Integration using STM32CubeIDE

User Guide

Integration using STM32CubeIDE

User Guide

Author: Acconeer AB

Version:a111-v2.15.2

Acconeer AB August 18, 2023

© 2023 by Acconeer AB - All rights reserved

Contents

© 2023 by Acconeer AB - All rights reserved

1 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 XC112/XR112, including the A111 radar sensor. There is some extra logic on the XC112 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 connect a Sparkfun A111 Pulsed Radar Breakout board to the Nucleo-board.

STM32CubeIDE can be downloaded from their 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 1.9.0

© 2023 by Acconeer AB - All rights reserved

2 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.

Let's get started. Start STM32CubeIDE and click "Start new STM32 project". The option is also available under "File → New → STM32Project".

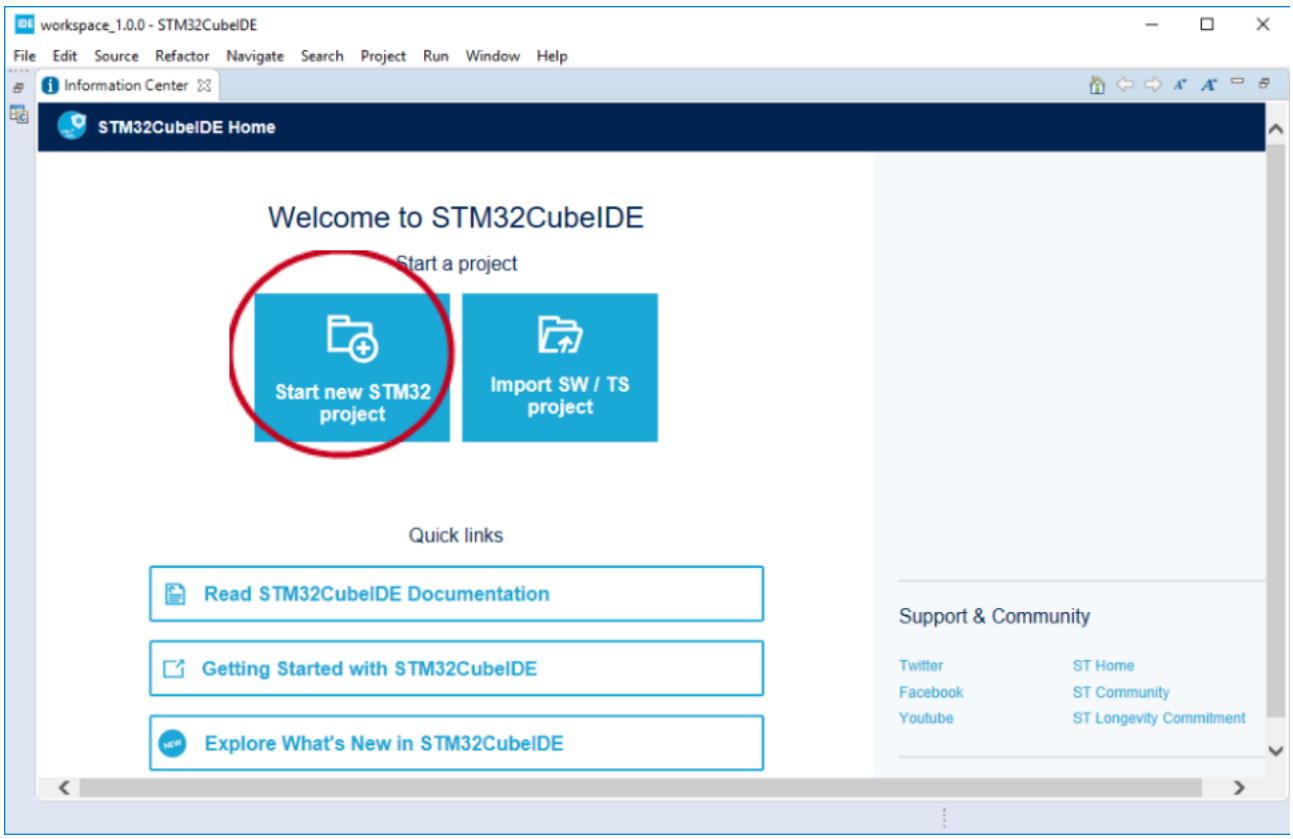

Figure 1: Start new STM32CubeIDE project

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

2.1 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 in the bottom of the page.

© 2023 by Acconeer AB - All rights reserved

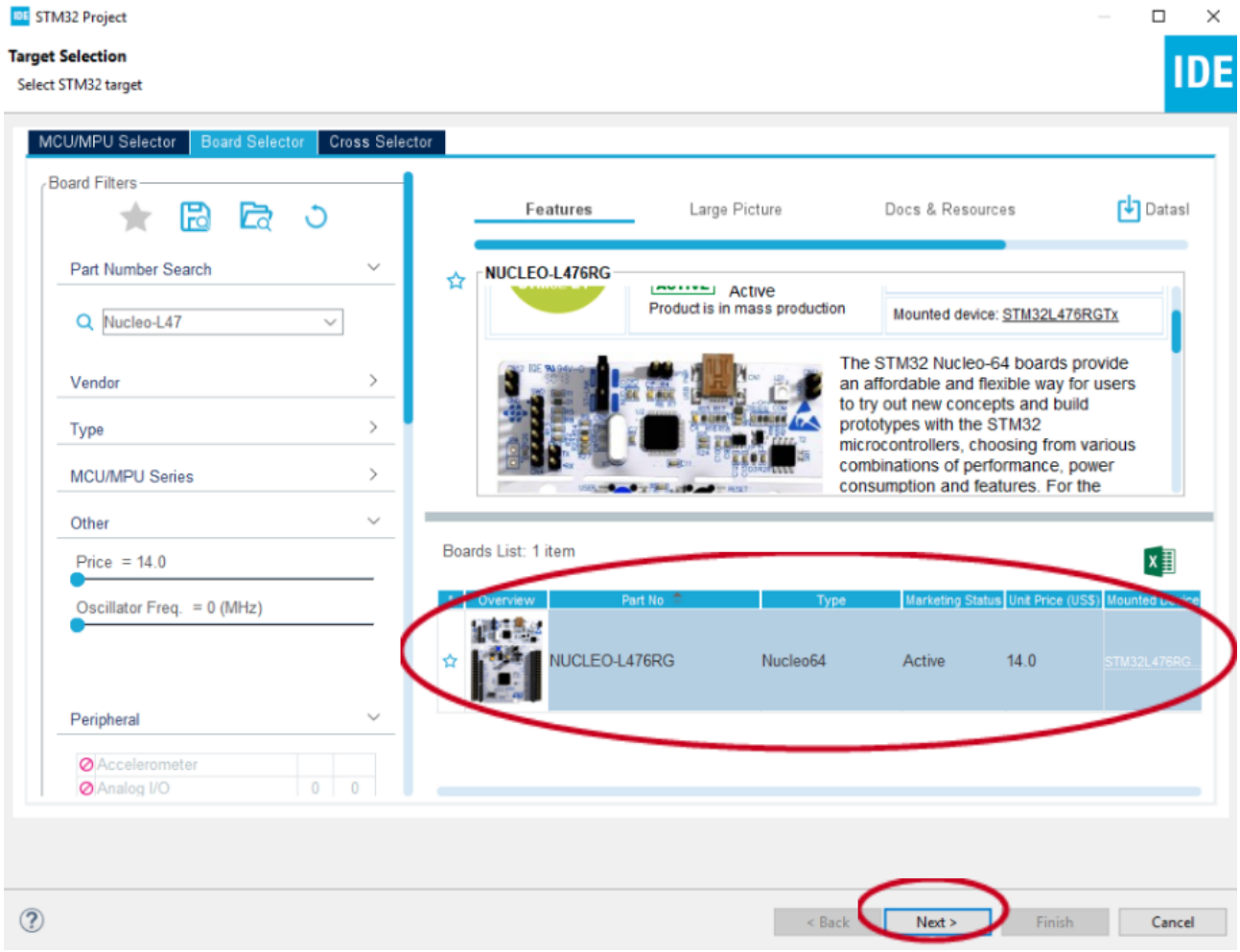

Figure 2: Target Selection

2.2 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 in the bottom of the page.

© 2023 by Acconeer AB - All rights reserved

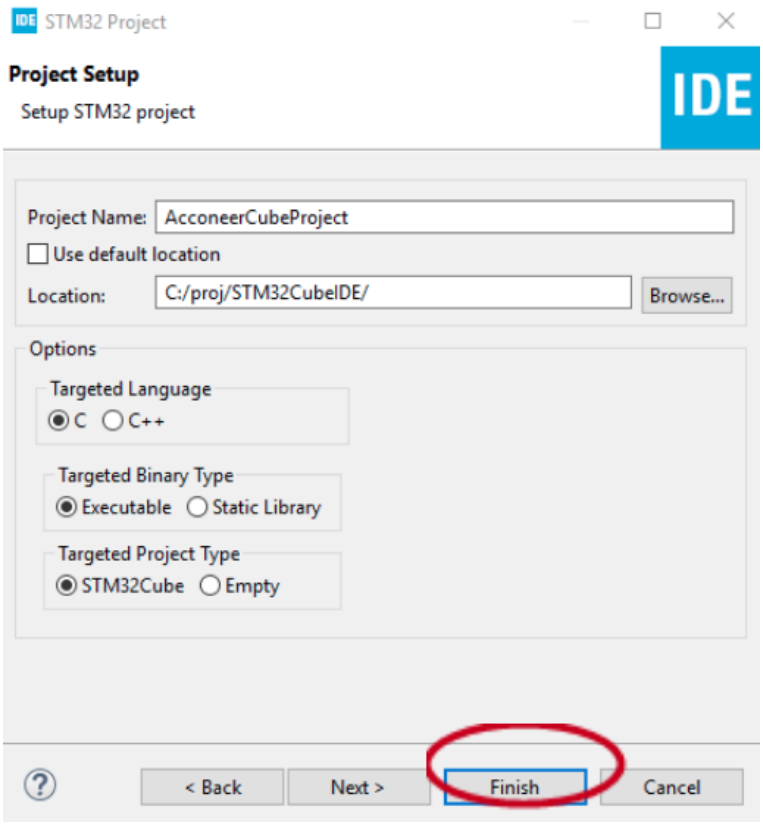

Figure 3: Project Setup

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

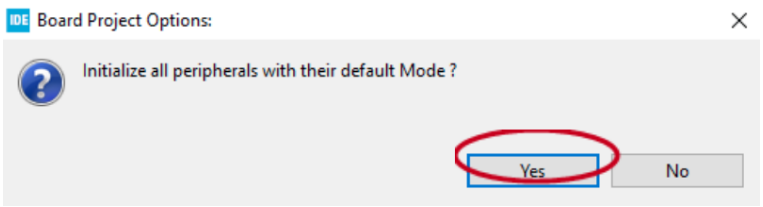

Figure 4: Initialize Peripherals

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

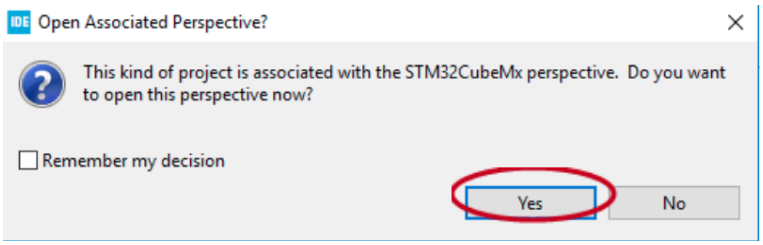

Figure 5: Open STM32CubeMX perspective

2.3 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 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.

© 2023 by Acconeer AB - All rights reserved

In our example we activate SPI3 with “Full-Duplex Master”-mode by going into “Connectivity” in the “Pinout and Configuration”-tab. Then by pressing “SPI3” the option of selecting mode is available, “Full-Duplex Master” is selected. When doing this we get the SPI GPIOs:

USER_LABEL	NUCLEO PIN
SPI3_MOSI	PC12
SPI3_MISO	PC11
SPI3_SCK	PC10

The User Labels for the SPI pins should be renamed to fit the Acconeer software, this is done by right clicking the pins and selecting “Enter User Label”. Rename the following pins:

OLD USER_LABEL	NEW USER_LABEL
SPI3_MOSI	A111_SPI_MOSI
SPI3_MISO	A111_SPI_MISO
SPI3_SCK	A111_SPI_SCK

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

2.3.1 Pin Configuration with XC112

The table below shows how the XC112 and the NUCLEO-board can be connected:

USER_LABEL	NUCLEO PIN	GPIO TYPE	XC112 PIN
A111_SENSOR_INTERRUPT	PA8	GPIO_EXTI8	38
A111_ENABLE	PA9	Output	16
A111_SPI_SS	PB6	Output	24
A111_SPI_MOSI	PC12	SPI3_MOSI	19
A111_SPI_MISO	PC11	SPI3_MISO	21
A111_SPI_SCK	PC10	SPI3_SCK	23
XC112_SPI_S1_ENABLE_N	PB4	Output	12
XC112_SPI_S2_ENABLE_N	PB10	Output	13
XC112_SPI_S3_ENABLE_N	PB2	Output	15
XC112_SPI_S4_ENABLE_N	PB1	Output	26
XC112_ENABLE_N	PB5	Output	31
XC112_PMU_ENABLE	PA10	Output	11
	+5V	5V	2
	+3V3	3V3	1
	GND	GND	20
	GND	GND	25

In the table above there are pins without User Labels, these are pins that do not need configuration in the STM32CubeMX tool, they just have to be connected.

2.3.2 Pin Configuration with Sparkfun

For the Sparkfun board the following pins can be used as an example, note that it is only important that the User Labels are named accordingly.

USER_LABEL	NUCLEO PIN	GPIO TYPE	SPARKFUN PIN
A111_SENSOR_INTERRUPT	PA8	GPIO_EXTI8	INT
A111_ENABLE	PB5	Output	EN
A111_CS_N	PB6	Output	$\overline{\text{CS}}$
A111_SPI_MOSI	PC12	SPI3_MOSI	MOSI
A111_SPI_MISO	PC11	SPI3_MISO	MISO
A111_SPI_SCK	PC10	SPI3_SCK	SCLK
	+3V3	5V	5V
	GND	GND	GND

In the table above there are pins without User Labels, these are pins that do not need configuration in the STM32CubeMX tool, they just have to be connected.

© 2023 by Acconeer AB - All rights reserved

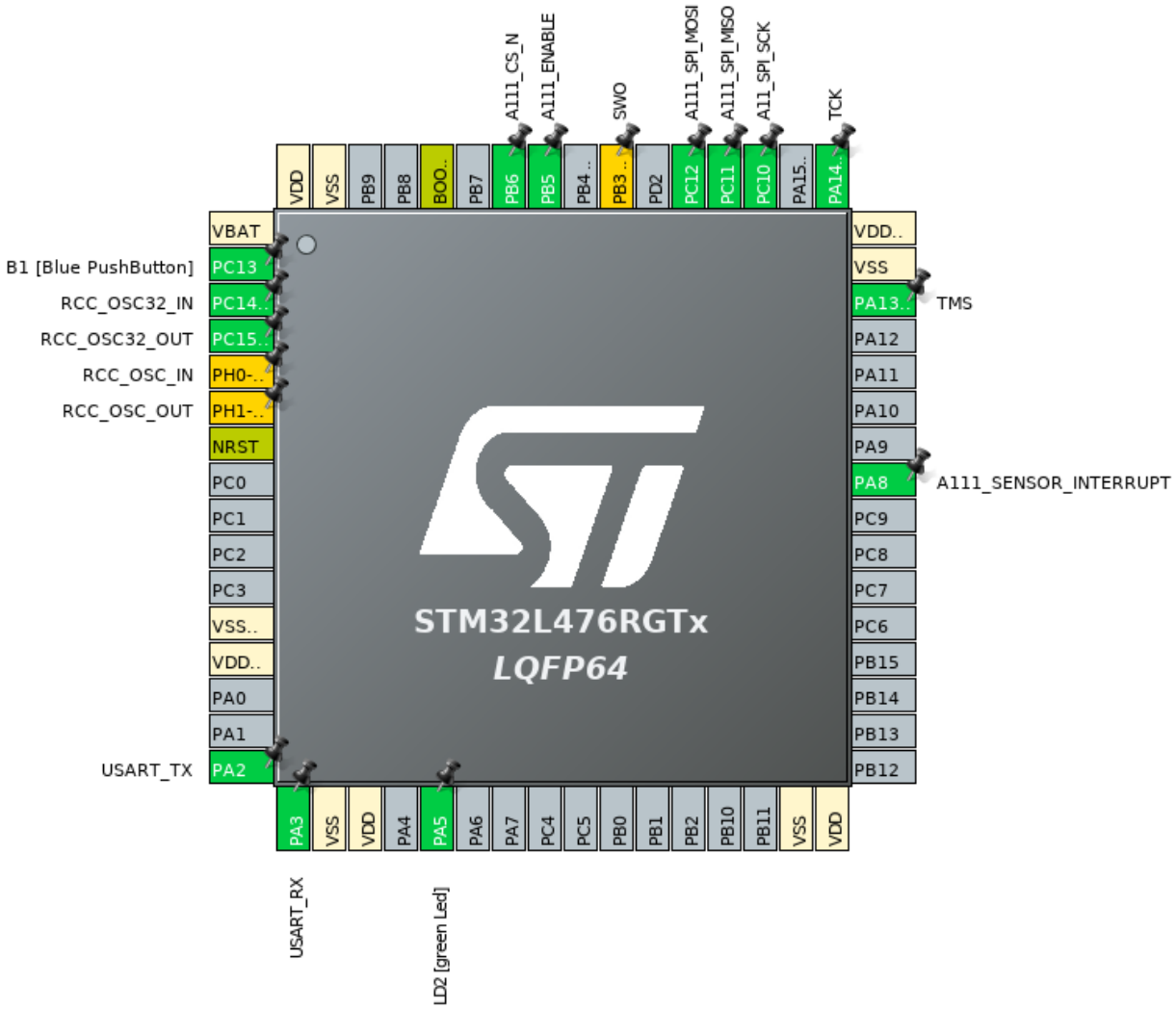

Figure 6: Example pin configuration for Sparkfun

2.3.3 Pin Configuration with Custom PCB

If you have made custom PCB with a STM32 micro controller and an A111 radar sensor we recommend that you use the same user label names as in the Sparkfun configuration above.

2.4 Interrupt Configuration

The NVIC interrupt for the A111_SENSOR_INTERRUPT pin should be enabled, in this case PA8 is configured as GPIO_EXTI8.

© 2023 by Acconeer AB - All rights reserved

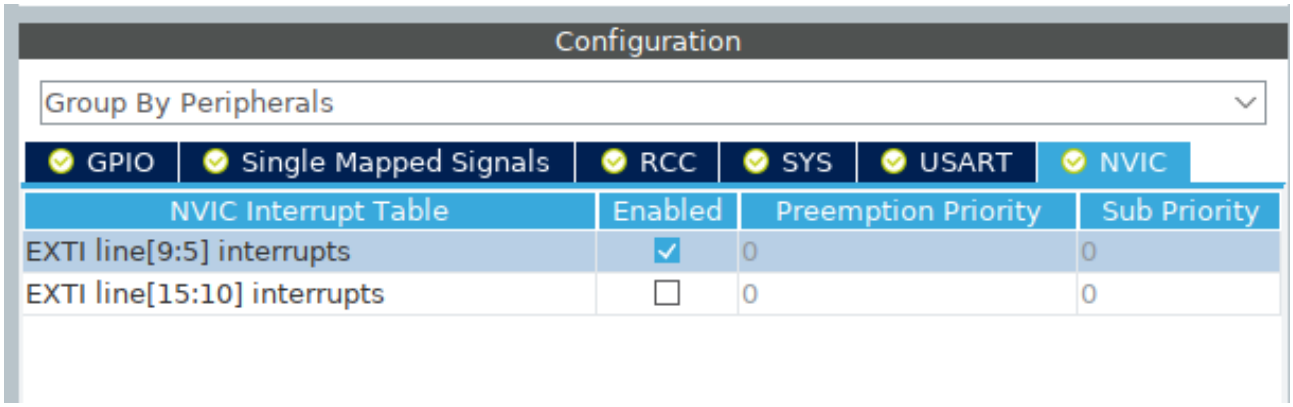

Figure 7: NVIC Configuration

2.5 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 A111 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.

© 2023 by Acconeer AB - All rights reserved

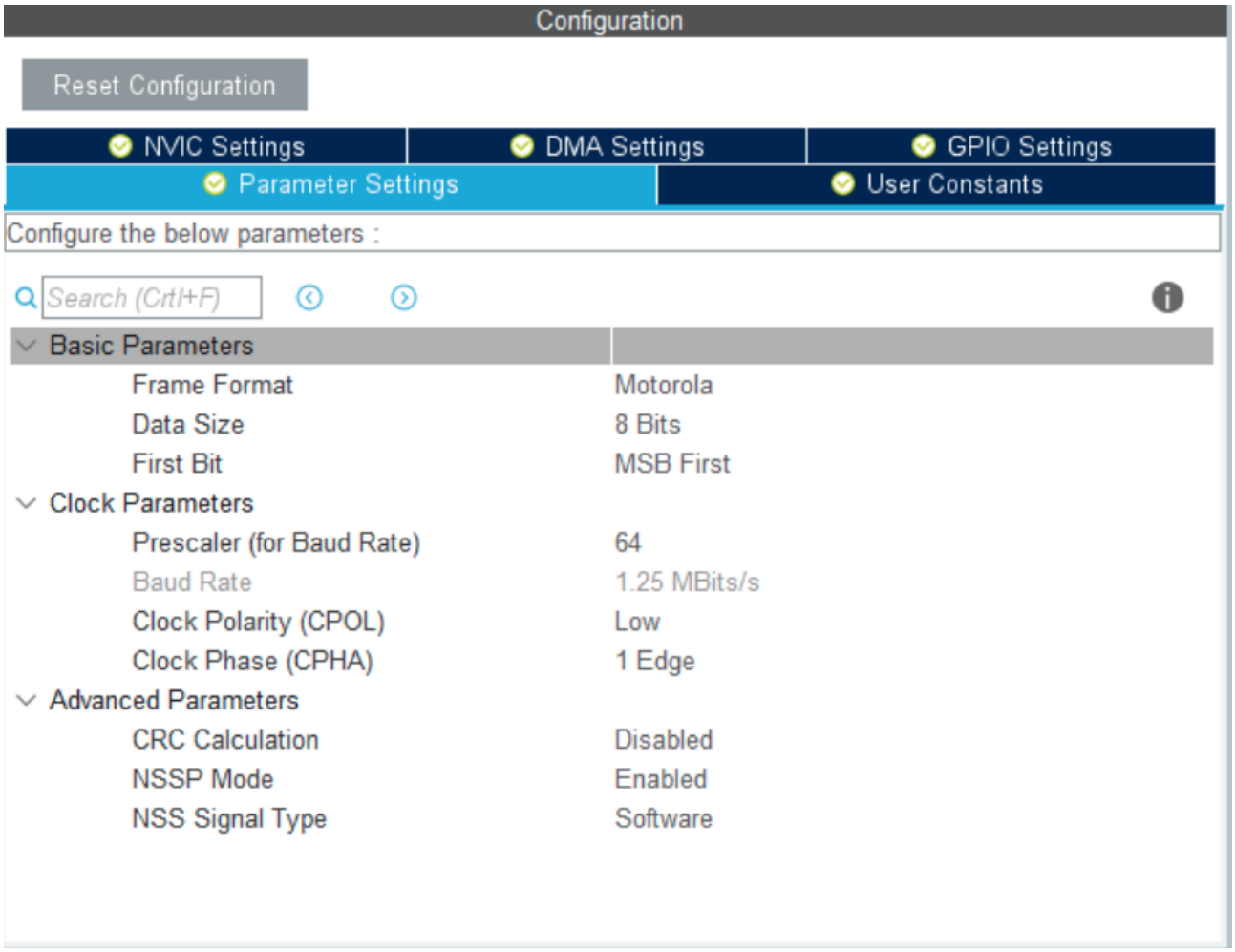

Figure 8: Example SPI configuration

2.6 Code Generation

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

© 2023 by Acconeer AB - All rights reserved

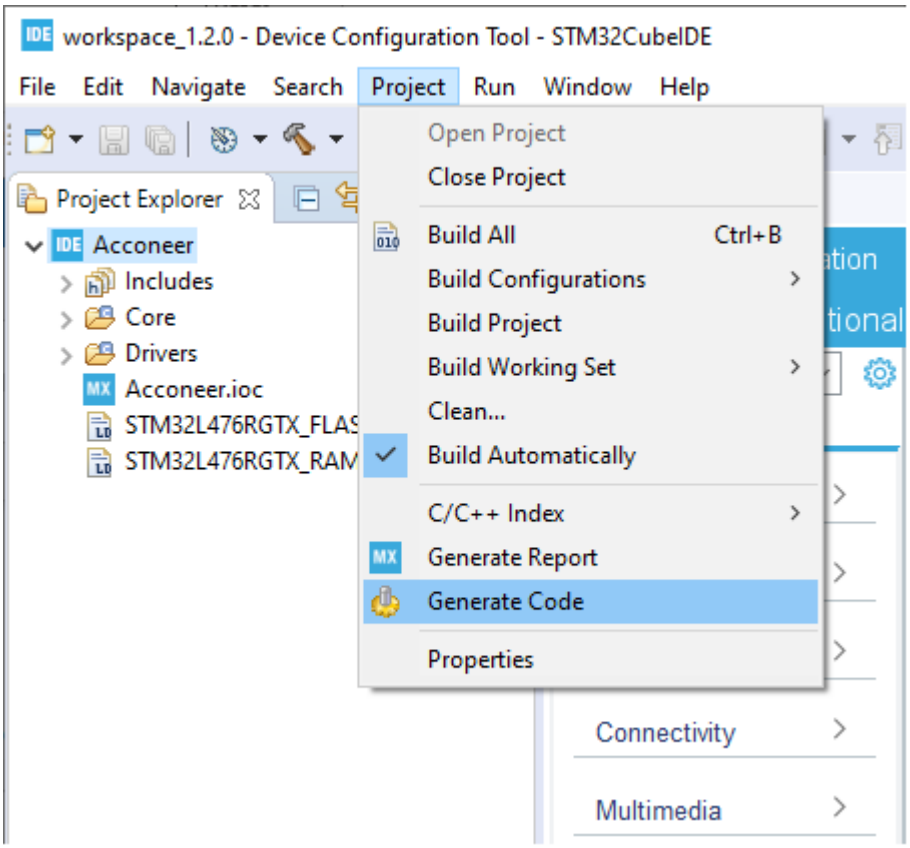

Figure 9: Generate code

© 2023 by Acconeer AB - All rights reserved

3 Configuring Project for Acconeer Software

3.1 Adding Acconeer Software

There are four different folders in the SDK zip:

- doc
- examples
- integration
- rss

In the “doc”-folder you can find reference documentation of the Acconeer software. The “examples”-folder example programs and their header-files. 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 connects 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.

3.1.1 Source-files

Now the integration-file which you want to use needs to be selected from the “cortex_m4/integration”-folder, since you can only use one. If you are using Sparkfun, select the “acc_hal_integration_stm32cube_sparkfun_a111”-file.

When a selection has been made, copy/move the integration-file you have selected into “Core/Src”-folder. If you want any examples, copy them from “cortex_m4/examples” to “Core/Src”-folder.

Also move the "acc_integration_stm32.c" and "acc_integration_log.c" file from the “cortex_m4/integration”-folder to the “Core/Src”-folder.

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

3.1.2 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/rss/include”-folder in your project

Repeat this procedure for the “cortex_m4/integration”-folder and the “cortex_m4/examples”-folder.

3.1.3 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/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"
4. Click "OK"

If you want to add the “acc_detector_distance” library, simply repeat the procedure above and exchange “acconeer” for “acc_detector_distance”. If you want to add Make sure that the detector is being added before the “acconeer”-library by moving “acconeer” down using the “Move Down” button when “acconeer” is selected. Do the same for "acc_detector_presence" and "acc_rf_certification_test_a111" if these libraries are desired.

© 2023 by Acconeer AB - All rights reserved

3.2 Project Settings

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

3.3 Adding Print Functionality through SWV

Add the following code in the “main.c”-file in the Src-folder. Note that “main.h” needs to be included in order to use the function “ITM_SendChar”.

```
int _write(int file, char *ptr, int len)
{
  int i = 0;

  for(i = 0; i < len; i++)
  {
    ITM_SendChar((*ptr++));
  }
  return len;
}
```

Make sure this code is written within the field “Private user code” for example, between the “USER CODE BEGIN 0” and “USER CODE END 0” comments.

Click Debug icon. You might have to set what type of project it is, in our case we had to select “STM32 MCU application”. After this, you should be able to enter “Debug Configurations → Debugger” and under “Serial Wire Viewer (SWV)” press the box to enable SWV.

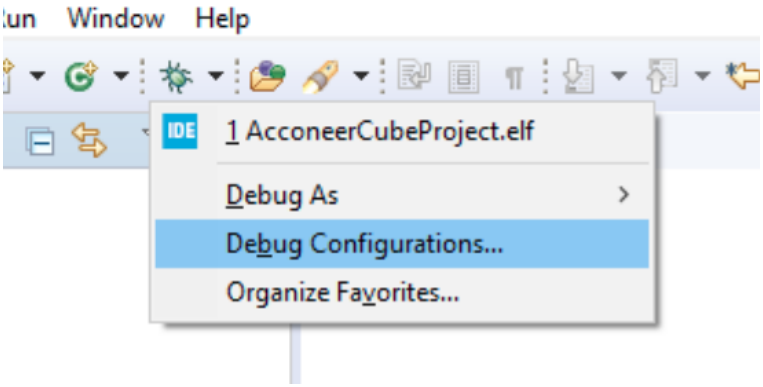

Figure 10: Debug button

© 2023 by Acconeer AB - All rights reserved

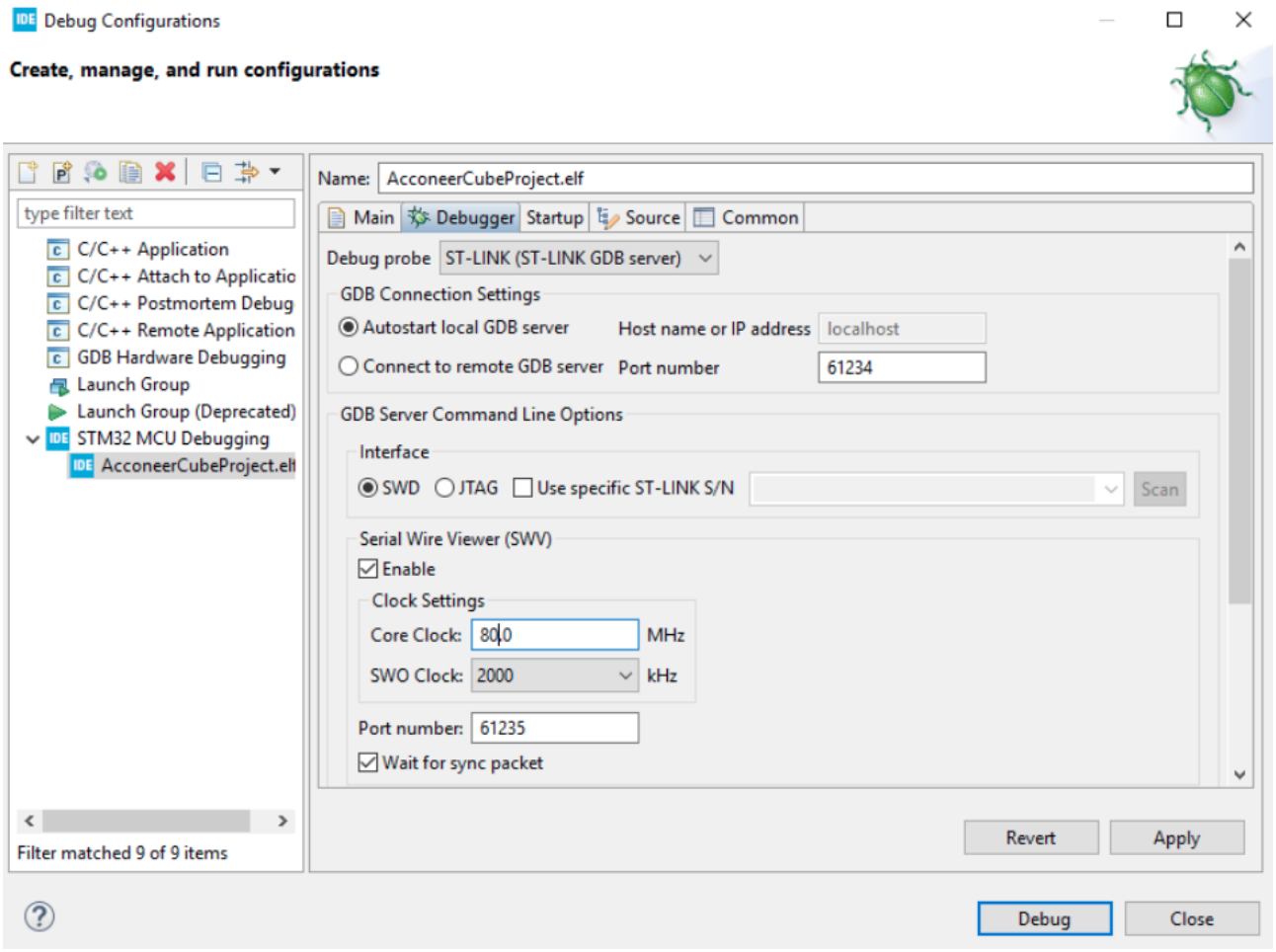

Figure 11: Debug configuration

Make sure to also match the Core clock with the HCLK, which can be found in the device configuration tool (STM32CubeMX perspective) in the “Clock Configuration”-tab. In our case it is set to 80.0 MHz.

Start the debug. When the debug is paused, open the SWV Console view by going into “Window → Show View → SWV → SWV ITM Data Console”. In this console, press "Configure Trace" (marked in the picture with red circle) and make sure that the ITM Stimulus Port 0 is ticked. Then press the "Start trace"-button.

© 2023 by Acconeer AB - All rights reserved

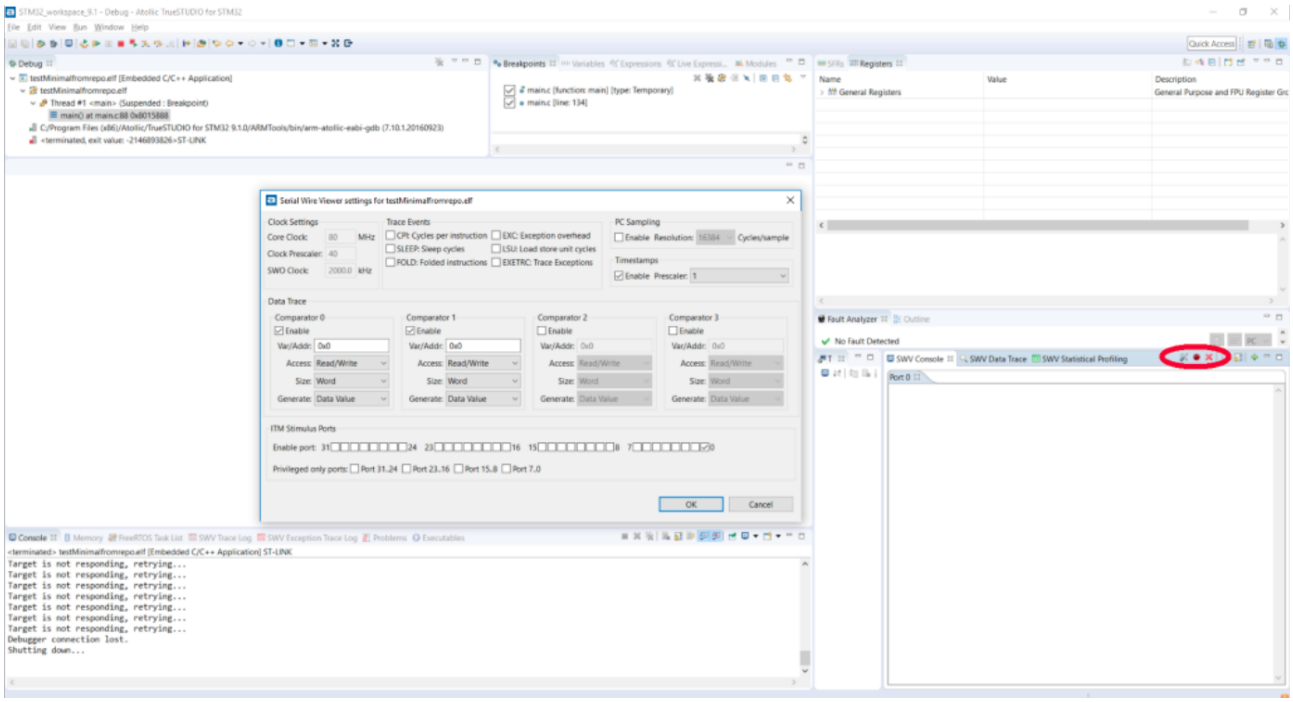

Figure 12: Configure Trace

3.4 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, ptr, len, 0xFFFF);
  return len;
}
```

The UART/USART-handle is generally called something like “huart1”, “huart2” or “huart3” from the auto generated drivers.

© 2023 by Acconeer AB - All rights reserved

4 HAL Integration File

4.1 Selecting the Appropriate HAL-integration File

First of you need to pick what HAL integration file to use. The functions in the HAL integration file acts 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.

4.1.1 Configurations with XC112

Select the file “acc_hal_integration_stm32cube_xc112.c“.

4.1.2 Configurations with the Sparkfun board

Select the file “acc_hal_integration_stm32cube_sparkfun_a111.c“.

4.1.3 Configurations with custom PCB

Select the file “acc_hal_integration_stm32cube_sparkfun_a111.c“ as a staring point.

Some of the functions in the file, such as acc_hal_integration_sensor_power_on and acc_hal_integration_sensor_transfer, may have to updated to match how the components on your PCB has to be configured in order to power on the sensor or initiate an SPI transfer.

4.2 A111_SPI_HANDLE

Define "A111_SPI_HANDLE" as hspi3 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 A111_SPI_HANDLE hspi3
/* USER CODE END Private defines */
```

If you are using another SPI handle than hspi3, simply switch out hspi3, for example hspi2.

4.3 Sensor Crystal Frequency

The ACC_BOARD_REF_FREQ define assumes 24MHz crystal on the XC112. Meanwhile the Sparkfun integration file assumes 26MHz. This value needs to be changed if a crystal with a different frequency is used with the A111 radar sensor.

© 2023 by Acconeer AB - All rights reserved

5 Running a Radar Sensor Example

As a first radar example program to run, we selected the program "example_assembly_test.c". This program is a good program to run the first time (especially if you have your own PCB), since it tests and analyzes both hardware and software integration.

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

```
/* USER CODE BEGIN Includes */
#include "example_assembly_test.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_assembly_test(0, NULL);
/* USER CODE END 2 */
```

If you want to run acc_ref_app_rf_certification_test you need to declare the following function in the main.c file:

```
/* USER CODE BEGIN PFP */
extern int acc_ref_app_rf_certification_test(int argc, char *argv[]);
/* USER CODE END PFP */
```

and call the following function:

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

© 2023 by Acconeer AB - All rights reserved

6 Troubleshooting and FAQ

6.1 Creation of Service Fails

The “acc_service_create” function can fail for different reasons, here are a few common reasons.

6.1.1 Service Creation Returns NULL

The function acc_service_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 pins are incorrectly connected, drivers are incorrect, or the signals sent to the radar are in the wrong order. See section 6.3 Troubleshooting SPI Communication for more information.

6.1.2 Creating the Service Hardfaults

The function acc_service_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.

6.2 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.

6.3 Troubleshooting SPI Communication

The following function can be used to find problems in the SPI communication with the radar sensor. It is not depending on any Acconeer libraries but needs a working HAL integration, i.e. there must be an implementation of the acc_hal_integration_get_implementation(). If you have made a custom PCB design, you need to implement this function yourself. You may find the code in the file acc_hal_integration_stm32cube_sparkfun_a111.c useful as a starting point.

```
bool hal_test_spi_read_chipid(void)
{
  const uint32_t sensor = 1;
  uint8_t buffer[6] = {0x30, 0x0, 0x0, 0x0, 0x0, 0x0};
  acc_hal_t hal = acc_hal_integration_get_implementation();

  hal.sensor_device.power_on(sensor);
  hal.sensor_device.transfer(sensor, buffer, sizeof(buffer));
  hal.sensor_device.power_off(sensor);

  if (buffer[4] == 0x11 && buffer[5] == 0x12)
  {
    printf("Test OK!\n");
    return true;
  }
  else
  {
    printf("Cannot read chip id!\n");
    return false;
  }
}
```

When the program is executed, the signals to the A111 should look as in figure 13.

© 2023 by Acconeer AB - All rights reserved

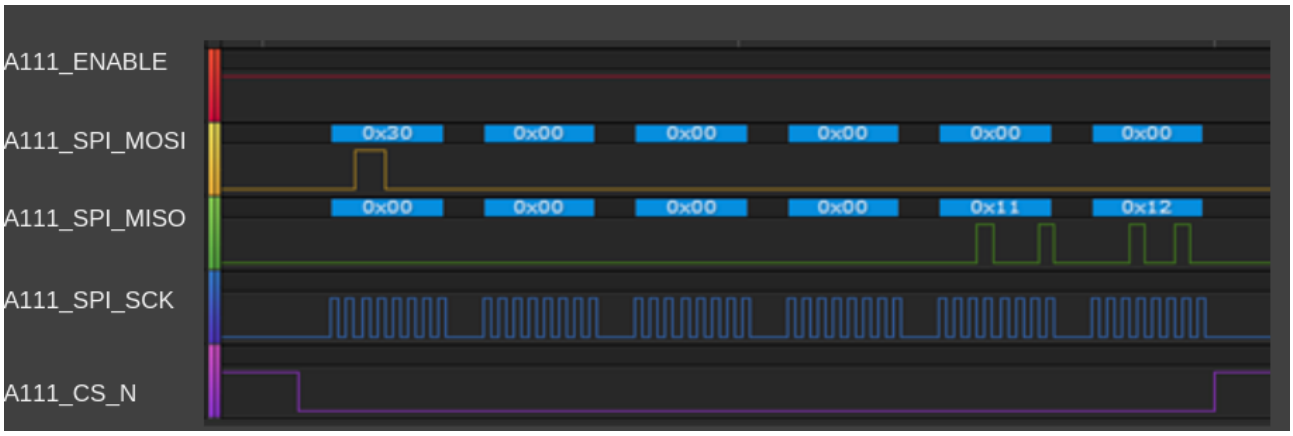

Figure 13: SPI transfer example

Note that the A111 enable signal must be set high at least 2 ms before the SPI transfer.

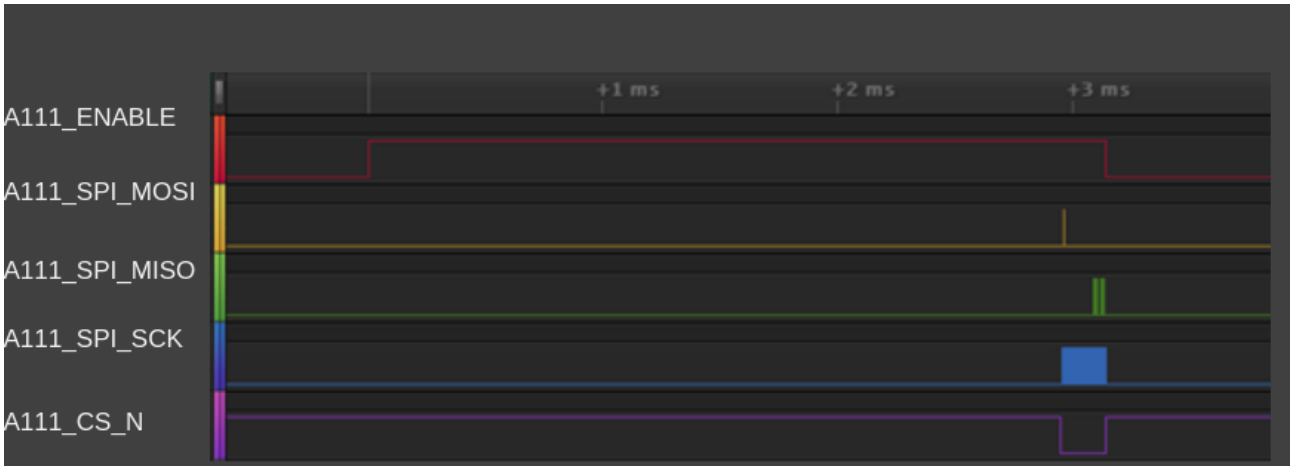

Figure 14: A111 Enable Signal

6.4 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:

```
hlpuart1.Init.BaudRate = 115200;
hlpuart1.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.

6.5 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 3.1.3 Libraries.

6.6 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

© 2023 by Acconeer AB - All rights reserved

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;
}
```

© 2023 by Acconeer AB - All rights reserved

7 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.

© 2023 by Acconeer AB - All rights reserved