_hid_thread, "hid_usb_app_thread_entry", usb_hid_thread_entry, 1,  
                        pointer, USBX_APP_STACK_SIZE, 20, 20, 1, TX_AUTO_START);
```

- Error check used in templates for each tx\_\* function are skipped in slides
- Steps 1-4 are needed for functionality of one class
  - Additionally step 7 is needed to init USB peripheral



# Template changes – app\_usb\_x\_device

- MX\_USBX\_Device\_Init
- For creating composite device next interface is added

## 5) Link CDC class to USB device as interface 1

```
cdc_acm_parameter.ux_slave_class_cdc_acm_instance_activate = CDC_Init_FS;  
cdc_acm_parameter.ux_slave_class_cdc_acm_instance_deactivate = CDC_DeInit_FS;  
cdc_acm_parameter.ux_slave_class_cdc_acm_parameter_change = ux_app_parameters_change;  
  
/* registers a slave class to the slave stack. The class is connected with interface 1 */  
  
ret = ux_device_stack_class_register(_ux_system_slave_class_cdc_acm_name,  
                                     ux_device_class_cdc_acm_entry, 1, 1,  
                                     (VOID *)&cdc_acm_parameter);
```



# Template changes – app\_usb\_x\_device

- MX\_USBX\_Device\_Init

## 6) CDC interface need dedicated thread for communication handling

- 1 thread is used for simplification in this demo, for regular application separate thread should be considered for Tx and Rx direction

```
ret = tx_thread_create(&ux_cdc_read_thread, "cdc_acm_read_usb_x_app_thread_entry",  
                      usb_x_cdc_acm_read_thread_entry, 2,  
                      pointer, USBX_APP_STACK_SIZE, 22, 22, TX_NO_TIME_SLICE,  
                      TX_AUTO_START);
```



# Template changes – app\_usbx\_device

7) Usbx\_app\_thread\_entry -This thread function initialize USB peripheral, run only once after reset

- Then USB events are handled in interrupt

```
void MX_USB_Device_Init(void)
{
    HAL_PWREx_EnableUSBVoltageDetector();

    MX_USB_OTG_HS_PCD_Init();
    HAL_PCDEx_SetRxFiFo(&hpcd_USB_OTG_HS, 0x200);
    HAL_PCDEx_SetTxFiFo(&hpcd_USB_OTG_HS, 0, 0x40);
    HAL_PCDEx_SetTxFiFo(&hpcd_USB_OTG_HS, 1, 0x40);
    HAL_PCDEx_SetTxFiFo(&hpcd_USB_OTG_HS, 2, 0x40);
    HAL_PCDEx_SetTxFiFo(&hpcd_USB_OTG_HS, 3, 0x40);

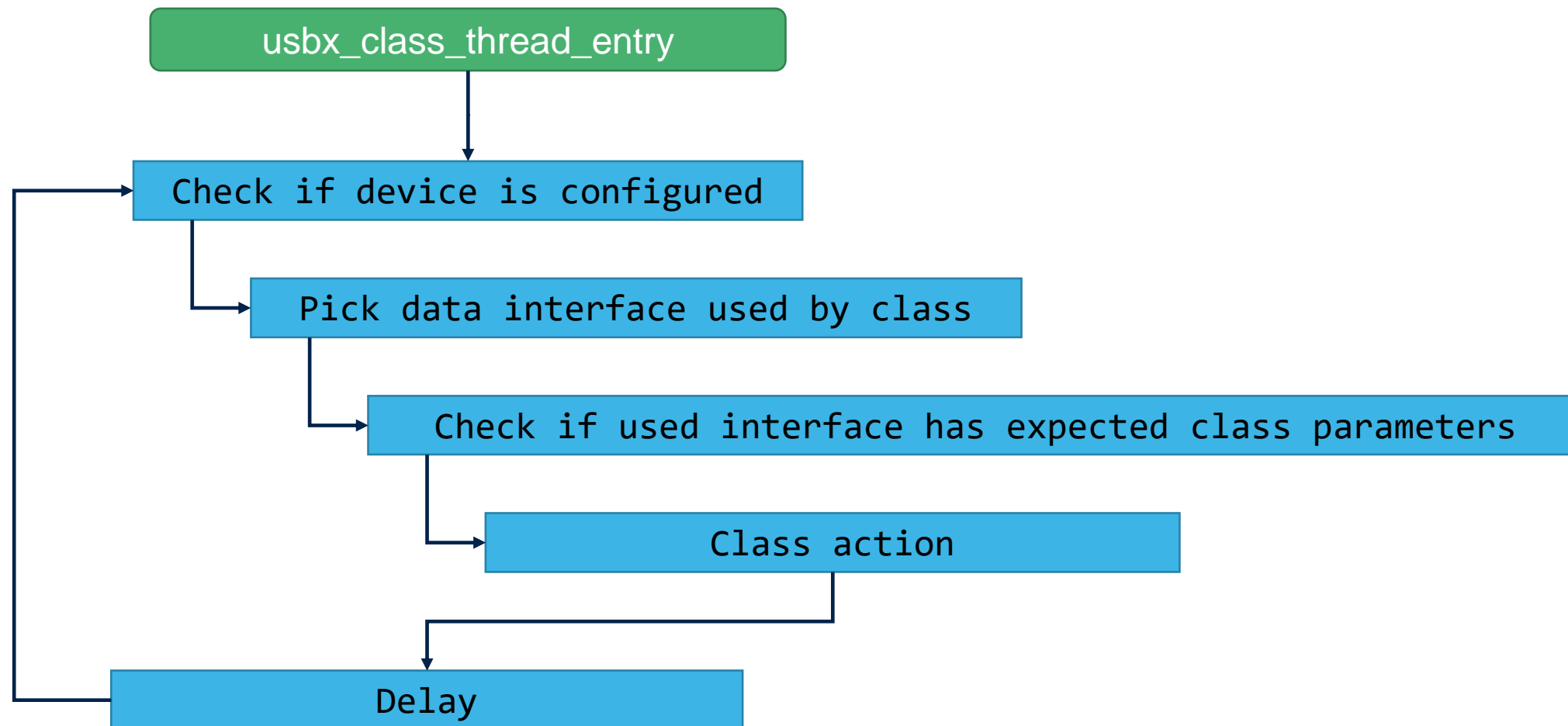
    _ux_dcd_stm32_initialize((ULONG)USB_OTG_HS, (ULONG)&hpcd_USB_OTG_HS);

    HAL_PCD_Start(&hpcd_USB_OTG_HS);
}
```

```
void usbx_app_thread_entry(ULONG arg)
{
    /* Sleep for 100 ms */
    tx_thread_sleep(0.1 * TX_TIMER_TICKS_PER_SECOND);
    MX_USB_Device_Init();
}
```



# USBX class thread function flow







# USBX class thread function flow

```
void usbx_class_thread_entry(ULONG arg)
{
    UX_SLAVE_DEVICE *device;
    UX_SLAVE_INTERFACE *data_interface;
    UX_SLAVE_CLASS_CDC_ACM *cdc_acm;
    device = &_ux_system_slave->ux_system_slave_device;
    while (1)
    {
        /* Check if device is configured */
        if (device->ux_slave_device_state == UX_DEVICE_CONFIGURED)
        {
            data_interface = device->ux_slave_device_first_interface[INTERFACE_NUMBER].ux_slave_interface_next_interface;

            ux_status = ux_utility_memory_compare(data_interface->ux_slave_interface_class->ux_slave_class_name,
                _ux_system_slave_class_cdc_acm_name, ux_utility_string_length_get(_ux_system_slave_class_cdc_acm_name));
            if (ux_status == UX_SUCCESS)
            {
                /* Perform class action*/
            }
            tx_thread_sleep(x);
        }
    }
}
```



# USBX class thread get data interface

- Class data interface use same number like was given in `ux_device_stack_class_register` function

```
void usbx_class_thread_entry(ULONG arg)
{
    UX_SLAVE_INTERFACE *data_interface;
    ...
    data_interface = device->ux_slave_device_first_interface[INTERFACE_NUMBER].ux_slave_interface_next_interface;
    ...
}
```

```
ret = ux_device_stack_class_register(_ux_system_slave_class_name,
                                     ux_device_class_entry, 1, INTERFACE_NUMBER,
                                     (VOID *)&class_parameter);
```



# Template changes – ux\_device\_mouse

- Function *usbx\_hid\_thread\_entry* handle HID action when device connected
  - Device connection check
  - Pick correct interface
  - Class action - moves mouse cursor
  - 5 second delay



# Template changes – `ux_device_cdc_acm`

- Function `usbx_cdc_acm_thread_entry` handle CDC data transfer when device connected
  - Echo functionality of this example – read message from host and immediately write it back
  - In more complex application additional write thread should be considered
- Function `ux_app_parameters_change` manage line coding
  - CTS, RTS and other virtual COM port events may be handled here as well



# CDC HID composite memory consumption

- Overall memory consumption (GCC, Optimize for size)

	RAM	Flash
Default STM32CubeMX parameters	27.09 kB	36.22 kB
Optimized STM32CubeMX parameters	14.73 kB	36.20 kB

- USBX and USBX tight threads RAM memory usage

	Description	Allocated memory [bytes]
USBX allocation		15104
Threads memory	cdc_acm_usbx_app_thread_entry	1024
	hid_usbx_app_thread_entry	1024
	main_usbx_app_thread_entry	1024
	ux_slave_hid_thread**	1024**
	Ux_slave_class_cdc_acm_bulkin_thread**	1024**
	Ux_slave_class_cdc_acm_bulkout_thread**	1024**
Total		18224

\*\*allocated already inside USBX allocation



# Application memory consumption

	Description	Allocated memory [bytes]
ux_system_initialize	ux_system memory structures	576
	margin for initial memory pointer	32
ux_device_stack_initialize	memory for the classes	576
	transfer request buffer	272
	pool for the interfaces	192
	pool for the endpoints	576
	endpoint data memory	8256
HID - ux_device_stack_class_register	hid class memory	128
	hid class stack	1040
	hide event array	720
CDC - ux_device_stack_class_register	cdc class memory	576
	Tx and Rx CDC threads	2064
ux_dcd_stm32_initialize	USB peripheral structure	96
Total		15104



# USBX references

- [Microsoft Azure RTOS USBX documentation](#)
- [STM32 wiki USBX](#)
- [Example project in X-CUBE-AZRTOS-H7 package](#)