Embedded Systems

CS 397

TRIMESTER 3, AY 2021/22

Hands-On 1-1
STM32CubeIDE: Program GPIO and USART

Dr. LIAW Hwee Choo

Department of Electrical and Computer Engineering

DigiPen Institute of Technology Singapore

HweeChoo.Liaw@DigiPen.edu

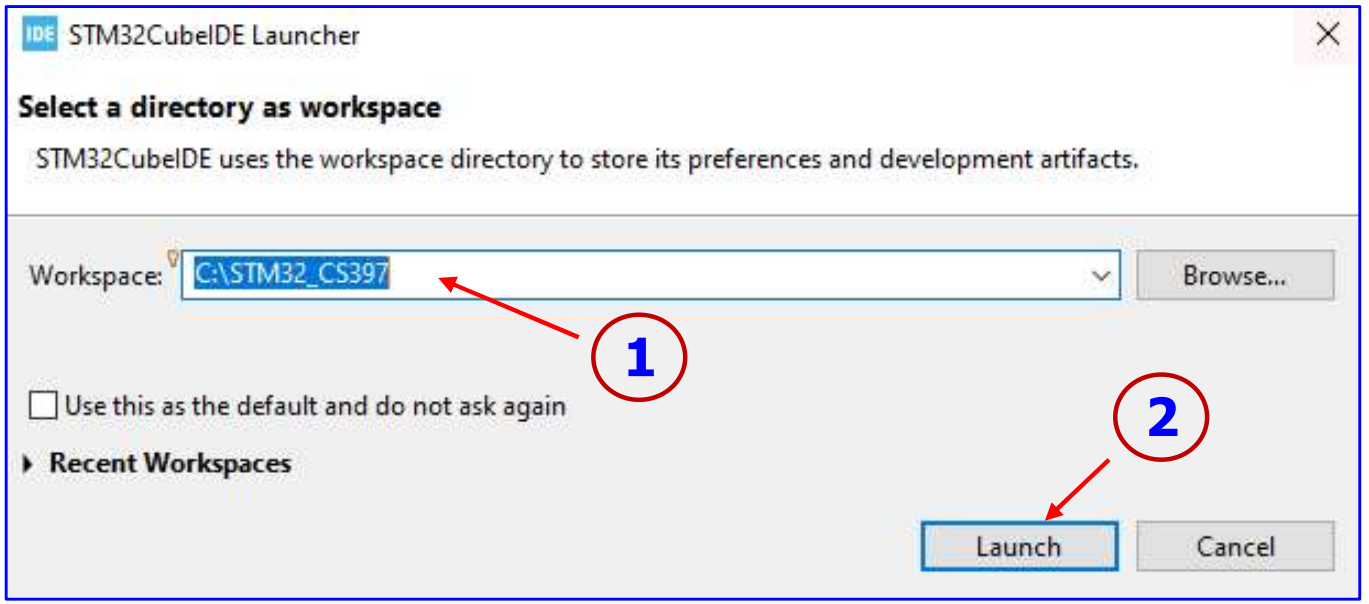
Objectives

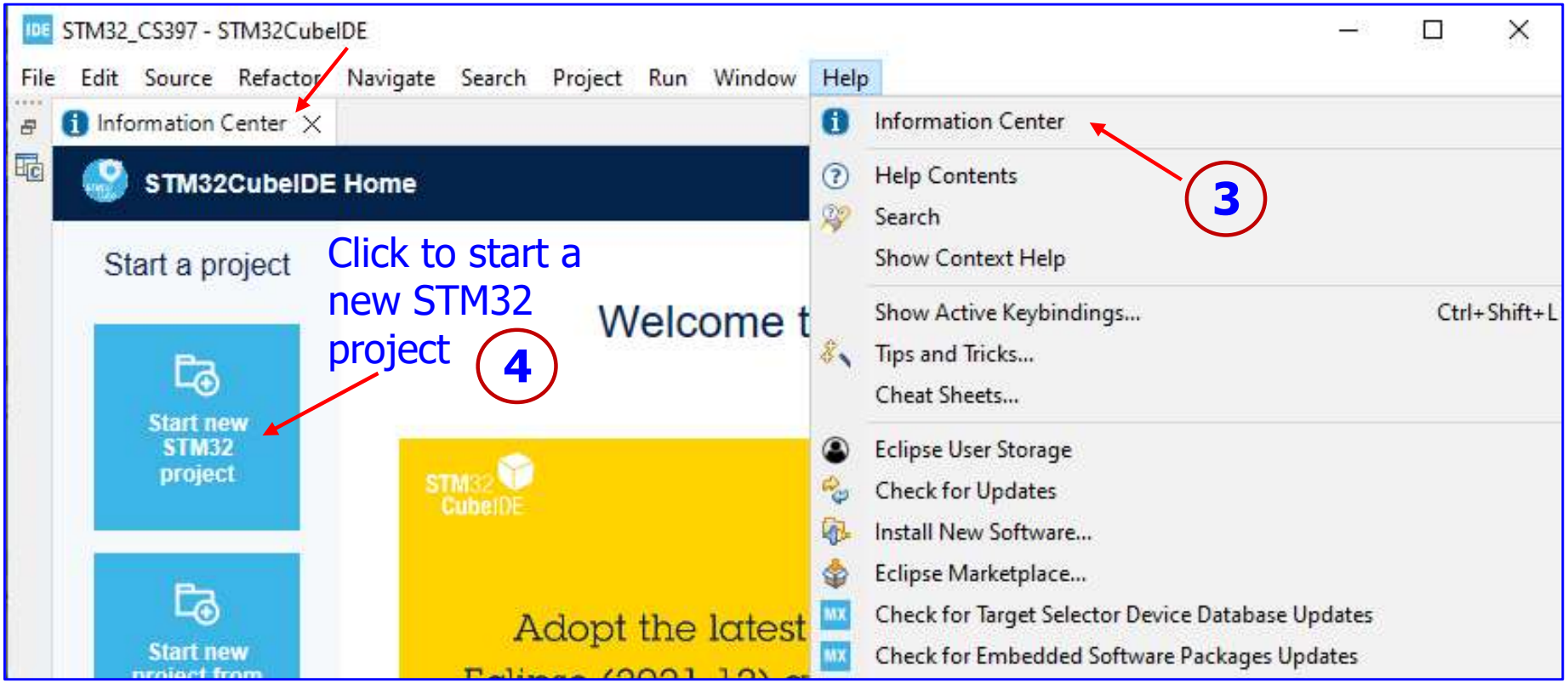
The aims of this session are to

- create a STM32 (STM32CubeIDE) project
- understand the requirements and to set up a development system for embedded applications using the STM32F767 microcontroller
- program the GPIOs including an interrupt for a GPIO input
- program the USART channel and enable it for program debugging
- test developed programs using the "TM Terminal" or "RealTerm", a serial terminal (COM port) software
- build-up the knowledge of embedded microcontrollers and their software development

STM32CubeIDE

Run STM32CubeIDE

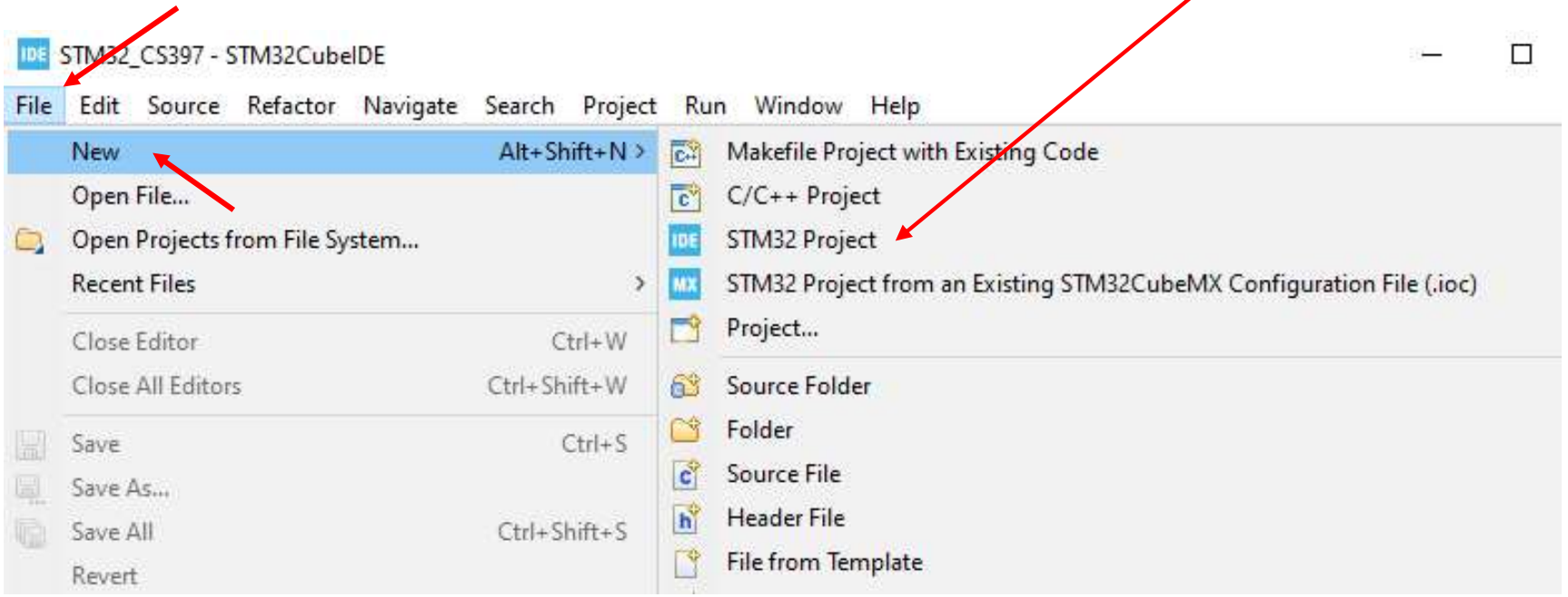




STM32CubeIDE

Or, Run STM32CubeIDE

Click to start a new STM32 project



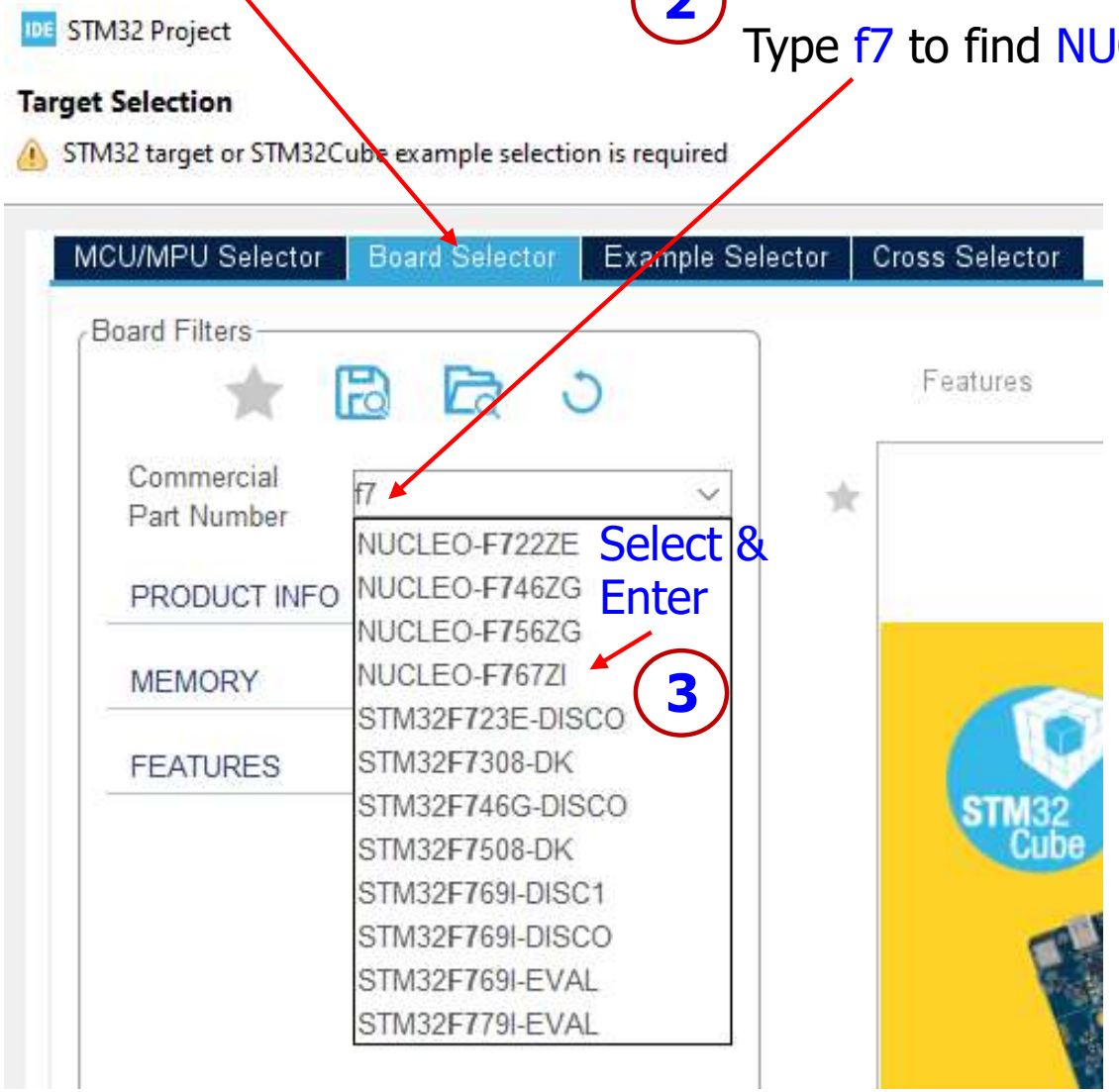
STM32CubeIDE

1

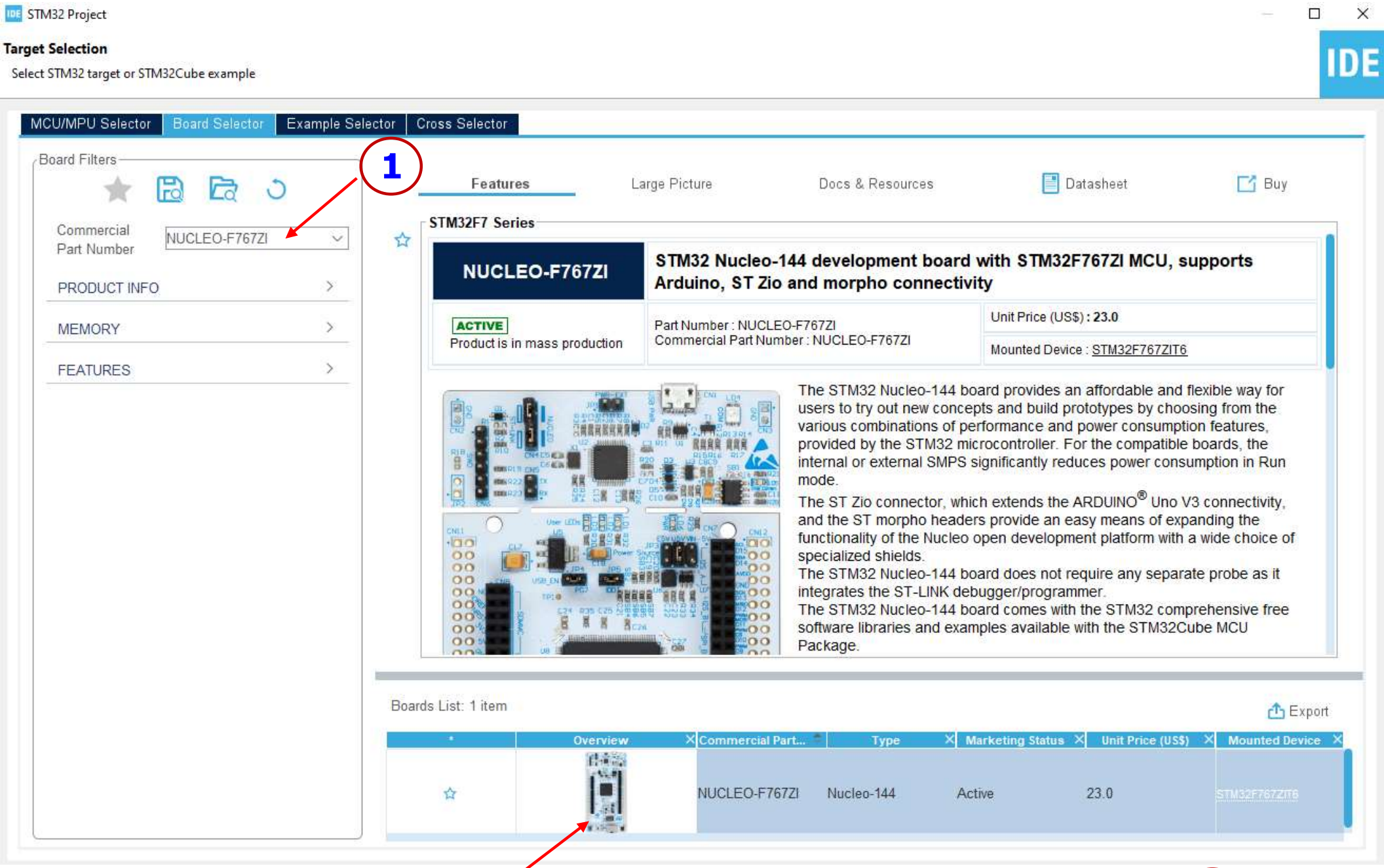
Choose Board Selector

2

Type f7 to find NUCLEO-F767ZI

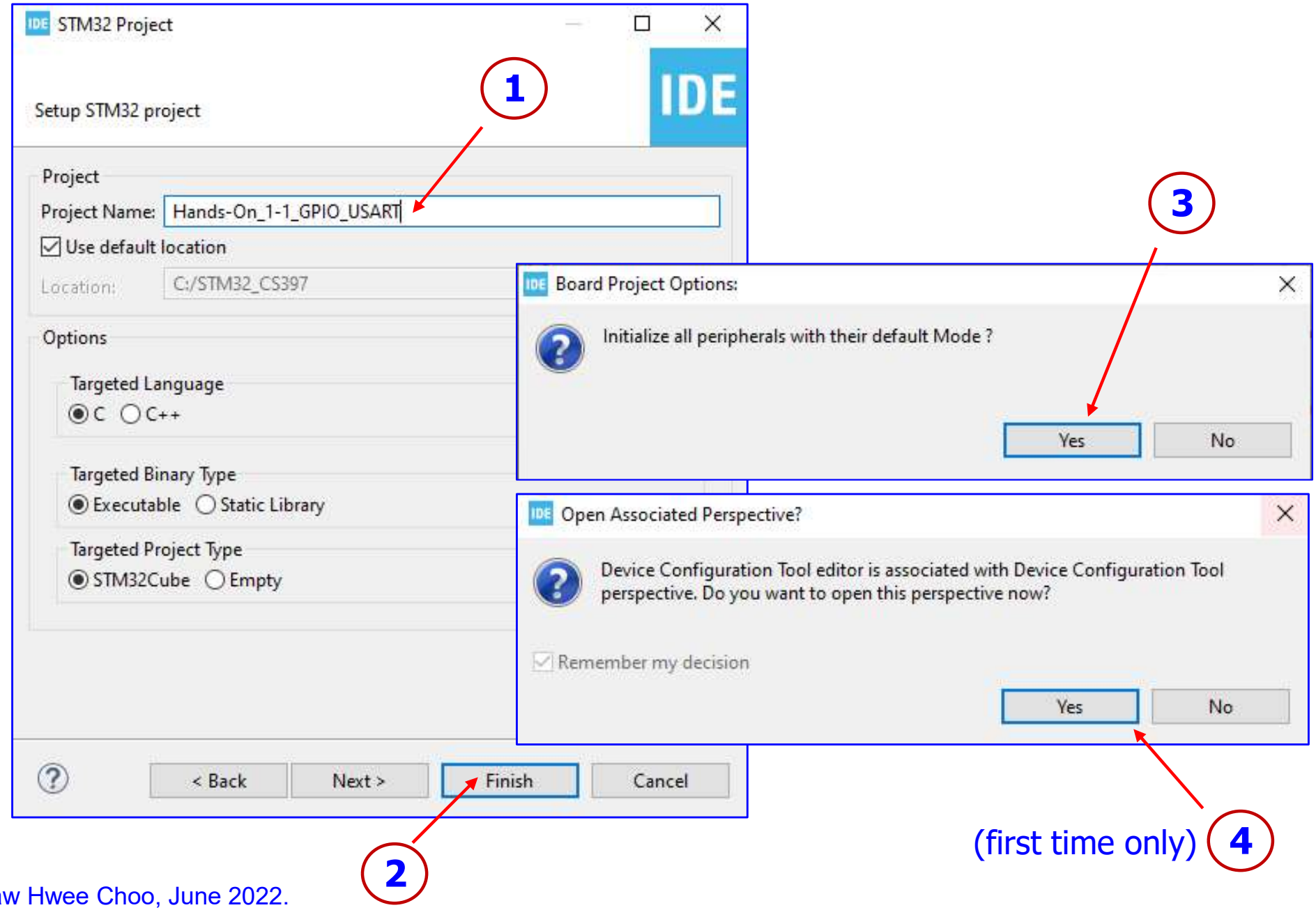


STM32CubeIDE

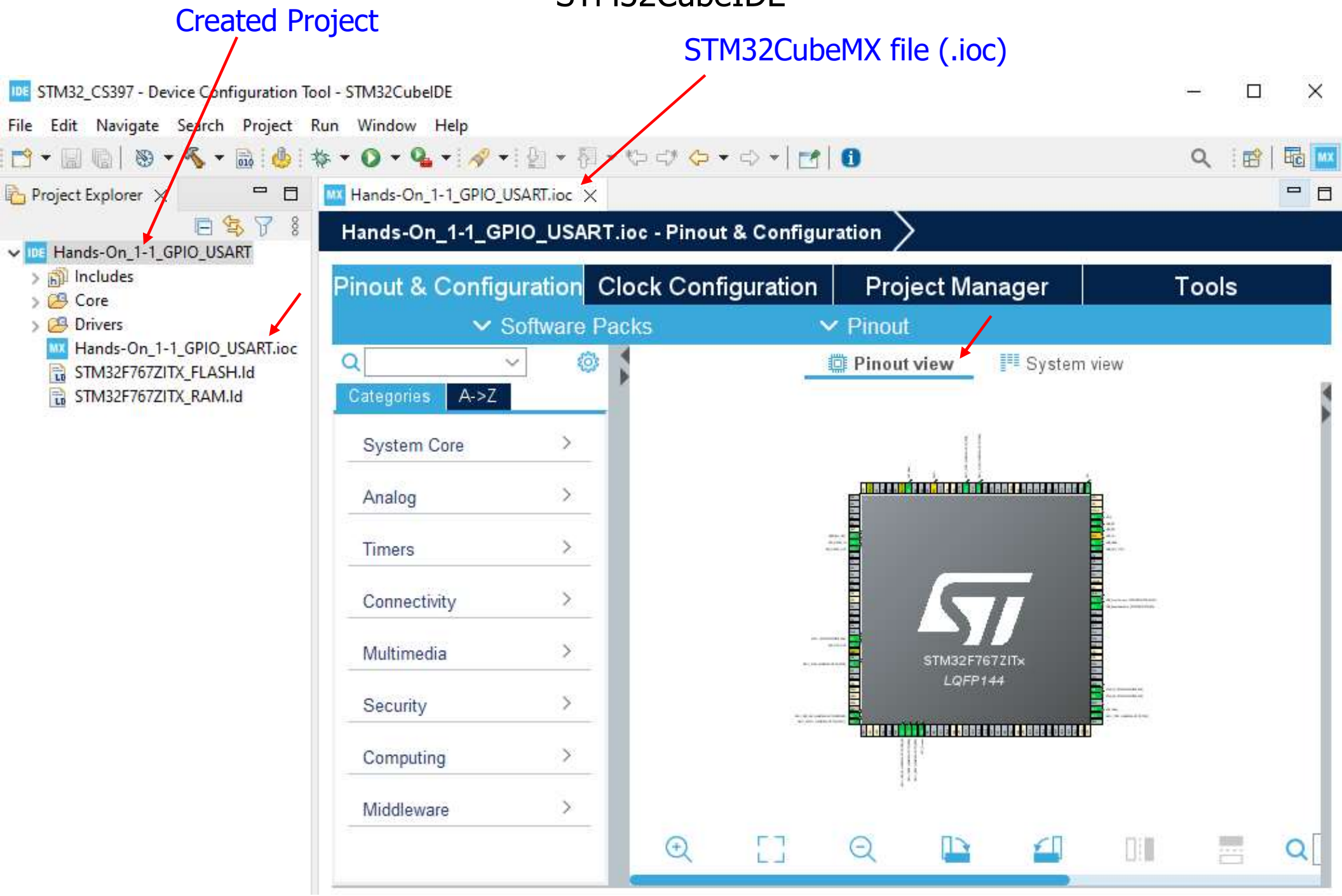


STM32CubeIDE

Enter Project Name: Hands-On_1-1_GPIO_USART

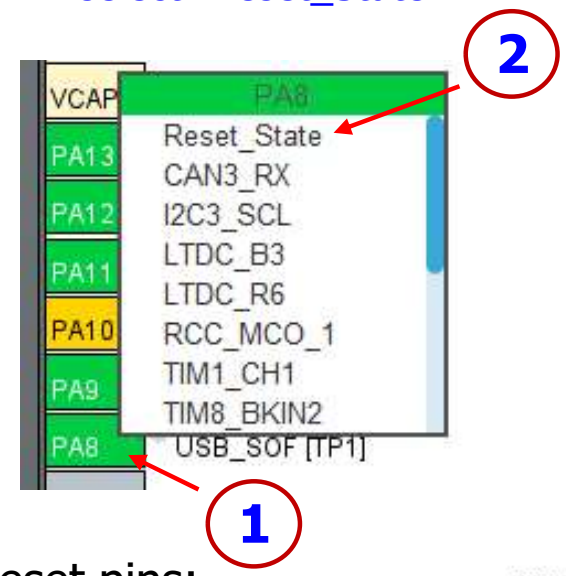


STM32CubeIDE



Reset Unused Pins

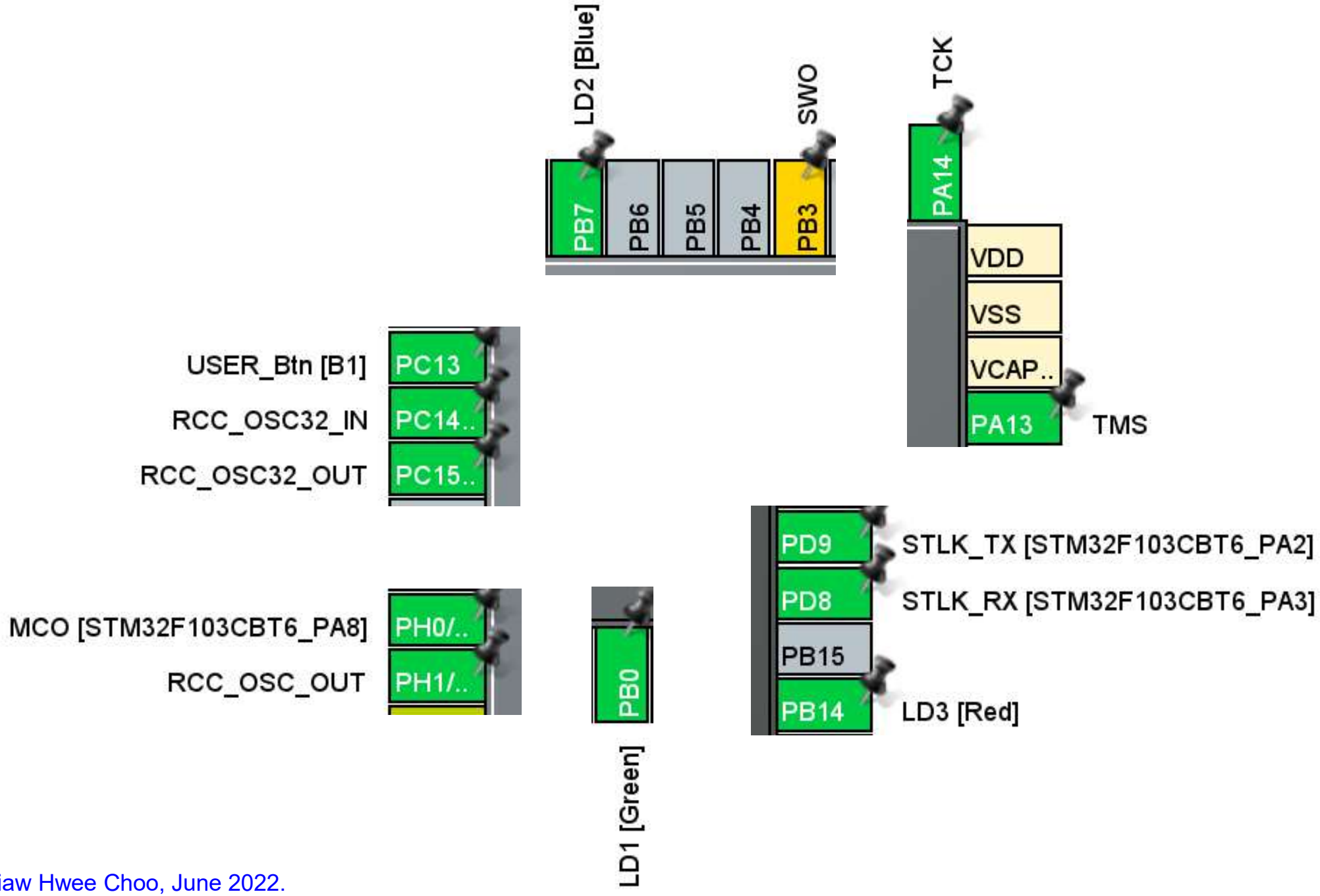
For example, PA8 is reset by
- left-click on the pin PA8, and
- select "Reset_State"



Reset pins:
- USB related:
  PA8, PA9, PA10, PA11,
  PA12, PG6, & PG7.
- ETH (RMII) related:
  PA1, PA2, PA7, PB13,
  PC1, PC4, PC5,
  PG11, & PG13.

Note that do not use the above pins for other functions/purposes.

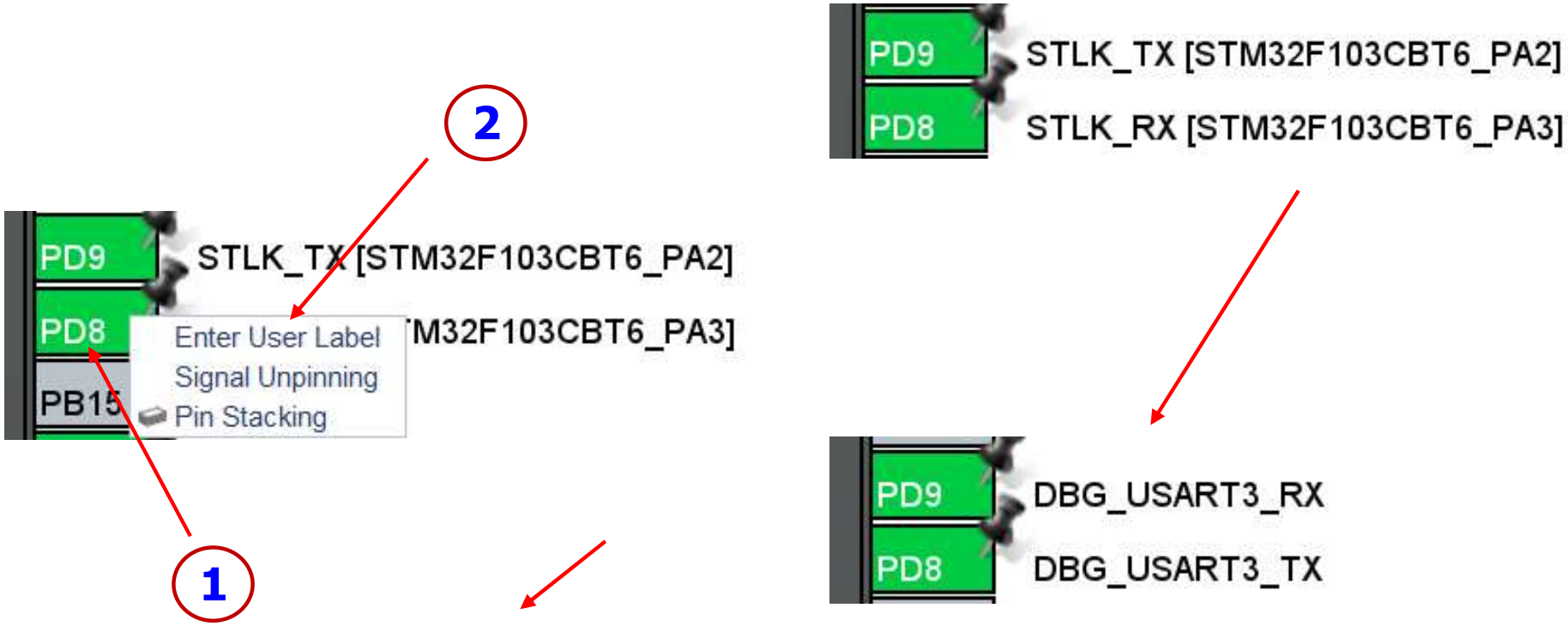
Check the Configured Pins at Pinout View



Rename PD8 and PD9 for USART3

For example, the pin PD8 can be renamed by

1. Right-click on the pin PD8, and
2. Select "Enter User Label"

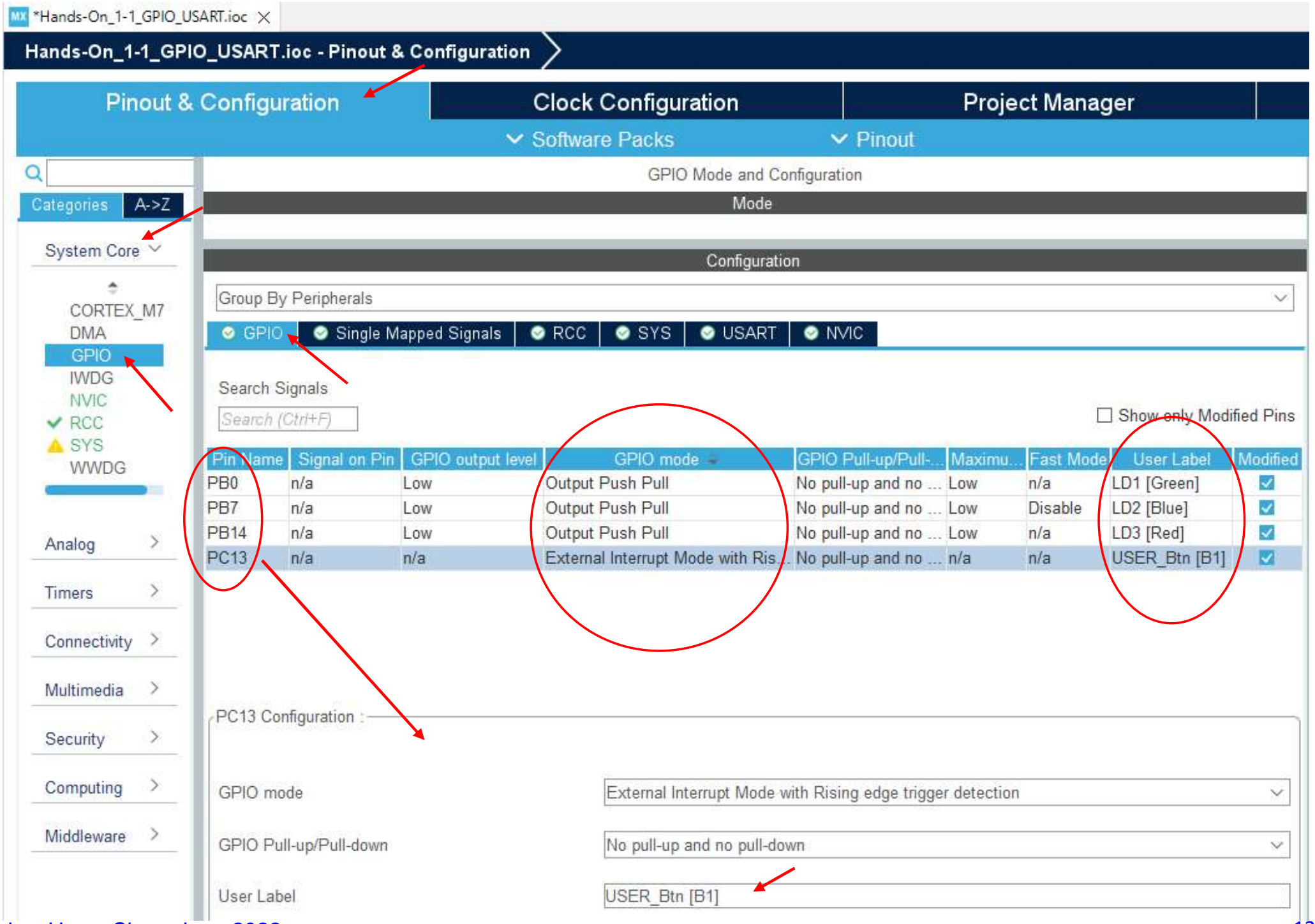


PD9: DBG_USART3_RX
PD8: DBG_USART3_TX

Note: Ctrl-c and Ctrl-v can be used to cut & paste the pin names.

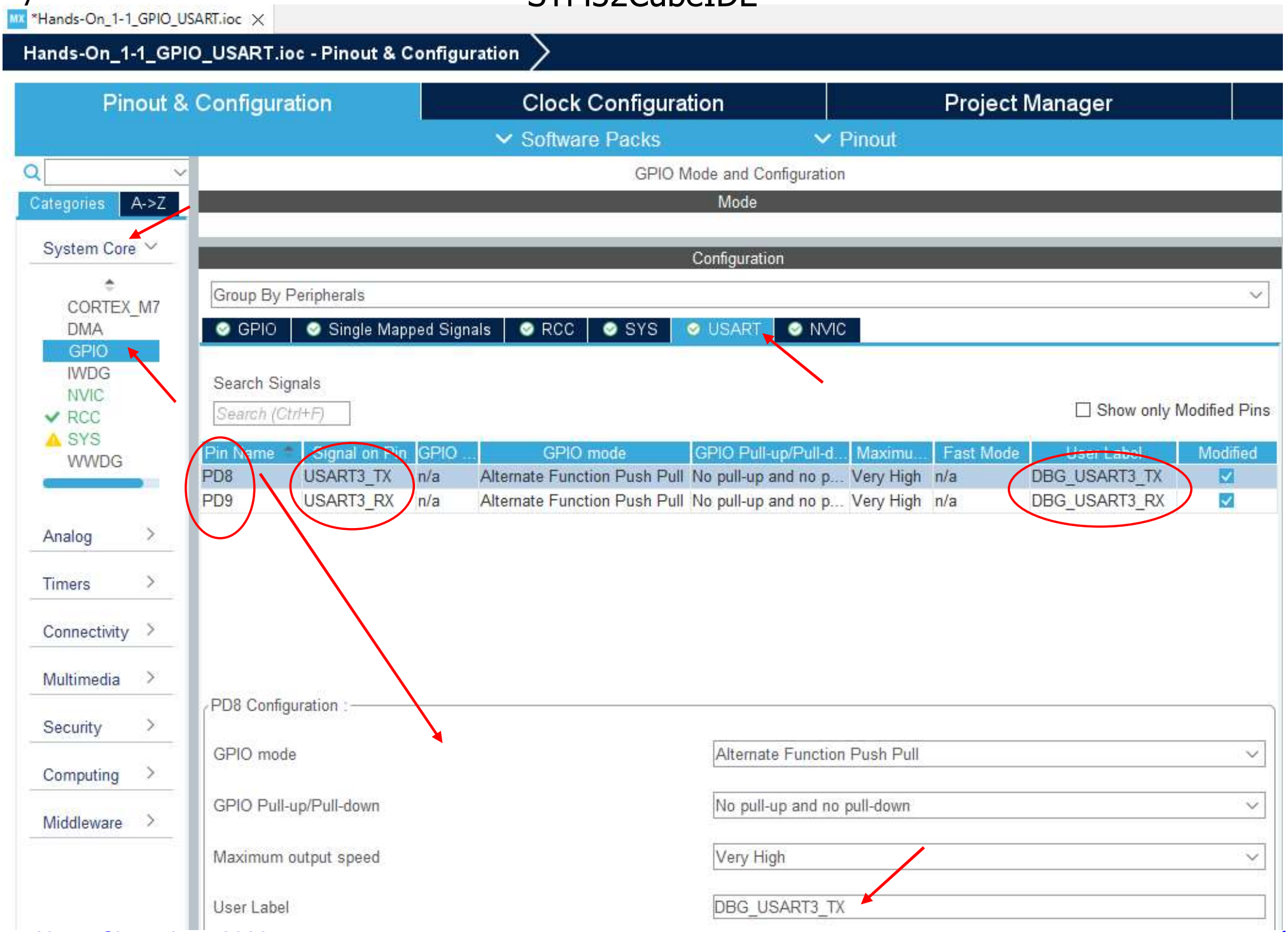
System Core – GPIO – GPIO

STM32CubeIDE



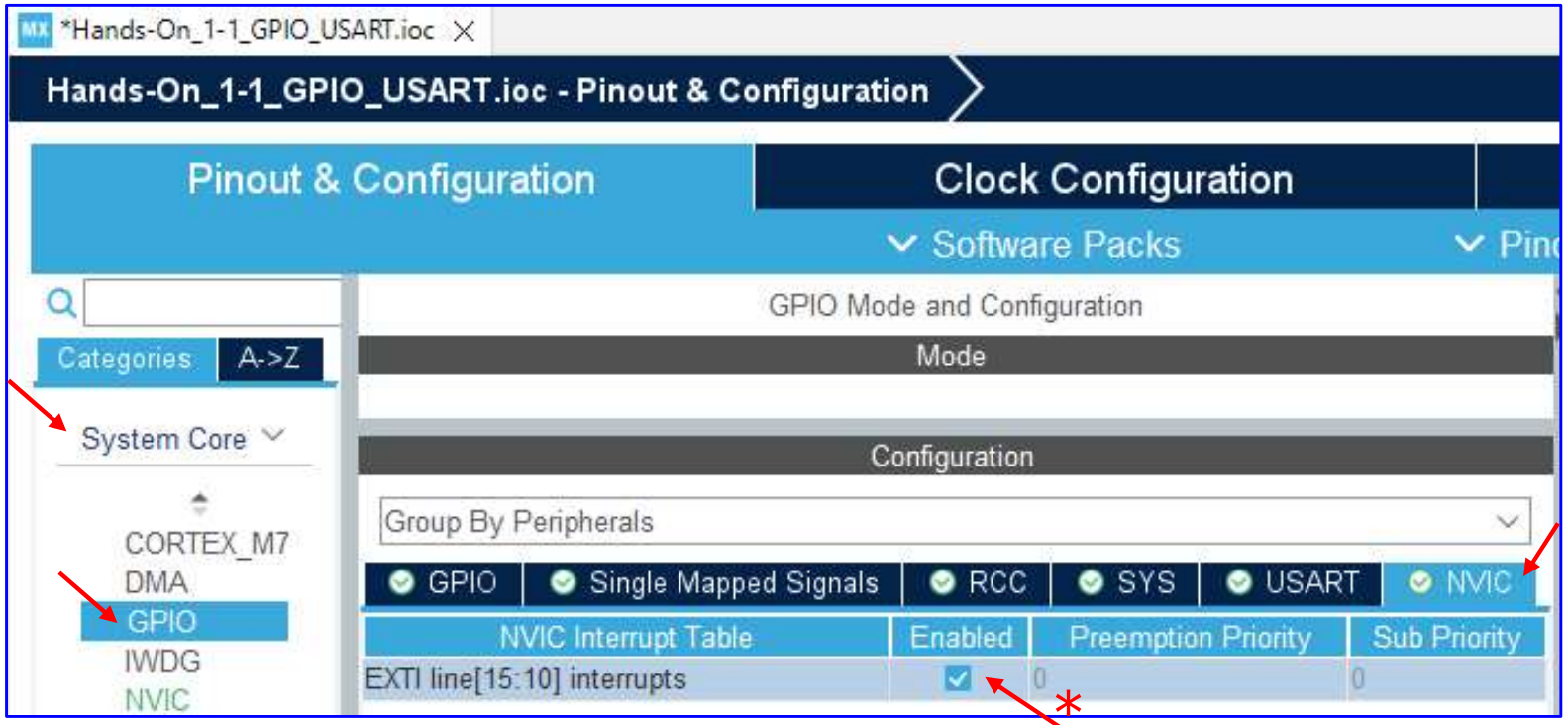
System Core – GPIO – USART

STM32CubeIDE



STM32CubeIDE

Enable the EXTI line[15:10] interrupts for PC13, USER_Btn [B1], input



USER_Btn [B1] PC13

RCC_OSC32_IN PC14..

RCC_OSC32_OUT PC15..

Check the RCC selection

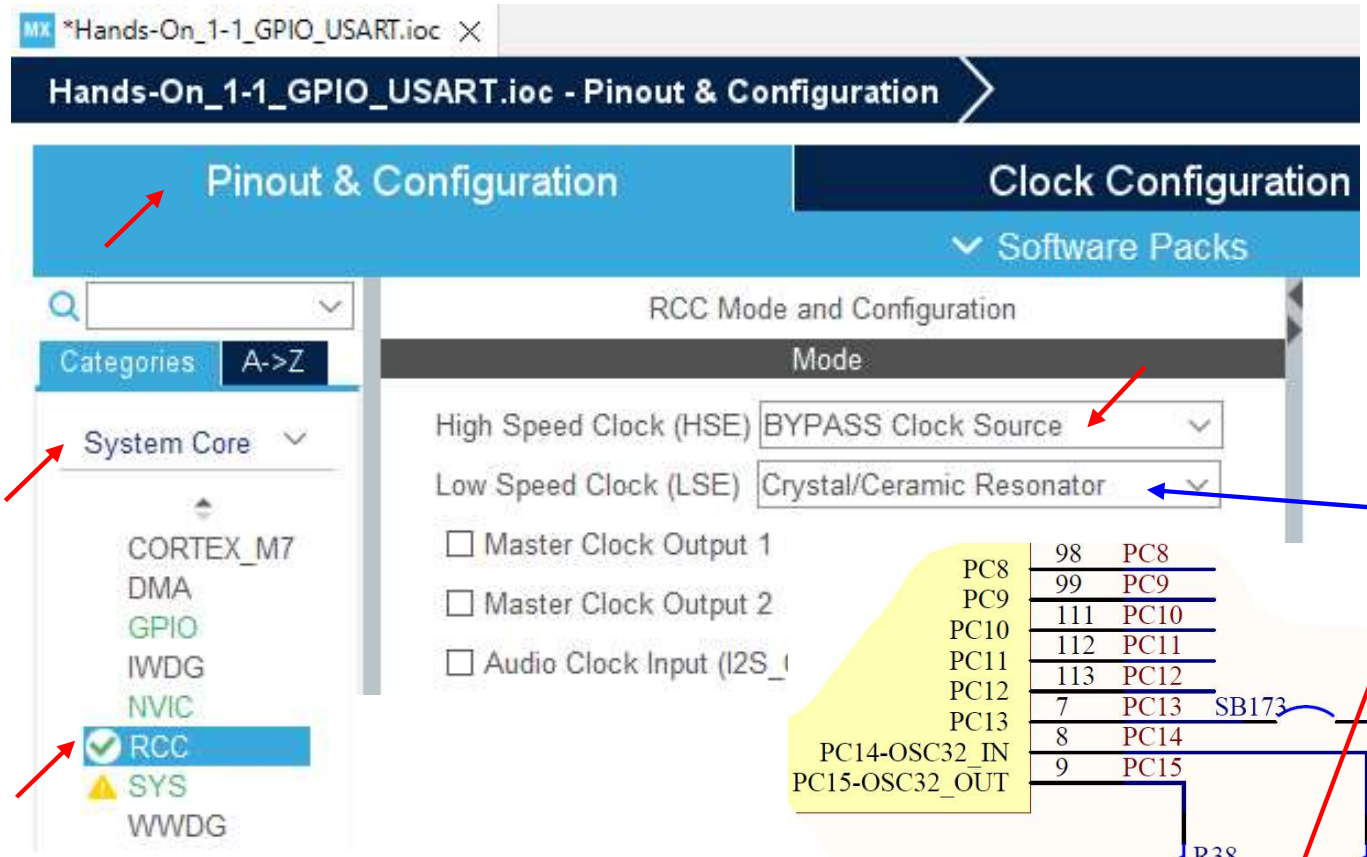
MCO [STM32F103CBT6_PA8] PH0/..

RCC_OSC_OUT PH1/..

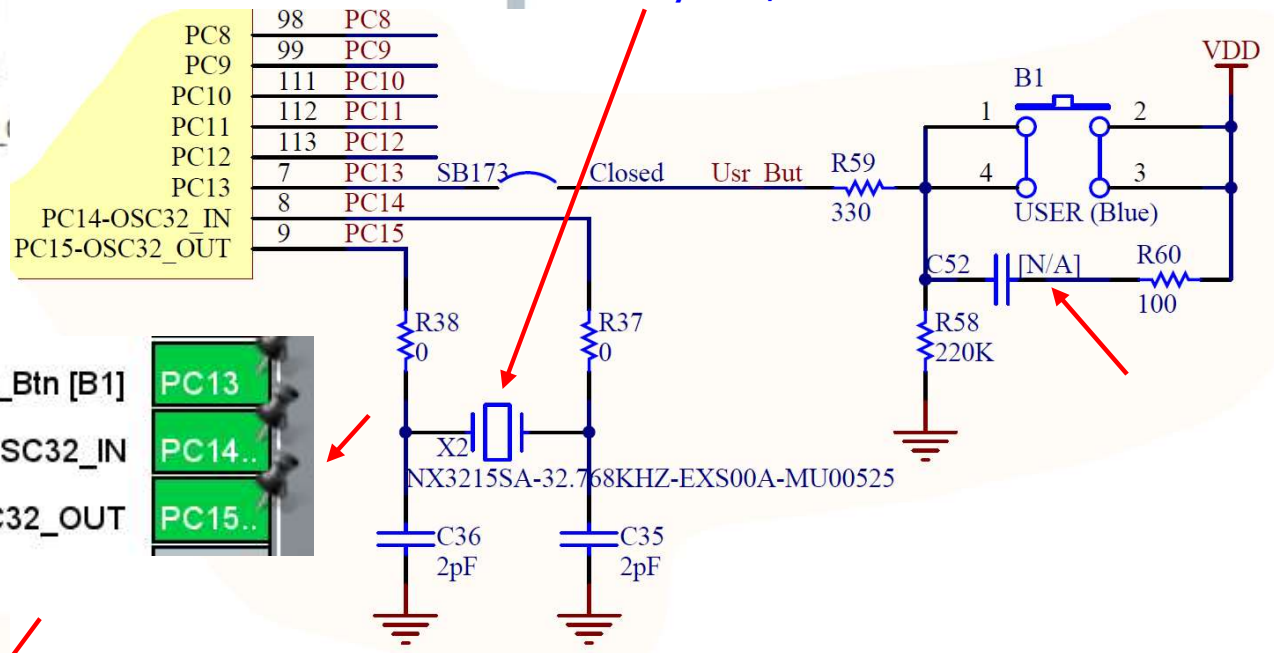
RCC = Reset and Clock Control

STM32CubeIDE

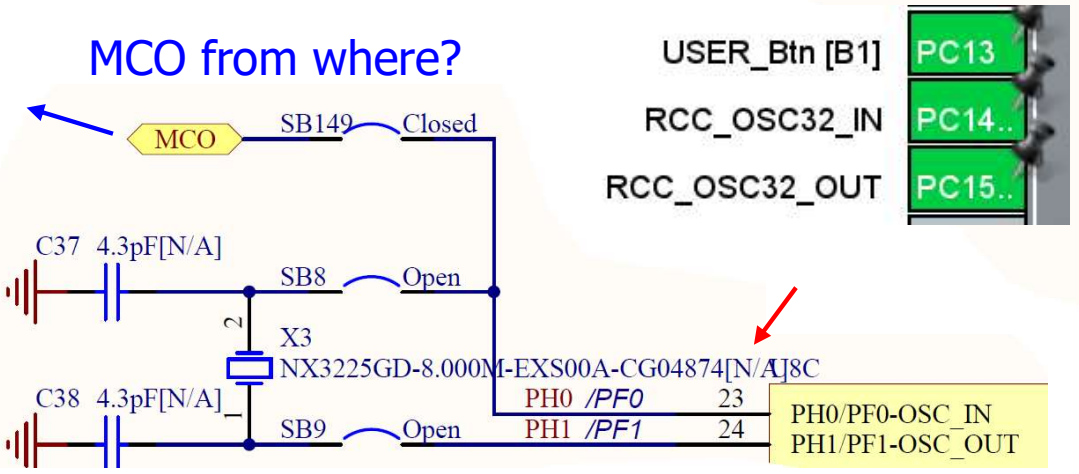
RCC (Reset and Clock Control) Mode and Configuration

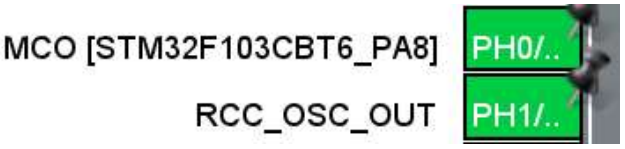


Crystal, LSE = 32.768 kHz



MCO from where?

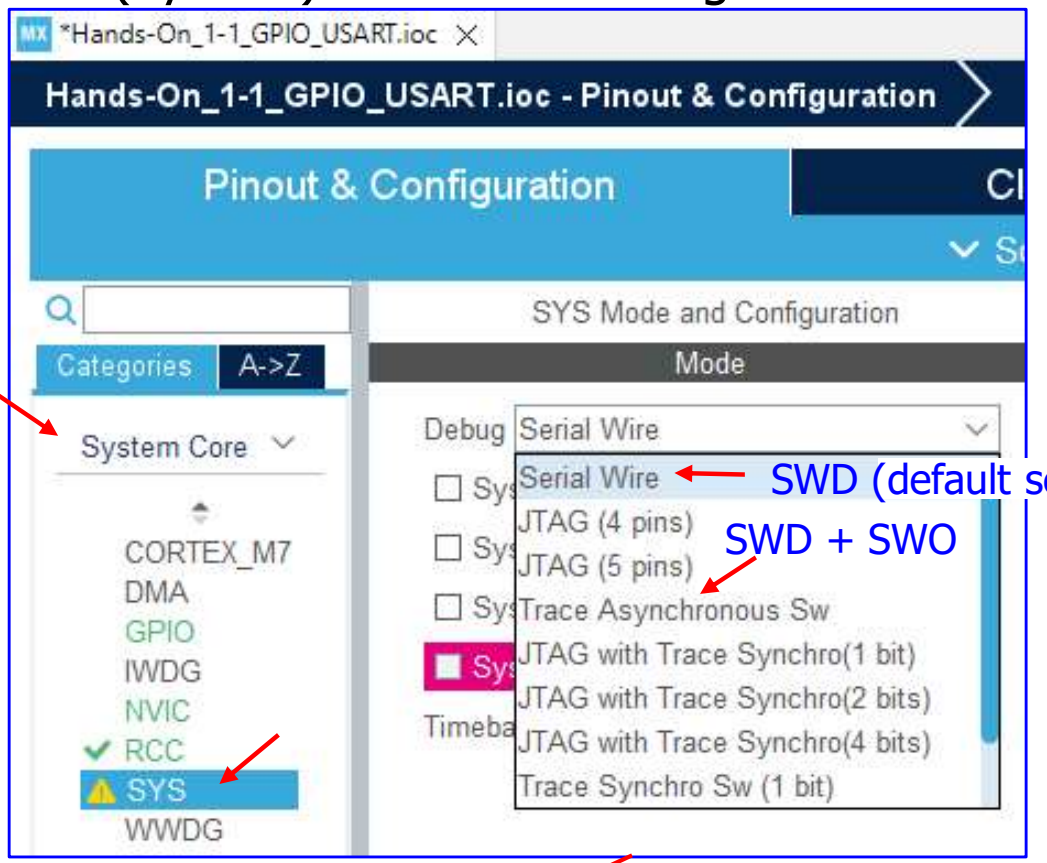


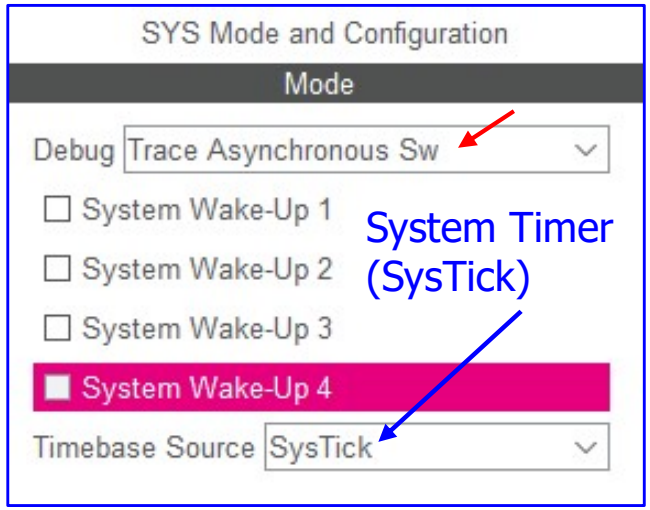


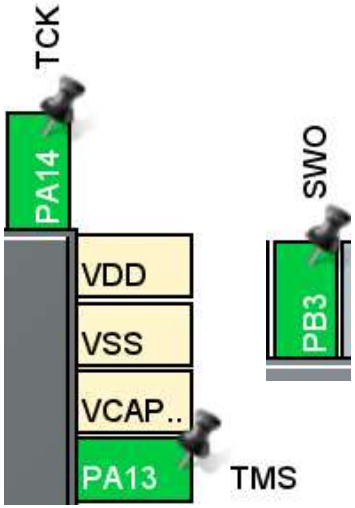
STM32CubeIDE

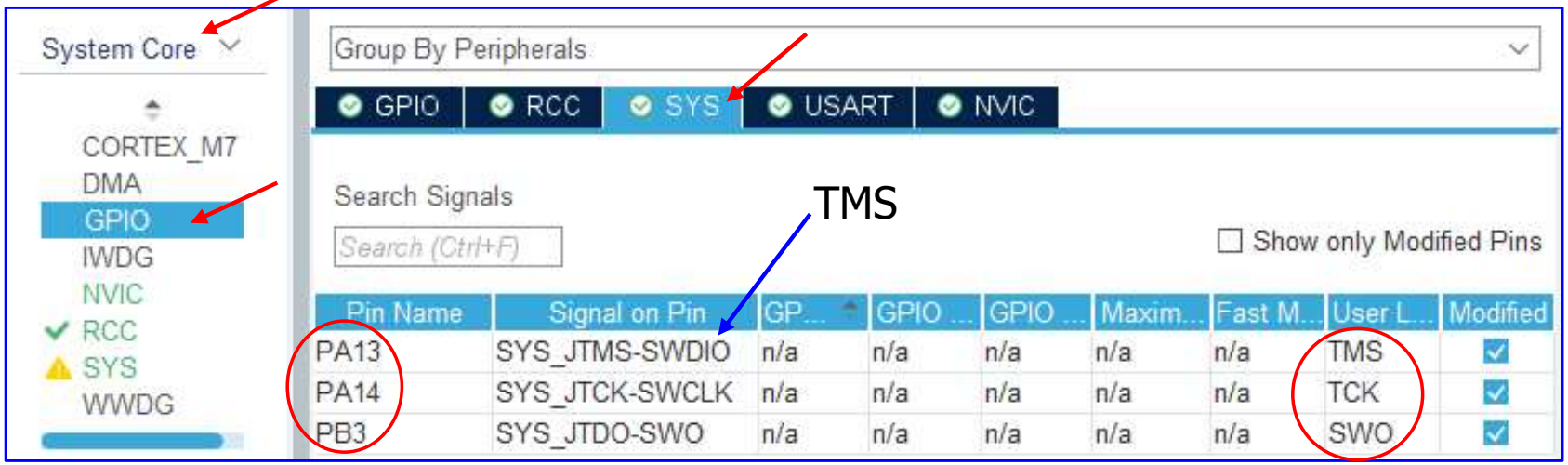
SYS (System) Mode and Configuration

TCK = Test Clock
TMS = Test Mode Select
SWD = Serial Wire Debug
SWO = Single/Serial Wire Output
SWV = Single/Serial Wire Viewer





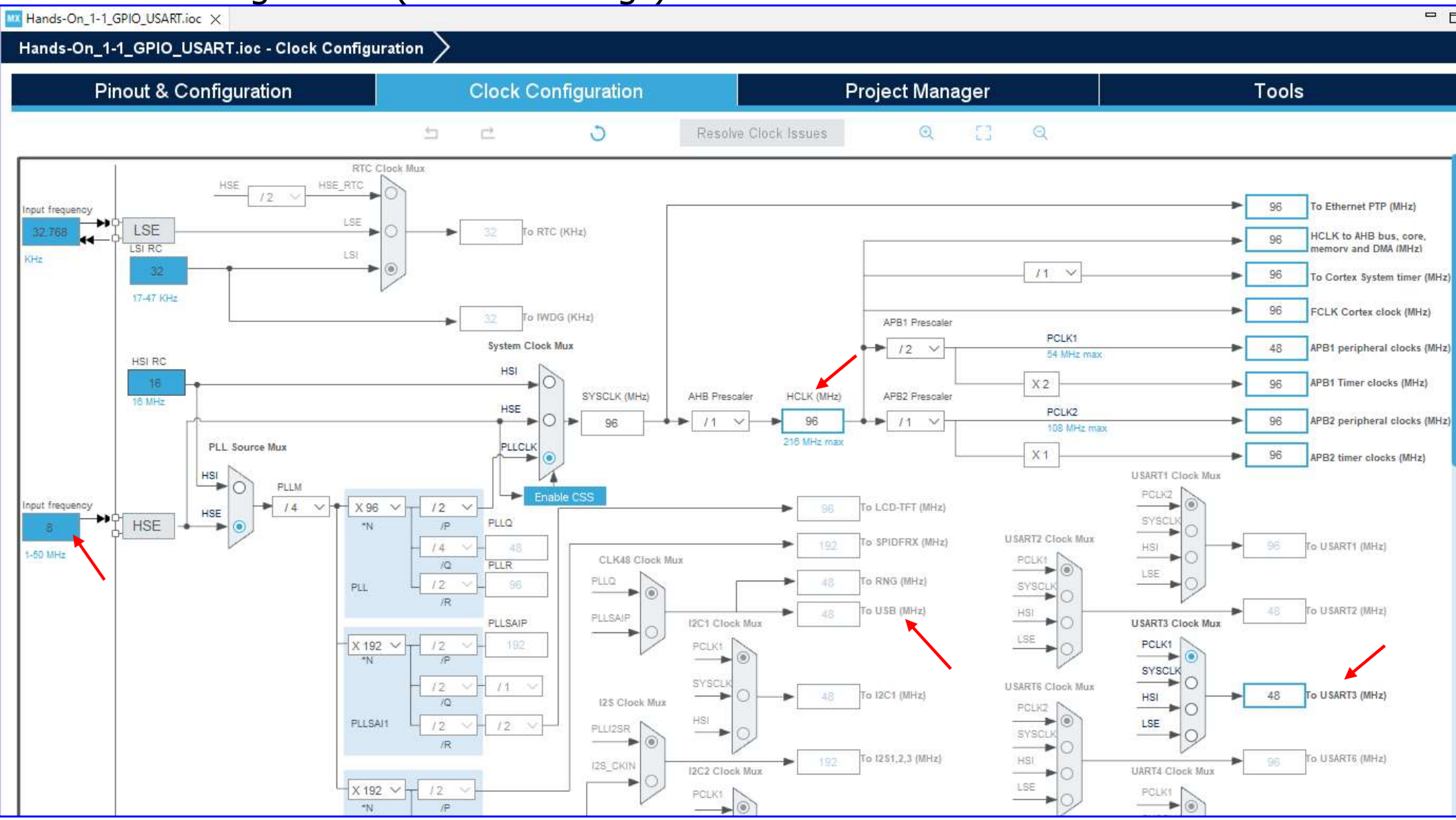




Note: PA13 and PA14 (SWD) are for programming.

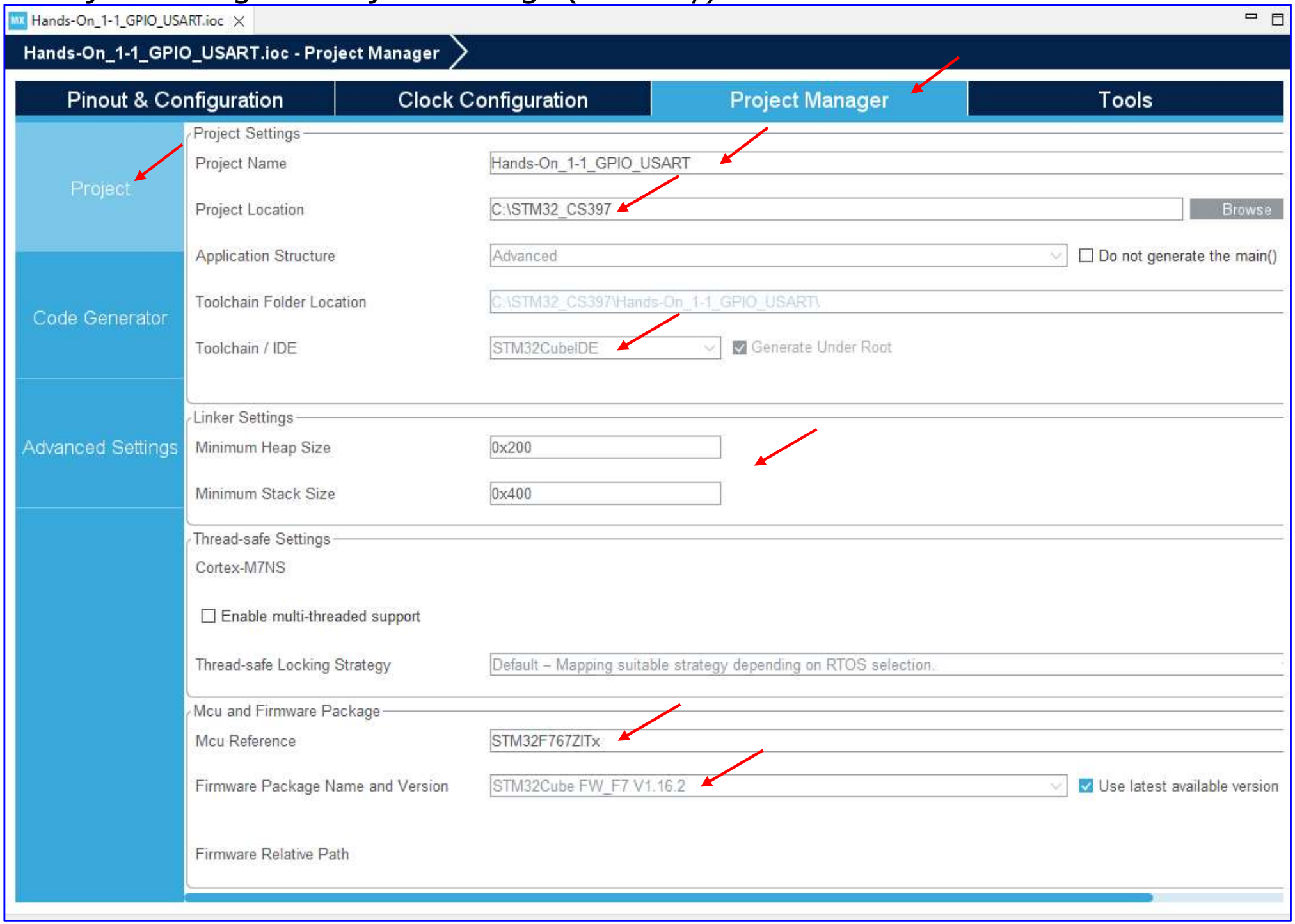
STM32CubeIDE

Clock Configuration (default settings)



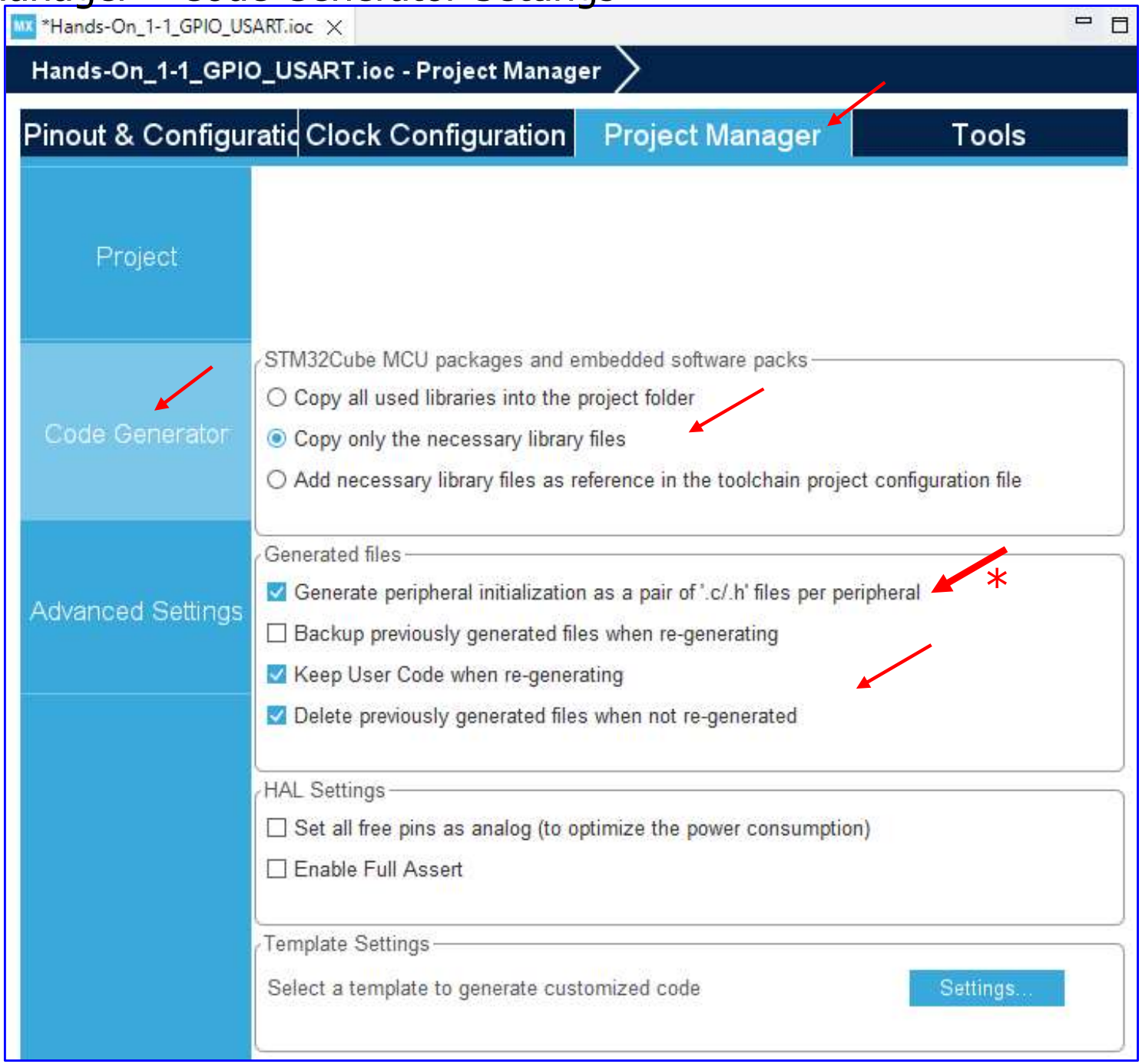
Note: Input Frequency = 8 MHz,
SYSCLK = HCLK (216 MHz max) = PCLK2 (108 MHz max) = 96 MHz,
PCLK1 (54 MHz max) = USART3 Clock = 48 MHz, and
(USB Clock = 48 MHz)

Project Manager – Project Settings (info only) STM32CubeIDE



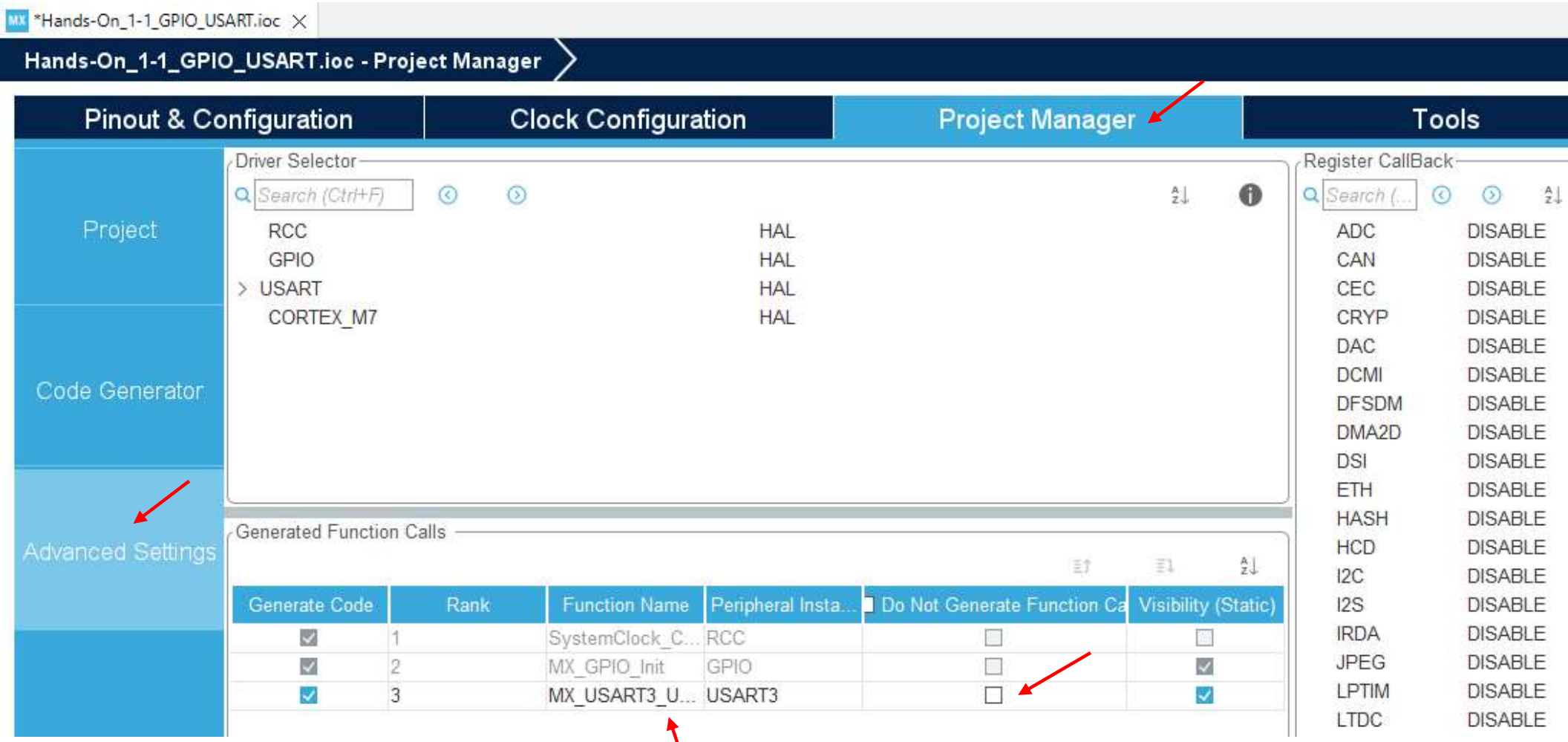
STM32CubeIDE

Project Manager – Code Generator Settings



STM32CubeIDE

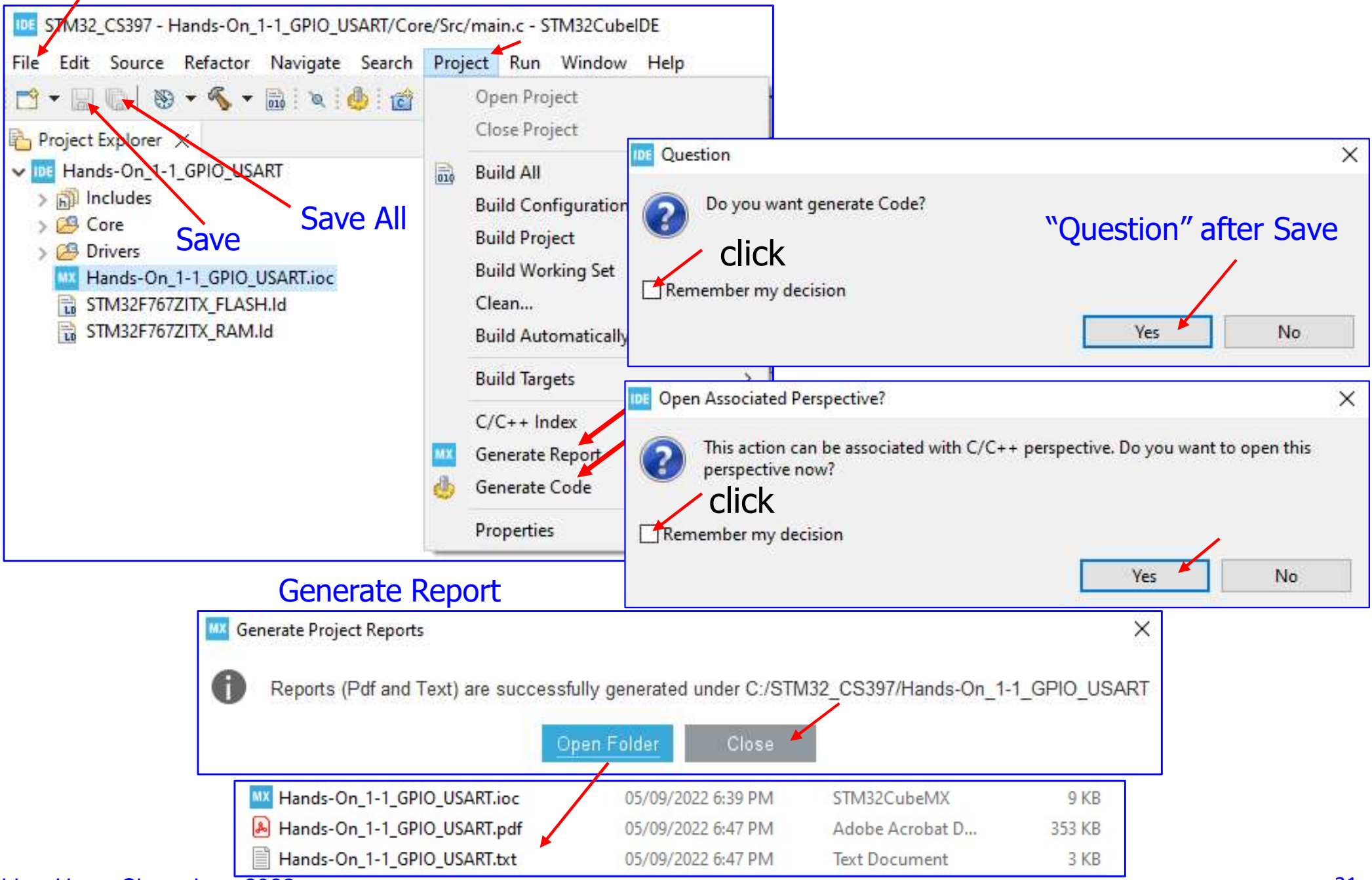
Project Manager – Advanced Settings



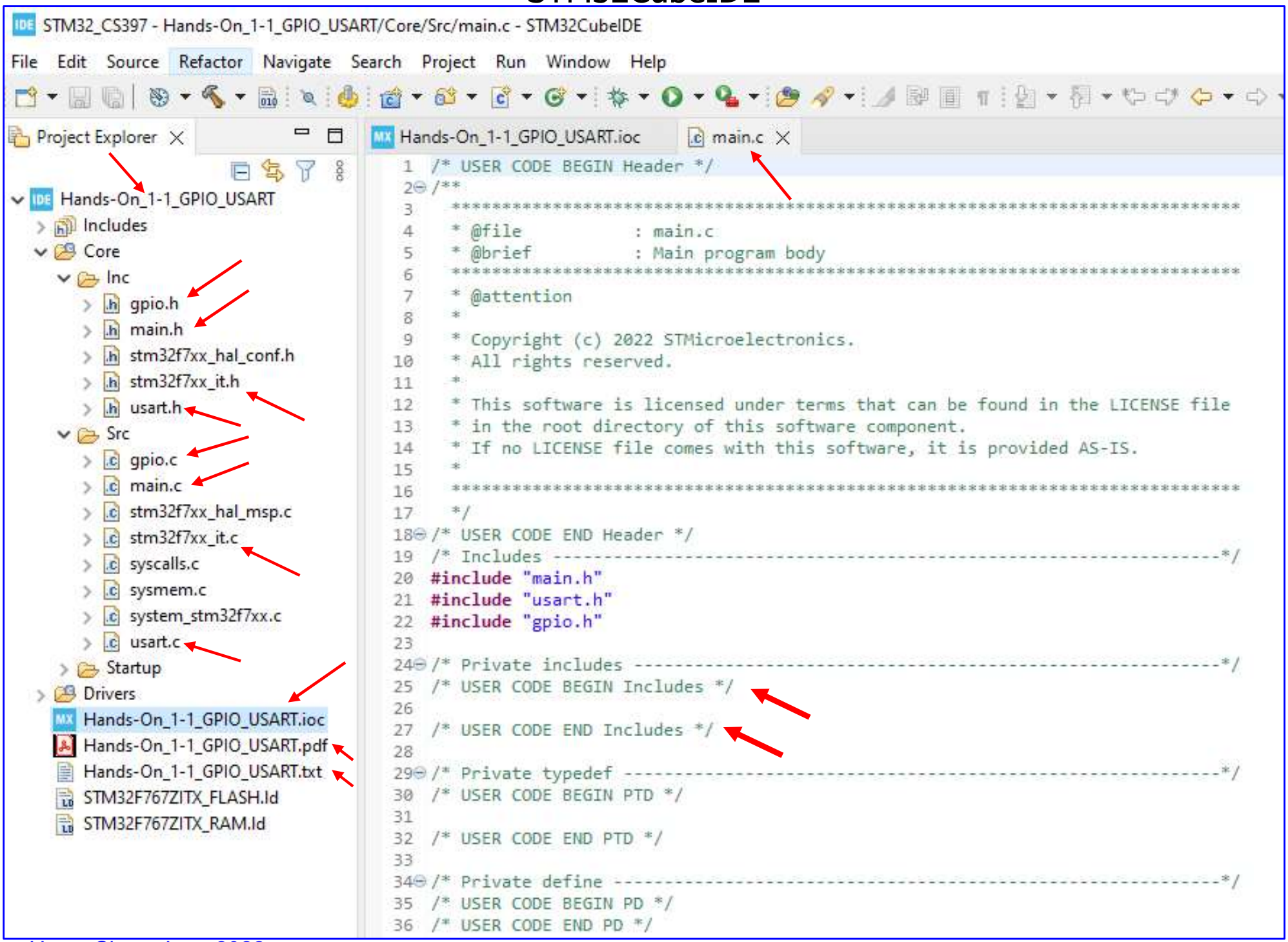
MX_USART3_UART_Init

STM32CubeIDE

Save Project, Generate Report, and Generate Code

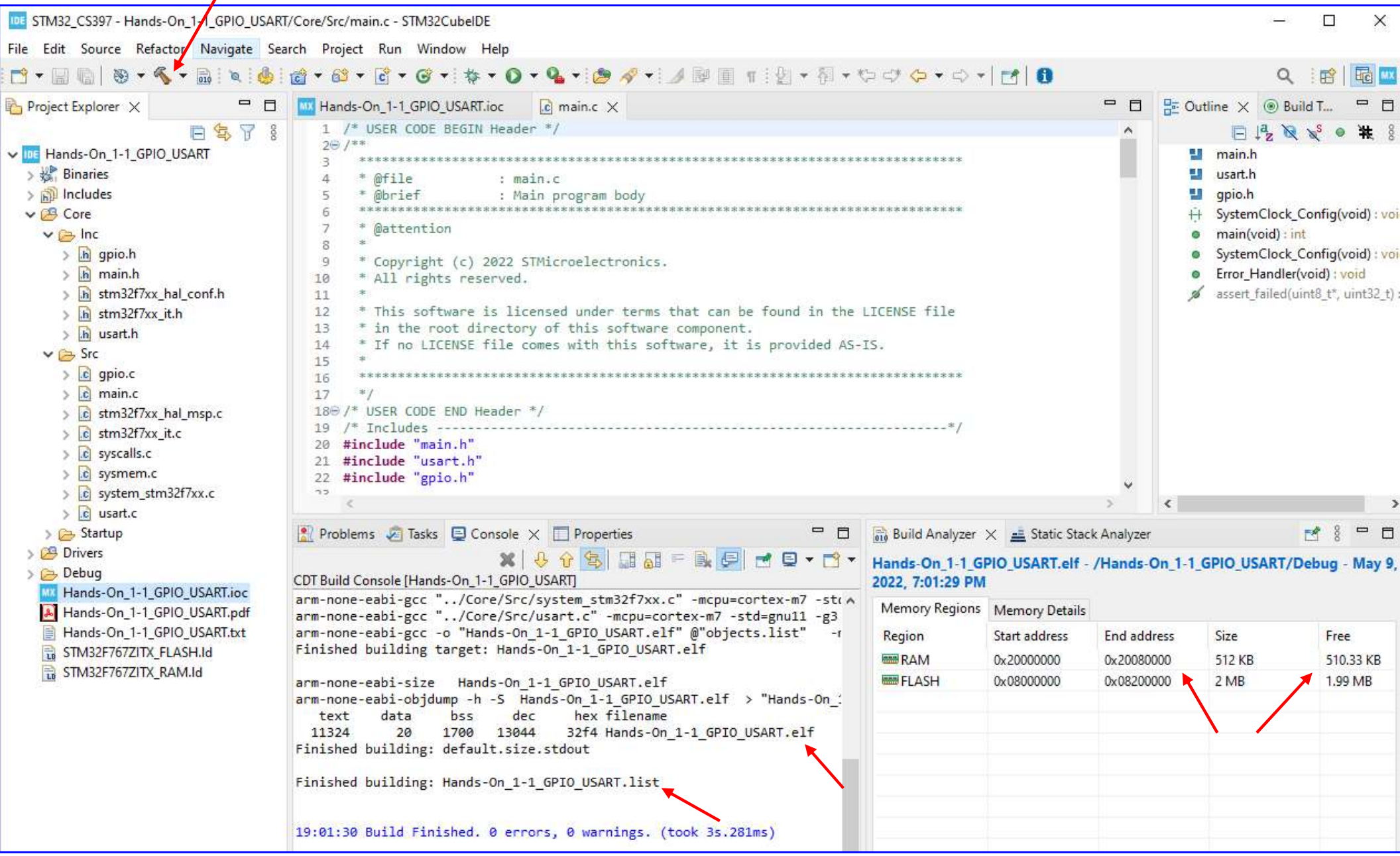


STM32CubeIDE



STM32CubeIDE

Build Project



STM32CubeIDE – gpio.c

```c
/* gpio.c */
/* Includes */
#include "gpio.h"
/* Configure GPIO */
void MX_GPIO_Init(void)
{
    GPIO_InitTypeDef GPIO_InitStruct = {0};
    /* GPIO Ports Clock Enable */
    __HAL_RCC_GPIOC_CLK_ENABLE();
    __HAL_RCC_GPIOH_CLK_ENABLE();
    __HAL_RCC_GPIOB_CLK_ENABLE();
    __HAL_RCC_GPIOD_CLK_ENABLE();
    __HAL_RCC_GPIOA_CLK_ENABLE();

    /*Configure GPIO pin Output Level */
    HAL_GPIO_WritePin(GPIOB, LD1_Pin|LD3_Pin|LD2_Pin,
                      GPIO_PIN_RESET);
    /*Configure GPIO pin : PtPin */
    GPIO_InitStruct.Pin = USER_Btn_Pin;
    GPIO_InitStruct.Mode = GPIO_MODE_IT_RISING;
    GPIO_InitStruct.Pull = GPIO_NOPULL;
    HAL_GPIO_Init(USER_Btn_GPIO_Port, &GPIO_InitStruct);

    /*Configure GPIO pins : PBPin PBPin PBPin */
    GPIO_InitStruct.Pin = LD1_Pin|LD3_Pin|LD2_Pin;
    GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
    GPIO_InitStruct.Pull = GPIO_NOPULL;
    GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
    HAL_GPIO_Init(GPIOB, &GPIO_InitStruct);

    /* EXTI interrupt init*/
    HAL_NVIC_SetPriority(EXTI15_10_IRQn, 0, 0);
    HAL_NVIC_EnableIRQ(EXTI15_10_IRQn);
}
```

```c
/* gpio.h */
/* Define to prevent recursive
inclusion */
#ifndef __GPIO_H__
#define __GPIO_H__

#ifdef __cplusplus
  extern "C" {
#endif

#include "main.h"

void MX_GPIO_Init(void);

#ifdef __cplusplus
}
#endif

#endif /*__GPIO_H__ */
```

Hover mouse pointer on this function to find its detail.

STM32CubeIDE – gpio.c & gpio.h

```
/**
  * @brief  Sets the priority of an interrupt.
  * @param  IRQn External interrupt number.
  *         This parameter can be an enumerator of IRQn_Type enumeration
  *         (For the complete STM32 Devices IRQ Channels list, please refer to the
  *          appropriate CMSIS device file (stm32f7xxxx.h))
  * @param  PreemptPriority The preemption priority for the IRQn channel.
  *         This parameter can be a value between 0 and 15
  *         A lower priority value indicates a higher priority
  * @param  SubPriority the subpriority level for the IRQ channel.
  *         This parameter can be a value between 0 and 15
  *         A lower priority value indicates a higher priority.
  * @retval None
  */
void HAL_NVIC_SetPriority(IRQn_Type IRQn, uint32_t PreemptPriority, uint32_t SubPriority)
{
    uint32_t prioritygroup = 0x00;

    /* Check the parameters */
    assert_param(IS_NVIC_SUB_PRIORITY(SubPriority));
    assert_param(IS_NVIC_PREEMPTION_PRIORITY(PreemptPriority));

    prioritygroup = NVIC_GetPriorityGrouping();

    NVIC_SetPriority(IRQn, NVIC_EncodePriority(prioritygroup, PreemptPriority, SubPriority));
}
```

STM32CubeIDE
– USART3
(default settings)

Pinout & Configuration | Clock Configuration | Project Mar

Software Packs | Pinout

Categories | A->Z

Connectivity

CAN1
CAN2
CAN3
ETH
FMC
I2C1
I2C2
I2C3
I2C4
MDIOS
QUADSPI
SDMMC1
SDMMC2
PI1
PI2
PI3
PI4
SPI5
SPI6
UART4
UART5
UART7
UART8
USART1
USART2
USART3
USART6
USB_OTG_FS
USB_OTG_HS

USART3 Mode and Configuration

Mode

Mode: Asynchronous

Hardware Flow Control (RS232): Disable

Hardware Flow Control (RS485)

Configuration

Reset Configuration

NVIC Settings | DMA Settings | GPIO Settings

Parameter Settings | User Constants

Configure the below parameters :

Search (Ctrl+F)

Basic Parameters

Baud Rate	115200 Bits/s
Word Length	8 Bits (including Parity)
Parity	None
Stop Bits	1

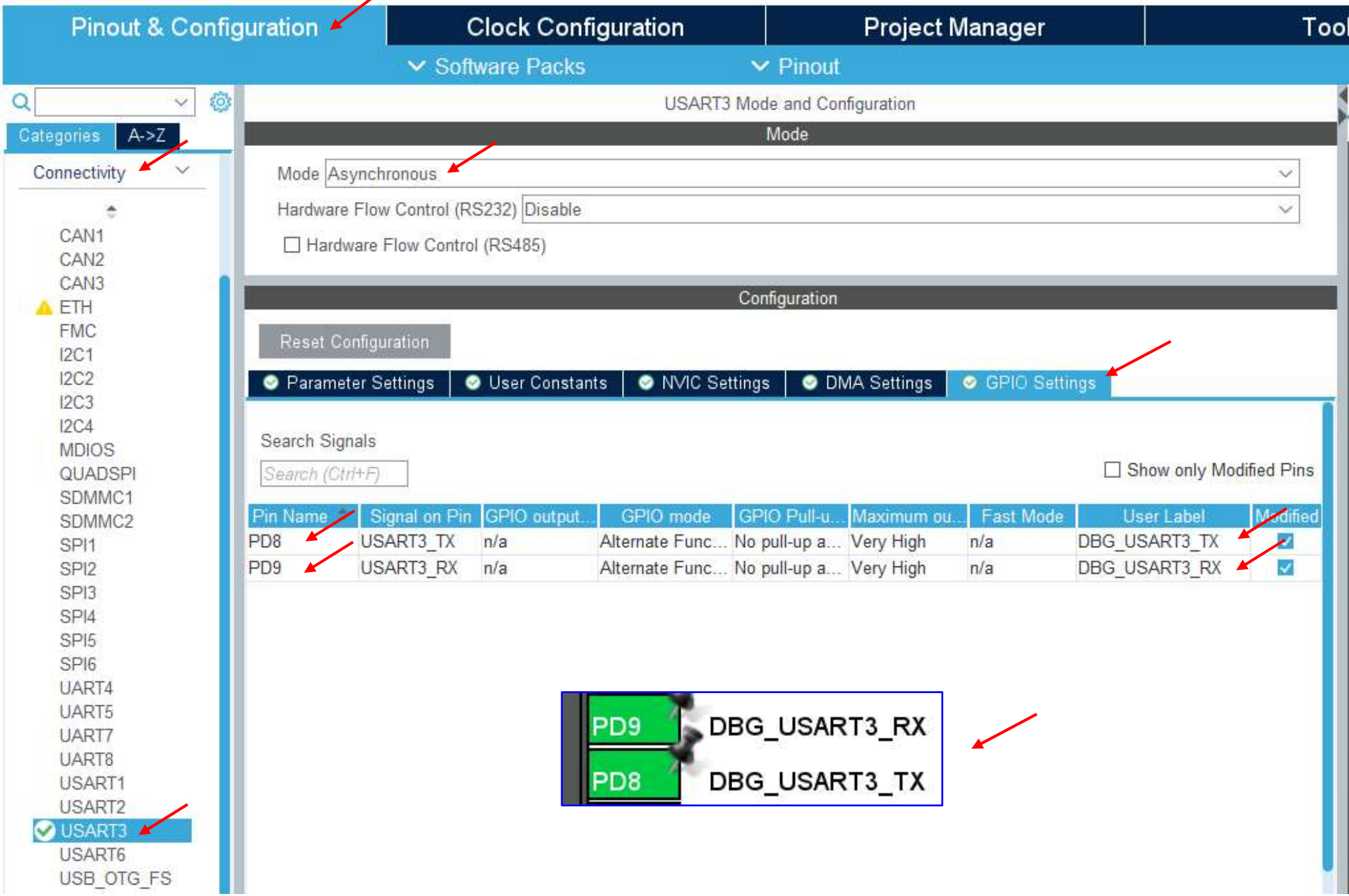
Advanced Parameters

Data Direction	Receive and Transmit
Over Sampling	16 Samples
Single Sample	Disable

Advanced Features

Auto Baudrate	Disable
TX Pin Active Level Inversion	Disable
RX Pin Active Level Inversion	Disable
Data Inversion	Disable
TX and RX Pins Swapping	Disable
Overrun	Enable
DMA on RX Error	Enable
MSB First	Disable

STM32CubeIDE – USART3

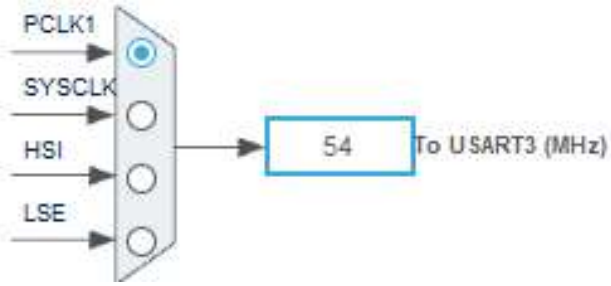


STM32CubeIDE – usart.c

```
/* usart.c */

#include "usart.h"

UART_HandleTypeDef huart3;

/* USART3 init function */
void MX_USART3_UART_Init(void)
{
    huart3.Instance = USART3;
    huart3.Init.BaudRate = 115200;
    huart3.Init.WordLength = UART_WORDLENGTH_8B;
    huart3.Init.StopBits = UART_STOPBITS_1;
    huart3.Init.Parity = UART_PARITY_NONE;
    huart3.Init.Mode = UART_MODE_TX_RX;
    huart3.Init.HwFlowCtl = UART_HWCONTROL_NONE;
    huart3.Init.OverSampling = UART_OVERSAMPLING_16;
    huart3.Init.OneBitSampling = UART_ONE_BIT_SAMPLE_DISABLE;
    huart3.AdvancedInit.AdvFeatureInit = UART_ADVFEATURE_NO_INIT;
    if (HAL_UART_Init(&huart3) != HAL_OK)
    {
        Error_Handler();
    }
}
```

STM32CubeIDE – usart.c & usart.h

```c
void HAL_UART_MspInit(UART_HandleTypeDef* uartHandle)
{
    GPIO_InitTypeDef GPIO_InitStruct = {0};
    RCC_PeriphCLKInitTypeDef PeriphClkInitStruct = {0};
    if(uartHandle->Instance==USART3)
    {
        /* Initializes the peripherals clock */
        PeriphClkInitStruct.PeriphClockSelection = RCC_PERIPHCLK_USART3;
        PeriphClkInitStruct.Usart3ClockSelection = RCC_USART3CLKSOURCE_PCLK1;
        if (HAL_RCCEx_PeriphCLKConfig(&PeriphClkInitStruct) != HAL_OK)
        {
            Error_Handler();
        }
        /* USART3 clock enable */
        __HAL_RCC_USART3_CLK_ENABLE();
        __HAL_RCC_GPIOD_CLK_ENABLE();
        GPIO_InitStruct.Pin = DBG_USART3_TX_Pin|DBG_USART3_RX_Pin;
        GPIO_InitStruct.Mode = GPIO_MODE_AF_PP;
        GPIO_InitStruct.Pull = GPIO_NOPULL;
        GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_VERY_HIGH;
        GPIO_InitStruct.Alternate = GPIO_AF7_USART3;
        HAL_GPIO_Init(GPIOD, &GPIO_InitStruct);
    }
}

void HAL_UART_MspDeInit(UART_HandleTypeDef* uartHandle)
{
    if(uartHandle->Instance==USART3)
    {
        /* Peripheral clock disable */
        __HAL_RCC_USART3_CLK_DISABLE();
        HAL_GPIO_DeInit(GPIOD, DBG_USART3_TX_Pin|DBG_USART3_RX_Pin);
    }
}
```




```c
/* usart.h */
/* Define to prevent recursive
inclusion */
#ifndef __USART_H__
#define __USART_H__

#ifdef __cplusplus
 extern "C" {
#endif

#include "main.h"

extern UART_HandleTypeDef huart3;

void MX_USART3_UART_Init(void);

#ifdef __cplusplus
}
#endif

#endif /*__USART_H__ */
```

STM32CubeIDE – main.h

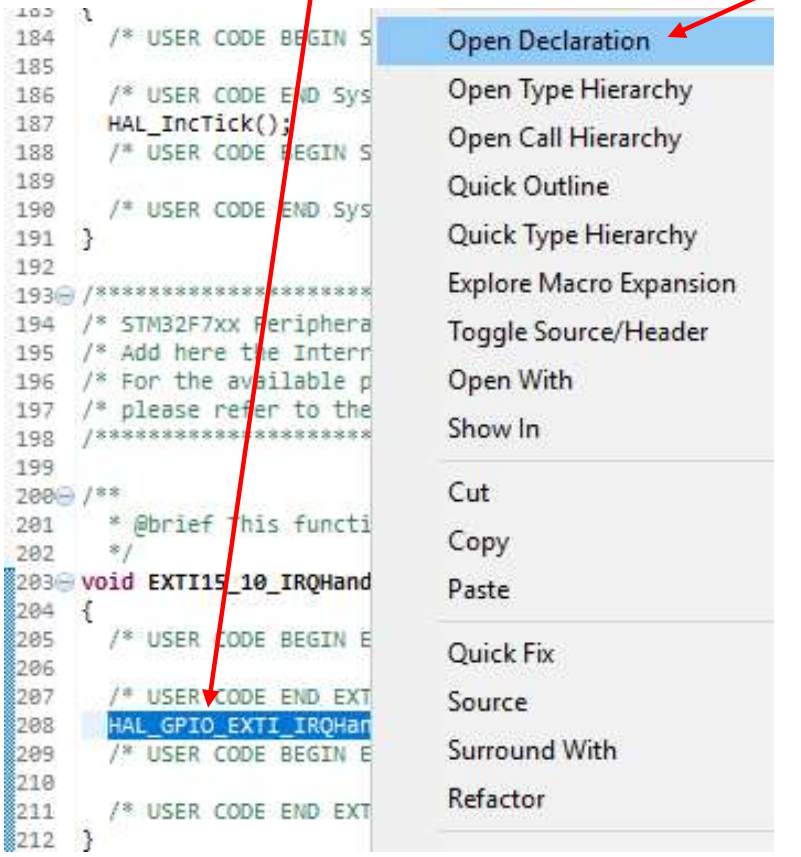
```
/* main.h */
/* Define to prevent recursive inclusion */
#ifndef __MAIN_H
#define __MAIN_H

#ifdef __cplusplus
extern "C" {
#endif

/* Includes */
#include "stm32f7xx_hal.h"

/* Exported functions prototypes */
void Error_Handler(void);

/* Private defines */
#define USER_Btn_Pin GPIO_PIN_13
#define USER_Btn_GPIO_Port GPIOC
#define USER_Btn_EXTI_IRQn EXTI15_10_IRQn
#define MCO_Pin GPIO_PIN_0
#define MCO_GPIO_Port GPIOH
#define LD1_Pin GPIO_PIN_0
#define LD1_GPIO_Port GPIOB
#define LD3_Pin GPIO_PIN_14
#define LD3_GPIO_Port GPIOB
#define DBG_USART3_TX_Pin GPIO_PIN_8
#define DBG_USART3_TX_GPIO_Port GPIOD
#define DBG_USART3_RX_Pin GPIO_PIN_9
#define DBG_USART3_RX_GPIO_Port GPIOD
#define TMS_Pin GPIO_PIN_13
#define TMS_GPIO_Port GPIOA
#define TCK_Pin GPIO_PIN_14
#define TCK_GPIO_Port GPIOA
#define SWO_Pin GPIO_PIN_3
#define SWO_GPIO_Port GPIOB
#define LD2_Pin GPIO_PIN_7
#define LD2_GPIO_Port GPIOB

#ifdef __cplusplus
}
#endif

#endif /* __MAIN_H */
```

STM32CubeIDE – HAL GPIO EXTI Callback

stm32f7xx_it.c (Interrupt Service Routines)

```
/**
  * @brief This function handles EXTI line[15:10]
interrupts.
  */
void EXTI15_10_IRQHandler(void)
{
    HAL_GPIO_EXTI_IRQHandler(USER_Btn_Pin);
}
```

Place the mouse pointer or cursor on this function and right click. Get the pop-up and select "Open Declaration".



stm32f7xx_hal_gpio.c

```
void HAL_GPIO_EXTI_IRQHandler(uint16_t GPIO_Pin)
{
  /* EXTI line interrupt detected */
  if(__HAL_GPIO_EXTI_GET_IT(GPIO_Pin) != RESET)
  {
    __HAL_GPIO_EXTI_CLEAR_IT(GPIO_Pin);
    HAL_GPIO_EXTI_Callback(GPIO_Pin);
  }
}

/* @brief  EXTI line detection callbacks */
__weak void HAL_GPIO_EXTI_Callback(uint16_t GPIO_Pin)
{
  /* Prevent unused argument(s) compilation warning */
  UNUSED(GPIO_Pin);

  /* NOTE: This function Should not be modified, when
the callback is needed, the HAL_GPIO_EXTI_Callback
could be implemented in the user file  */
}
```

STM32CubeIDE – main.c (1/3)

```c
/* main.c */
/* Includes */
#include "main.h"
#include "usart.h"
#include "gpio.h"

/* Private includes */
/* USER CODE BEGIN Includes */
/* USER CODE END Includes */

/* Private typedef */
/* USER CODE BEGIN PTD */
/* USER CODE END PTD */

/* Private define */
/* USER CODE BEGIN PD */
/* USER CODE END PD */

/* Private macro */
/* USER CODE BEGIN PM */
/* USER CODE END PM */

/* Private variables */
/* USER CODE BEGIN PV */
/* USER CODE END PV */

/* Private function prototypes */
void SystemClock_Config(void);
/* USER CODE BEGIN PFP */
/* USER CODE END PFP */

/* Private user code */
/* USER CODE BEGIN 0 */
/* USER CODE END 0 */
```

```c
int main(void)
{
  /* USER CODE BEGIN 1 */
  /* USER CODE END 1 */

  /* MCU Configuration: Reset of all peripherals,
     Initializes the Flash interface and the Systick. */
  HAL_Init();

  /* USER CODE BEGIN Init */
  /* USER CODE END Init */

  /* Configure the system clock */
  SystemClock_Config();

  /* USER CODE BEGIN SysInit */
  /* USER CODE END SysInit */

  /* Initialize all configured peripherals */
  MX_GPIO_Init();
  MX_USART3_UART_Init();
  /* USER CODE BEGIN 2 */
  /* USER CODE END 2 */

  /* Infinite loop */
  /* USER CODE BEGIN WHILE */
  while (1)
  {
    /* USER CODE END WHILE */

    /* USER CODE BEGIN 3 */
  }
  /* USER CODE END 3 */
}
```

STM32CubeIDE – main.c (2/3)

```
/* System Clock Configuration */
void SystemClock_Config(void)
{
    RCC_OscInitTypeDef RCC_OscInitStruct = {0};
    RCC_ClkInitTypeDef RCC_ClkInitStruct = {0};

    /* Configure LSE Drive Capability */
    HAL_PWR_EnableBkUpAccess();

    /** Configure the main internal regulator output voltage */
    __HAL_RCC_PWR_CLK_ENABLE();
    __HAL_PWR_VOLTAGESCALING_CONFIG(PWR_REGULATOR_VOLTAGE_SCALE3);

    /** Initializes the RCC Oscillators according to the specified parameters
     * in the RCC_OscInitTypeDef structure. */
    RCC_OscInitStruct.OscillatorType = RCC_OSCILLATORTYPE_HSE;
    RCC_OscInitStruct.HSEState = RCC_HSE_BYPASS;
    RCC_OscInitStruct.PLL.PLLState = RCC_PLL_ON;
    RCC_OscInitStruct.PLL.PLLSource = RCC_PLLSOURCE_HSE;
    RCC_OscInitStruct.PLL.PLLM = 4;
    RCC_OscInitStruct.PLL.PLLN = 96;
    RCC_OscInitStruct.PLL.PLLP = RCC_PLLP_DIV2;
    RCC_OscInitStruct.PLL.PLLQ = 4;
    RCC_OscInitStruct.PLL.PLLR = 2;
    if (HAL_RCC_OscConfig(&RCC_OscInitStruct) != HAL_OK)
    {
        Error_Handler();
    }

    /* Activate the Over-Drive mode */
    if (HAL_PWREx_EnableOverDrive() != HAL_OK)
    {
        Error_Handler();
    }
```

STM32CubeIDE – main.c (3/3)

```
    /* Initializes the CPU, AHB and APB buses clocks */
    RCC_ClkInitStruct.ClockType = RCC_CLOCKTYPE_HCLK |RCC_CLOCKTYPE_SYSCLK
                                  |RCC_CLOCKTYPE_PCLK1|RCC_CLOCKTYPE_PCLK2;
    RCC_ClkInitStruct.SYSCLKSource = RCC_SYSCLKSOURCE_PLLCLK;
    RCC_ClkInitStruct.AHBCLKDivider = RCC_SYSCLK_DIV1;
    RCC_ClkInitStruct.APB1CLKDivider = RCC_HCLK_DIV2;
    RCC_ClkInitStruct.APB2CLKDivider = RCC_HCLK_DIV1;

    if (HAL_RCC_ClockConfig(&RCC_ClkInitStruct, FLASH_LATENCY_3) != HAL_OK)
    {
        Error_Handler();
    }
}

/* USER CODE BEGIN 4 */
/* USER CODE END 4 */
/* This function is executed in case of error occurrence. */
void Error_Handler(void)
{
    /* USER CODE BEGIN Error_Handler_Debug */
    __disable_irq();
    while (1)
    {     }
    /* USER CODE END Error_Handler_Debug */
}

#ifdef  USE_FULL_ASSERT
void assert_failed(uint8_t *file, uint32_t line)
{
    /* USER CODE BEGIN 6 */
    /* User can add his own implementation to report the file name and line number,
       ex: printf("Wrong parameters value: file %s on line %d\r\n", file, line) */
    /* USER CODE END 6 */
}
#endif /* USE_FULL_ASSERT */
```

Add Code to USER CODE 4 (main.c)

```
/* USER CODE BEGIN 4 */

void HAL_GPIO_EXTI_Callback(uint16_t GPIO_Pin)
{
    if(GPIO_Pin == GPIO_PIN_13)
    {
        HAL_GPIO_TogglePin(GPIOB, LD2_Pin);
    }
}

int __io_putchar(int ch)
{
    uint8_t c[1];
    c[0] = ch & 0x00FF;
    HAL_UART_Transmit(&huart3, &*c, 1, 10);
    return ch;
}
// comment the function _write() below, re-compile and run to check the program
int _write(int file, char *ptr, int len)
{
    int DataIdx;
    for(DataIdx= 0; DataIdx< len; DataIdx++)
    {
        __io_putchar(*ptr++);
    }
    return len;
}

/* USER CODE END 4 */
```

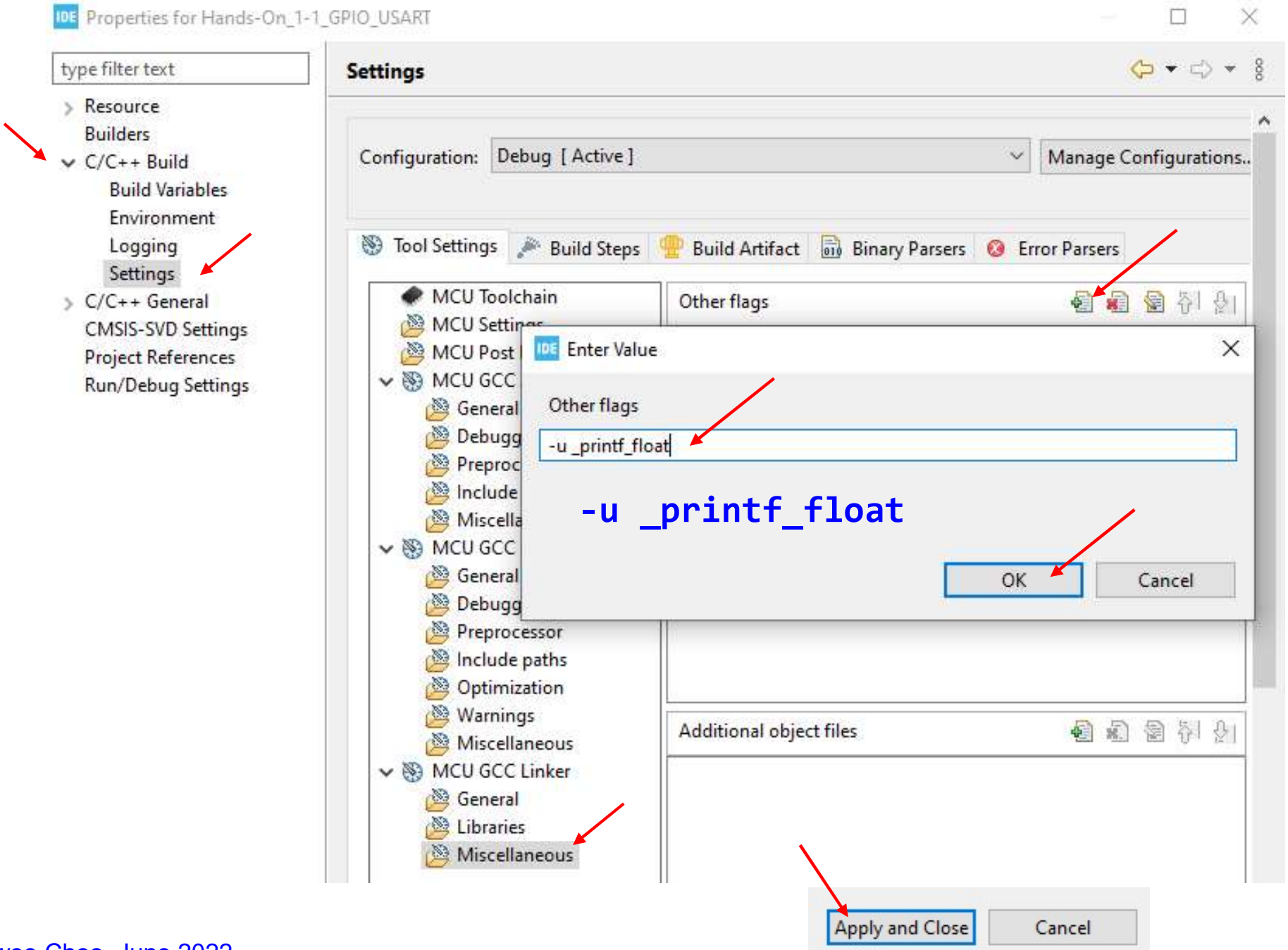
STM32CubeIDE

Add Code to USER CODE BEGIN Includes and WHILE (main.c)

```c
/* main.c */
/* Includes */
#include "main.h"
#include "usart.h"
#include "gpio.h"
/* Private includes */
/* USER CODE BEGIN Includes */
#include <stdio.h>
/* USER CODE END Includes */
void SystemClock_Config(void);
int main(void)
{
    HAL_Init();
    SystemClock_Config();
    /* Initialize all configured peripherals */
    MX_GPIO_Init();
    MX_USART3_UART_Init();
    /* Infinite loop */
    /* USER CODE BEGIN WHILE */
    while (1)
    {
        HAL_GPIO_TogglePin(LD3_GPIO_Port, LD3_Pin);
        printf("Print vales = %d  %ld  %4.3f \r\n", 33, 345UL, 8.456f);
        HAL_Delay(500);
        //  project -> properties -> C/C++ Build - Settings -> Tool Settings -> MCU GCC Linker -
        //  - Miscellaneous -> Other Flags: -u _printf_float
    /* USER CODE END WHILE */
    /* USER CODE BEGIN 3 */
  }
  /* USER CODE END 3 */
}
```

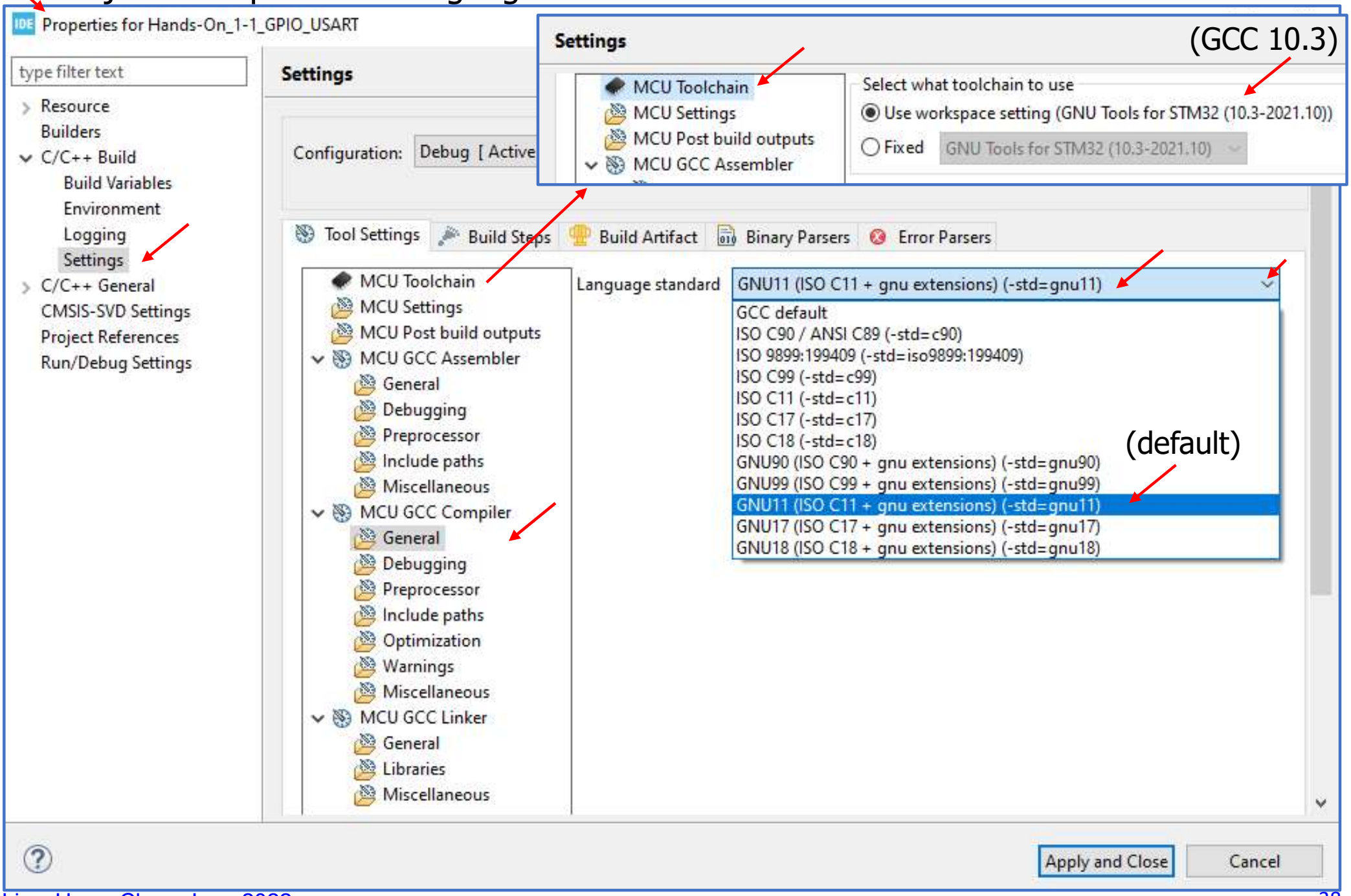
STM32CubeIDE

Project Properties: Add "-u _printf_float" to MCU GCC Linker – Miscellaneous – Other flags



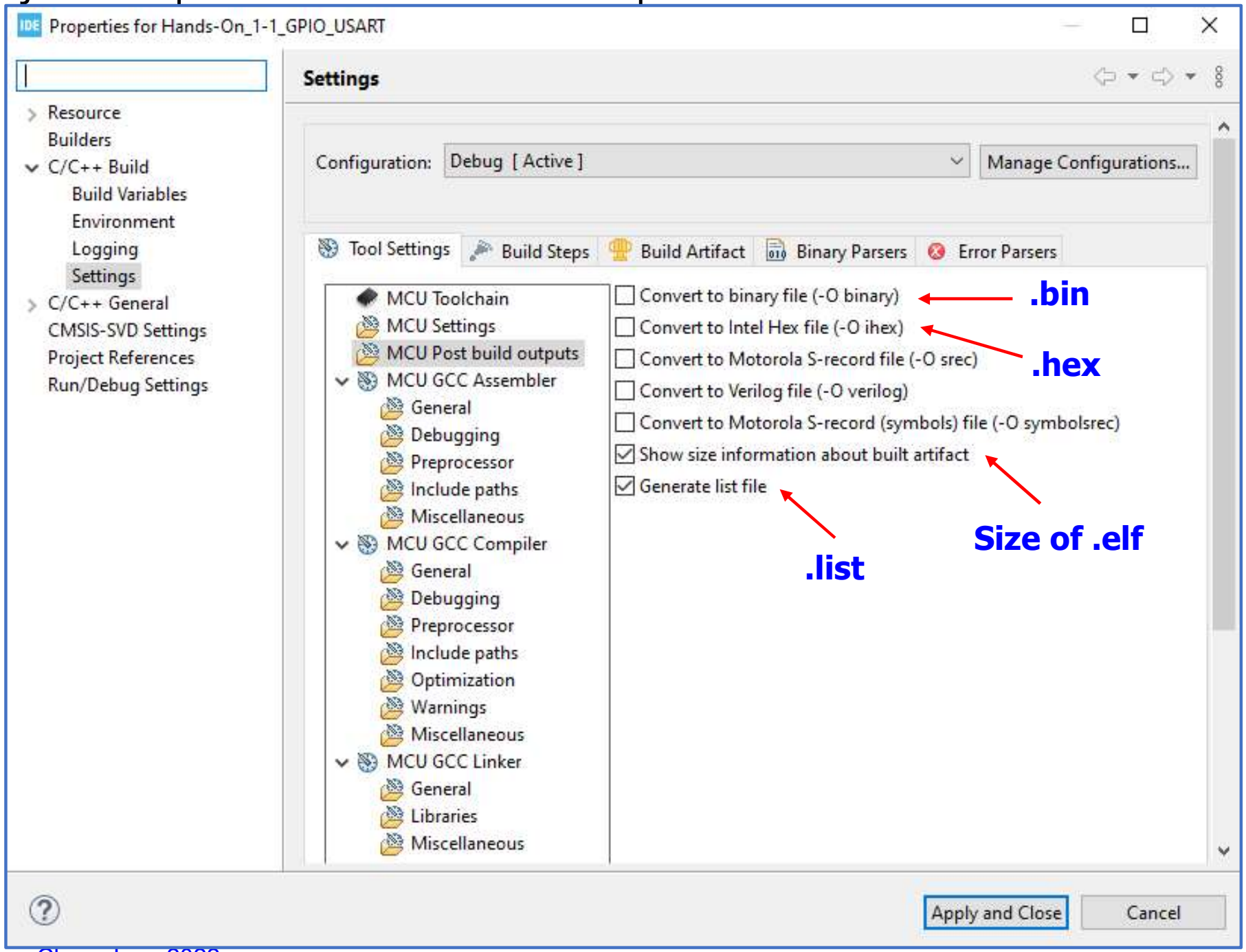
STM32CubeIDE

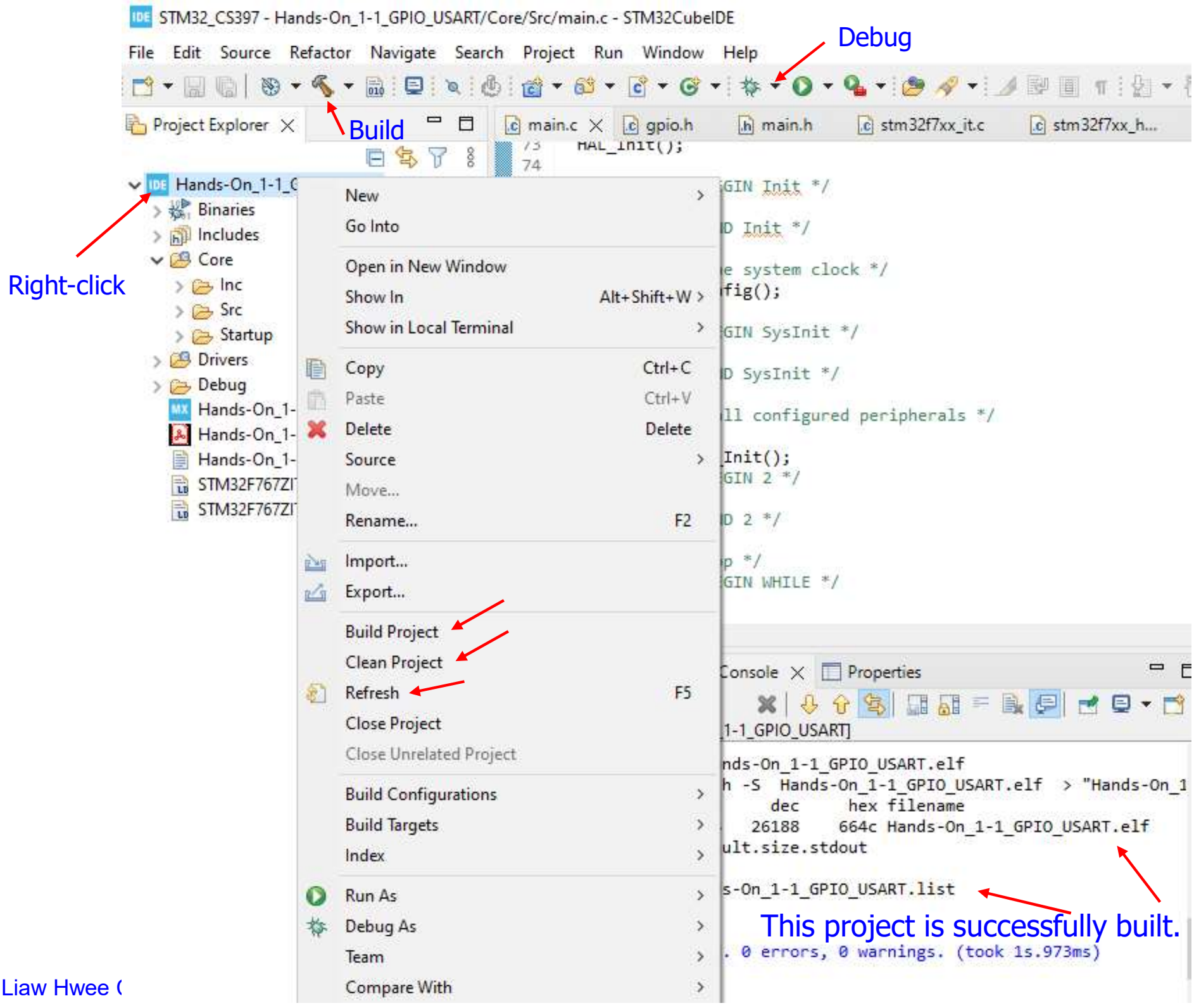
Project – Properties – Language Standard



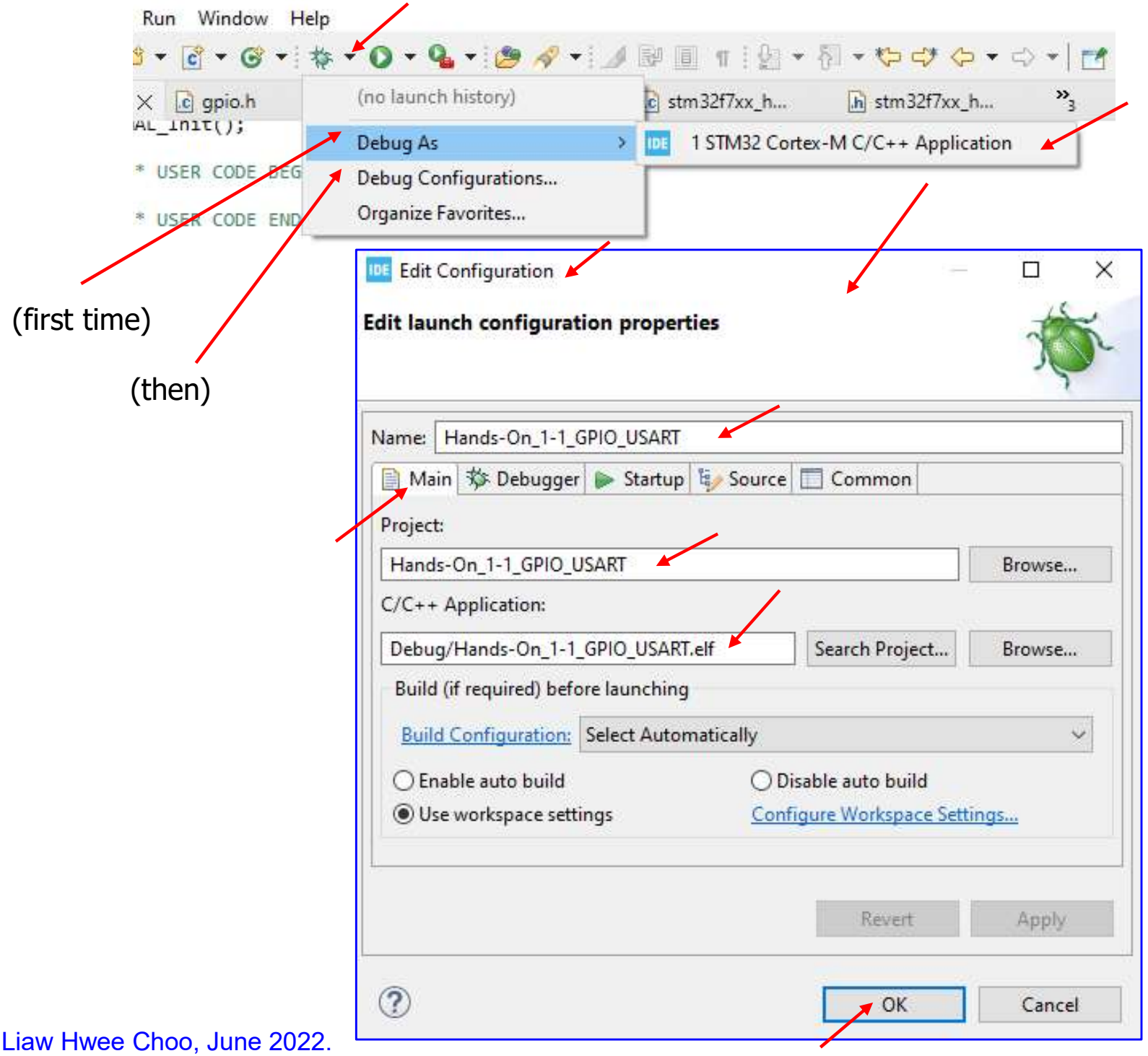
STM32CubeIDE

Project – Properties – MCU Post Build Outputs

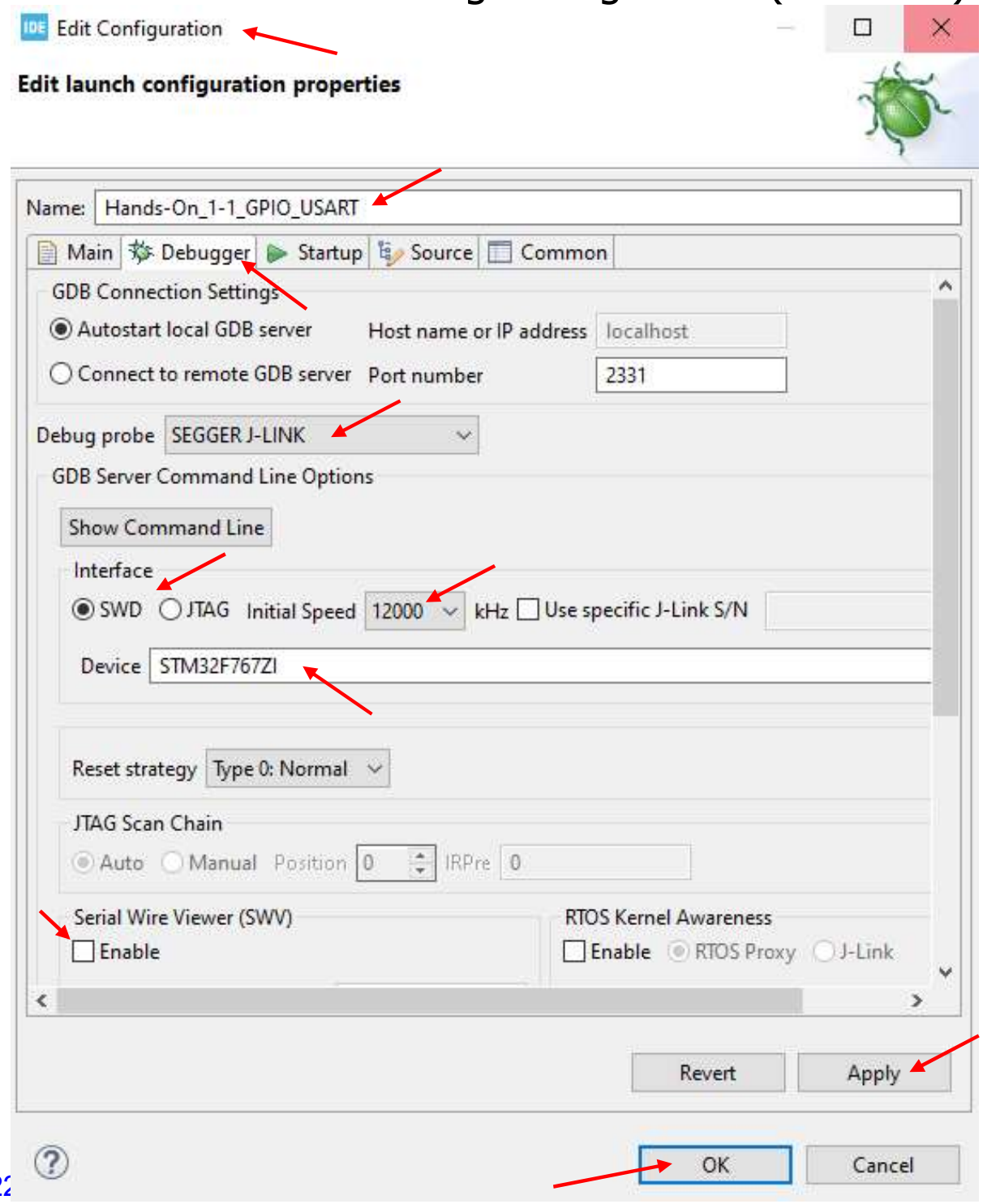



STM32_CS397 - Hands-On_1-1_GPIO_USART/Core/Src/main.c - STM32CubeIDE
File Edit Source Refactor Navigate Search Project Run Window Help
Debug
Build
Project Explorer
main.c
gpio.h
main.h
stm32f7xx_it.c
stm32f7xx_h...
Hands-On_1-1_G
Binaries
Includes
Core
Inc
Src
Startup
Drivers
Debug
Hands-On_1-
Hands-On_1-
Hands-On_1-
STM32F767ZI
STM32F767ZI
Right-click
New
Go Into
Open in New Window
Show In Alt+Shift+W
Show in Local Terminal
Copy Ctrl+C
Paste Ctrl+V
Delete Delete
Source
Move...
Rename... F2
Import...
Export...
Build Project
Clean Project
Refresh F5
Close Project
Close Unrelated Project
Build Configurations
Build Targets
Index
Run As
Debug As
Team
Compare With
HAL_Init();
74
GIN Init */
D Init */
e system clock */
fig();
GIN SysInit */
D SysInit */
ll configured peripherals */
_Init();
GIN 2 */
D 2 */
p */
GIN WHILE */
Console
Properties
1-1_GPIO_USART]
nds-On_1-1_GPIO_USART.elf
h -S Hands-On_1-1_GPIO_USART.elf > "Hands-On_1
dec hex filename
26188 664c Hands-On_1-1_GPIO_USART.elf
ult.size.stdout
s-On_1-1_GPIO_USART.list
This project is successfully built.
. 0 errors, 0 warnings. (took 1s.973ms)

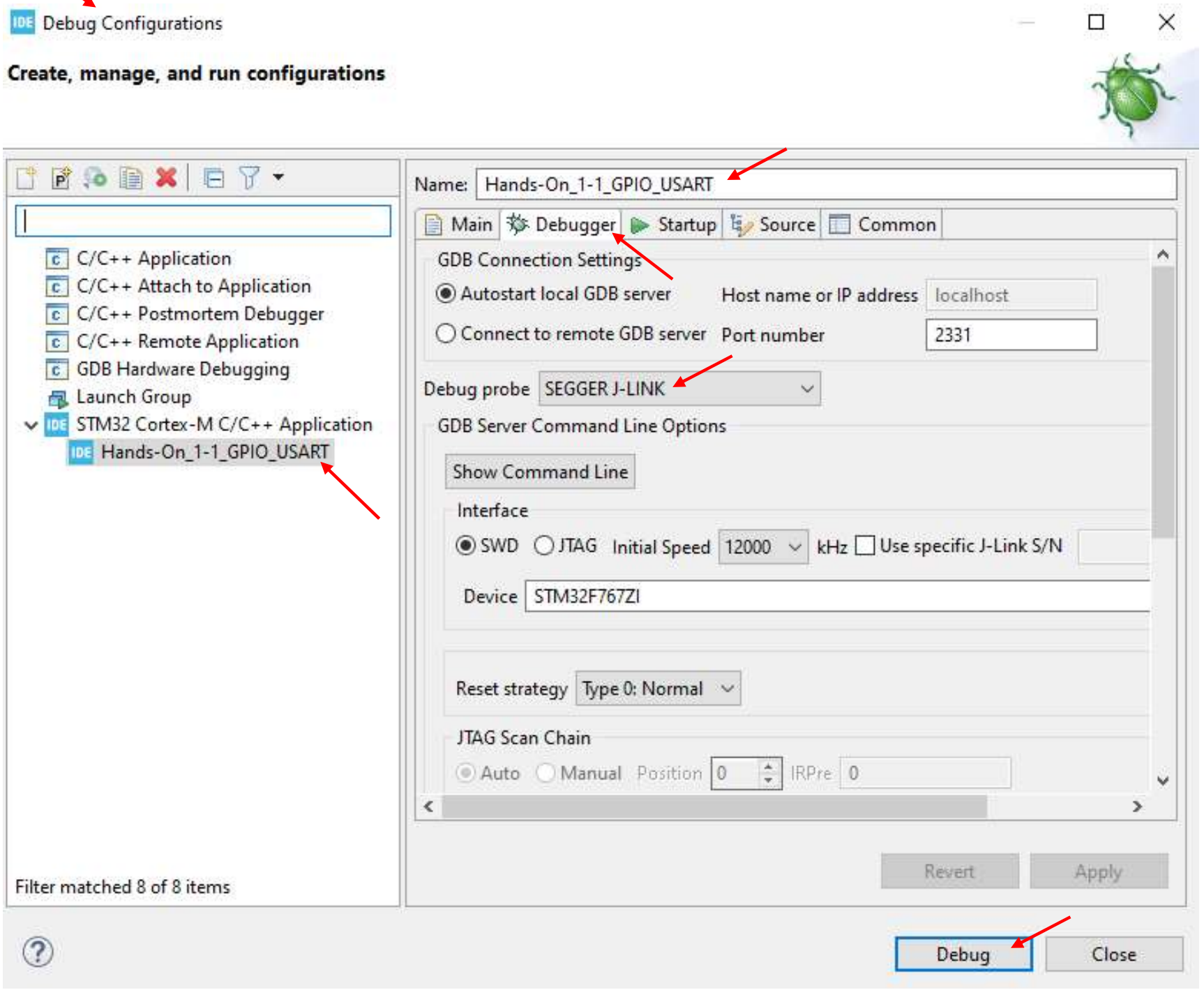
STM32CubeIDE – Debug As (Configuration)



STM32CubeIDE – Debug Configuration (first time)

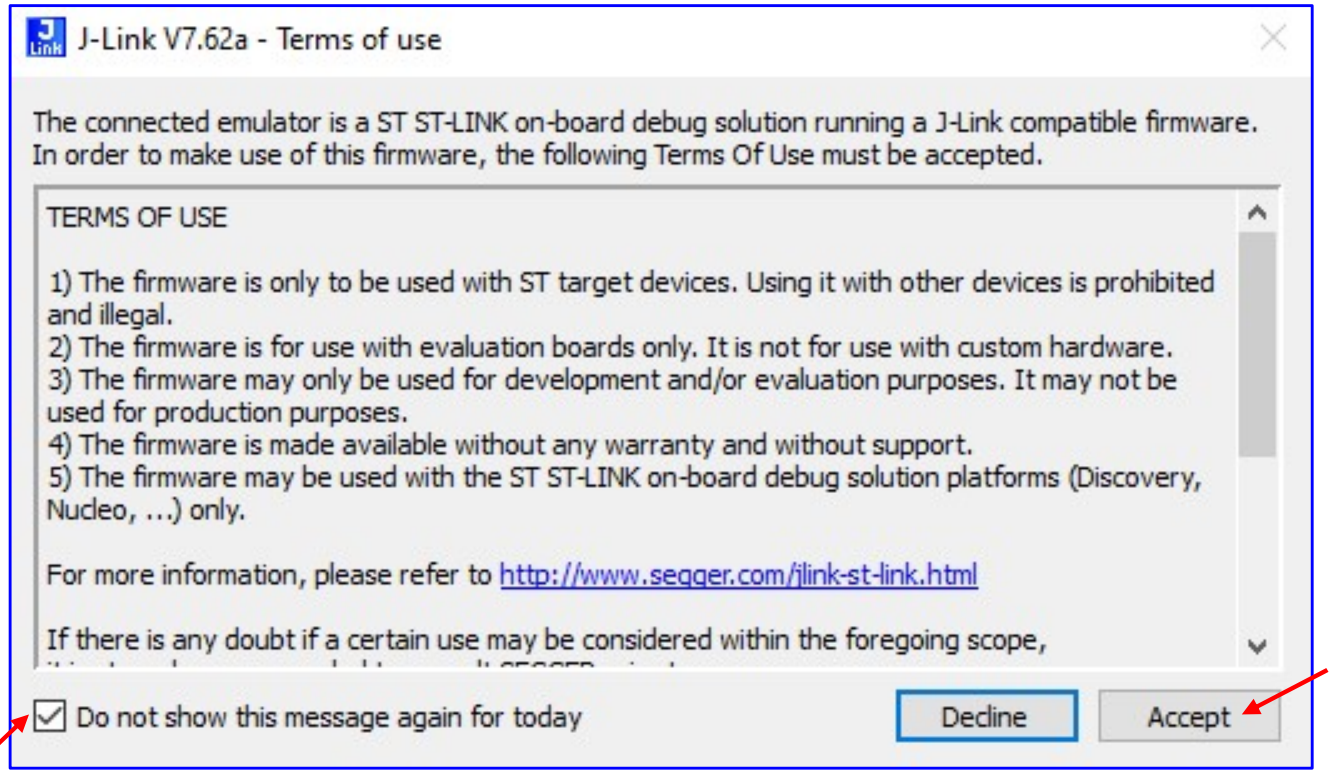


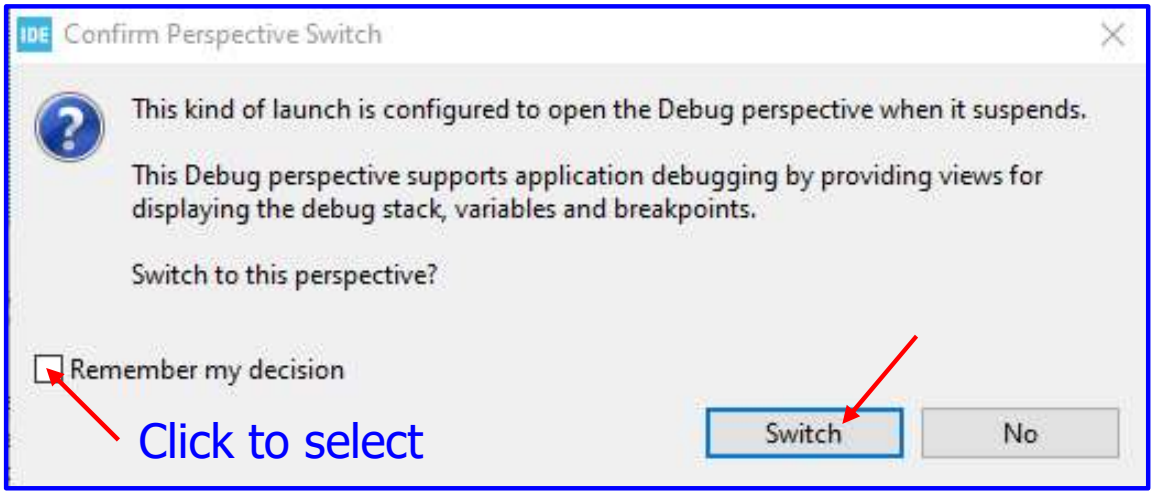
STM32CubeIDE – Debug Configuration (after first time)



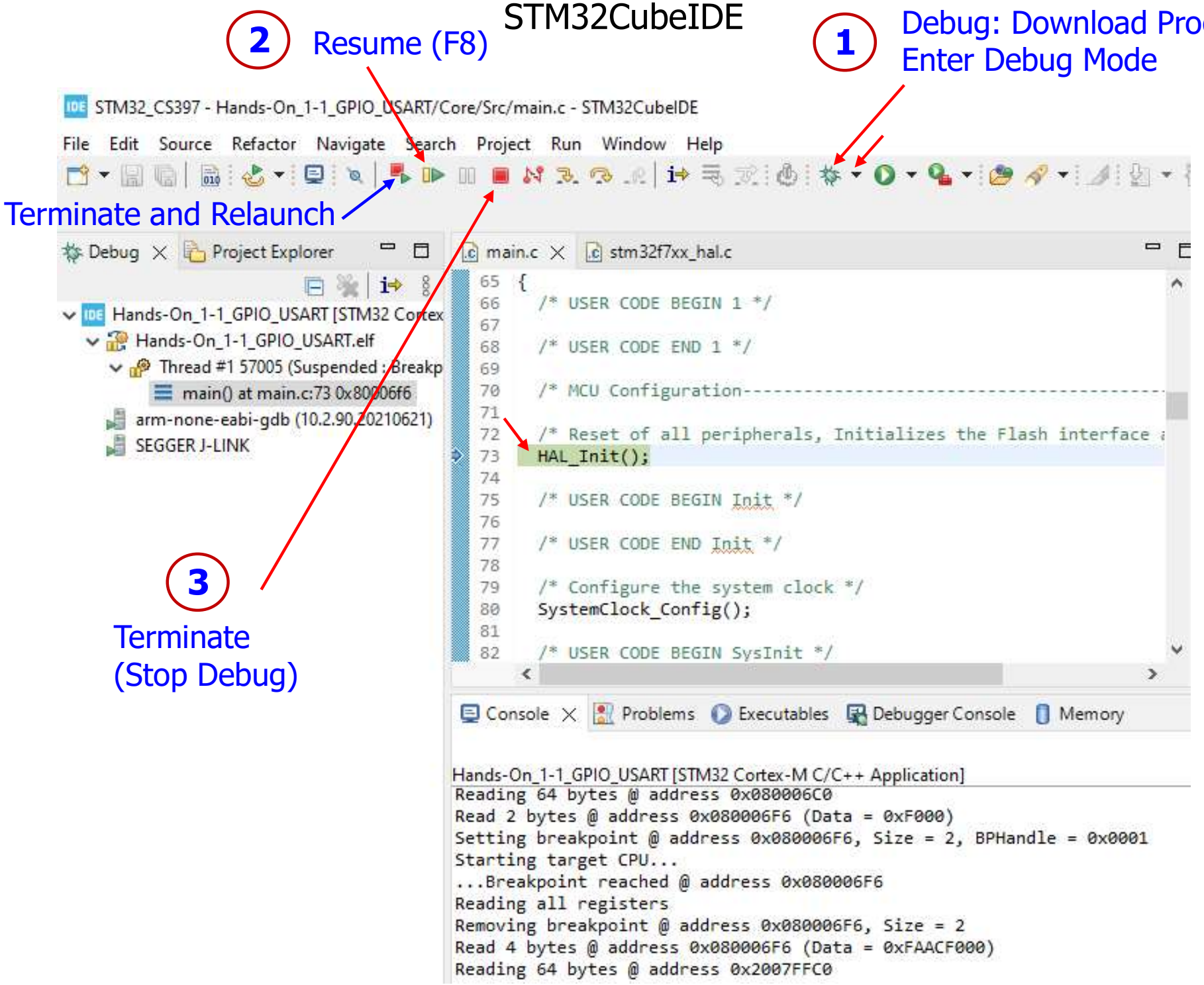
Make sure that the STM32 board is connected to USB cable, ...

STM32CubeIDE – Debug



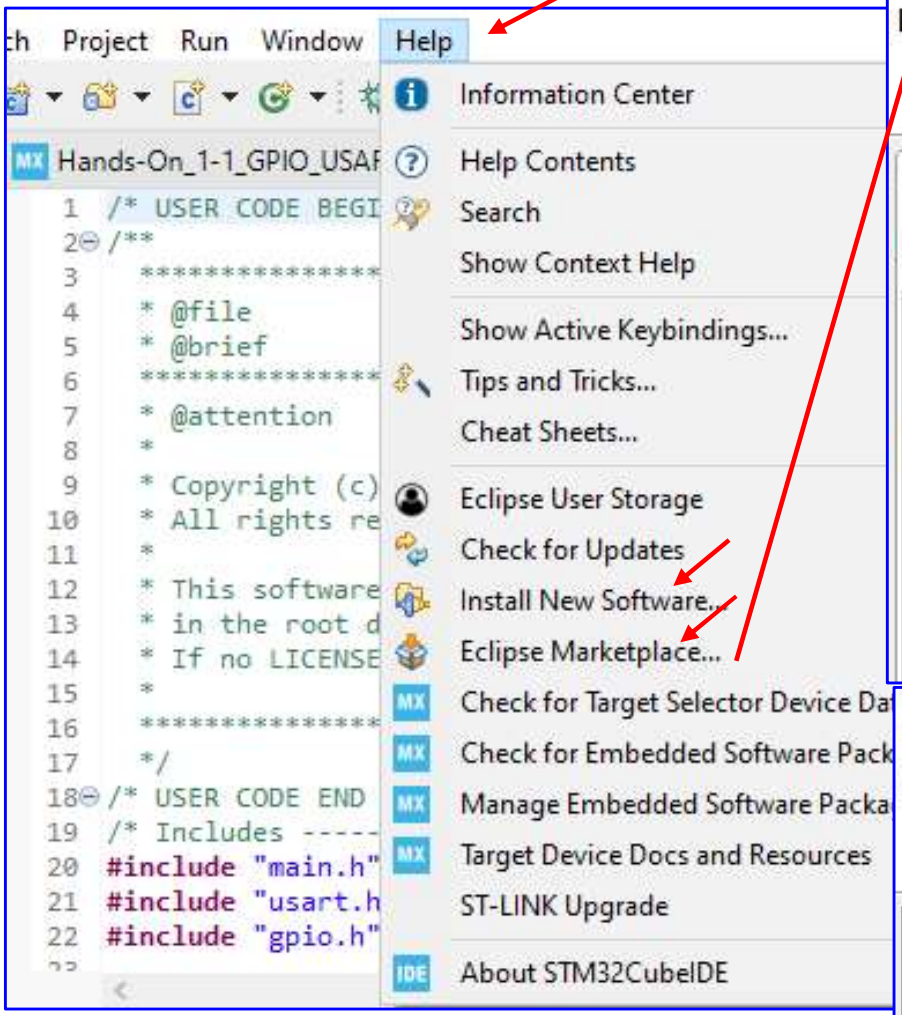


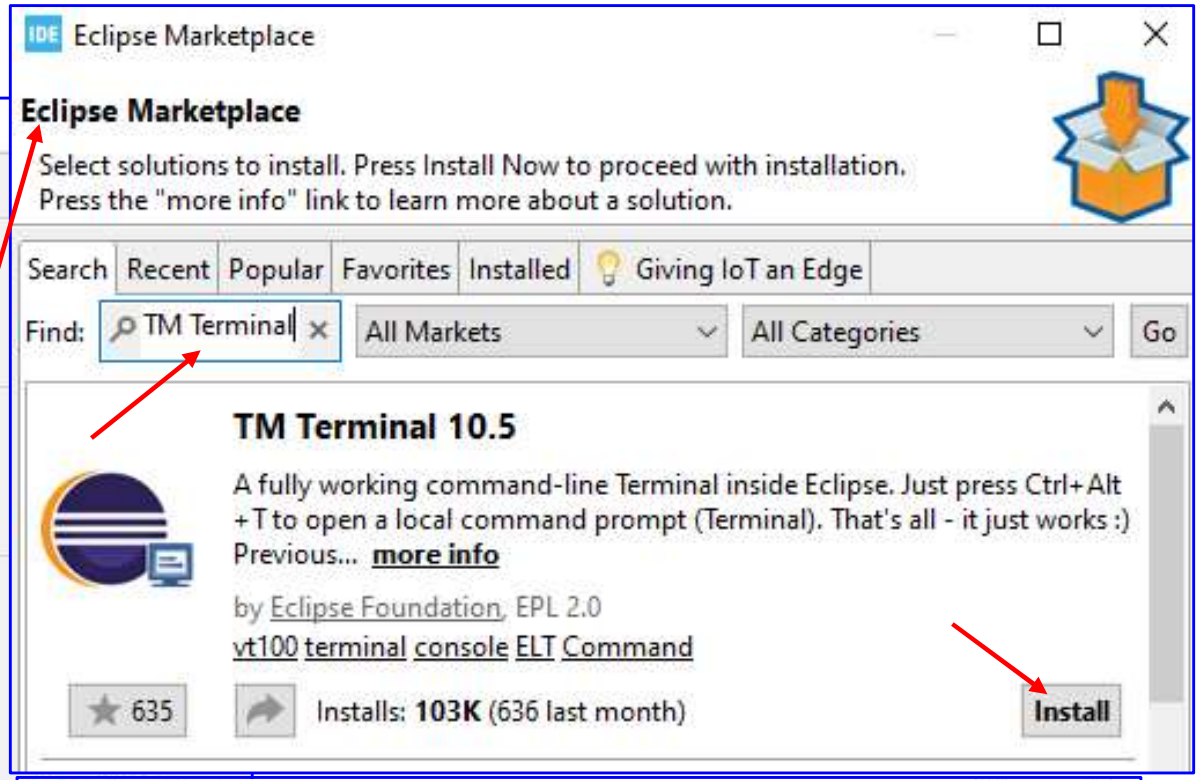
STM32CubeIDE

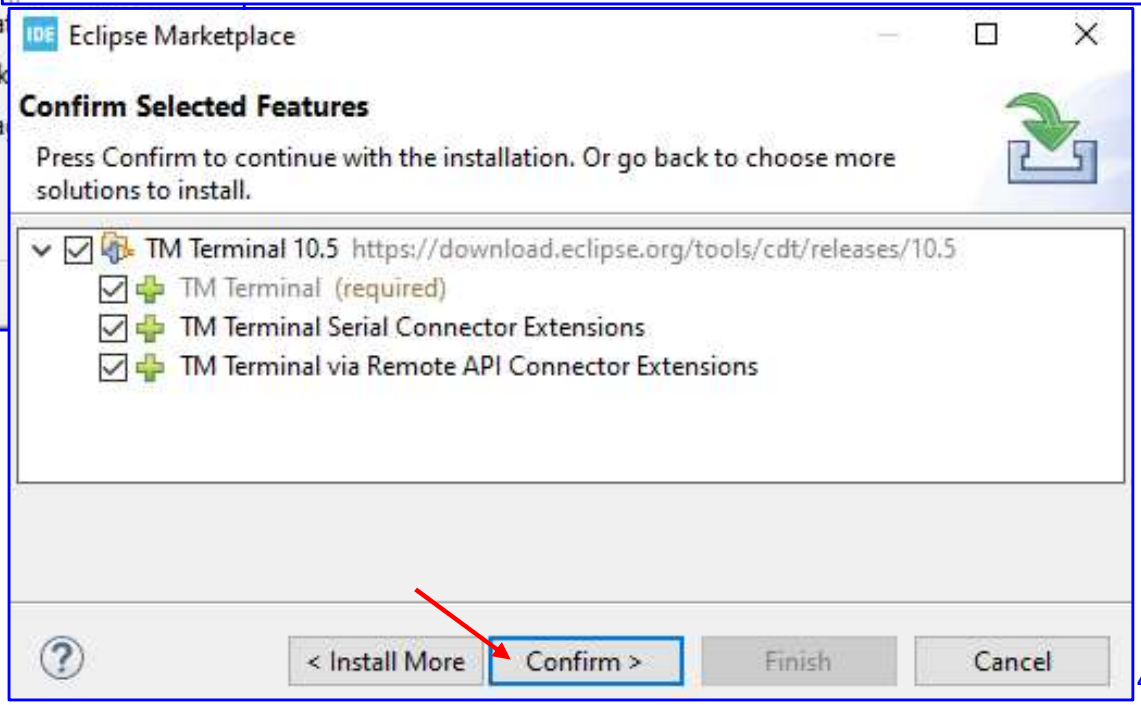


STM32CubeIDE

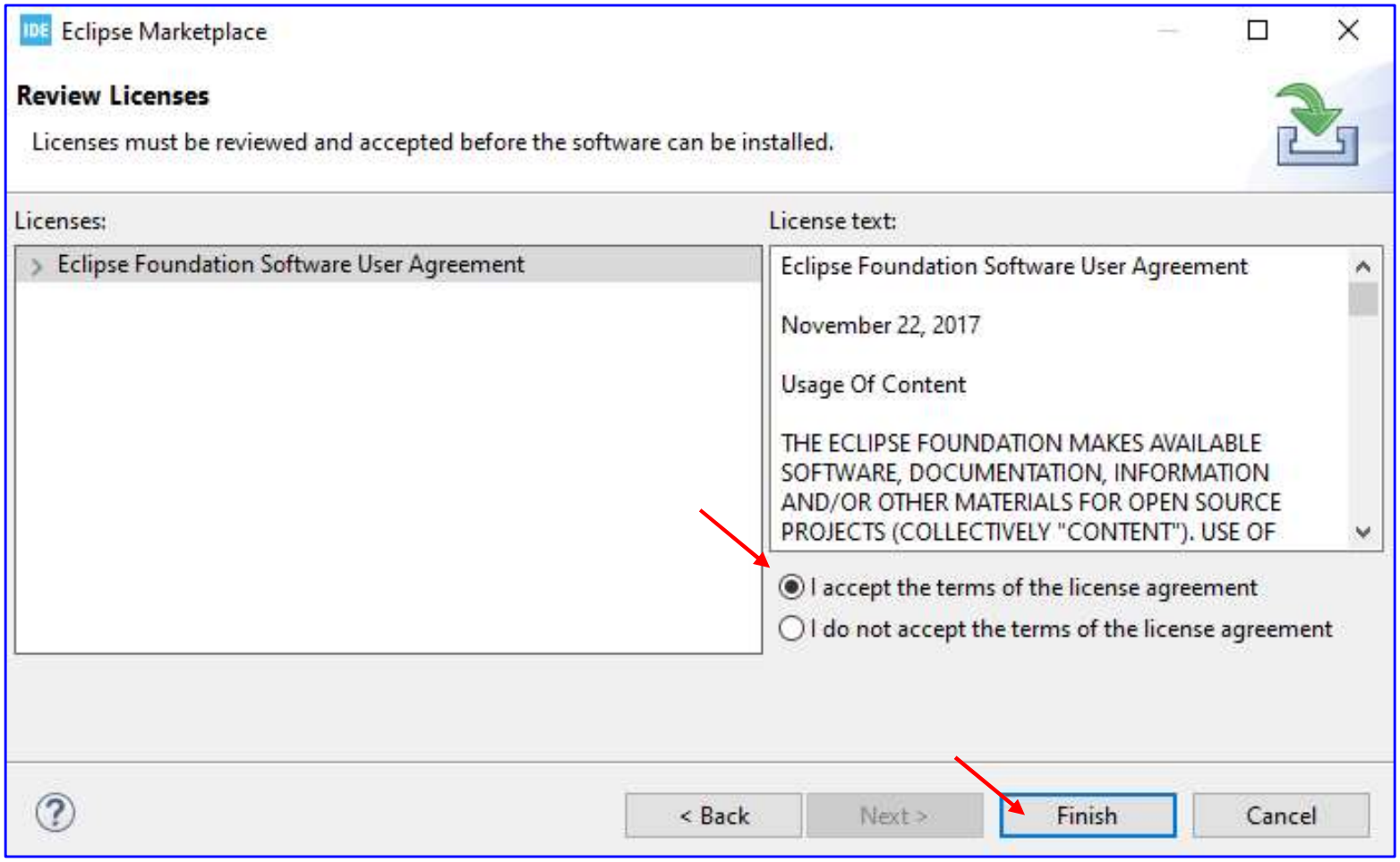
Install TM Terminal

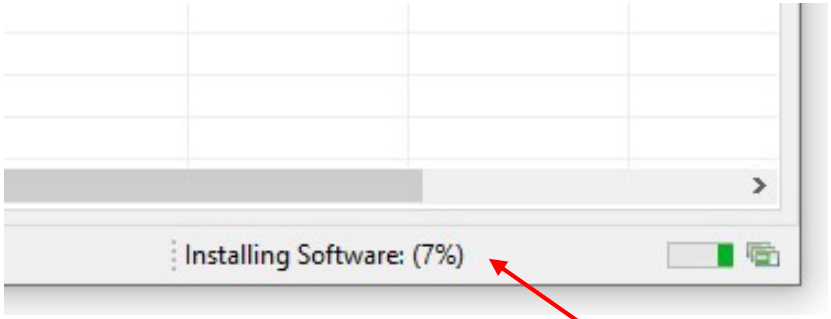


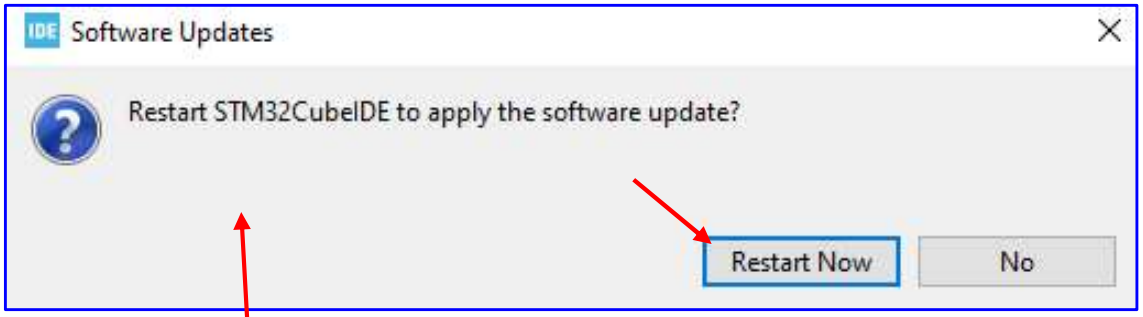




STM32CubeIDE



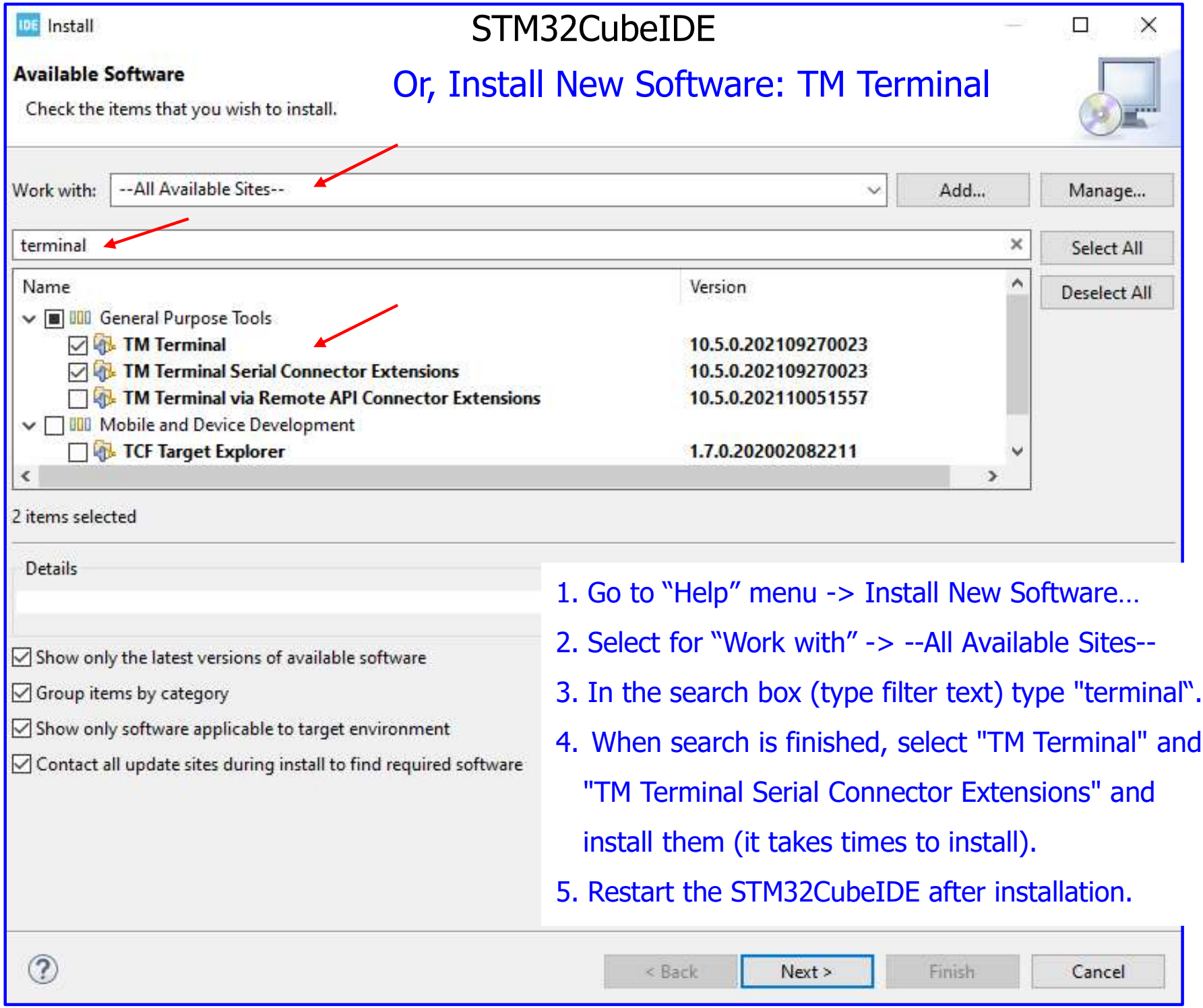




wait ...until software installed.

STM32CubeIDE

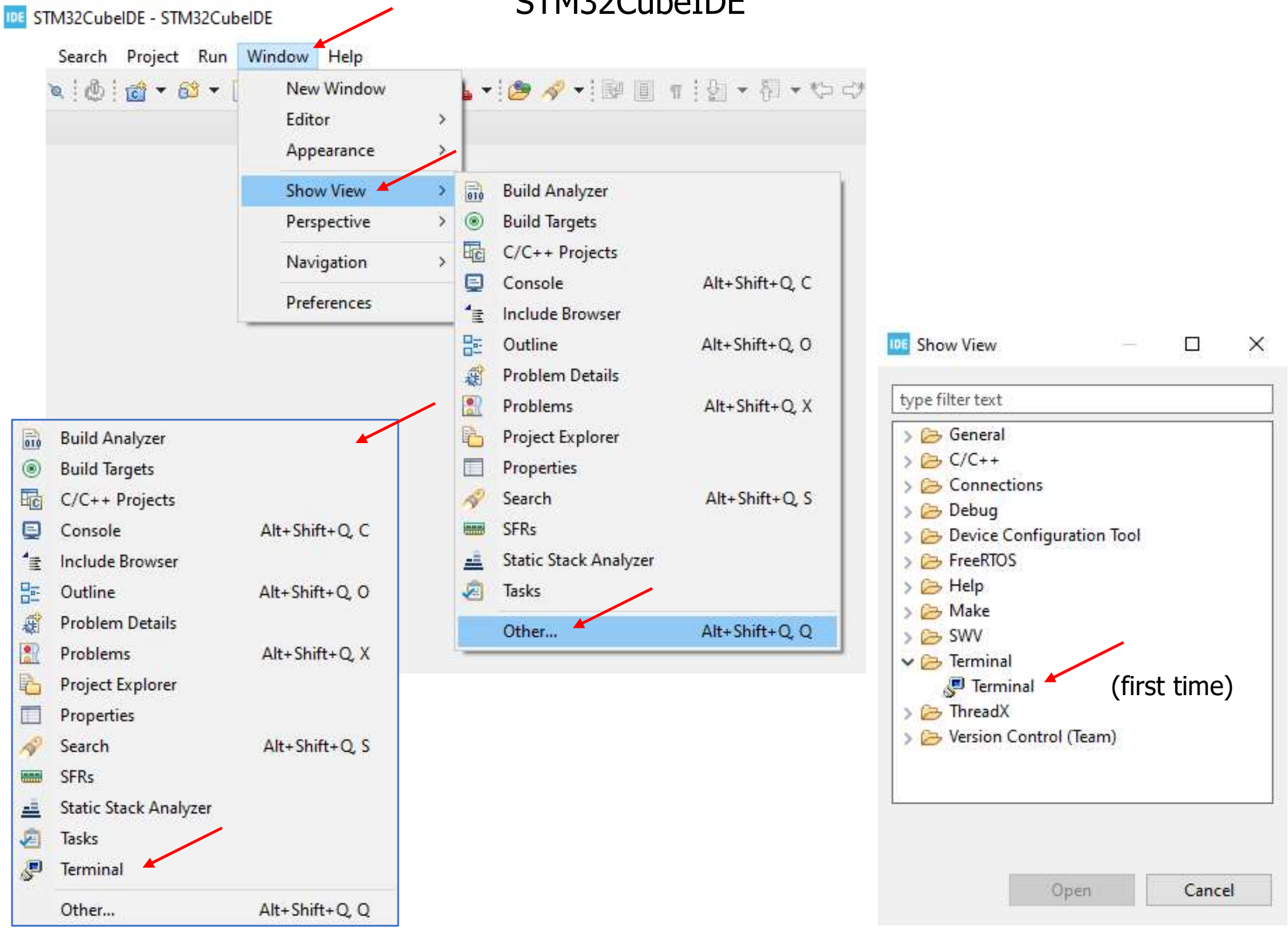
Or, Install New Software: TM Terminal



1. Go to "Help" menu -> Install New Software…
2. Select for "Work with" -> --All Available Sites--
3. In the search box (type filter text) type "terminal".
4. When search is finished, select "TM Terminal" and "TM Terminal Serial Connector Extensions" and install them (it takes times to install).
5. Restart the STM32CubeIDE after installation.

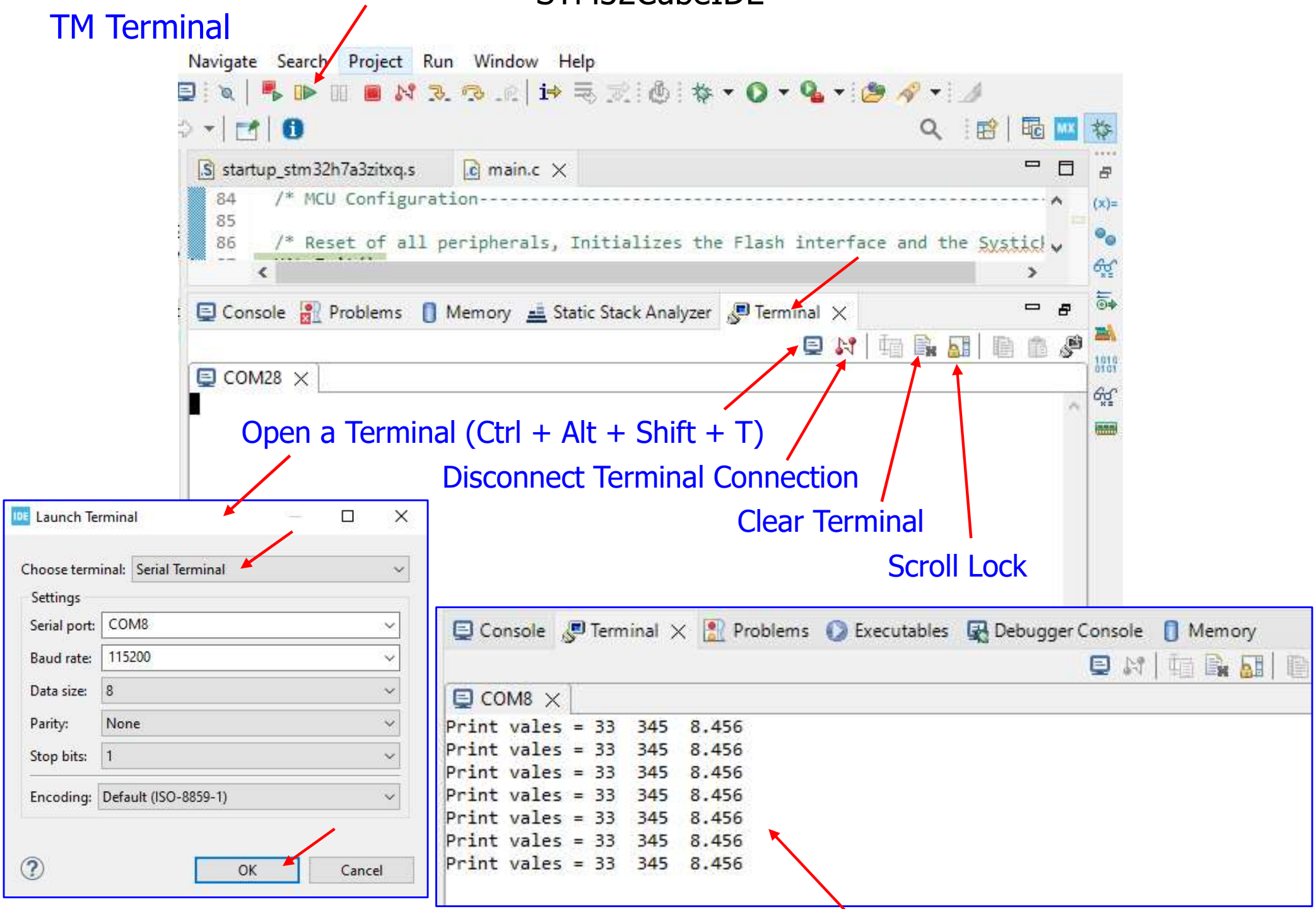
TM Terminal

STM32CubeIDE



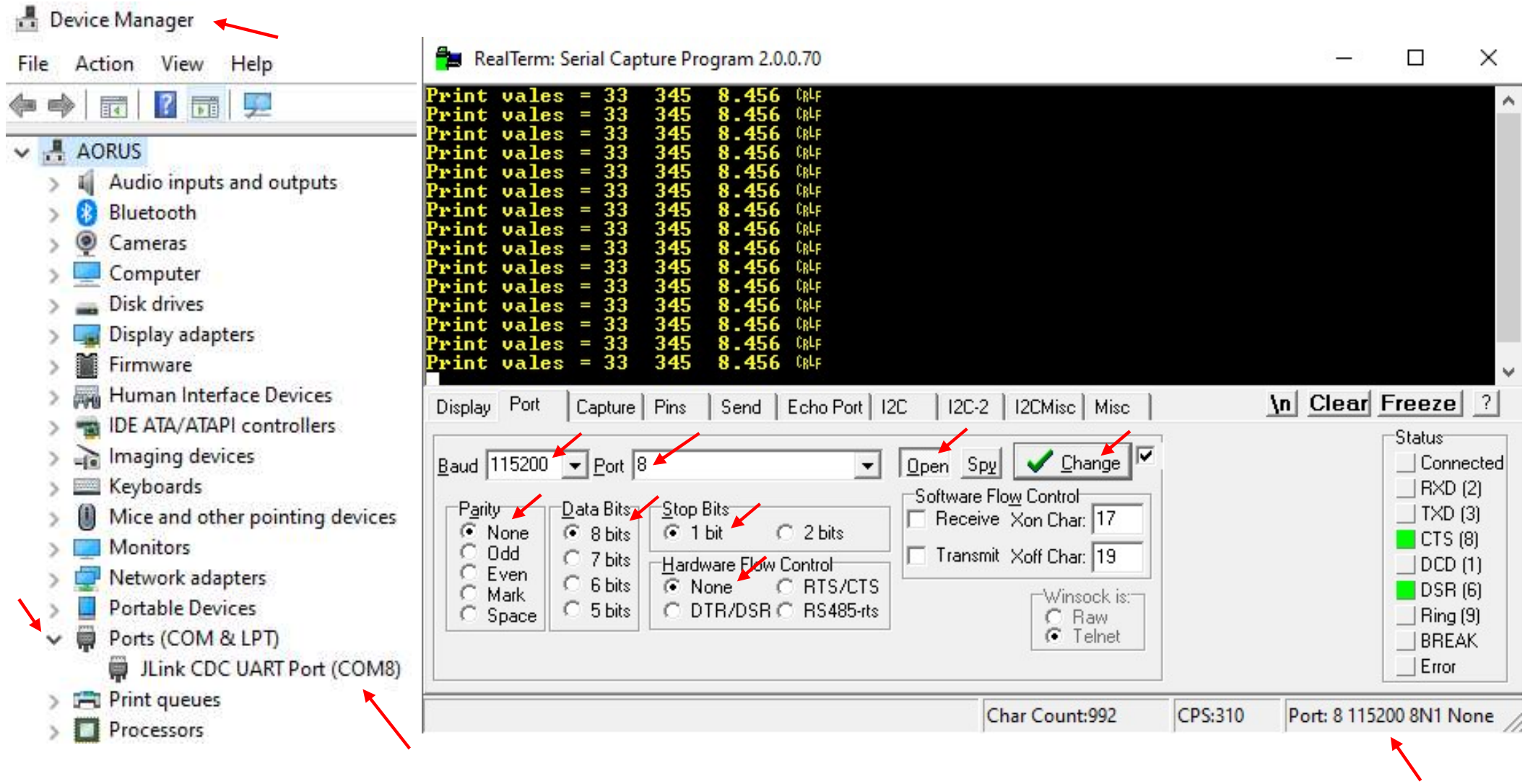
STM32CubeIDE

TM Terminal



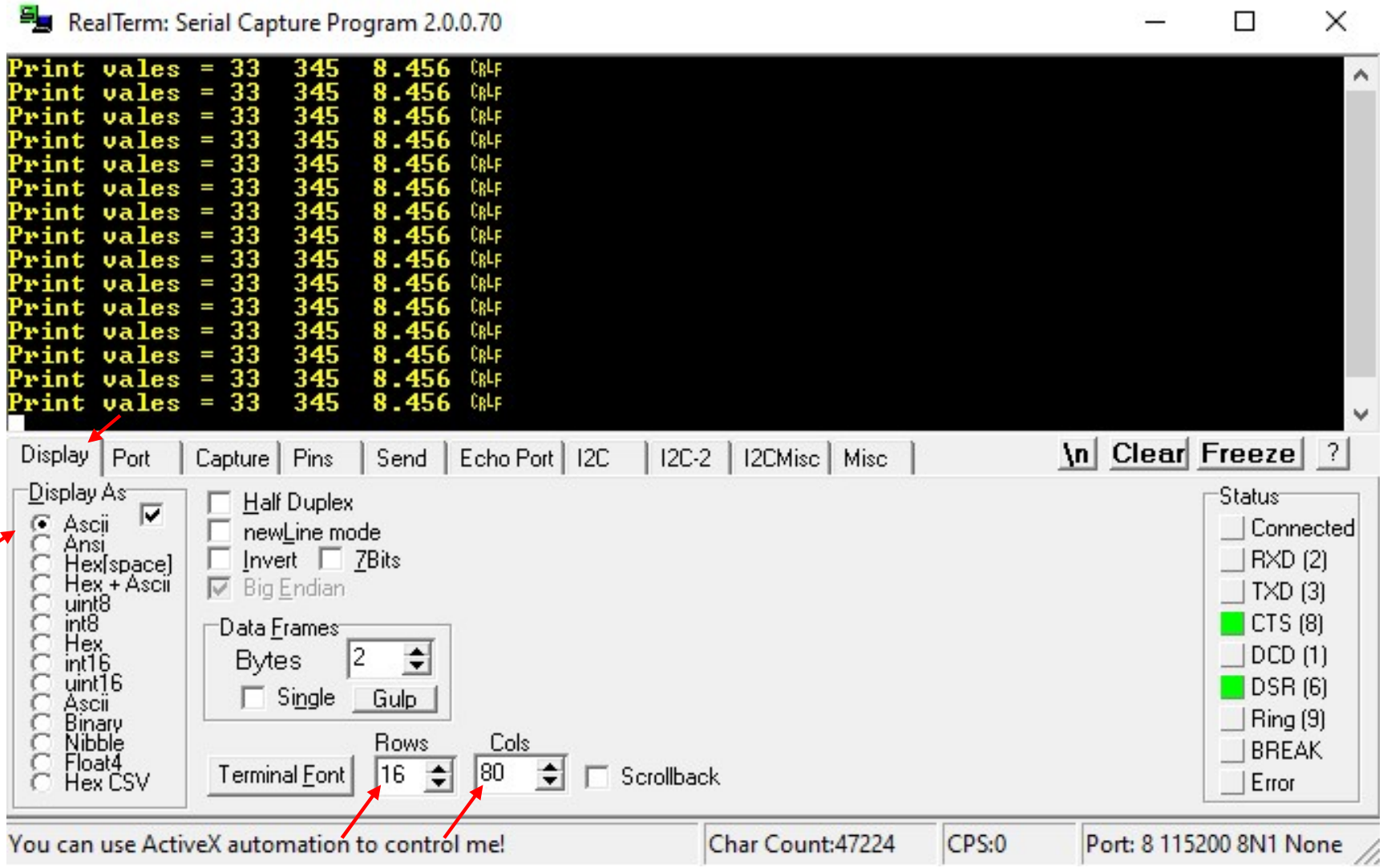
Test Program Running in STM32F767ZI

- Check "Device Manager, Ports (COM & LPT)", and find the connected JLink CDC UART Port
- Run "RealTerm", ensure proper settings, and click "Change", then "Open" to capture data



RealTerm, a Serial Terminal (COM Port) Software https://realterm.sourceforge.io/

RealTerm is an engineer's terminal software specially designed for capturing, controlling and debugging binary and other difficult data streams.



Test Program

- Check the values displayed on the TM Terminal or RealTerm and ensure that they are correct.
- Observe the flashing LD3 (Red LED) and measure the flashing rate.
- Press the User button (B1) a few times slowly to see the response of LD2 (Blue LED).
- Modify the program to make the values displayed on the TM Terminal or RealTerm changing accordingly to a programmed sequence.