SPI communication between a Nucleo-STM32L476 board and a BME280 sensor

Jean-Christophe Toussaint
Grenoble INP - Phelma

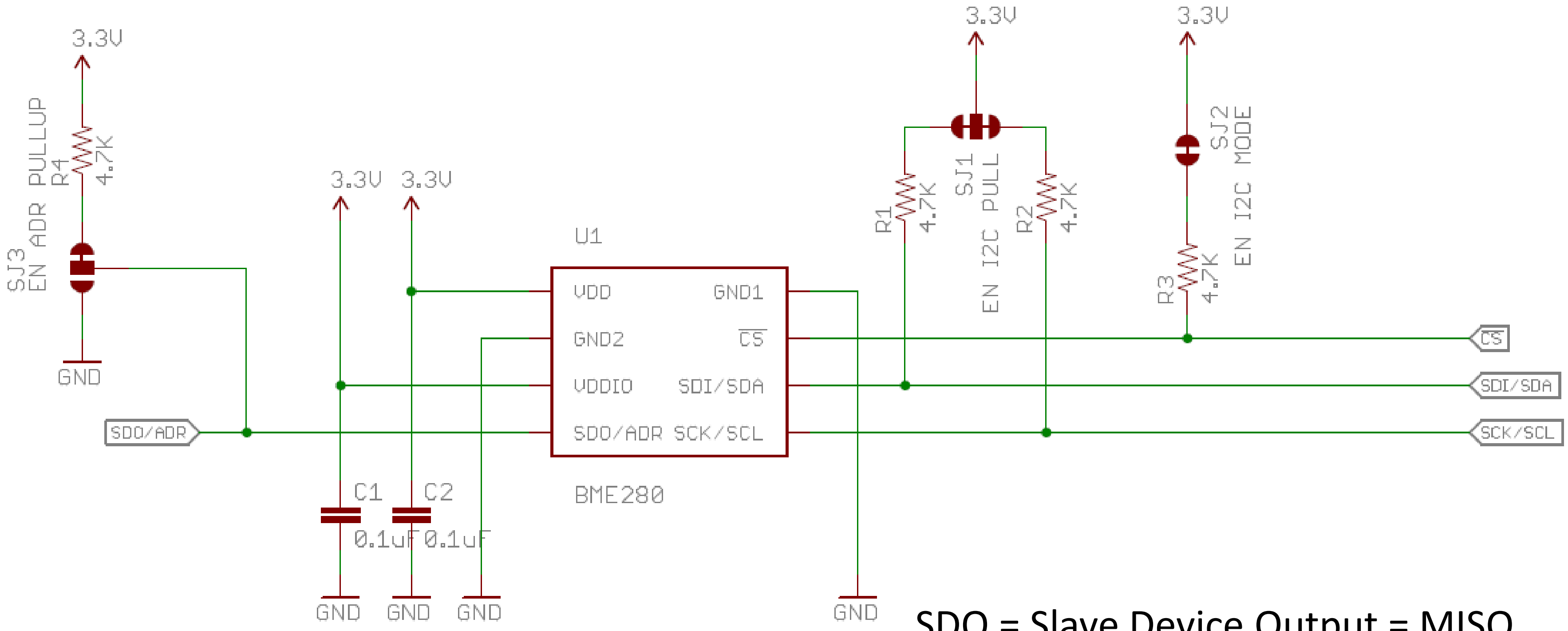
Nucleo64-STM32L476 : SPI communication with a sensor

The Serial Peripheral Interface (SPI) is a specification of serial, synchronous and full duplex communication between a master (or multiple masters) and a single or multiple slave devices.

This presentation is aimed at helping users to realize such a communication between a Nucleo board STM32L476RG and an SPI pressure-temperature-humidity Sensor BME280.

This example was developed using the Keil programing environment.

BME280 Wiring



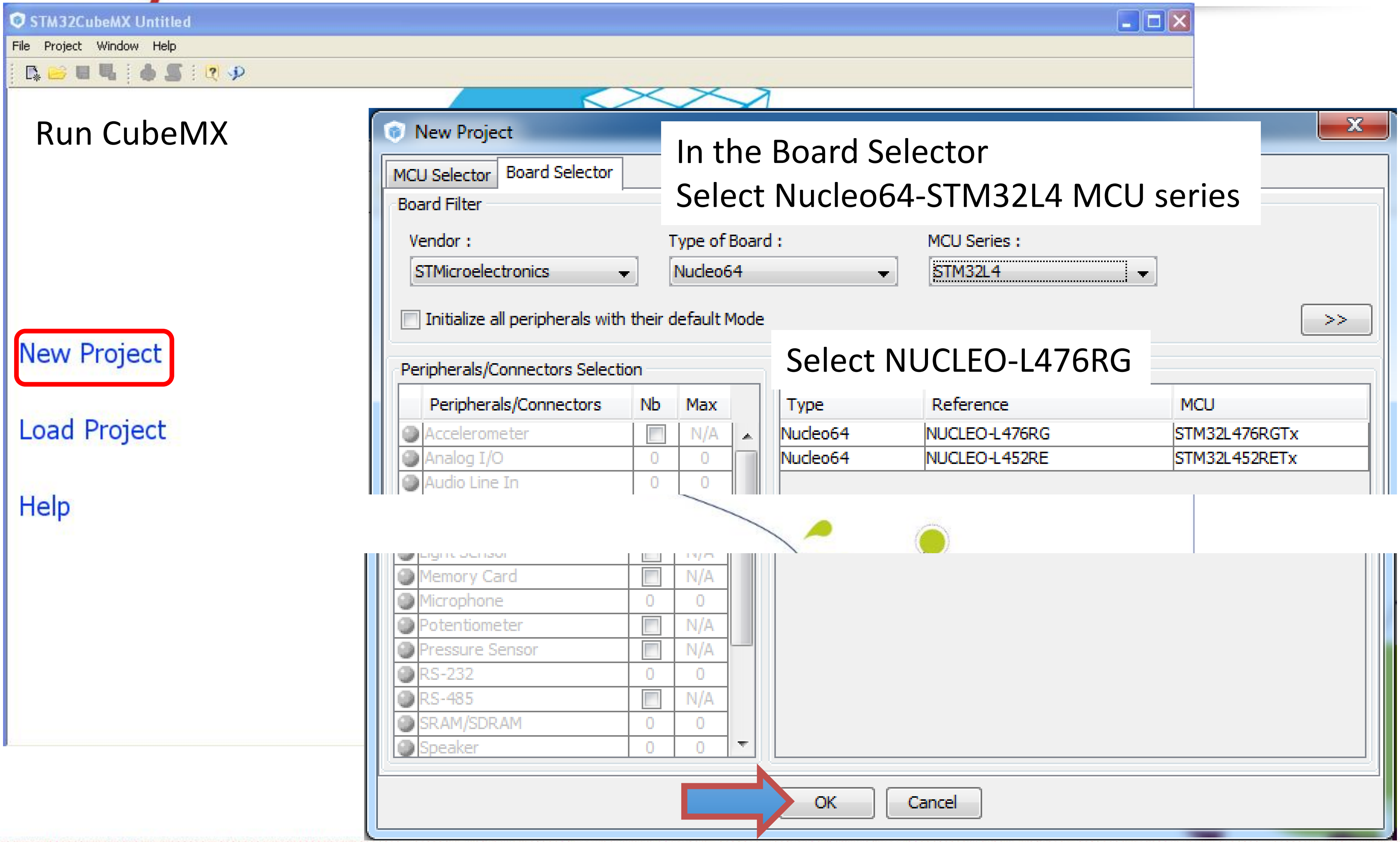
SDO = Slave Device Output = MISO

SDI = Slave Device Input = MOSI

SCK = Slave Clock

CS = Chip Select

Nucleo64-STM32L476RG : CubeMx configurator



Run CubeMX

New Project

Load Project

Help

In the Board Selector

Select Nucleo64-STM32L4 MCU series

Select NUCLEO-L476RG

Vendor : STMicroelectronics

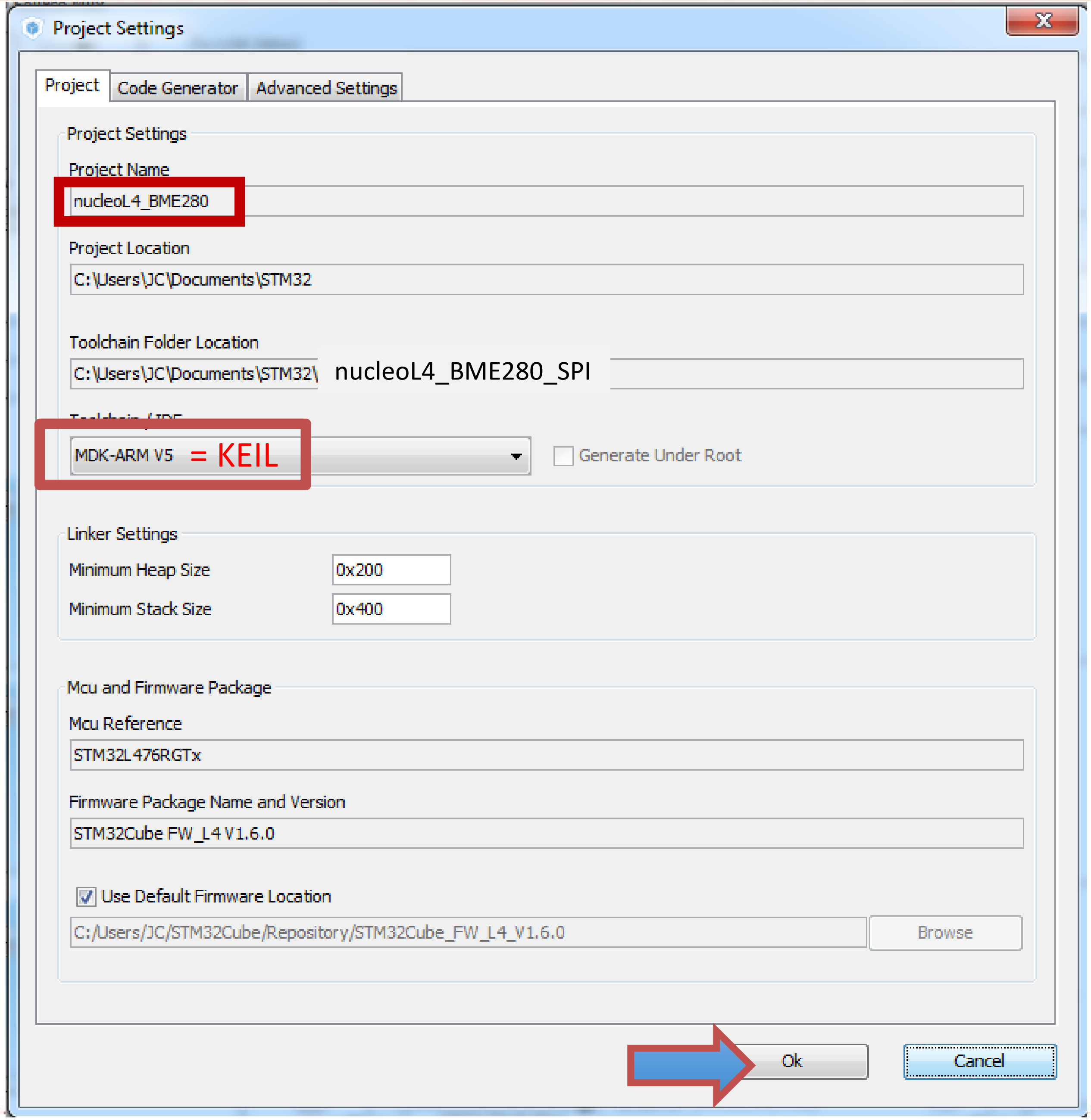
Type of Board : Nucleo64

MCU Series : STM32L4

□ Initialize all peripherals with their default Mode

Peripherals/Connectors	Nb	Max
Accelerometer	□	N/A
Analog I/O	0	0
Audio Line In	0	0
Light Sensor	□	N/A
Memory Card	□	N/A
Microphone	0	0
Potentiometer	□	N/A
Pressure Sensor	□	N/A
RS-232	0	0
RS-485	□	N/A
SRAM/SDRAM	0	0
Speaker	0	0

Type	Reference	MCU
Nucleo64	NUCLEO-L476RG	STM32L476RGTx
Nucleo64	NUCLEO-L452RE	STM32L452RETx



Project Settings

Project Name: nucleoL4_BME280

Project Location: C:\Users\JC\Documents\STM32

Toolchain Folder Location: C:\Users\JC\Documents\STM32\ nucleoL4_BME280_SPI

Toolchain / IDE: MDK-ARM V5 = KEIL

□ Generate Under Root

Linker Settings

Minimum Heap Size: 0x200

Minimum Stack Size: 0x400

Mcu and Firmware Package

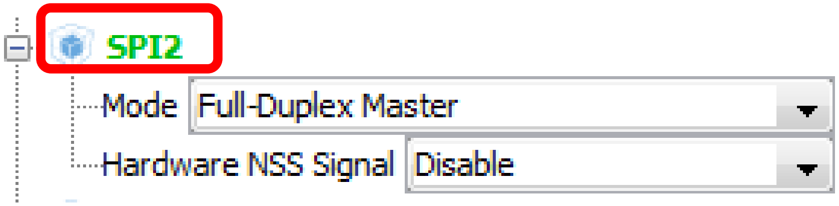
Mcu Reference: STM32L476RGTx

Firmware Package Name and Version: STM32Cube FW_L4 V1.6.0

☑ Use Default Firmware Location

C:/Users/JC/STM32Cube/Repository/STM32Cube_FW_L4_V1.6.0

Nucleo64-STM32L476RG : SPI and USART

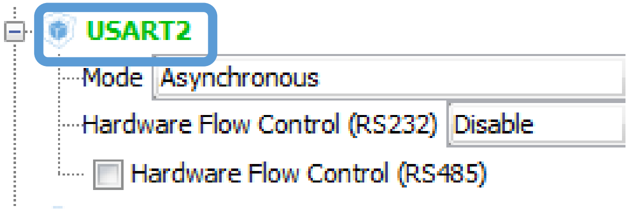


SPI2

Mode: Full-Duplex Master

Hardware NSS Signal: Disable

Choose SPI2 as the SPI bus.
PB10 and (PC2, PC3) are then automatically selected as circled beside.
PB15 is used as a software Chip Select pin. Change its name to SPI2_CS1.



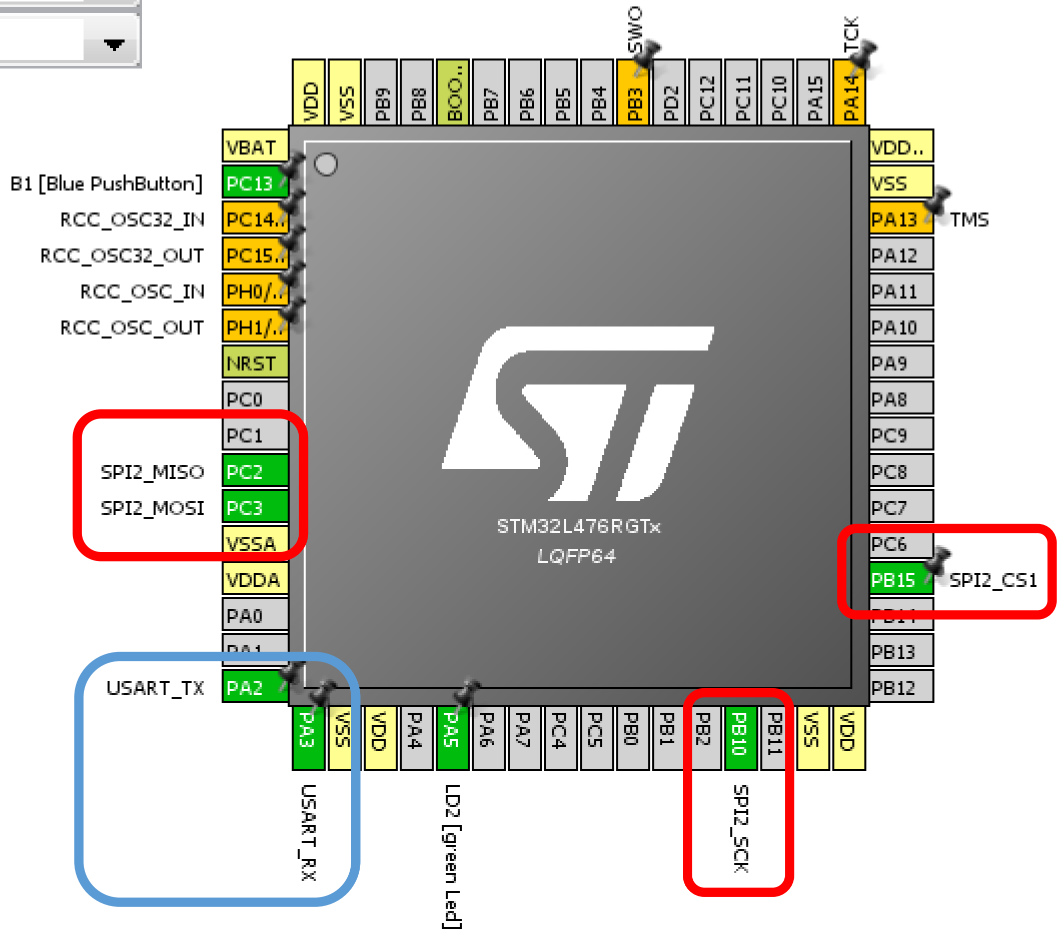
USART2

Mode: Asynchronous

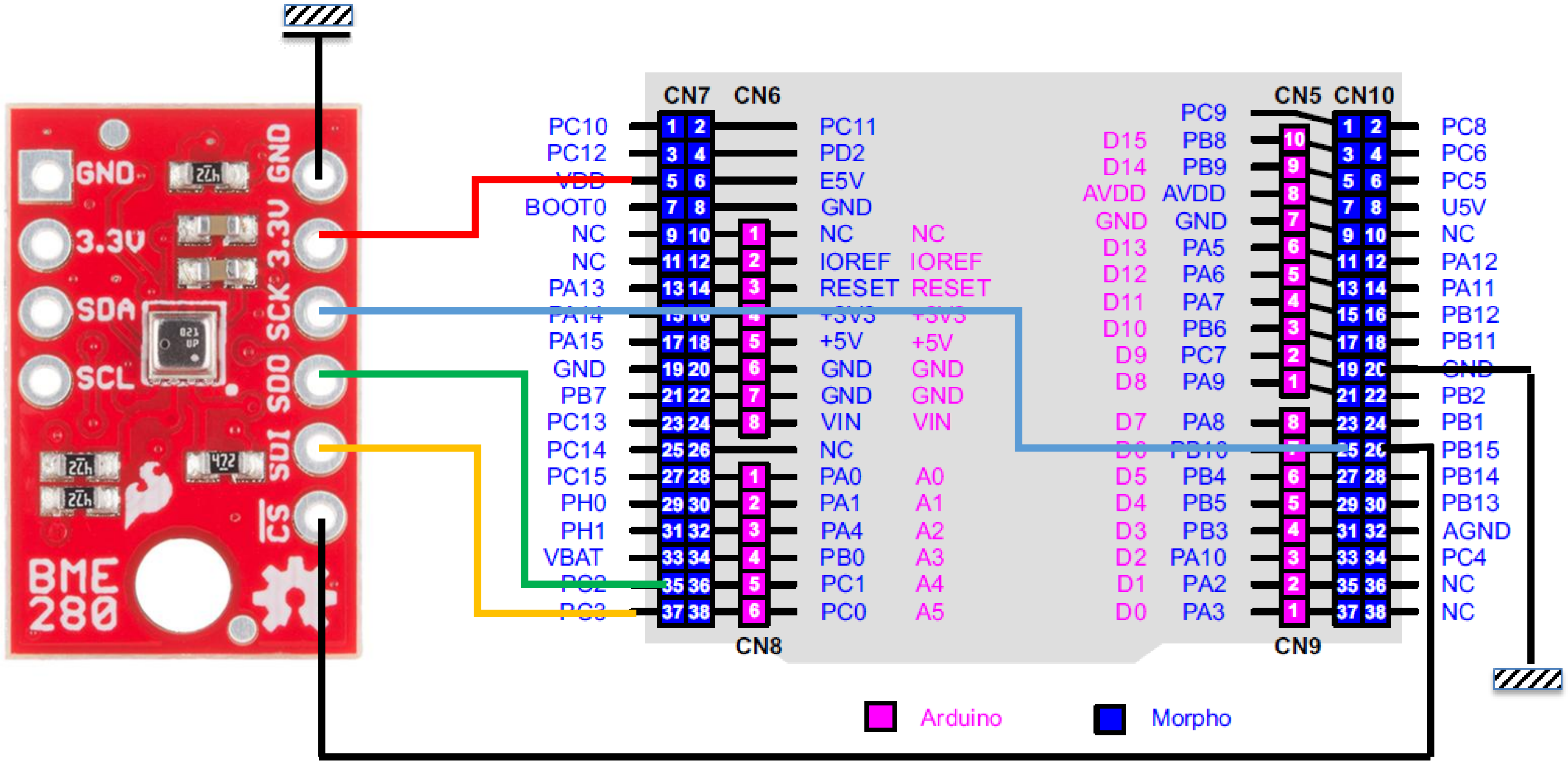
Hardware Flow Control (RS232): Disable

□ Hardware Flow Control (RS485)

Choose USART2 as the serial bus
PA2 and PA3 are then automatically selected
as circled beside



Nucleo-L476RG and BME280 Wiring


GND
3.3V
SDA
SCL
BME 280
GND
3.3V
SCK
SDO
SDI
CS
CN7
CN6
CN5
CN10
PC10 PC12 VDD BOOT0 NC NC PA13 PA14 PA15 GND PB7 PC13 PC14 PC15 PH0 PH1 VBAT PC2 PC3
PC11 PD2 E5V GND NC IOREF RESET +3V3 +5V GND GND VIN NC PA0 PA1 PA4 PB0 PC1 PC0
NC IOREF RESET +3V3 +5V GND GND VIN A0 A1 A2 A3 A4 A5
D15 D14 AVDD GND D13 D12 D11 D10 D9 D8 D7 D6 D5 D4 D3 D2 D1 D0
PC9 PB8 PB9 AVDD GND PA5 PA6 PA7 PB6 PC7 PA9 PA8 PB10 PB4 PB5 PB3 PA10 PA2 PA3
PC8 PC6 PC5 U5V NC PA12 PA11 PB12 PB11 GND PB2 PB1 PB15 PB14 PB13 AGND PC4 NC NC
CN8
CN9
Arduino
Morpho

/** SPI2 GPIO Configuration

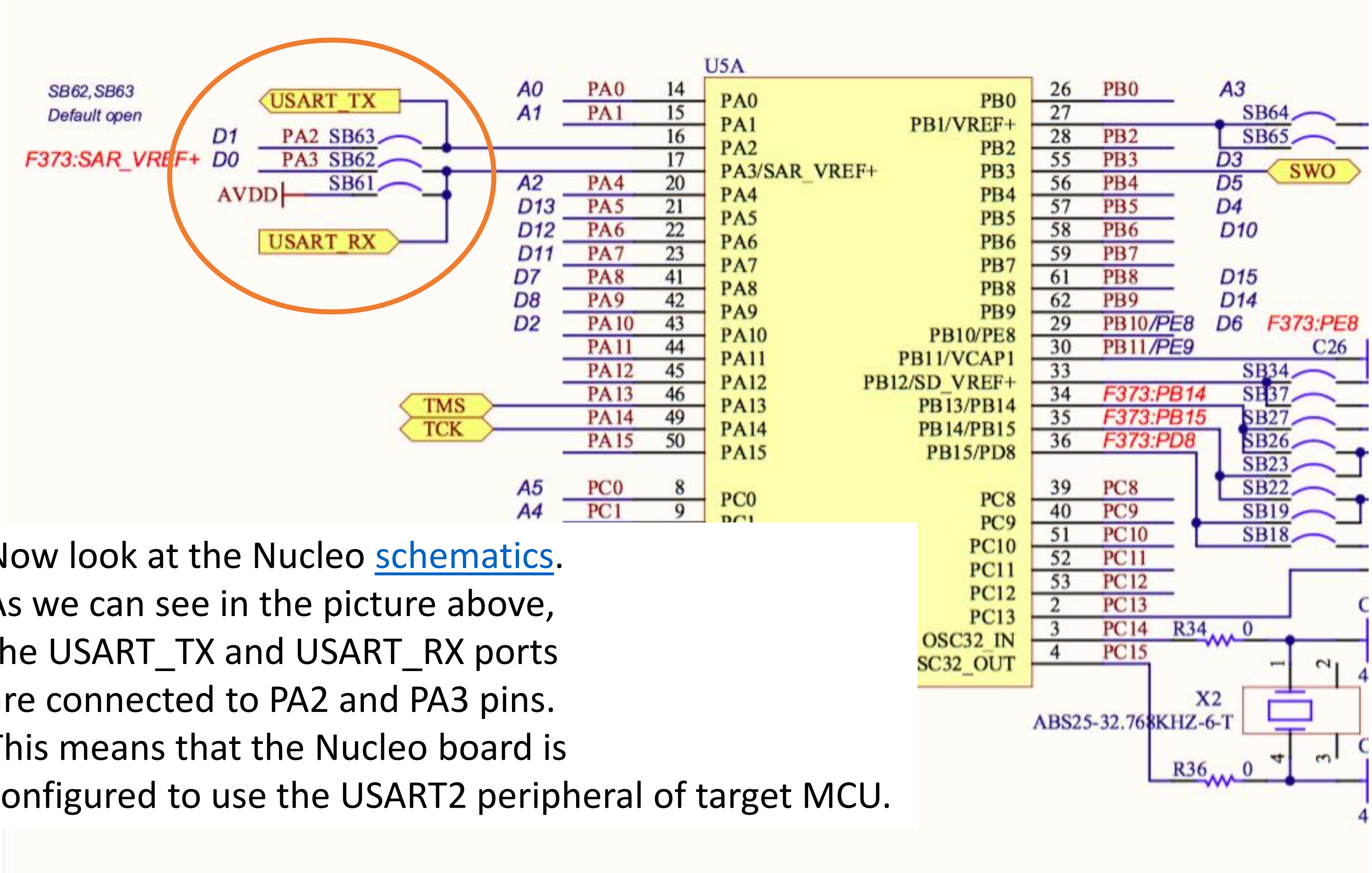
PC2 ------> SPI2_MISO

PC3 ------> SPI2_MOSI

PB10 ------> SPI2_SCK

DO NOT PLUG THE NUCLEO TO USB PORT
Before Double-Checking your wiring

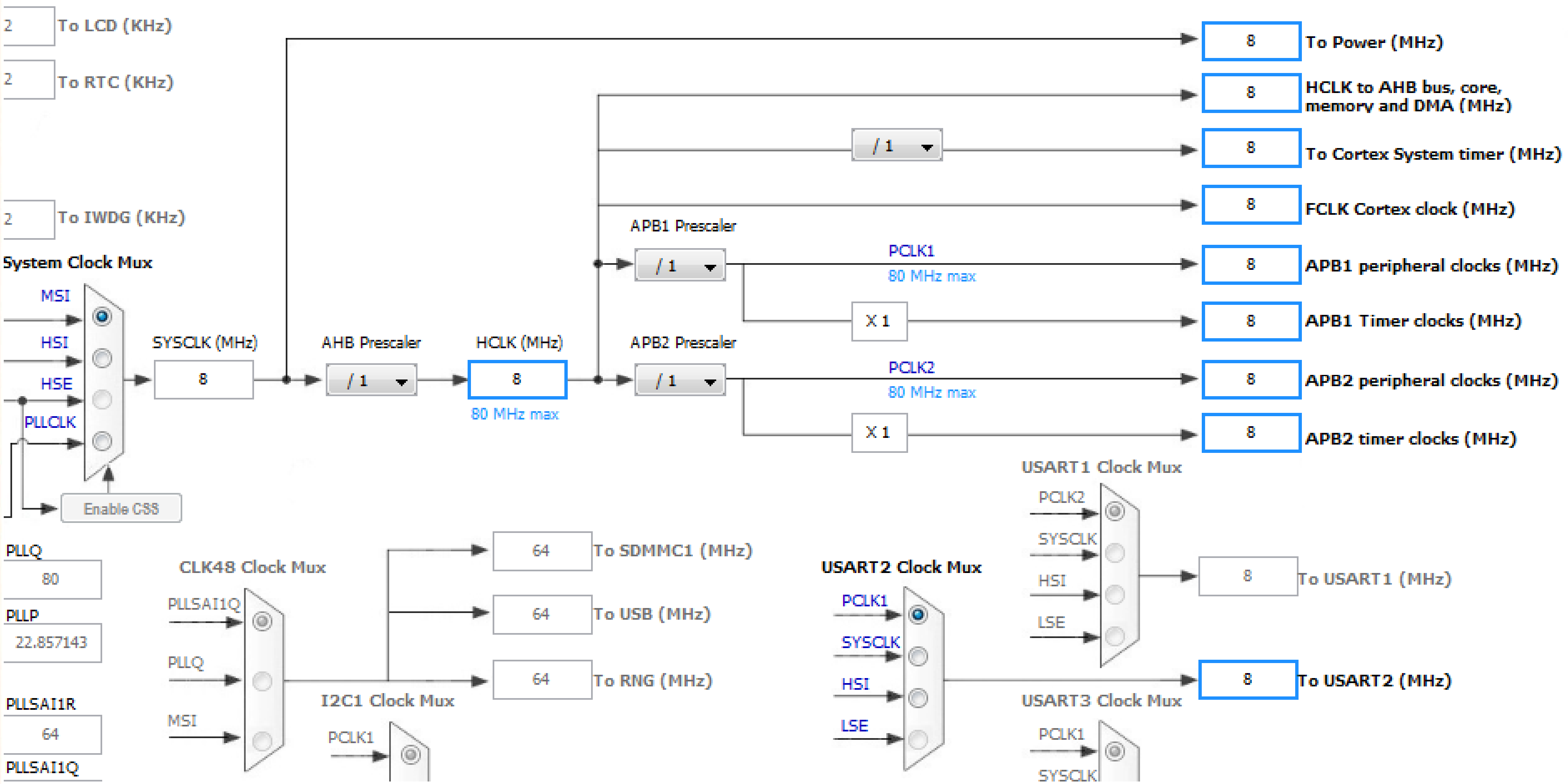
User manual UM1724 page 61



Now look at the Nucleo schematics.
As we can see in the picture above,
the USART_TX and USART_RX ports
are connected to PA2 and PA3 pins.
This means that the Nucleo board is
configured to use the USART2 peripheral of target MCU.

Nucleo64-STM32L476RG : clock configuration

The clocks configuration used for this project are :



2 To LCD (KHz)
2 To RTC (KHz)
2 To IWDG (KHz)

SYSCLK (MHz): 8
AHB Prescaler: / 1
HCLK (MHz): 8 (80 MHz max)
/ 1
APB1 Prescaler: / 1
APB2 Prescaler: / 1

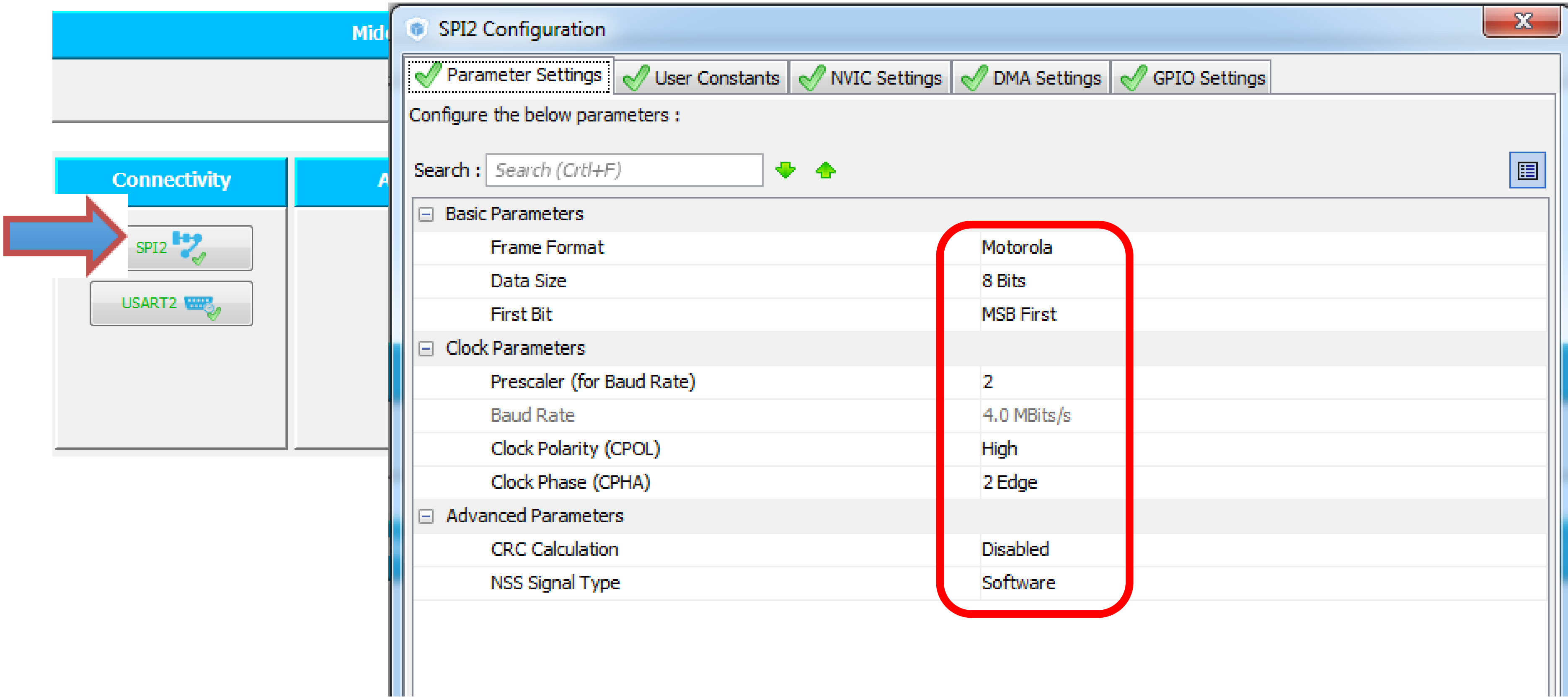
8 To Power (MHz)
8 HCLK to AHB bus, core, memory and DMA (MHz)
8 To Cortex System timer (MHz)
8 FCLK Cortex clock (MHz)
8 APB1 peripheral clocks (MHz)
8 APB1 Timer clocks (MHz)
8 APB2 peripheral clocks (MHz)
8 APB2 timer clocks (MHz)

PLLQ: 80
PLLP: 22.857143
PLLSAI1R: 64
PLLSAI1Q

64 To SDMMC1 (MHz)
64 To USB (MHz)
64 To RNG (MHz)

8 To USART1 (MHz)
8 To USART2 (MHz)

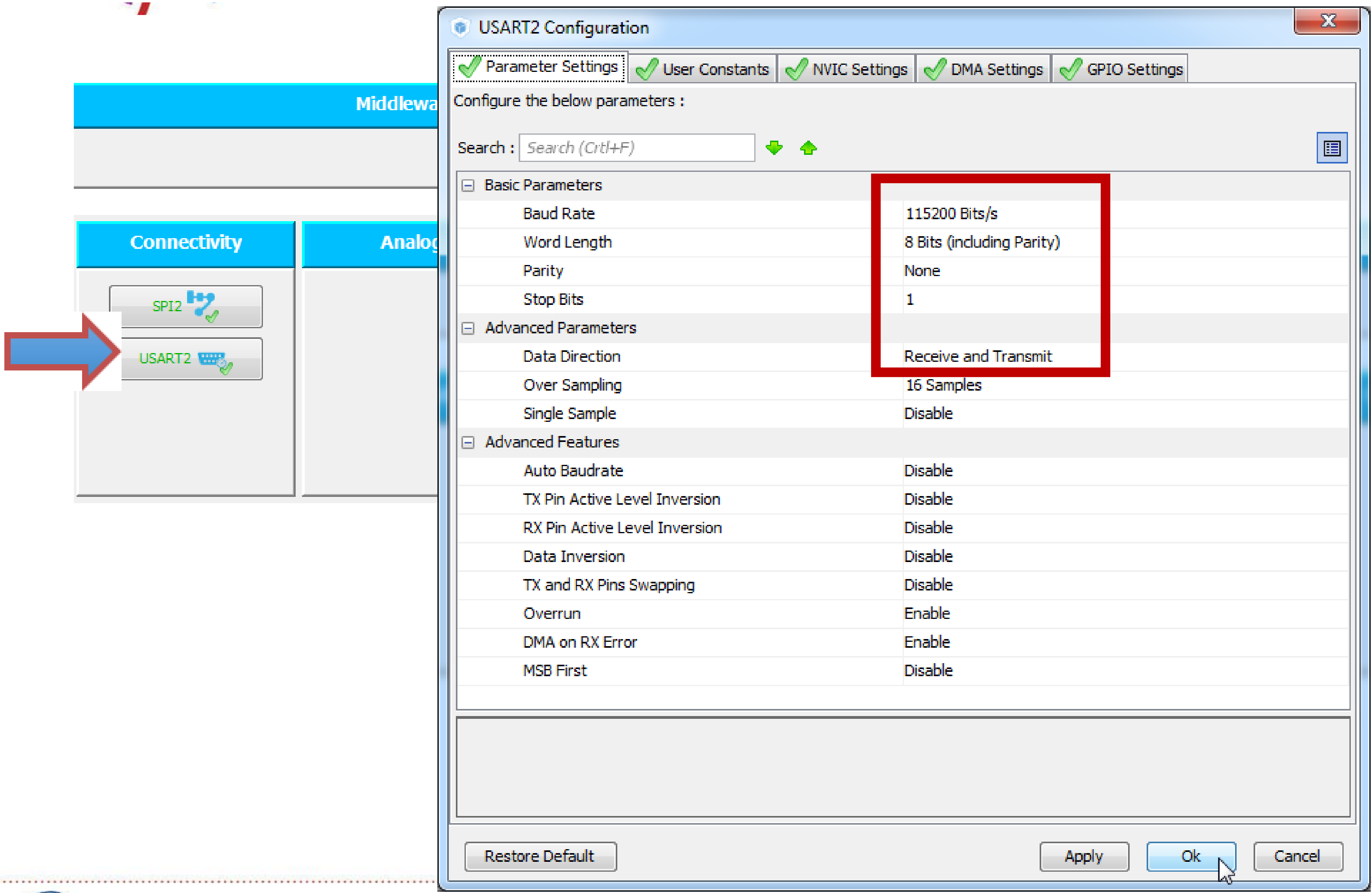
Nucleo64-STM32L476RG : SPI2 and USART2 configuration



Basic Parameters	
Frame Format	Motorola
Data Size	8 Bits
First Bit	MSB First
Clock Parameters	
Prescaler (for Baud Rate)	2
Baud Rate	4.0 MBits/s
Clock Polarity (CPOL)	High
Clock Phase (CPHA)	2 Edge
Advanced Parameters	
CRC Calculation	Disabled
NSS Signal Type	Software

Baud Rate should be less 10 Mbits/s for BME280 sensor

Nucleo64-STM32L476RG : USART2 configuration



Configure the below parameters :

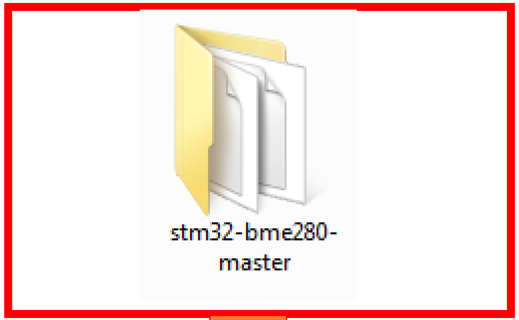
Basic Parameters	
Baud Rate	115200 Bits/s
Word Length	8 Bits (including Parity)
Parity	None
Stop Bits	1
Advanced Parameters	
Data Direction	Receive and Transmit
Over Sampling	16 Samples
Single Sample	Disable
Advanced Features	
Auto Baudrate	Disable
TX Pin Active Level Inversion	Disable
RX Pin Active Level Inversion	Disable
Data Inversion	Disable
TX and RX Pins Swapping	Disable
Overrun	Enable
DMA on RX Error	Enable
MSB First	Disable

SPI BME280 Library for STM32

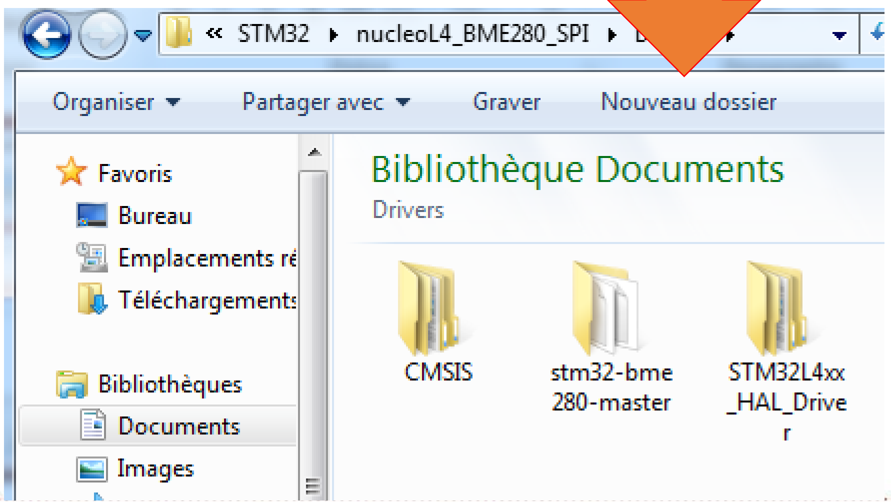
The BME280 Library for STM32F446 developed by Bosch was adapted for STM32L4

Download http://communication.minatec.inpg.fr/toussaint/STM32/stm32-bme280-master.tar.gz

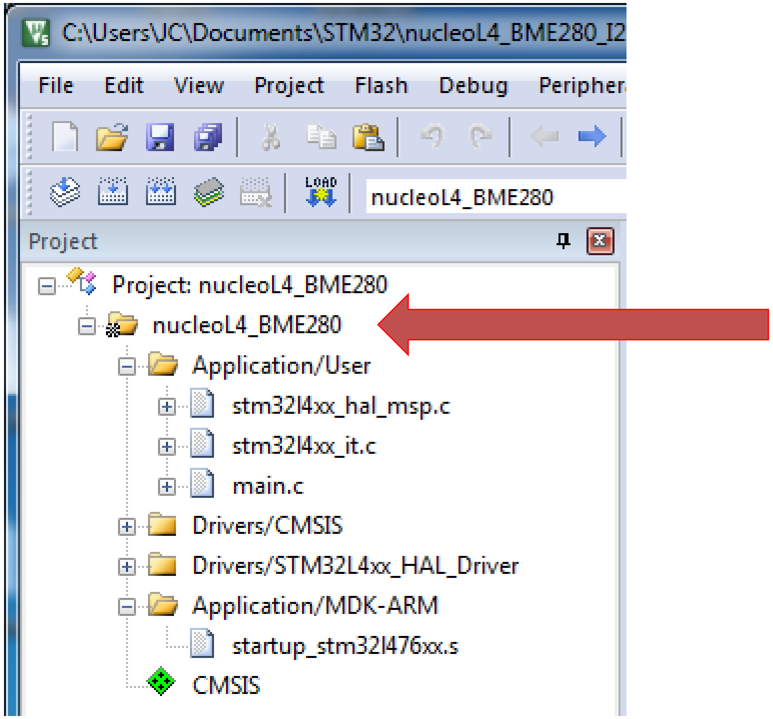
Uncompress it with winzip or equivalent

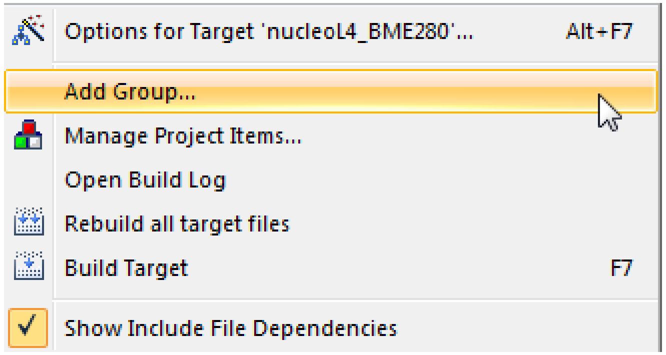


To be copied into the Drivers subdirectory of the project

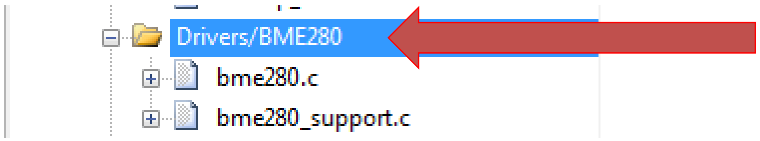


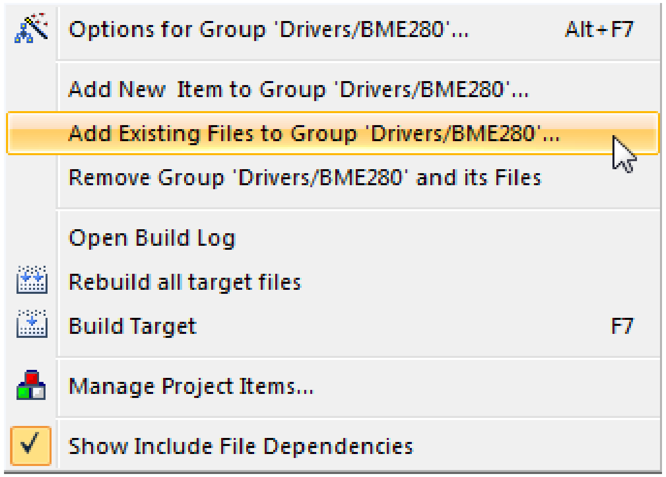
In the Keil environment
Create the group Drivers/BME280



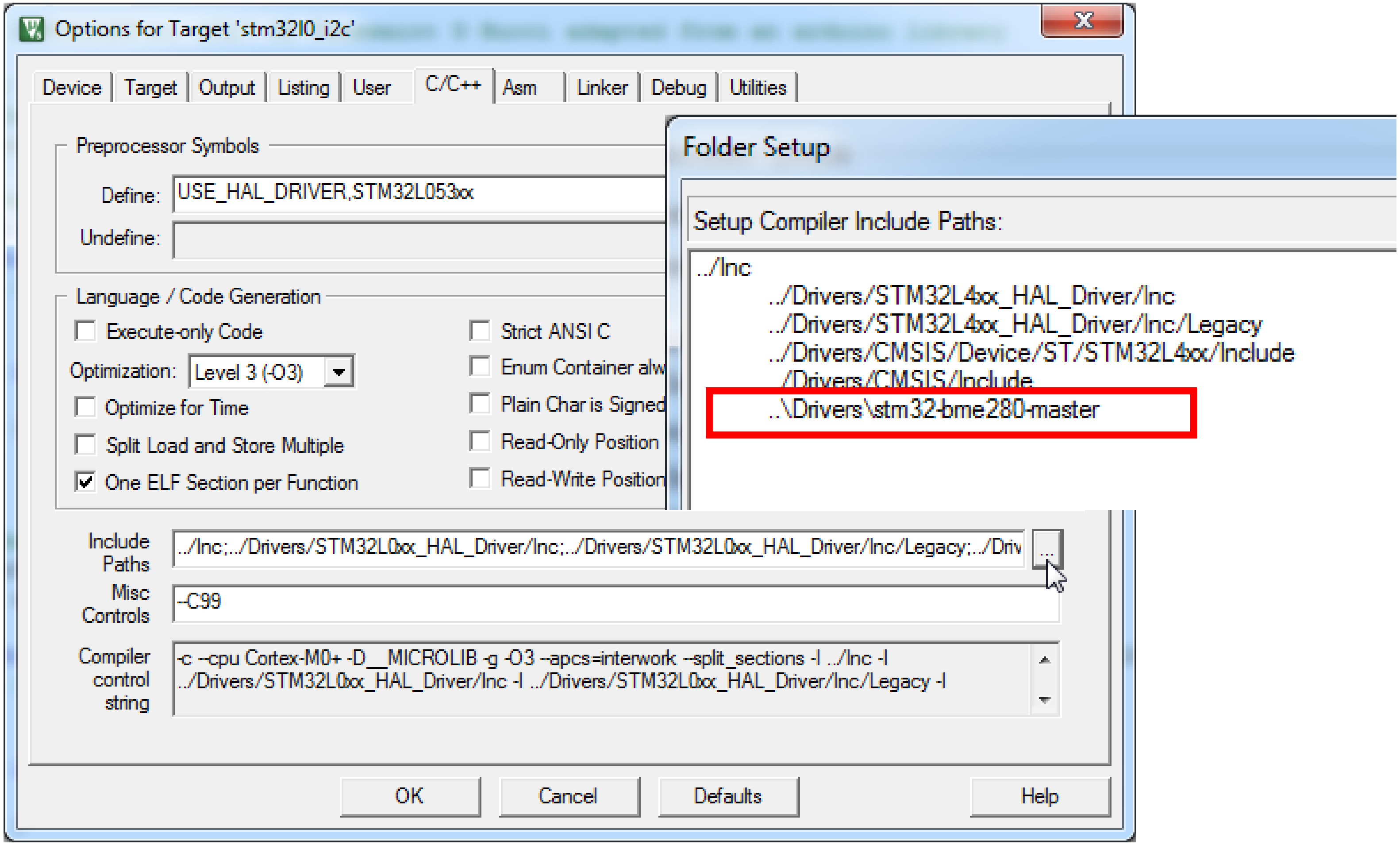


Add the following files into BME280 folder.




Options for Group 'Drivers/BME280'... Alt+F7
Add New Item to Group 'Drivers/BME280'...
Add Existing Files to Group 'Drivers/BME280'...
Remove Group 'Drivers/BME280' and its Files
Open Build Log
Rebuild all target files
Build Target F7
Manage Project Items...
Show Include File Dependencies

Adding paths to the « include » files driving the BME280 components



Define: USE_HAL_DRIVER,STM32L053xx

Undefine:

☐ Execute-only Code

Optimization: Level 3 (-O3)

☐ Optimize for Time

☐ Split Load and Store Multiple

☑ One ELF Section per Function

☐ Strict ANSI C

☐ Enum Container alw

☐ Plain Char is Signed

☐ Read-Only Position

☐ Read-Write Position

Include Paths: ../Inc;../Drivers/STM32L0xx_HAL_Driver/Inc;../Drivers/STM32L0xx_HAL_Driver/Inc/Legacy;../Driv

Misc Controls: -C99

Compiler control string:

```
-c --cpu Cortex-M0+ -D__MICROLIB -g -O3 --apcs=interwork --split_sections -I ../Inc -I
../Drivers/STM32L0xx_HAL_Driver/Inc -I ../Drivers/STM32L0xx_HAL_Driver/Inc/Legacy -I
```

Setup Compiler Include Paths:

```
../Inc
../Drivers/STM32L4xx_HAL_Driver/Inc
../Drivers/STM32L4xx_HAL_Driver/Inc/Legacy
../Drivers/CMSIS/Device/ST/STM32L4xx/Include
../Drivers/CMSIS/Include
..\Drivers\stm32-bme280-master
```

Part I : measuring temperature and pressure in polling mode

Includes to be added in file main.c

```
/* USER CODE BEGIN Includes */
#include <stdio.h>
#include "bme280.h"
/* USER CODE END Includes */

/* USER CODE BEGIN PV */
struct bme280_t mybme280;
/* USER CODE END PV */

/* USER CODE BEGIN PFP */
/* Private function prototypes ---------*/
s32 bme280_data_readout_template(void);
/* USER CODE END PFP */
```

Lines to be added in file main.c

The measurements are done in polling mode and therefore waste a lot of CPU time and power consumption.

```
/* Infinite loop */
  /* USER CODE BEGIN WHILE */
  while (1){
  /* USER CODE END WHILE */
  /* USER CODE BEGIN 3 */
       int status=bme280_data_readout_template();
        printf("status %d\n", status);
        HAL_Delay(1000);
  }
/* USER CODE END 3 */

/* USER CODE BEGIN 4 */
int fputc(int ch, FILE *f){
  uint8_t c=(uint8_t) (ch & 0x00FF);
  HAL_UART_Transmit(&huart2,&c,1,10);
  return ch;
  }
/* USER CODE END 4 */
```

Running the NucleoL4_BME280 project

- Compile the project sources within

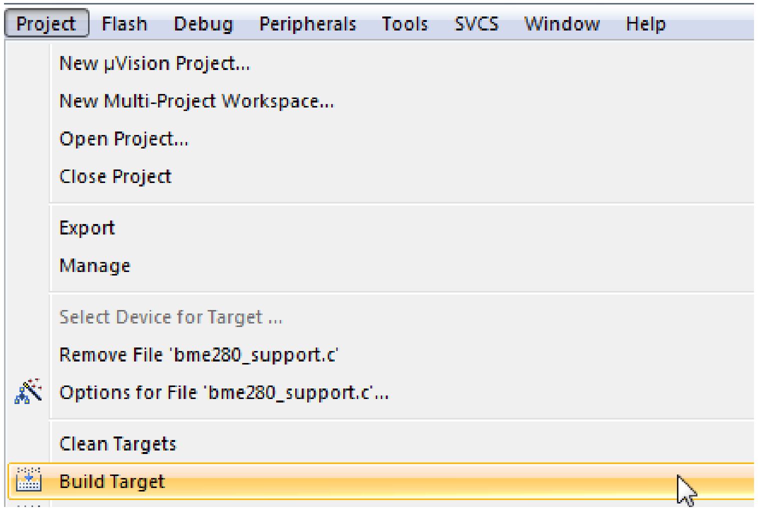
the keil environment

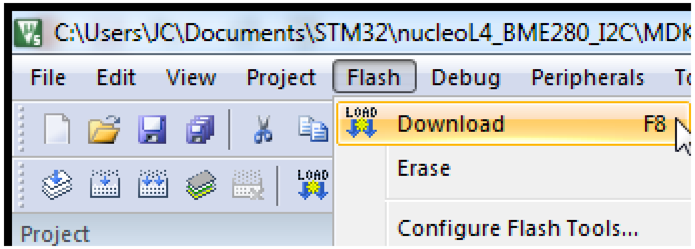
- Download the binary into the MCU

- Install a serial terminal like Termite

https://www.compuphase.com/software_termite.htm

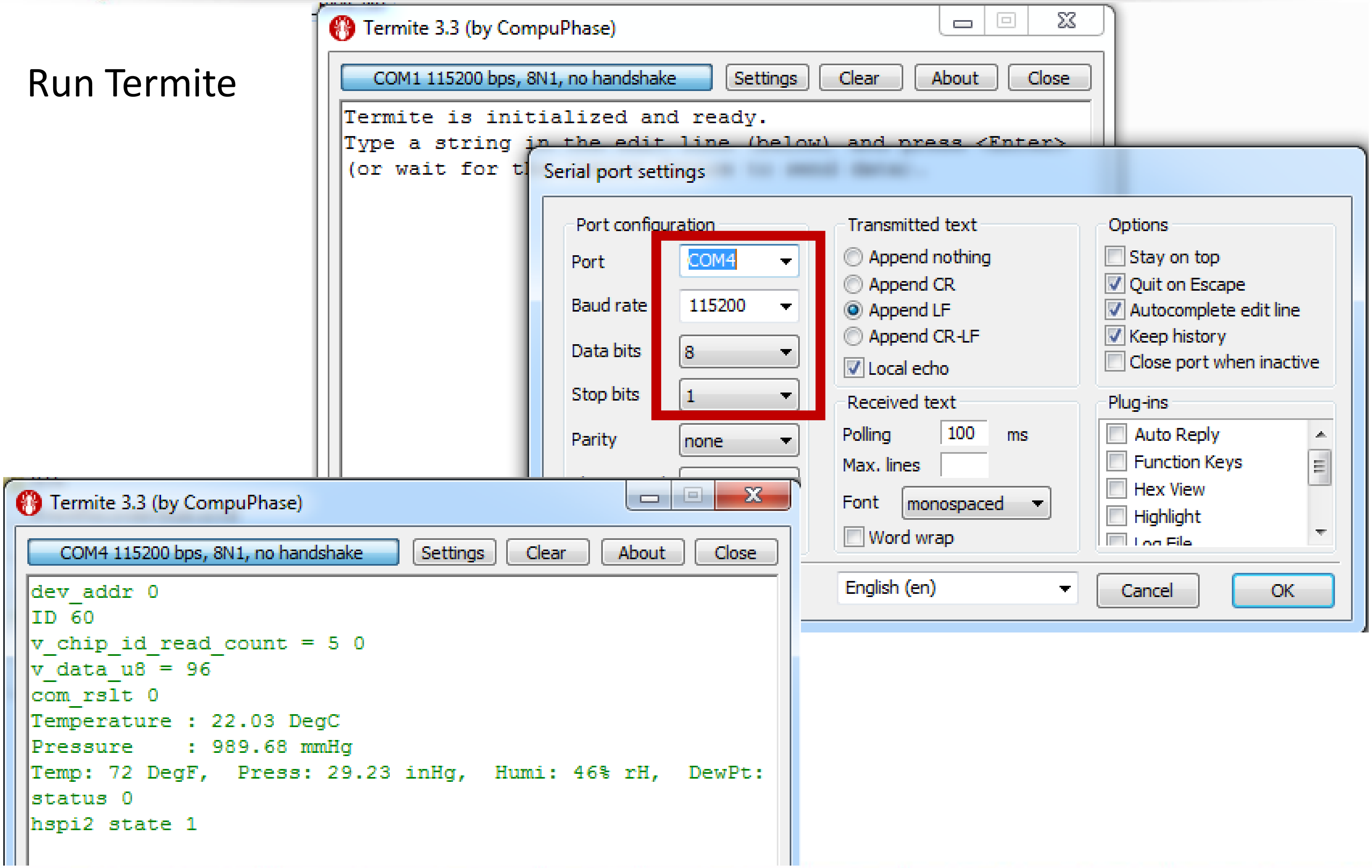
- Reset the MCU for running and enjoy





Serial Terminal Configuration

- Run Termite



Termite 3.3 (by CompuPhase)

```
Termite is initialized and ready.
Type a string in the edit line (below) and press <Enter>
(or wait for t
```

Serial port settings

Port configuration

- Port: COM4
- Baud rate: 115200
- Data bits: 8
- Stop bits: 1
- Parity: none

Transmitted text

- ○ Append nothing
- ○ Append CR
- ● Append LF
- ○ Append CR-LF
- ☑ Local echo

Options

- □ Stay on top
- ☑ Quit on Escape
- ☑ Autocomplete edit line
- ☑ Keep history
- □ Close port when inactive

Received text

- Polling: 100 ms
- Max. lines:
- Font: monospaced
- □ Word wrap

Plug-ins

- □ Auto Reply
- □ Function Keys
- □ Hex View
- □ Highlight
- □ Log File

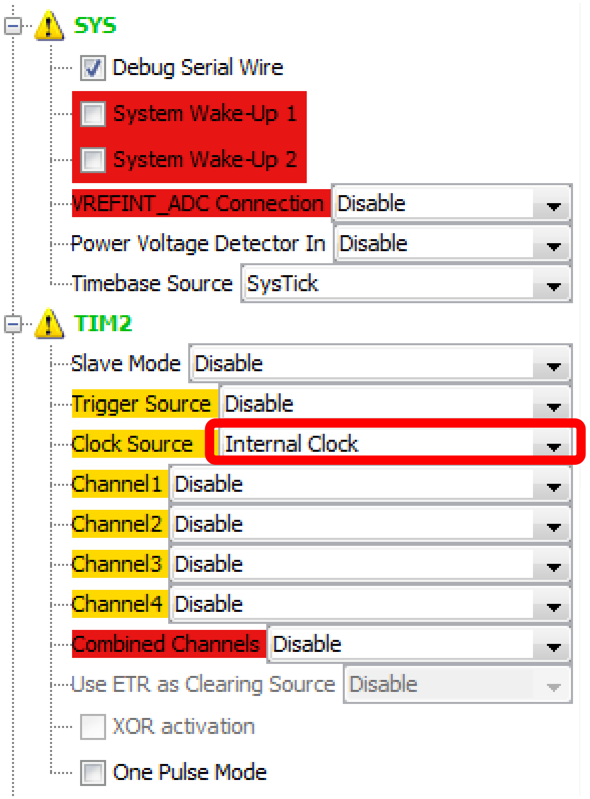
Termite 3.3 (by CompuPhase)

```
dev_addr 0
ID 60
v_chip_id_read_count = 5 0
v_data_u8 = 96
com_rslt 0
Temperature : 22.03 DegC
Pressure    : 989.68 mmHg
Temp: 72 DegF,  Press: 29.23 inHg,  Humi: 46% rH,  DewPt:
status 0
hspi2 state 1
```

Part II - Interrupt managed by a timer

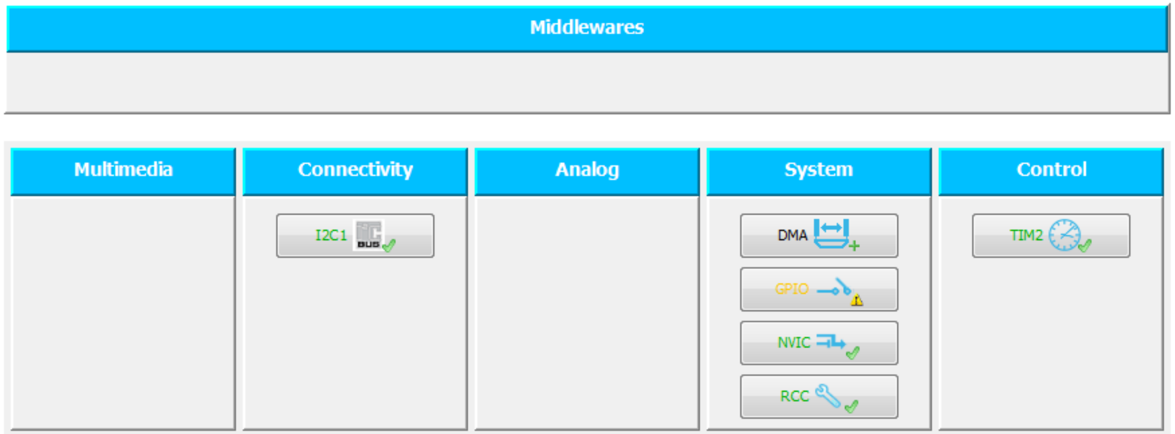
The goal is to reduce the power consumption. One measures temperature and pressure at regular intervals and puts the MCU in a sleep mode between two successive measurements.

A timer with a counter is used. An update Event is generated when counter reaches a given value.



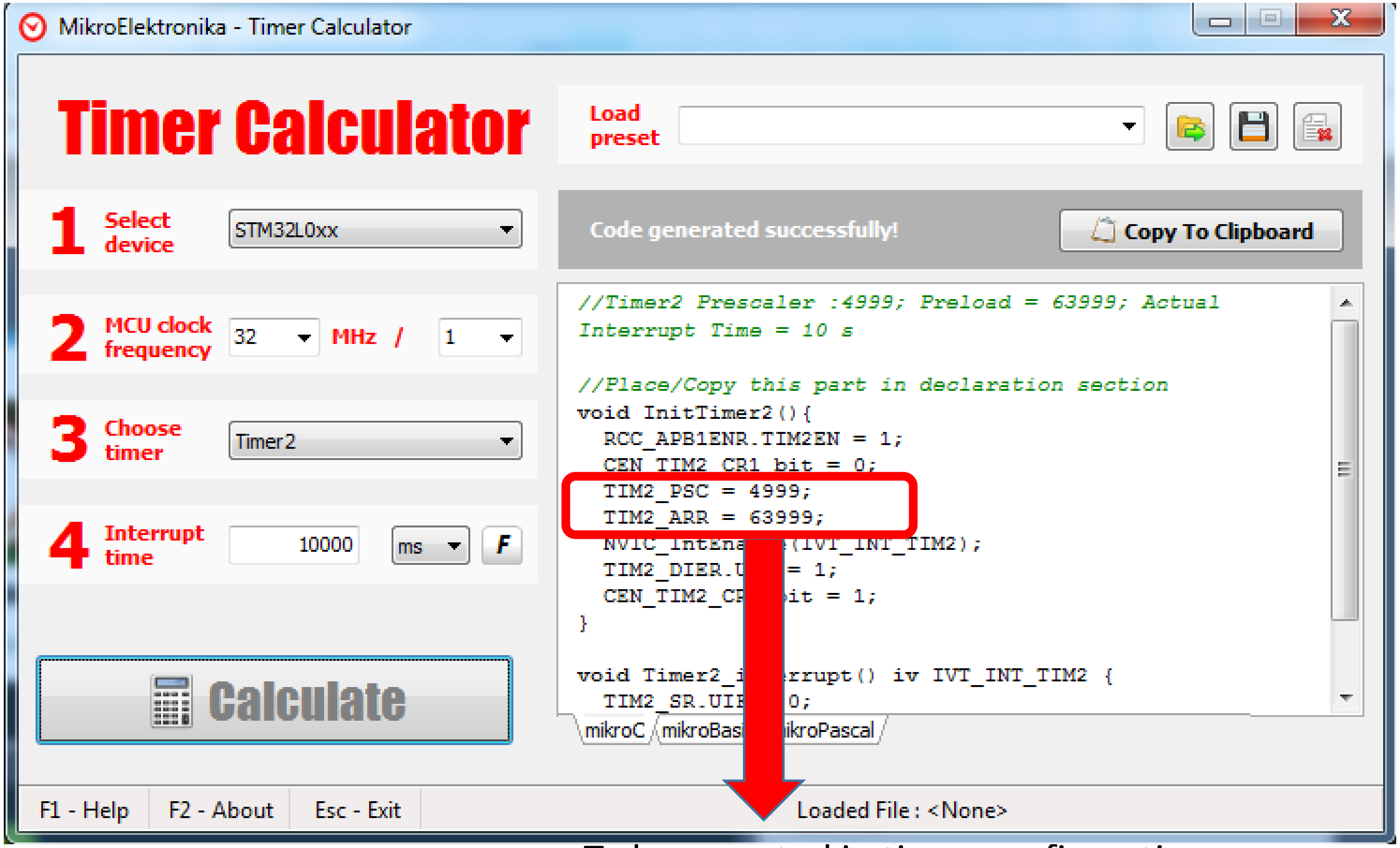
SYS
- ☑ Debug Serial Wire
- ☐ System Wake-Up 1
- ☐ System Wake-Up 2
- VREFINT_ADC Connection: Disable
- Power Voltage Detector In: Disable
- Timebase Source: SysTick

TIM2
- Slave Mode: Disable
- Trigger Source: Disable
- Clock Source: Internal Clock
- Channel1: Disable
- Channel2: Disable
- Channel3: Disable
- Channel4: Disable
- Combined Channels: Disable
- Use ETR as Clearing Source: Disable
- ☐ XOR activation
- ☐ One Pulse Mode



The calculations of “Prescaler” (PSC) and “Counter Period” (ARR) parameters are carried out using the MikroElektronika application “timer calculator”

An interrupt is generated each T=10s, using SysClock =32 MHz
ARR et PSC are such as T = (ARR+1)*(PSC+1)/32e6



MikroElektronika - Timer Calculator

Timer Calculator

Load preset

1 Select device: STM32L0xx

2 MCU clock frequency: 32 MHz / 1

3 Choose timer: Timer2

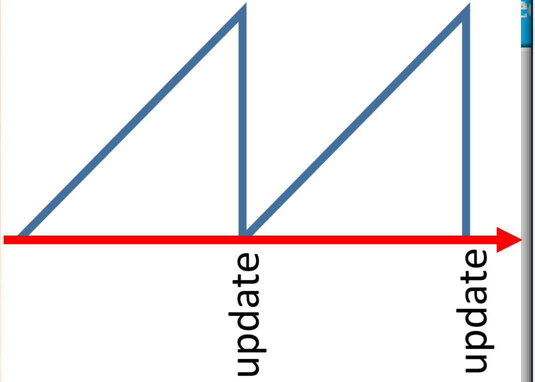
4 Interrupt time: 10000 ms

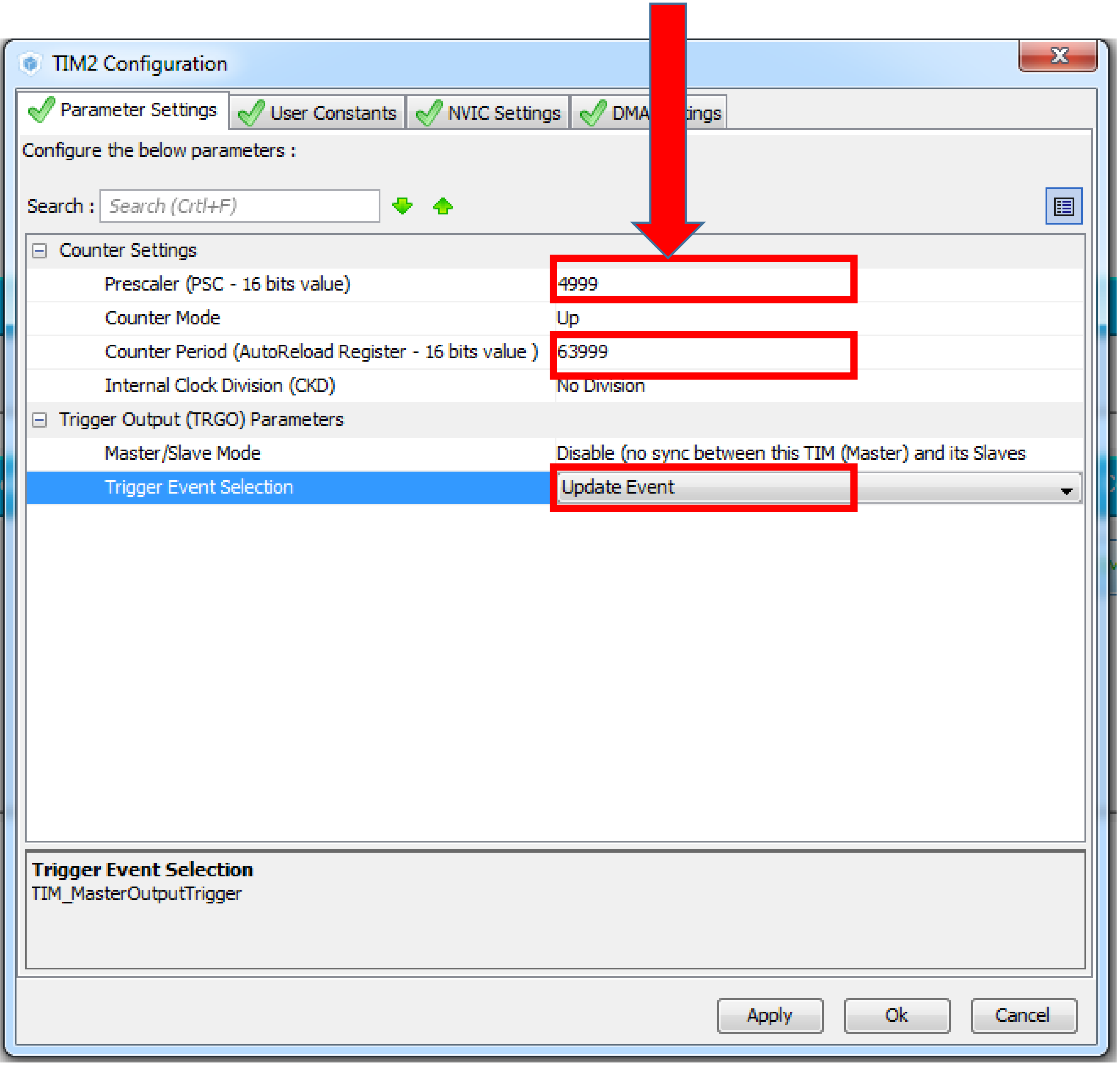
Code generated successfully!

```
//Timer2 Prescaler :4999; Preload = 63999; Actual
Interrupt Time = 10 s

//Place/Copy this part in declaration section
void InitTimer2(){
  RCC_APB1ENR.TIM2EN = 1;
  CEN_TIM2_CR1_bit = 0;
  TIM2_PSC = 4999;
  TIM2_ARR = 63999;
  NVIC_IntEna  (IVT_INT_TIM2);
  TIM2_DIER.U  = 1;
  CEN_TIM2_CR  it = 1;
}

void Timer2_i  rrupt() iv IVT_INT_TIM2 {
  TIM2_SR.UII  0;
```

To be reported in timer configuration

TIM2





Configure the below parameters :

Counter Settings

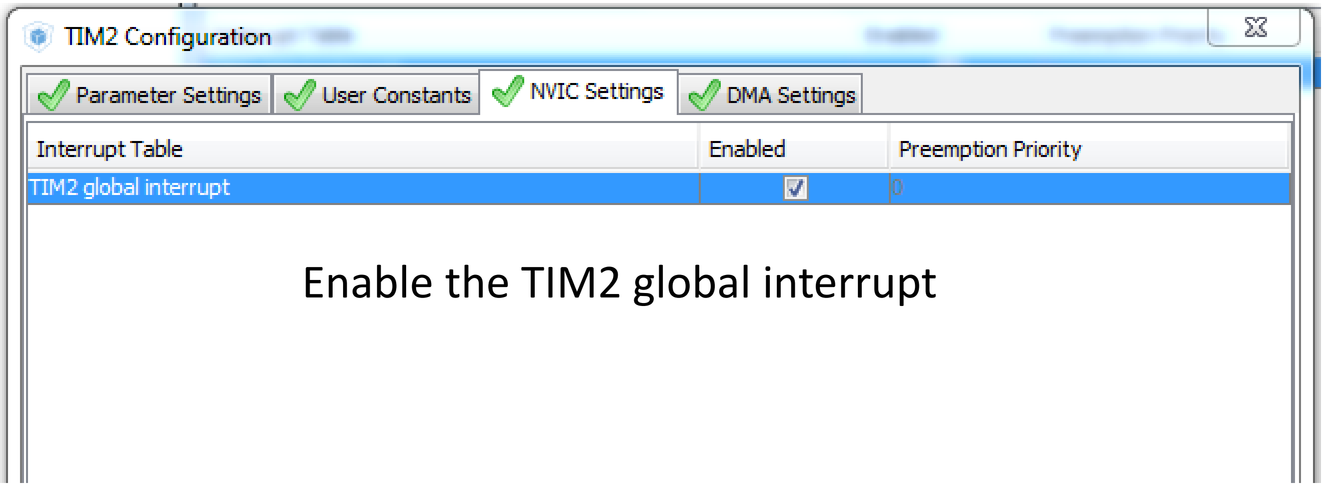
Prescaler (PSC - 16 bits value)	4999
Counter Mode	Up
Counter Period (AutoReload Register - 16 bits value)	63999
Internal Clock Division (CKD)	No Division

Trigger Output (TRGO) Parameters

Master/Slave Mode	Disable (no sync between this TIM (Master) and its Slaves
Trigger Event Selection	Update Event

Trigger Event Selection
TIM_MasterOutputTrigger

NVIC Settings

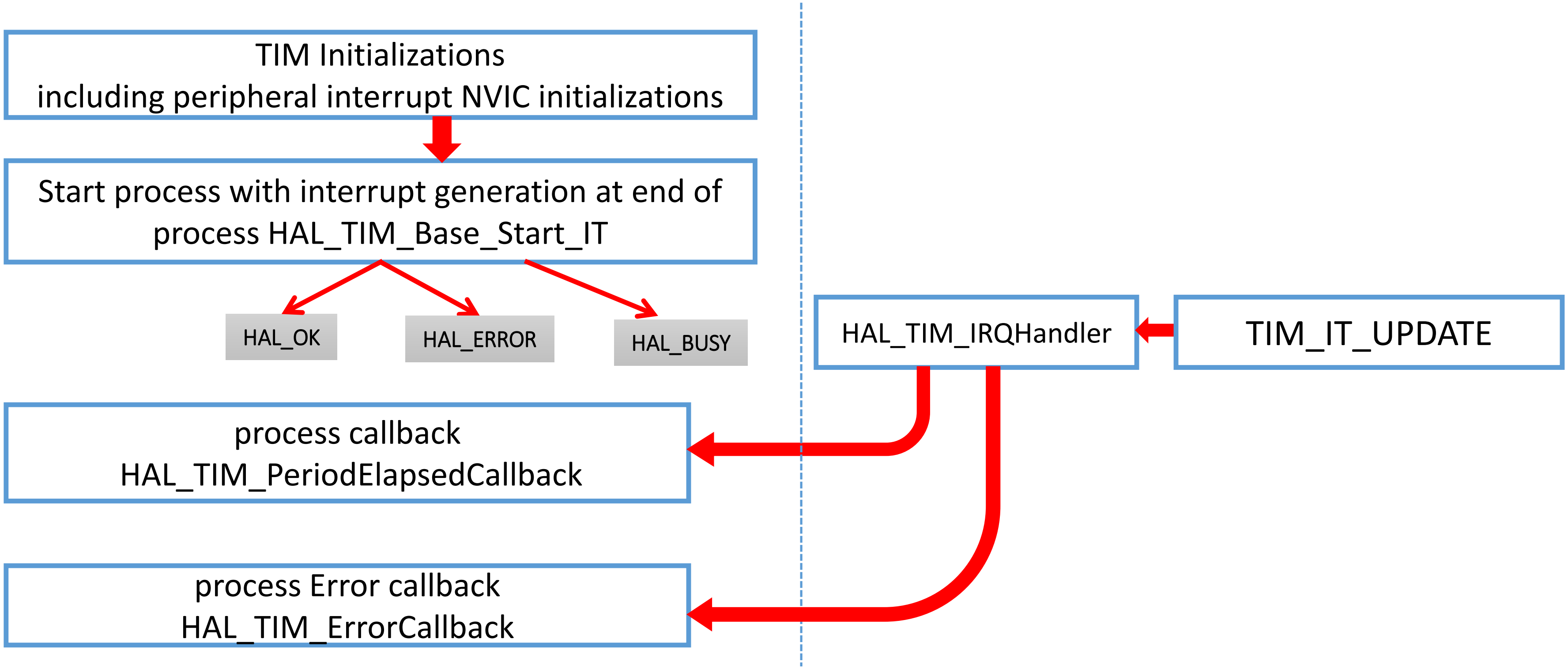


TIM2 Configuration

Interrupt Table	Enabled	Preemption Priority
TIM2 global interrupt	☑	0

Enable the TIM2 global interrupt

HAL Library TIM with IT flow



TIM Initializations
including peripheral interrupt NVIC initializations

Start process with interrupt generation at end of process HAL_TIM_Base_Start_IT

HAL_OK

HAL_ERROR

HAL_BUSY

HAL_TIM_IRQHandler

TIM_IT_UPDATE

process callback
HAL_TIM_PeriodElapsedCallback

process Error callback
HAL_TIM_ErrorCallback

```
/* USER CODE BEGIN Includes */
/* USER CODE END Includes */

/* USER CODE BEGIN 2 */
HAL_TIM_Base_Start_IT(&htim2);
/* USER CODE END 2 */

/* USER CODE BEGIN 4 */
void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim){
  __NOP();    // Only for debugging
  }

/* USER CODE END 4 */
```

Details of the Infinite loop in main

```
/* Infinite loop */
 /* USER CODE BEGIN WHILE */

 while (1)
   {
   /* USER CODE BEGIN 3 */
   HAL_SuspendTick();
   HAL_PWR_EnterSLEEPMode(PWR_LOWPOWERREGULATOR_ON, PWR_SLEEPENTRY_WFI);

   HAL_ResumeTick();
   int status=bme280_data_readout_template();
   printf("status %d\n", status);
   }
 /* USER CODE END WHILE */
```

Running the NucleoL4_BME280 project

- Compile the project sources within

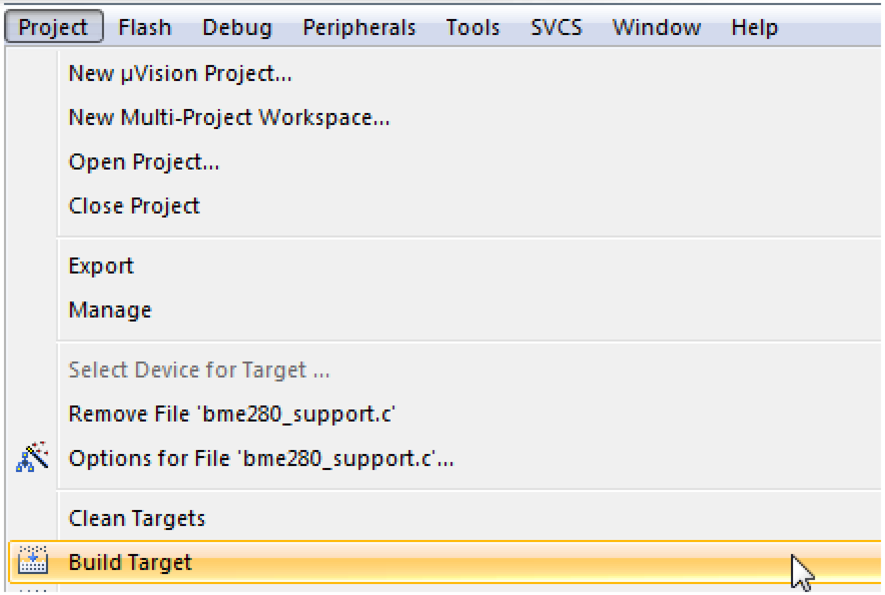
the keil environment

- Download the binary into the MCU

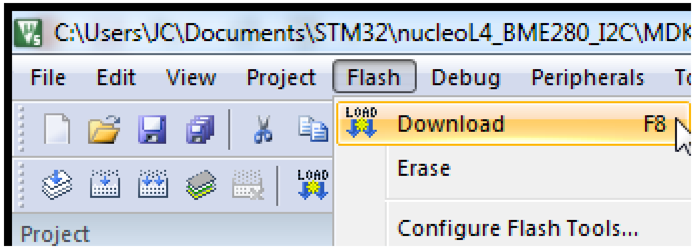
- Install a serial terminal for windows like

Termite

- Reset the MCU for running and enjoy



Project
Flash
Debug
Peripherals
Tools
SVCS
Window
Help
New µVision Project...
New Multi-Project Workspace...
Open Project...
Close Project
Export
Manage
Select Device for Target ...
Remove File 'bme280_support.c'
Options for File 'bme280_support.c'...
Clean Targets
Build Target



C:\Users\JC\Documents\STM32\nucleoL4_BME280_I2C\MDK
File
Edit
View
Project
Flash
Debug
Peripherals
To
LOAD
Download
F8
Erase
Configure Flash Tools...
LOAD
Project