A121 STM32CubeIDE

User Guide

A121 STM32CubeIDE

User Guide

Author: Acconeer AB

Version:a121-v1.3.0

Acconeer AB October 6, 2023

Contents

1 Acconeer SDK Documentation Overview

To better understand what SDK document to use, a summary of the documents are shown in the table below.

Table 1: SDK document overview.

Name	Description	When to use
	RSS API documentation (html)	
rss_api	The complete C API documentation.	- RSS application implementation - Understanding RSS API functions
	User guides (PDF)	
A121 Assembly Test	Describes the Acconeer assembly test functionality.	- Bring-up of HW/SW - Production test implementation
A121 Breathing Reference Application	Describes the functionality of the Breathing Reference Application.	- Working with the Breathing Reference Application
A121 Distance Detector	Describes usage and algorithms of the Distance Detector.	- Working with the Distance Detector
A121 SW Integration	Describes how to implement each integration function needed to use the Acconeer sensor.	- SW implementation of custom HW integration
A121 Presence Detector	Describes usage and algorithms of the Presence Detector.	- Working with the Presence Detector
A121 Smart Presence Reference Application	Describes the functionality of the Smart Presence Reference Application.	- Working with the Smart Presence Reference Application
A121 Sparse IQ Service	Describes usage of the Sparse IQ Service.	- Working with the Sparse IQ Service
A121 Tank Level Reference Application	Describes the functionality of the Tank Level Reference Application.	- Working with the Tank Level Reference Application
A121 STM32CubeIDE	Describes the flow of taking an Acconeer SDK and integrate into STM32CubeIDE.	- Using STM32CubeIDE
A121 Raspberry Pi Software	Describes how to develop for Raspberry Pi.	- Working with Raspberry Pi
A121 Ripple	Describes how to develop for Ripple.	- Working with Ripple on Raspberry Pi
XM125 Software	Describes how to develop for XM125.	- Working with XM125
I2C Distance Detector	Describes the functionality of the I2C Distance Detector Application.	- Working with the I2C Distance Detector Application
I2C Presence Detector	Describes the functionality of the I2C Presence Detector Application.	- Working with the I2C Presence Detector Application
	Handbook (PDF)	
Handbook	Describes different aspects of the Acconeer offer, for example radar principles and how to configure	- To understand the Acconeer sensor - Use case evaluation
	Readme (txt)	
[README	Various target specific information and links	- After SDK download

2 Introduction

In this document there will be a short guide with example on how to generate a project and setup the Acconeer software in STM32CubeIDE.

The MCU board used as an example in this guide is a Nucleo-L476RG. We will show how to connect an XE121, including the A121 radar sensor. There is some extra logic on the XE121 board to support multiple sensors that typically is not present on boards with only one sensor. To show how a typical single sensor integration can be done, we have also included some notes on how to use the XE121 in a single sensor setup.

STM32CubeIDE can be downloaded from the ST website at:
`https://www.st.com/en/development-tools/stm32cubeide.html`

This guide has been verified in both Ubuntu 20.04 and Windows with STM32CubeIDE 1.11.2

3 Getting Started with STM32CubeIDE

This section will cover how to setup a project in STM32CubeIDE, and make sure that the code works with the Acconeer software.

Start STM32CubeIDE and click “Start new STM32 project”. The option is also available under “File → New → STM32Project”.

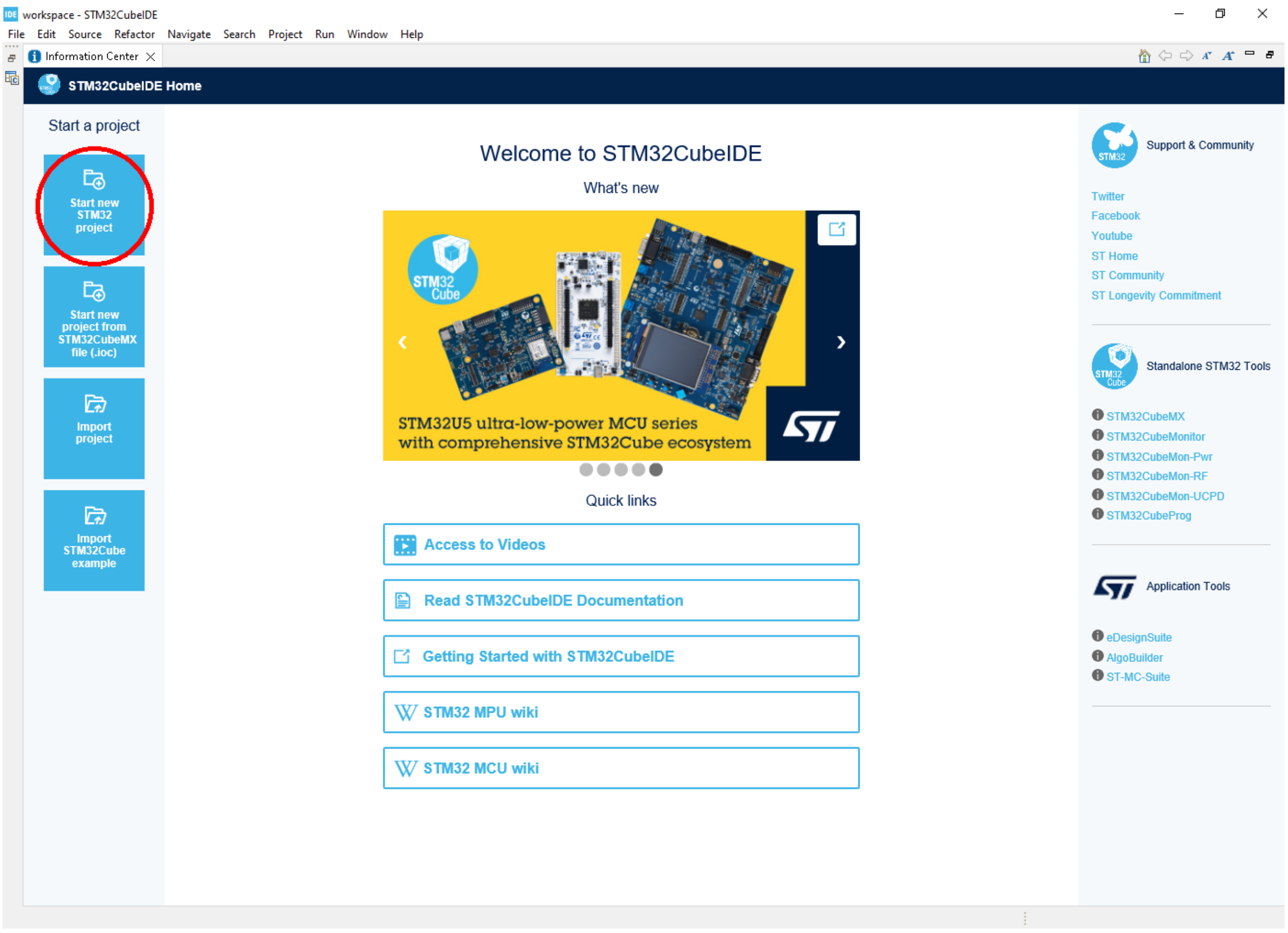


Figure 1: Start new STM32CubeIDE project

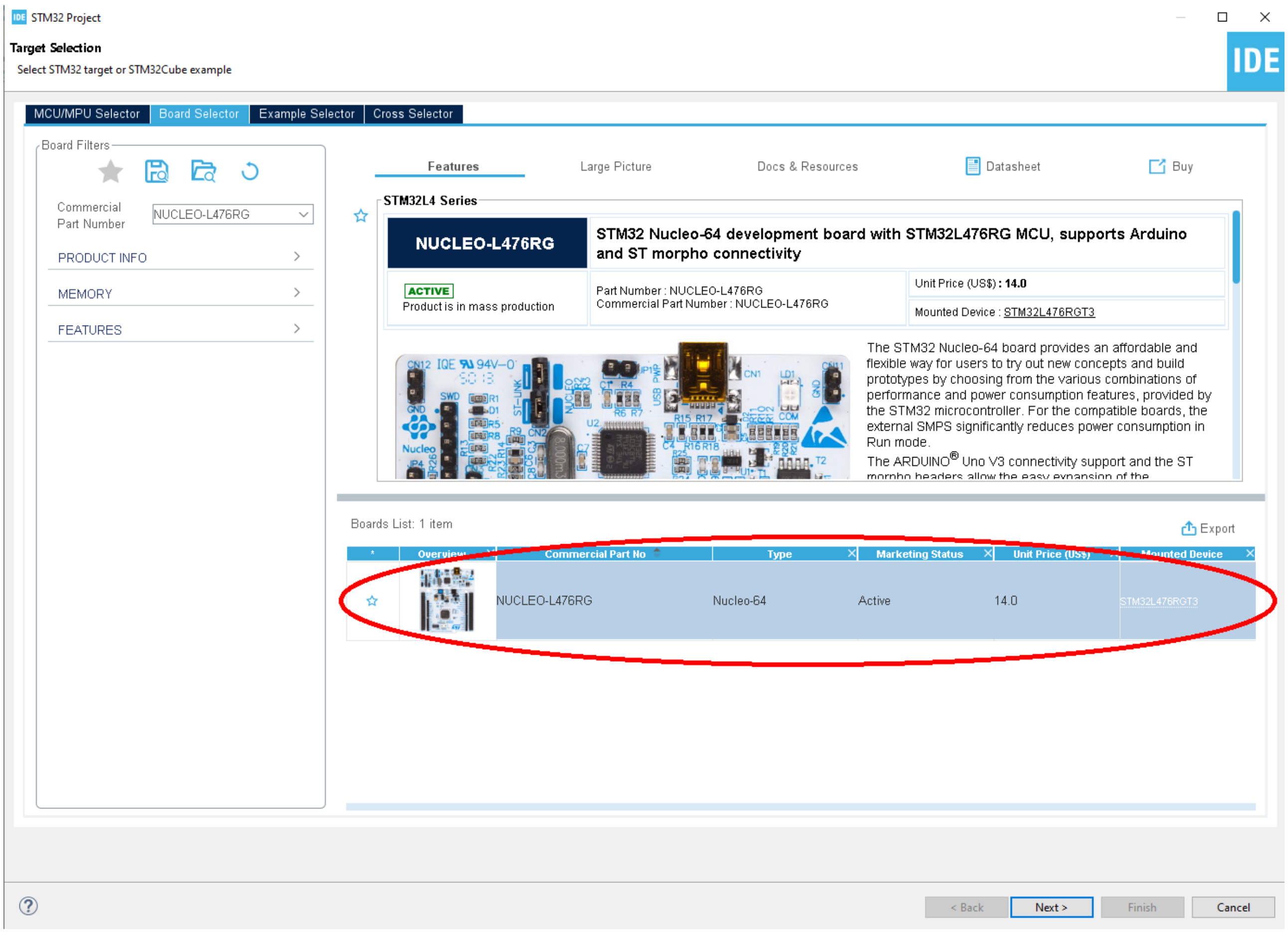
When running on Linux/Ubuntu you might be asked about Connection Parameters, generally, you can skip this part by selecting “No proxy”.

3.1 MCU/Board Selection

Search for the MCU or Board you are working with in the MCU Selector/Board Selector tab. In the example in this document we use the board “NUCLEO-L476RG”. Start off by searching for “NUCLEO-L47” in the “Part Number Search”-option at the top left in the Board Selector tab.

The board will show up in the “Boards list” at the bottom of the page. Click it. When clicking the board or MCU, you will be given some information about it.

Make sure the “Nucleo-L476RG” board is selected and press the “Next”-button at the bottom of the page.



Target Selection

Select STM32 target or STM32Cube example

Commercial Part Number: NUCLEO-L476RG

STM32L4 Series

NUCLEO-L476RG

STM32 Nucleo-64 development board with STM32L476RG MCU, supports Arduino and ST morpho connectivity

ACTIVE
Product is in mass production

Part Number : NUCLEO-L476RG
Commercial Part Number : NUCLEO-L476RG

Unit Price (US$) : **14.0**

Mounted Device : STM32L476RGT3

The STM32 Nucleo-64 board provides an affordable and flexible way for users to try out new concepts and build prototypes by choosing from the various combinations of performance and power consumption features, provided by the STM32 microcontroller. For the compatible boards, the external SMPS significantly reduces power consumption in Run mode.

The ARDUINO® Uno V3 connectivity support and the ST morpho headers allow the easy expansion of the

Boards List: 1 item

*	Overview	Commercial Part No	Type	Marketing Status	Unit Price (US$)	Mounted Device
☆		NUCLEO-L476RG	Nucleo-64	Active	14.0	STM32L476RGT3

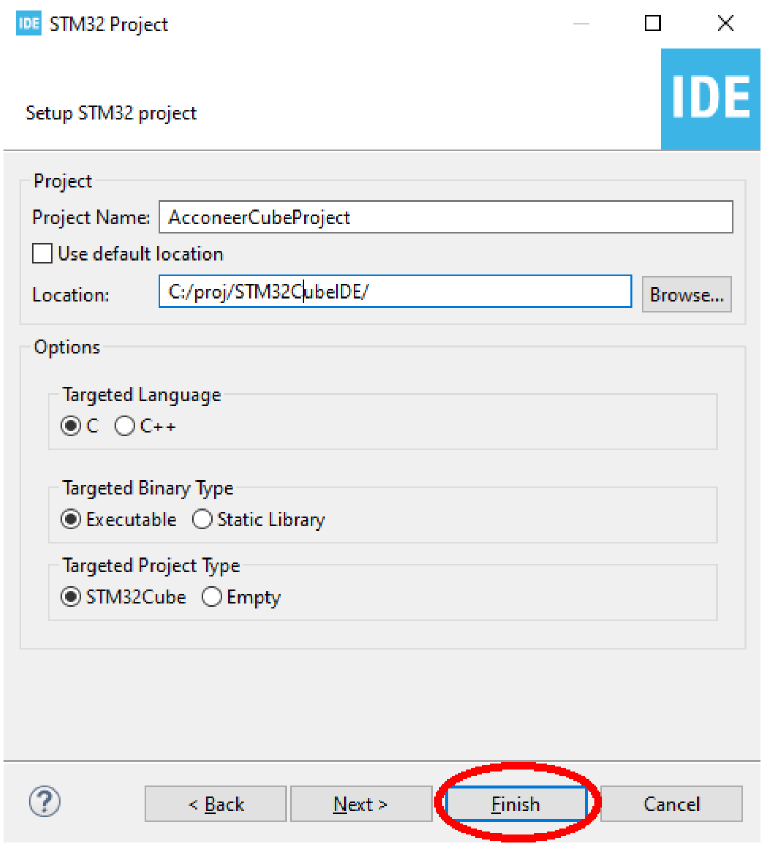
Figure 2: Target Selection

3.2 Project Setup

Select a name and location for your project and select the following options:

- Target language: C
- Target Binary type: Executable
- Target Project Type: STM32Cube

Finally, press the “Finish”-button at the bottom of the page.


STM32 Project

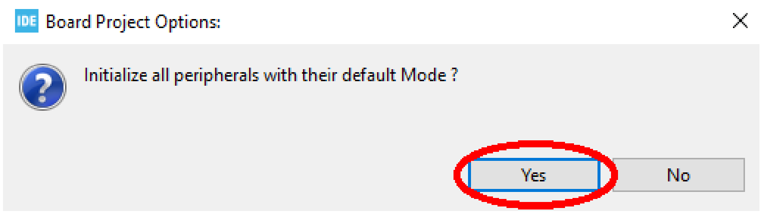
Setup STM32 project

Project
Project Name: AcconeerCubeProject
☐ Use default location
Location: C:/proj/STM32CubeIDE/

Options
Targeted Language: ● C ○ C++
Targeted Binary Type: ● Executable ○ Static Library
Targeted Project Type: ● STM32Cube ○ Empty

Figure 3: Project Setup

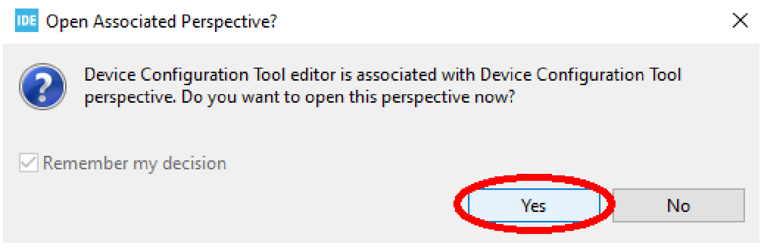
Press “Yes” when you are asked if you want to initialize peripherals to their default mode.


Board Project Options:

Initialize all peripherals with their default Mode ?

Figure 4: Initialize Peripherals

Press “Yes” when you are asked if you want to open the STM32CubeMx perspective.


Open Associated Perspective?

Device Configuration Tool editor is associated with Device Configuration Tool perspective. Do you want to open this perspective now?

☑ Remember my decision

Figure 5: Open STM32CubeMX perspective

3.3 Pin Configuration

The Pinout is flexible – however it is important that the pins communicating with the radar have the right user labels and that SPI is configured with “Full-Duplex Master”-mode.

In order to perform pin configuration you need to have the “.ioc”-file open that is named after your project.

The “LD2 [green Led]“ on the Nucleo board shares functionality with “SPI1_CLK“. Before we add SPI functionality we need to remove the “LD2 [green Led]“ configuration from PA5. The pin configuration is removed by doing a left mouse click on PA5 / “LD2 [green Led]“ and then selecting “Reset_State“.

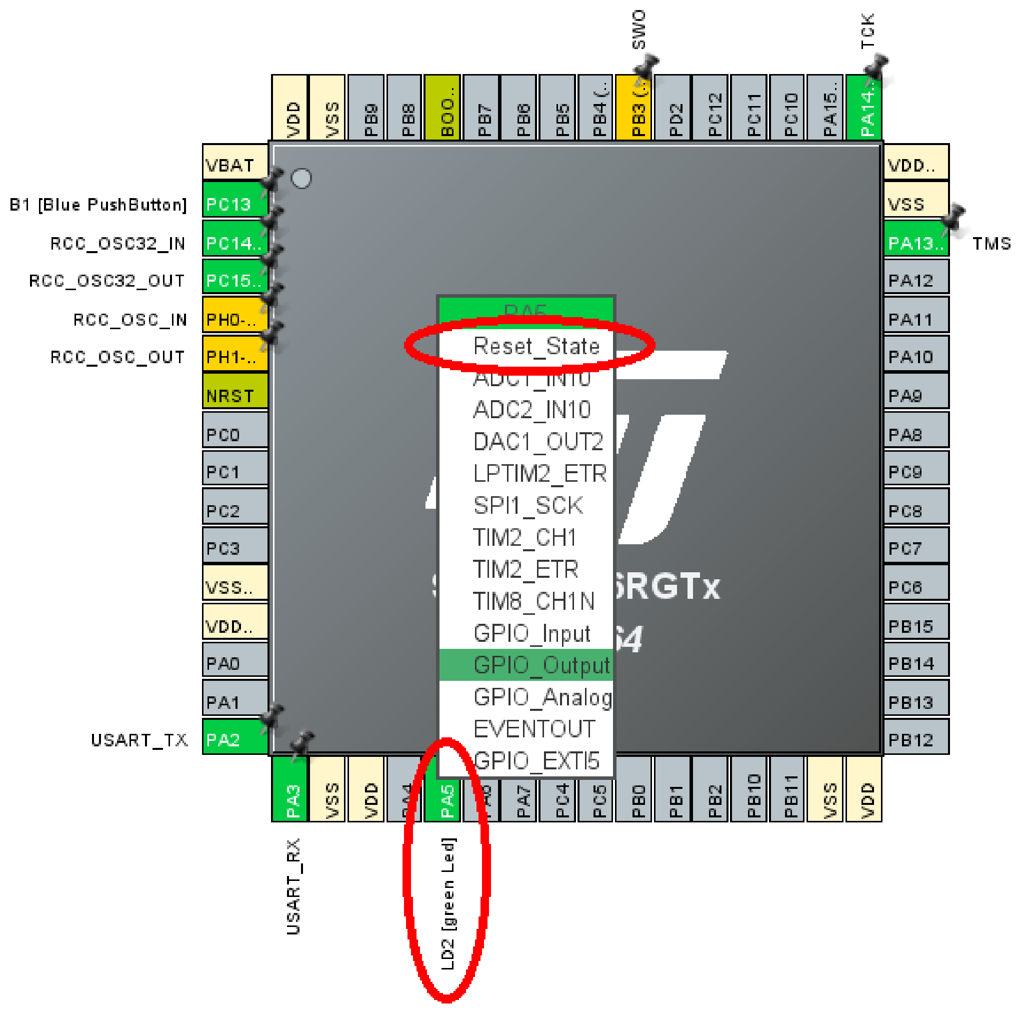


Figure 6: Remove LD2 pin

The “B1 [Blue PushButton]“ on the Nucleo board is not used on our examples and can be removed. The pin configuration is removed by doing a left mouse click on PC13 / “B1 [Blue PushButton]“ and then selecting “Reset_State“.

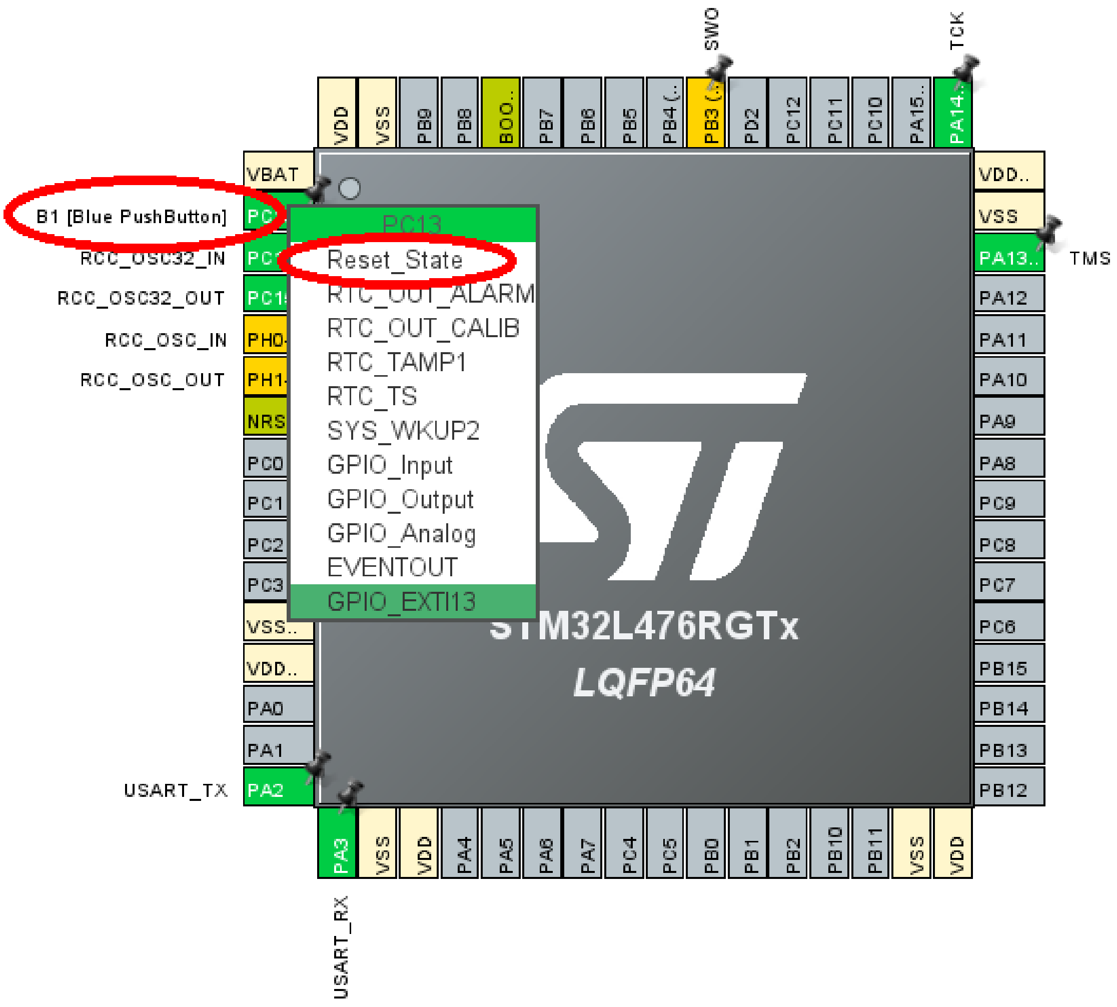
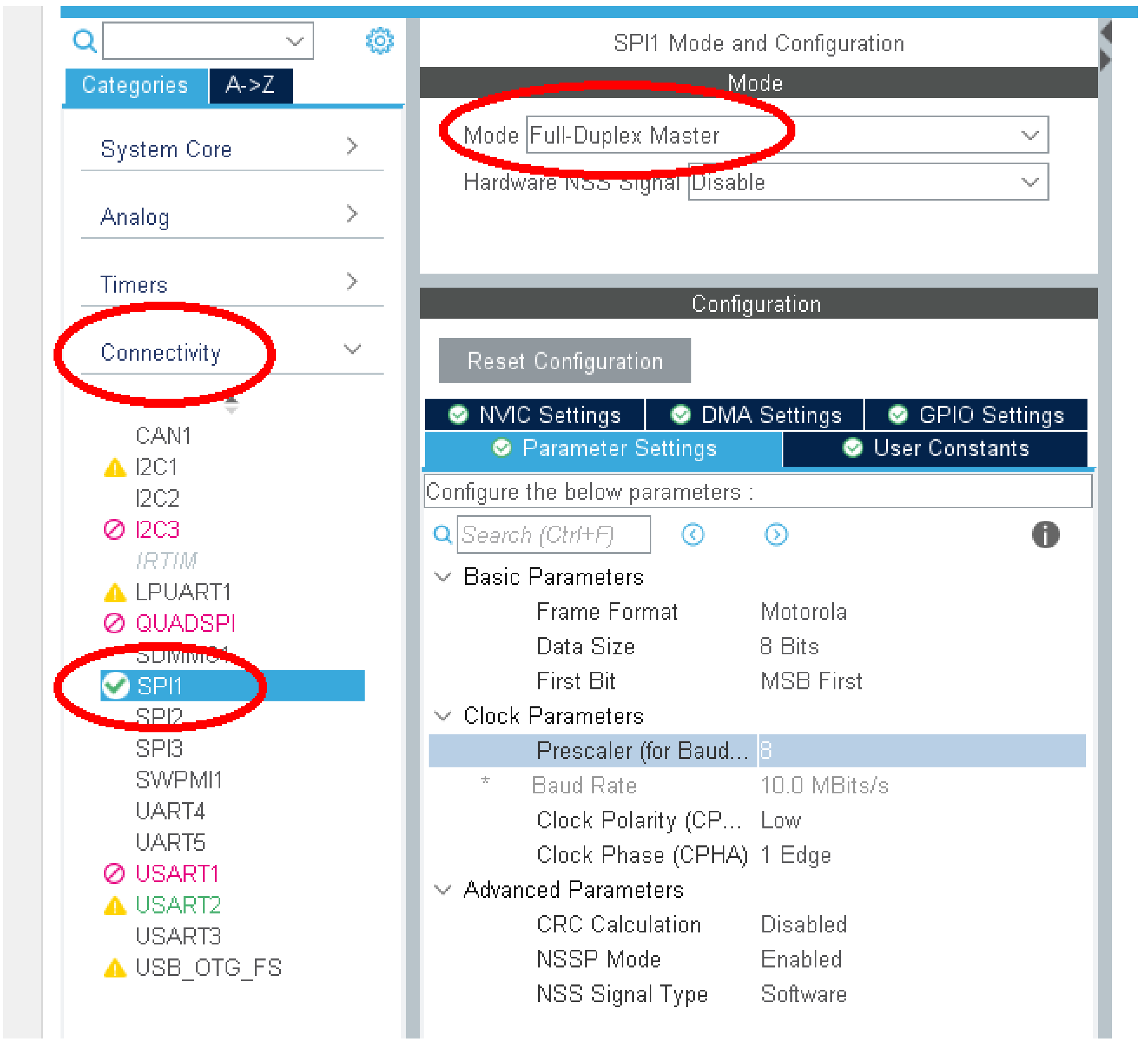


Figure 7: Remove B1 pin

In our example we activate SPI1 with “Full-Duplex Master”-mode by going into “Connectivity” in the “Pinout and Configuration”-tab. Then by pressing “SPI1” the option of selecting mode is available, “Full-Duplex Master” is selected. When doing this we get the SPI GPIOs:



SPI1 Mode and Configuration

Mode

Mode: Full-Duplex Master

Hardware NSS Signal: Disable

Configuration

Configure the below parameters :

Basic Parameters	
Frame Format	Motorola
Data Size	8 Bits
First Bit	MSB First
Clock Parameters	
Prescaler (for Baud...	8
* Baud Rate	10.0 MBits/s
Clock Polarity (CP...	Low
Clock Phase (CPHA)	1 Edge
Advanced Parameters	
CRC Calculation	Disabled
NSSP Mode	Enabled
NSS Signal Type	Software

Figure 8: SPI Setup

USER_LABEL	NUCLEO PIN
SPI1_MOSI	PA7
SPI1_MISO	PA6
SPI1_SCK	PA5

3.3.1 Pin Configuration with XE121

In order to set new GPIOs, you can left click the desired pin and selected the desired function of the pin.

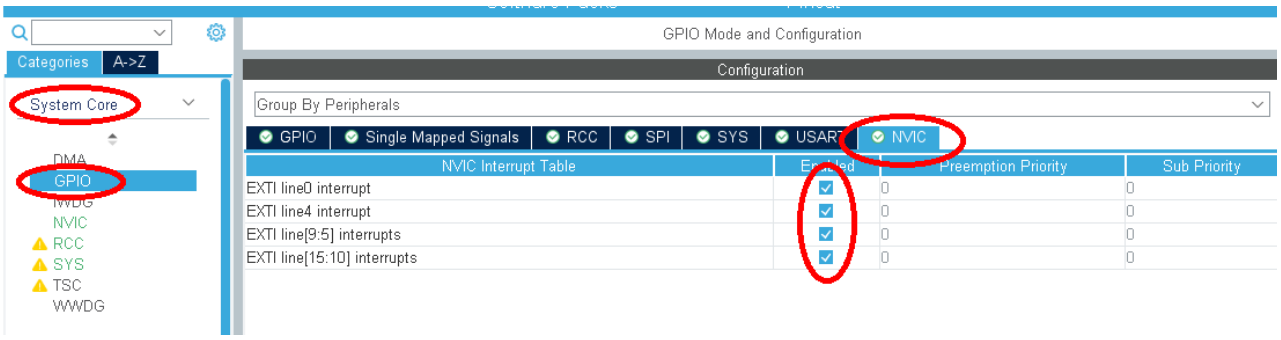
The table below shows how the XE121 and the NUCLEO-board can be connected:

USER_LABEL	NUCLEO PIN	GPIO TYPE	COMMENT
SEN_EN1	PB0	GPIO_Output	
SEN_EN2	PB5	GPIO_Output	Only needed for multi-sensor
SEN_EN3	PB3	GPIO_Output	Only needed for multi-sensor
SEN_EN4	PA10	GPIO_Output	Only needed for multi-sensor
SEN_EN5	PC1	GPIO_Output	Only needed for multi-sensor
SPI1_MOSI	PA7	SPI1_MOSI	
SPI1_MISO	PA6	SPI1_MISO	
SPI1_SCK	PA5	SPI1_SCK	
A121_SPI_SS	PB6	GPIO_Output	
SPI_SEL0	PA4	GPIO_Output	
SPI_SEL1	PA1	GPIO_Output	
SPI_SEL2	PA0	GPIO_Output	
SEN_INT1	PA9	GPIO_EXTI9	
SEN_INT2	PA8	GPIO_EXTI8	Only needed for multi-sensor
SEN_INT3	PB10	GPIO_EXTI10	Only needed for multi-sensor
SEN_INT4	PB4	GPIO_EXTI4	Only needed for multi-sensor
SEN_INT5	PC0	GPIO_EXTI0	Only needed for multi-sensor

3.4 Interrupt Configuration

The NVIC interrupt for the SEN_INT1, SEN_INT2, SEN_INT3, SEN_INT4, SEN_INT5 pins should be enabled.

1. Select the **GPIO** item under the **System Core**.
2. Select the **NVIC** tab.
3. Tick the checkbox **EXTI line0 interrupt**.
4. Tick the checkbox **EXTI line4 interrupt**.
5. Tick the checkbox **EXTI line[9:5] interrupt**.
6. Tick the checkbox **EXTI line[15:10] interrupt**.



NVIC Interrupt Table	Enabled	Preemption Priority	Sub Priority
EXTI line0 interrupt	✓	0	0
EXTI line4 interrupt	✓	0	0
EXTI line[9:5] interrupts	✓	0	0
EXTI line[15:10] interrupts	✓	0	0

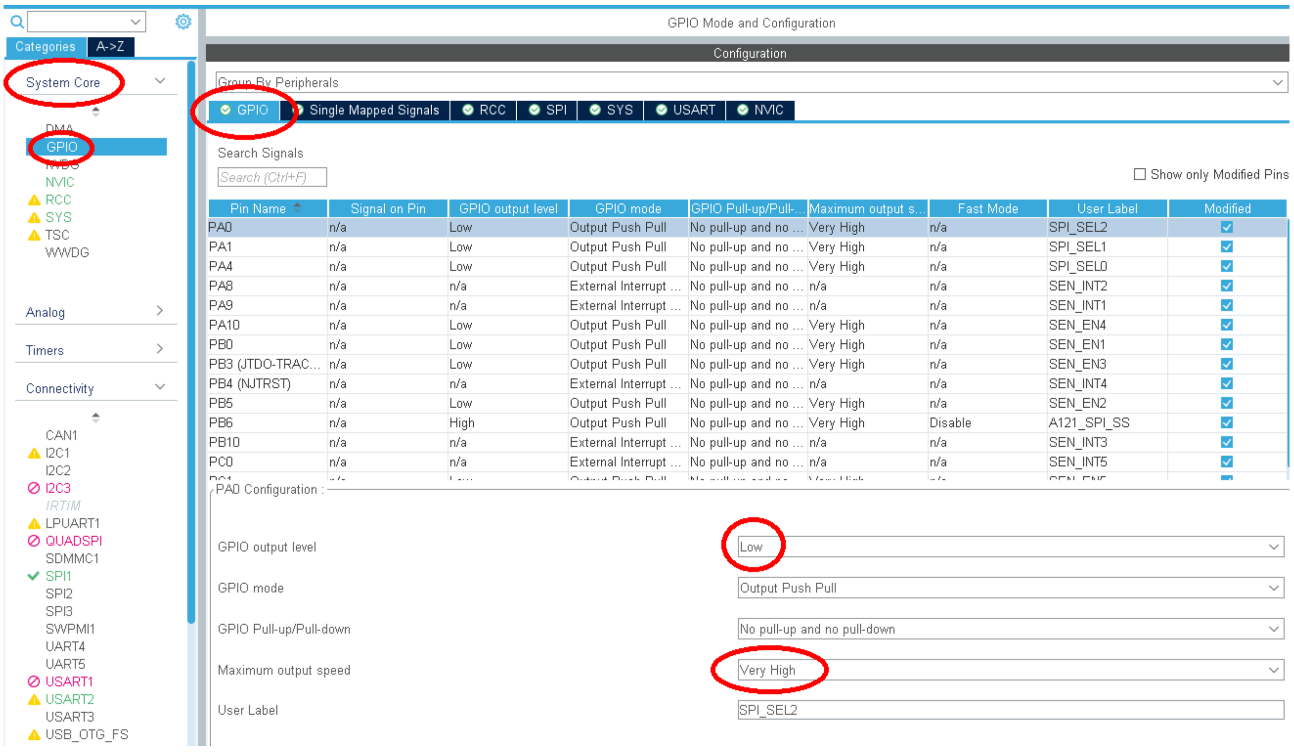
Figure 9: NVIC Configuration

3.5 GPIO Configuration

The GPIO signals from the processor to the sensor should be setup as outputs with very high drive strength.

1. Select the **GPIO** item under the **System Core**.
2. Select the **GPIO** tab.
3. Configure the GPIO pins accoring to the table below.

USER_LABEL	NUCLEO PIN	GPIO Output Level	Maximum Output speed
SEN_EN1	PB0	Low	Very High
SEN_EN2	PB5	Low	Very High
SEN_EN3	PB3	Low	Very High
SEN_EN4	PA10	Low	Very High
SEN_EN5	PC1	Low	Very High
A121_SPI_SS	PB6	High	Very High
SPI_SEL0	PA4	Low	Very High
SPI_SEL1	PA1	Low	Very High
SPI_SEL2	PA0	Low	Very High



Pin Name	Signal on Pin	GPIO output level	GPIO mode	GPIO Pull-up/Pull-...	Maximum output s...	Fast Mode	User Label	Modified
PA0	n/a	Low	Output Push Pull	No pull-up and no ...	Very High	n/a	SPI_SEL2	✓
PA1	n/a	Low	Output Push Pull	No pull-up and no ...	Very High	n/a	SPI_SEL1	✓
PA4	n/a	Low	Output Push Pull	No pull-up and no ...	Very High	n/a	SPI_SEL0	✓
PA8	n/a	n/a	External Interrupt ...	No pull-up and no ...	n/a	n/a	SEN_INT2	✓
PA9	n/a	n/a	External Interrupt ...	No pull-up and no ...	n/a	n/a	SEN_INT1	✓
PA10	n/a	Low	Output Push Pull	No pull-up and no ...	Very High	n/a	SEN_EN4	✓
PB0	n/a	Low	Output Push Pull	No pull-up and no ...	Very High	n/a	SEN_EN1	✓
PB3 (JTDO-TRAC...	n/a	Low	Output Push Pull	No pull-up and no ...	Very High	n/a	SEN_EN3	✓
PB4 (NJTRST)	n/a	n/a	External Interrupt ...	No pull-up and no ...	n/a	n/a	SEN_INT4	✓
PB5	n/a	Low	Output Push Pull	No pull-up and no ...	Very High	n/a	SEN_EN2	✓
PB6	n/a	High	Output Push Pull	No pull-up and no ...	Very High	Disable	A121_SPI_SS	✓
PB10	n/a	n/a	External Interrupt ...	No pull-up and no ...	n/a	n/a	SEN_INT3	✓
PC0	n/a	n/a	External Interrupt ...	No pull-up and no ...	n/a	n/a	SEN_INT5	✓

PA0 Configuration :

GPIO output level	Low
GPIO mode	Output Push Pull
GPIO Pull-up/Pull-down	No pull-up and no pull-down
Maximum output speed	Very High
User Label	SPI_SEL2

Figure 10: GPIO Configuration

3.6 XE121 Sensor Selection

The XE121 board has support for adding up to 4 extra A121 sensors by using FFC cables and XS121 boards. The selection of which sensor to communicate with is done in the acc_hal_integration_xe121_multi_sensor.c file.

SPI_SEL2	SPI_SEL1	SPI_SEL0	Sensor Selection
LOW	LOW	LOW	Sensor 1 (XE121)
LOW	LOW	HIGH	Sensor 2 (XS121 S2)
LOW	HIGH	LOW	Sensor 3 (XS121 S3)
LOW	HIGH	HIGH	Sensor 4 (XS121 S4)
HIGH	LOW	LOW	Sensor 5 (XS121 S5)

3.7 XE121 Single Sensor Setup

If the XE121 board is used in a single sensor setup there is no need to connect the following signals SEN_INT2, SEN_INT3, SEN_INT4, SEN_INT5, SEN_EN2, SEN_EN3, SEN_EN4 and SEN_EN5.

Please note that SEN_SEL0, SEN_SEL1 and SEN_SEL2 need to be kept low to select the sensor onboard the XE121.

The acc_hal_integration_xe121_multi_sensor.c should be replaced with the acc_hal_integration_xe121_single_sensor.c file in the STM32CubeIDE project.

3.8 SPI Configuration

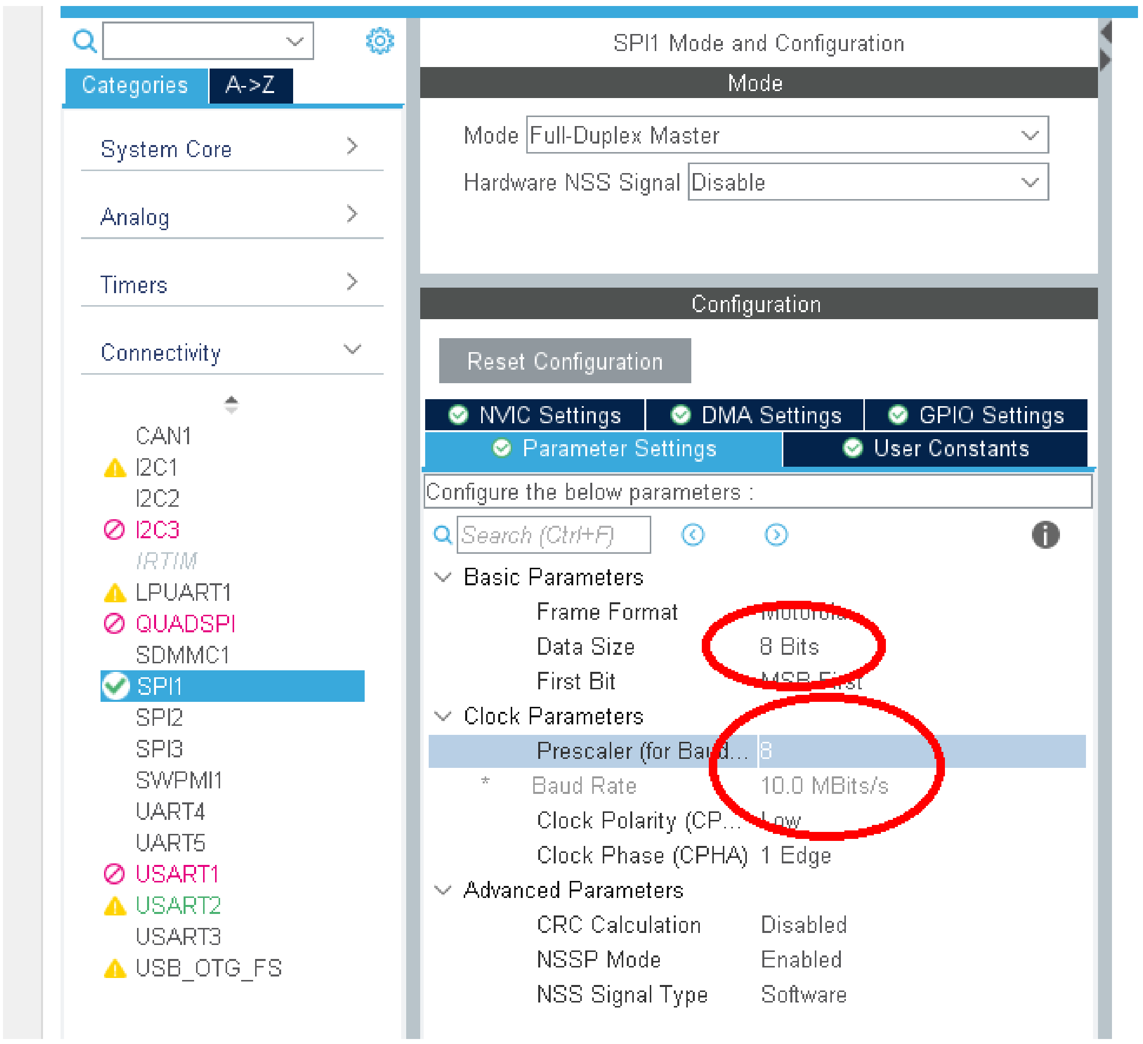
To make the SPI interface work properly with the Acconeer software you might need to set the Prescaler (for Baud Rate) and Data Size.

Press the SPI you are using and under the Configuration menu you can change parameters. Under “Basic Parameters” you can find that “Data Size” is set to 4 Bits by default, change this to 8 Bits.

Under “Clock Parameters” you will find “Prescaler”, this controls the frequency of the SPI bus. The higher the prescaler value is, the lower the frequency will be. Depending on your setup, you might have to use different prescaler values to fit your project. The A121 sensor supports SPI frequencies up to 50 MHz. In experimental configurations where the sensor and MCU are not mounted on the same PCB, the maximum SPI frequency is often significantly lower and typically around 10 MHz.

It is recommended to use the highest prescaler to begin with in order to make sure the SPI-communication is stable. Once the communication works properly you can start trying lower prescalers in order to increase the frequency.

You might have to save the project and restart the program in order to access these settings.



Mode: Full-Duplex Master
Hardware NSS Signal: Disable

Basic Parameters
- Frame Format: Motorola
- Data Size: 8 Bits
- First Bit: MSB First

Clock Parameters
- Prescaler (for Baud...: 8
- * Baud Rate: 10.0 MBits/s
- Clock Polarity (CP...: Low
- Clock Phase (CPHA): 1 Edge

Advanced Parameters
- CRC Calculation: Disabled
- NSSP Mode: Enabled
- NSS Signal Type: Software

Figure 11: SPI Master Config

3.9 Code Generation

Select Project/Generate Code to generate the driver and configuration MCU.

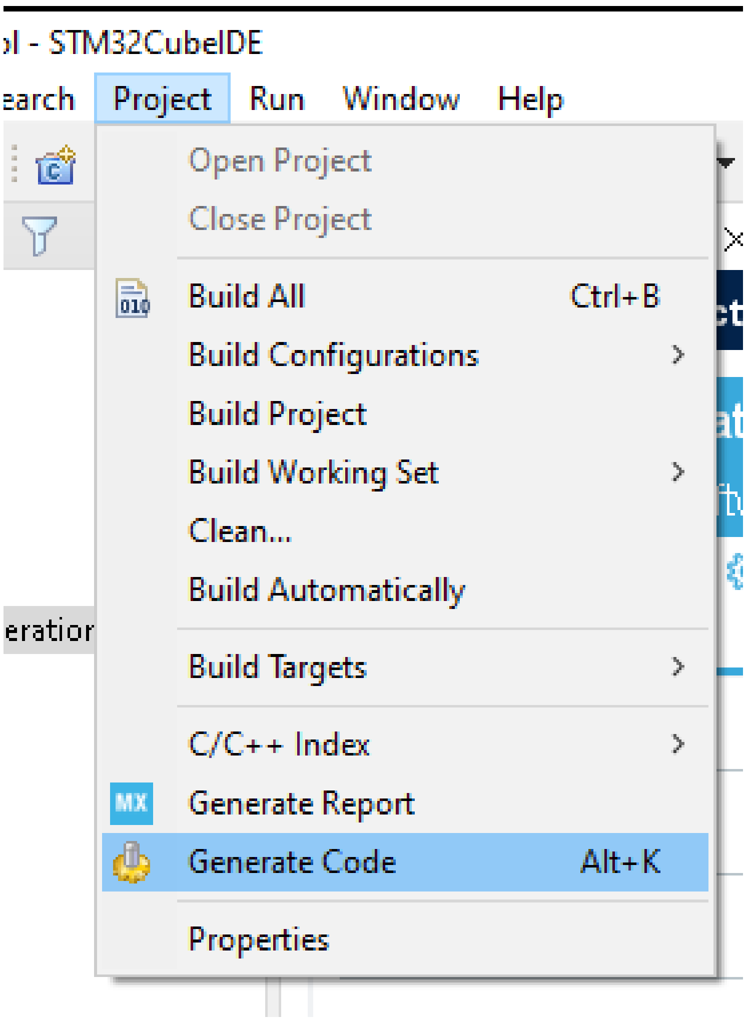


Figure 12: Generate code

4 Configuring Project for Acconeer Software

4.1 Adding Acconeer Software

There are different folders in the SDK zip:

- algorithms
- doc
- examples
- integration
- rss
- use_cases

In the “doc”-folder you can find reference documentation of the Acconeer software. The “examples”-folder contains examples of hwo to use the RSS API. The “use_cases”-folder contains example and reference apps for a specific use cases. The “rss”-folder contains two subfolders called “include” and “lib”. The “lib”-folder contains the Radar System Software (RSS) and the “include”-folder contains the header-files needed to use RSS. The “integration”-folder contains files which connect RSS with the drivers generated by STM32CubeIDE.

Start off by unpacking the zip-file into your project. Refresh the project by right clicking the project in the Project Explorer and click “Refresh”. Make sure you can see the folder in your project before continuing.

4.1.1 Source-files

Now the integration-file which you want to use needs to be selected from the “cortex_m4_gcc/integration”-folder,

Copy/move the “acc_hal_integration_stm32cube_xe121_multi_sensor.c”-file you have selected into “Core/Src”-folder. Also, copy content from “cortex_m4_gcc/algorithms” to “Core/Src”-folder. If you want any example or reference apps, copy them from “cortex_m4_gcc/examples” or from “cortex_m4_gcc/use_cases” to “Core/Src”-folder.

Also move the "acc_integration_stm32.c" and the "acc_integration_log.c" files from the “cortex_m4_gcc/integration”-folder to the “Core/Src”-folder.

For this guide we select “example_service.c” and move it to the “Core/Src”-folder.

4.1.2 Header-files

1. Select your project in the “Project Explorer”
2. Go into “Project → Properties → C/C++ General → Paths and Symbols → Includes”
3. Press “Add...” and then “Workspace...”
4. Select the “cortex_m4_gcc/rss/include”-folder in your project

Repeat this procedure for the “cortex_m4_gcc/integration”-folder and the “cortex_m4_gcc/examples”-folder.

4.1.3 Libraries

In order to set the path for the libraries, do the following:

1. Select your project in the “Project Explorer”
2. Go into “Project → Properties → C/C++ General → Paths and Symbols → Library Paths”
3. Press “Add...” and then “Workspace...”
4. Select the “cortex_m4_gcc/rss/lib”-folder in your project

Once the path is set, you can add the specific libraries by the following:

1. Go into “Project → Properties → C/C++ General → Paths and Symbols → Libraries”
2. Click “Add...”
3. Enter "acconeer_a121"
4. Click "OK"

If you want to add the “acc_detector_distance_a121” or “acc_detector_presence_a121” library, simply repeat the procedure above and exchange “acconeer_a121” for “acc_detector_distance_a121” or “acc_detector_presence_a121”. Make sure that the detector is being added before the “acconeer_a121”-library by moving “acconeer_a121” down using the “Move Down” button when “acconeer_a121” is selected. Do the same for "acc_rf_certification_test_a121" if this library is desired.

4.2 Project Settings

Set the project to gnu99-compiler by going into “Project → Properties → C/C++ Build → Settings → Tool Settings → MCU GCC Compiler → General”.

4.3 Adding Print Functionality with UART/USART

If an UART/USART has been added when setting up the project in the STM32CubeMX perspective and you want to use it for prints then you can simply add the following code to your project:

```
int _write(int file, char *ptr, int len)
{
    (void)file;
    HAL_UART_Transmit(&huart2, (uint8_t *)ptr, len, 0xFFFF);
    return len;
}
```

A terminal emulator, for example PuTTY, can be used to view the UART prints from the board.

Download PuTTY from https://putty.org and install it on your computer.

4.3.1 Find STM32 Board COM port

If using Windows, go to Device Manager to locate the COM port used by your STM32 Board.

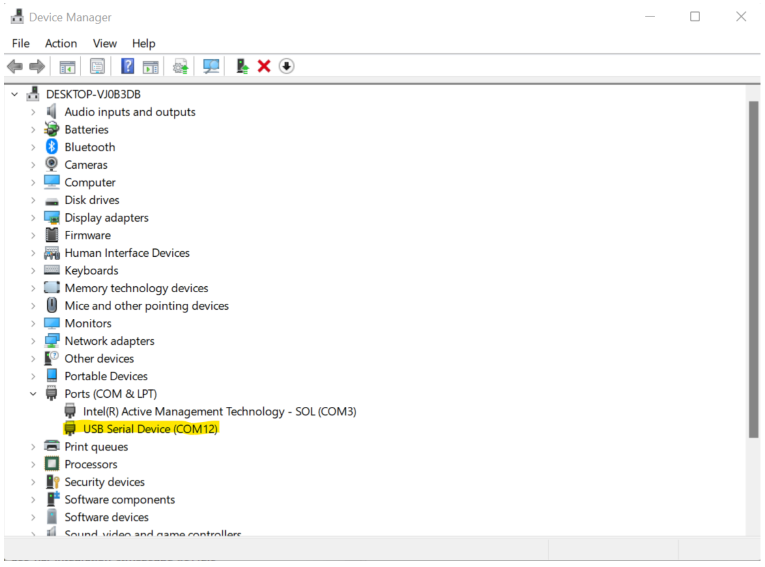
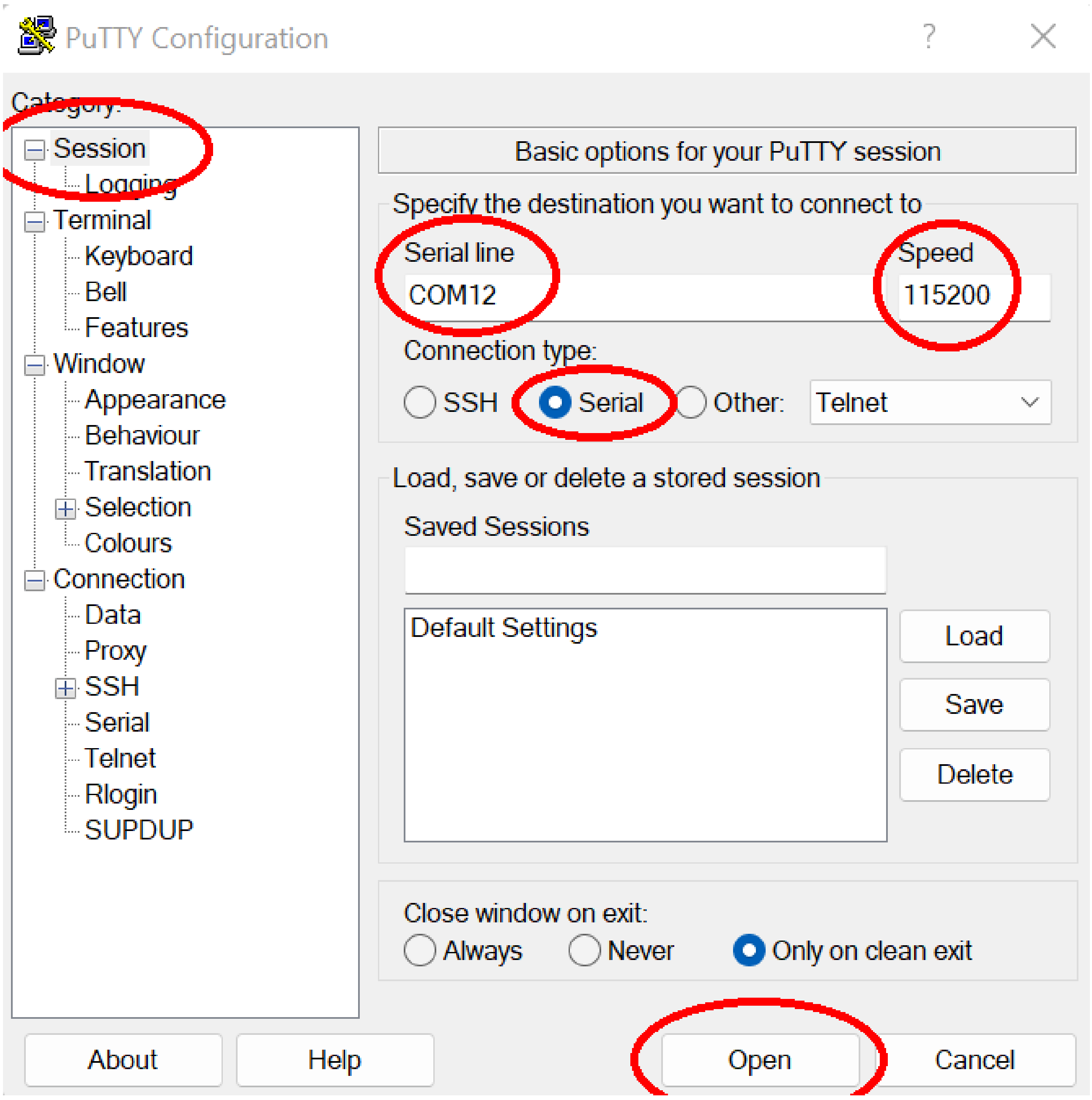


Figure 13: Windows Device Manager

4.3.2 Start and Configure PuTTY

1. Start the PuTTY application
2. Select **Session** in the **Category** window.
3. Set the **Connection Type** to **Serial**
4. Type the STM32 Board COM port in the **Serial line** text box.
5. Use the same **Speed** as used for the UART in the STM32CubeIDE project, default 115200.
6. Select **Terminal** in the **Category** window.

7. Tick the **Implicit CR in every LF** tickbox
8. Clock in **Open** to start the terminal



Specify the destination you want to connect to

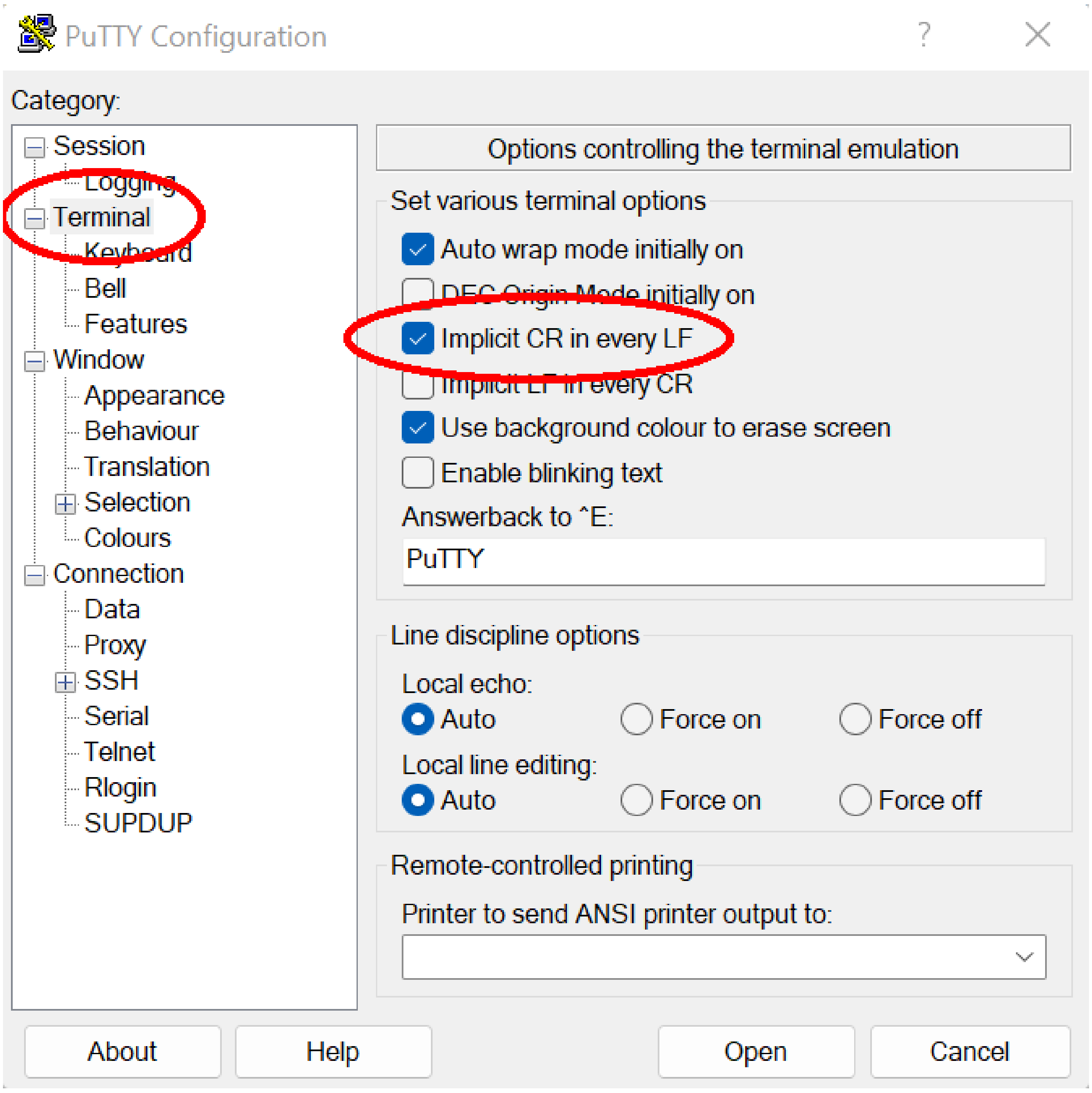
Serial line: COM12

Speed: 115200

Connection type: SSH [✓ Serial] Other: Telnet

Close window on exit: Always Never [✓ Only on clean exit]

Figure 14: PuTTY Session



Options controlling the terminal emulation

Set various terminal options

- [✓] Auto wrap mode initially on
- [ ] DEC Origin Mode initially on
- [✓] Implicit CR in every LF
- [ ] Implicit LF in every CR
- [✓] Use background colour to erase screen
- [ ] Enable blinking text

Answerback to ^E: PuTTY

Line discipline options

Local echo: [✓ Auto] Force on Force off

Local line editing: [✓ Auto] Force on Force off

Remote-controlled printing

Printer to send ANSI printer output to:

Figure 15: PuTTY Configuration

5 HAL Integration File

5.1 Selecting the Appropriate HAL-integration File

First off you need to pick what HAL integration file to use. The functions in the HAL integration file act as glue between the RSS radar stack and the drivers generated by the device configuration tool (STM32CubeMX perspective). Your hardware setup determines which HAL integration file to select as a starting point.

The files “acc_hal_integration_stm32cube_xe121_<single/multi>_sensor.c“ are HAL integrations prepared by Acconeer to handle the XE121 EVK board together with an STM32 processor.

The file “acc_hal_integration_stm32cube_xe121_single_sensor.c“ will only handle the onboard sensor on the XE121. The file “acc_hal_integration_stm32cube_xe121_multi_sensor.c“ will handle the onboard sensor on the XE121 aswell as the possiblity to use XS121 sensors connected in slot S2, S3, S4, S5..

5.2 A121_SPI_HANDLE

Define "A121_SPI_HANDLE" as hspi1 between the comments “USER CODE BEGIN Private defines” and “USER CODE END Private defines” in the file “Core/Inc/main.h”.

```
/* USER CODE BEGIN Private defines */
#define A121_SPI_HANDLE hspi1
/* USER CODE END Private defines */
```

6 Running a Radar Sensor Example

As a first radar example program to run, we selected the program "example_service.c".

To run the radar example program that we have chosen, simply include the header file “example_service.h” in the user code includes field in your "main.c"-file in the following manner:

```
/* USER CODE BEGIN Includes */
#include "example_service.h"
/* USER CODE END Includes */
```

After including the header-file, you can call the function from the "main.c"-file in the user code 2 field by the following call:

```
/* USER CODE BEGIN 2 */
acc_example_service(0, NULL);
/* USER CODE END 2 */
```

7 Troubleshooting and FAQ

7.1 Example Fails

The example program can fail for different reasons, here are a few common reasons.

7.1.1 Sensor Creation Returns NULL

The function acc_sensor_create returns NULL. This is most likely because the pins have either been connected wrong or some other pin fault. Usually due to an SPI communication problem. Could be due to incorrectly connected pins, drivers are incorrect, or the signals sent to the radar are in the wrong order. See section 7.3 Troubleshooting SPI Communication for more information.

7.1.2 Config Creation Hardfaults

The function acc_config_create hardfaults. Most likely due to memory problems. Depending on the memory of the MCU, heap and stack might overwrite each other. Or there is simply not enough memory.

Are you using FREERTOS? Make sure that the thread that is handling the Acconeer software has enough stack size to be able to run the software.

7.2 The Program is Stuck in HAL_Delay

If the program keeps entering HAL_Delay() or seems to be "stuck" there for longer periods of time, it might be because the interrupt pin is not connected or malfunctioning.

7.3 Troubleshooting SPI Communication

The following function can be used to find problems in the SPI communication with the radar sensor.

```
#include "acc_hal_integration_a121.h"

bool hal_test_spi_read_chipid(void)
{
  const uint32_t sensor = 1;

  const acc_hal_a121_t *hal = acc_hal_rss_integration_get_implementation();

  uint8_t buffer[6] = {0x30, 0x0, 0x0, 0x0, 0x0, 0x0 };

  acc_hal_integration_sensor_supply_on(sensor);
  acc_hal_integration_sensor_enable(sensor);

  hal->transfer(sensor, buffer, sizeof(buffer));

  acc_hal_integration_sensor_disable(sensor);
  acc_hal_integration_sensor_supply_off(sensor);

  if (buffer[4] == 0x12 && buffer[5] == 0x10)
  {
    printf("Test OK !\n");
    return true;
  }

  printf("Cannot read chip id !\n");
  return false;
}
```

When the program is executed, the signals to the A121 should look as in figure 16.

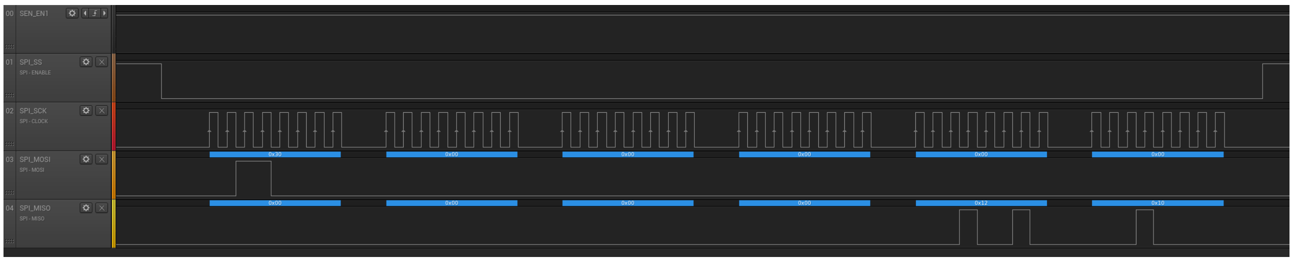


Figure 16: SPI transfer example

Note that the A121 enable signal must be set high at least 2 ms before the SPI transfer.

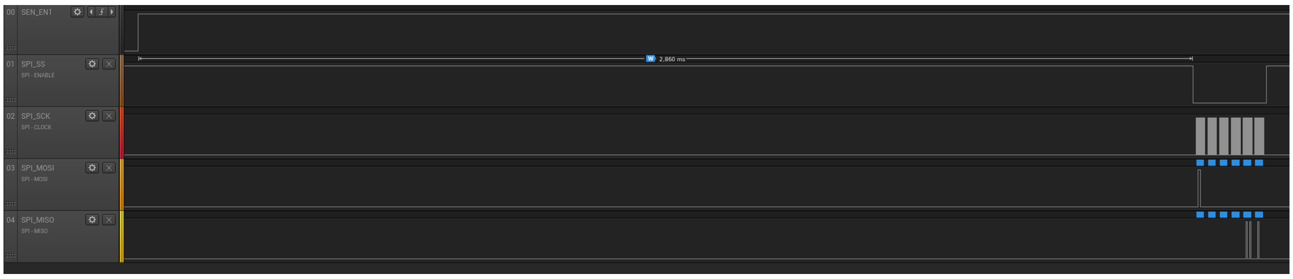


Figure 17: A121 Enable Signal

7.4 UART Problems

In order to verify the prints over UART we use picocom in Ubuntu:

```
$ picocom --imap lfcrlf --baud 115200 /dev/ttyACM0
```

We also had to make sure, in main.c, that the baudrate and word length is correct:

```
huart2.Init.BaudRate = 115200;
huart2.Init.WordLength = UART_WORDLENGTH_8B;
```

The linker might tell you that you have multiple definitions of the function "_write". If it happens, remove the implementation in “syscalls.c” and compile/link again.

7.5 Link Errors

Some users have experienced that STM32CubeIDE 1.0.0 forgets the link order of the libraries. Please check that the RSS libraries are listed in order stated in section 4.1.3 Libraries.

7.6 Heap Memory Corruption

When asking for more heap, the sbrk function will increase the heap towards the stack. However, it is imperative that the heap and stack never meet or overwrite each other. Unfortunately, the automatically generated code is using the stack pointer as border between them when it should use the maximum needed stack. This means that the heap might use some memory which is later overwritten when the stack is growing.

In the file "Src/sysmem.c" remove the line:

```
register char * stack_ptr asm("sp");
```

After removing the above line, change the function called "_sbrk" so that it looks like this:

```
caddr_t _sbrk(int incr)
{
  extern char end asm("end");
  extern char estack asm("_estack");
  extern char min_stack_size asm("_Min_Stack_Size");
  char *stack_limit = (char*)(&estack - &min_stack_size);
  static char *heap_end;
  char *prev_heap_end;

  if (heap_end == 0)
    heap_end = &end;

  prev_heap_end = heap_end;
  if (heap_end + incr > stack_limit)
  {
    errno = ENOMEM;
    return (caddr_t) -1;
  }

  heap_end += incr;

  return (caddr_t) prev_heap_end;
}
```

8 Disclaimer

The information herein is believed to be correct as of the date issued. Acconeer AB ("Acconeer") will not be responsible for damages of any nature resulting from the use or reliance upon the information contained herein. Acconeer makes no warranties, expressed or implied, of merchantability or fitness for a particular purpose or course of performance or usage of trade. Therefore, it is the user's responsibility to thoroughly test the product in their particular application to determine its performance, efficacy and safety. Users should obtain the latest relevant information before placing orders.

Unless Acconeer has explicitly designated an individual Acconeer product as meeting the requirement of a particular industry standard, Acconeer is not responsible for any failure to meet such industry standard requirements.

Unless explicitly stated herein this document Acconeer has not performed any regulatory conformity test. It is the user's responsibility to assure that necessary regulatory conditions are met and approvals have been obtained when using the product. Regardless of whether the product has passed any conformity test, this document does not constitute any regulatory approval of the user's product or application using Acconeer's product.

Nothing contained herein is to be considered as permission or a recommendation to infringe any patent or any other intellectual property right. No license, express or implied, to any intellectual property right is granted by Acconeer herein.

Acconeer reserves the right to at any time correct, change, amend, enhance, modify, and improve this document and/or Acconeer products without notice.

This document supersedes and replaces all information supplied prior to the publication hereof.