XM125 Software

User Guide

XM125 Software

User Guide

Author: Acconeer AB

Version:a121-v1.7.0

Acconeer AB June 4, 2024

Contents

1 Acconeer SDK Documentation Overview 3

2 Introduction 4

3 Installing Software Image 5
- 3.1 Windows COM port drivers . . . 5
- 3.2 Flash Over UART Using STM32CubeProgrammer . . . 5
  - 3.2.1 Boot the XM125 in bootloader mode . . . 5
  - 3.2.2 Program the XM125 . . . 5
- 3.3 Flash Over UART Using stm32loader . . . 6

4 Setting up a Development Environment 7
- 4.1 Using a Debugger . . . 7
- 4.2 Building From the Command Line . . . 7
  - 4.2.1 Download Software Using a J-Link . . . 7
- 4.3 STM32CubeIDE . . . 7
  - 4.3.1 Configuring Project for Acconeer Software . . . 8
  - 4.3.2 Project Settings . . . 10
  - 4.3.3 Running the Program . . . 10
  - 4.3.4 Debug Output . . . 10

5 Troubleshooting and FAQ 11
- 5.1 LTO wrapper fails . . . 11
- 5.2 Link errors in sysmem.c . . . 11

6 Disclaimer 12

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 Touchless Button Reference Application	Describes the functionality of the Touchless Button Reference Application.	- Working with the Touchless Button Reference Application
A121 Parking Reference Application	Describes the functionality of the Parking Reference Application.	- Working with the Parking 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
XM126 Software	Describes how to develop for XM126.	- Working with XM126
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
I2C Breathing Reference Application	Describes the functionality of the I2C Breathing Reference Application.	- Working with the I2C Breathing Reference 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

The Acconeer Software Development Kit (SDK) enables customers to develop their own software that can be executed on the module. This enables full control of all the peripherals and to maximize the performance and power consumption for a specific use case.

The SDK comes with a number of example applications that can be used as a starting point when developing your own application. These applications can be downloaded and executed using the methods described in “Installing Software Image” at page 5.

When developing your own application we recommend that you setup a development environment as described in “Setting up a Development Environment” at page 7.

This guide has been verified in Ubuntu 20.04 and Windows with STM32CubeIDE 1.11.2 and STM32CubeMX 6.8.0

3 Installing Software Image

The XM125 uses the STM32L431 MCU which contains a ROM bootloader. The MCU is configured to enable the bootloader during manufacturing.

Another option is to use a SWD debugger, this requires additional hardware which is suitable when developing your own applications.

3.1 Windows COM port drivers

If running on Windows, you might need to install a driver for the USB to UART Bridge. It can be downloaded here.

3.2 Flash Over UART Using STM32CubeProgrammer

Download and install STM32CubeProgrammer.

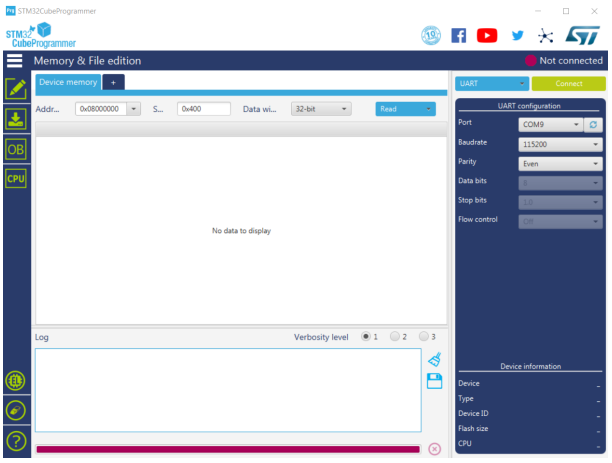
3.2.1 Boot the XM125 in bootloader mode

1. Connect the XE125 to your PC with a USB-C cable to the USB connector
2. Press and hold the “DFU” button on the board
3. Press the “RESET” button (still holding the “DFU” button)
4. Release the “RESET” button
5. Release the “DFU” button

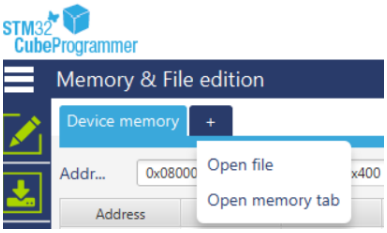
Your XM125 device is now in “DFU” mode waiting for a software upgrade procedure to be started.

3.2.2 Program the XM125

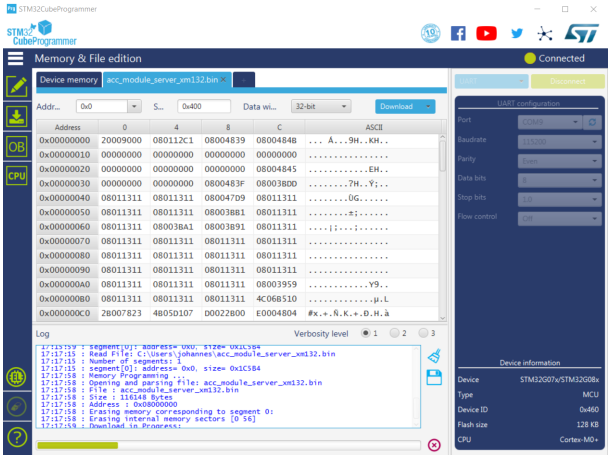
1. Start the STM32CubeProgrammer



2. Select correct port to the right. E.g. COM9.
3. Press “Connect” in the upper right corner
4. Press The “+” button and the “Open file”



5. Browse to and select the binary you like to program, e.g. “example_service.bin”
6. Press the “Download” button. The green progress bar in the bottom indicates the progress



7. Once programming is complete press the “Disconnect” button
8. Press the “RESET” button or do a power cycle to start the embedded application

3.3 Flash Over UART Using stm32loader

The stm32loader is a python program. See pypi.org/project/stm32loader/ for more information.

Install it using “pip install stm32loader”

1. Set the XM125 into bootloader mode, see above for how to do this
2. Program the device with “stm32loader -p /dev/ttyUSB0 -e -w -v example_service.bin”. Make sure to specify correct port.
3. Press “RESET” or power cycle the device to start the embedded application

4 Setting up a Development Environment

In order to develop your own applications you need to set up a development environment. The XM125 is based on a STM32L431 SoC by STMicroelectronics.

4.1 Using a Debugger

In order to debug your applications it is recommended to use a SWD debugger. We recommend that you use a SEGGER JLink debug probe e.g. J-Link BASE Compact or an ST-LINK debugger.

Figure 1: J-Link Base Compact

The J-Link BASE Compact can be used to set breakpoints and single step the program in an easy way.

4.2 Building From the Command Line

All example applications can be built from the command line using “make”.

1. Download the STM32Cube MCU Package for STM32L4 series (version 1.17.0) from www.st.com.
2. Extract the archive into a folder, e.g. “/home/acconeer/sdk/”
3. Download “GCC ARM Embedded 9-2020-q2-update” from developer.arm.com.
4. Extract the archive into a folder, e.g. “/home/acconeer/compilers/”
5. Download and extract the Acconeer SDK zip file, e.g. “/home/acconeer/xm125/”

```
$ cd /home/acconeer/xm125
$ export GNU_INSTALL_ROOT=/home/acconeer/compilers/gcc-arm-none-eabi-9-2020-
  q2-update/bin/
$ export STM32CUBE_FW_L4_ROOT="/home/acconeer/sdk/STM32Cube_FW_L4_V1.17.0/"
$ make -j10
```

The above will compile all example applications which can be downloaded to the target using any of the methods described in “Installing Software Image” at page 5

4.2.1 Download Software Using a J-Link

You can flash the software using a J-Link debugger from the command line. First install the “J-Link Software and Documentation Pack” from www.segger.com.

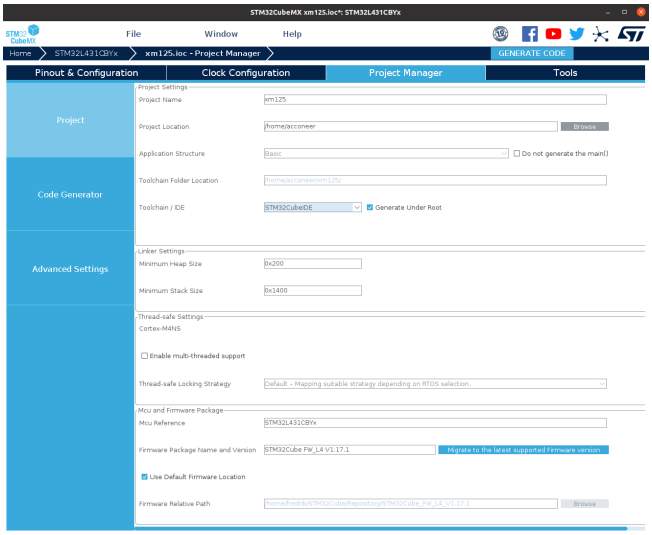
```
$ make flash_jlink_example_servic
```

4.3 STM32CubeIDE

If you prefer to use an integrated development environment we recommend that you use the STM32CubeIDE together with a SEGGER J-Link debug probe or an ST-Link debugger. The Acconeer SDK for XM125 includes an STM32CubeMX project file, ’xm125.ioc’. From this file it’s possible to generate an STM32CubeIDE project directly from the SDK.

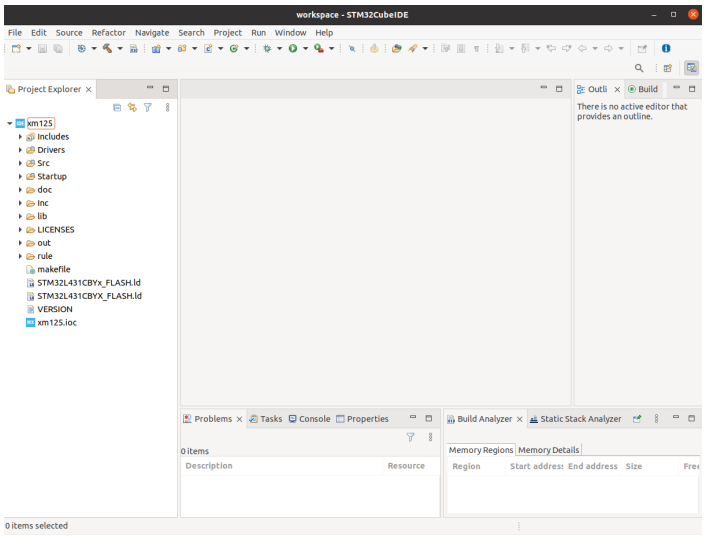
1. Download the latest version of STM32CubeMX from www.st.com.
2. Extract the archive into a temporary folder, e.g. “/home/acconeer/sdk/temp”
3. Run the installer for your preferred OS from “/home/acconeer/sdk/temp”
4. Download and install the latest version of STM32CubeIDE for your preferred OS from www.st.com.
5. Download and extract the Acconeer SDK zip file, e.g. “/home/acconeer/acconeer_xm125/”

6. Start STM32CubeMX, then select “File/Load Project. . . ” and browse to the folder where you unpacked the zip file, then select “xm125.ioc” and click on “Open”
7. Select the Project Manager tab and change “Toolchain / IDE” to “STM32CubeIDE” and press “GENERATE CODE”.
8. Select “Open Project” in the dialog to open the newly created project in STM32CubeIDE.



4.3.1 Configuring Project for Acconeer Software

Now when you have an STM32CubeIDE project you need to modify it to include the Acconeer SDK components.



Source Files The SDK includes many examples and applications. STM32CubeIDE will try to compile and link all source files in the SDK which will cause “multiple definition” errors when linking. To avoid this you should exclude the source files not needed from the build.

Select all source files starting with “applications”, “example” and “use_cases” except the file you want to use, right click and select “Resource Configurations → Exclude from build”.

For building “examples”, only the example source file is needed. For “use_cases”, the use case source file and “algorithms” are needed.

When building i2c-examples multiple files are needed.

For i2c_distance_detector.c the following files are needed:

- acc_reg_protocol.c
- distance_reg_protocol.c

- distance_reg_protocol_access.c
- i2c_application_system_stm32.c
- i2c_distance_detector.c

For i2c_presence_detector.c the following files are needed:

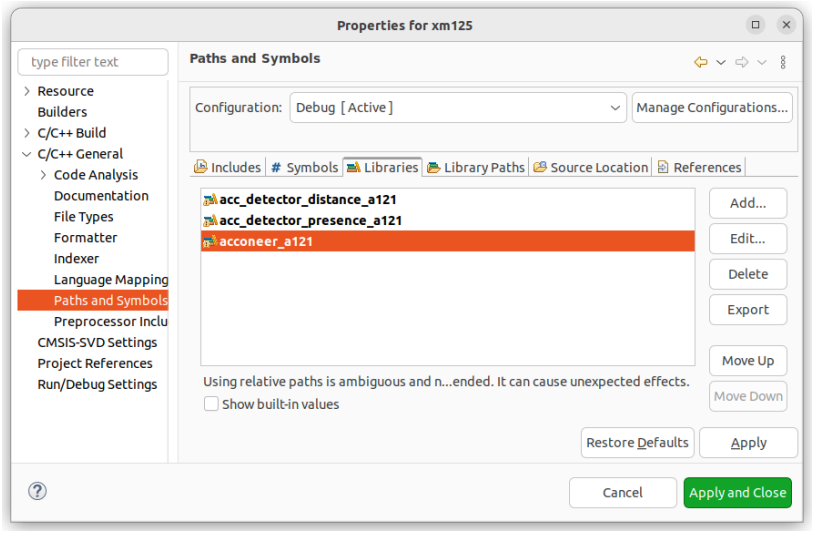
- acc_reg_protocol.c
- i2c_application_system_stm32.c
- i2c_presence_detector.c
- presence_reg_protocol.c
- presence_reg_protocol_access.c

Header-files You have to manually add the "Inc" folder to the project paths:

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 "Inc"-folder in your project

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 "lib"-folder in your project



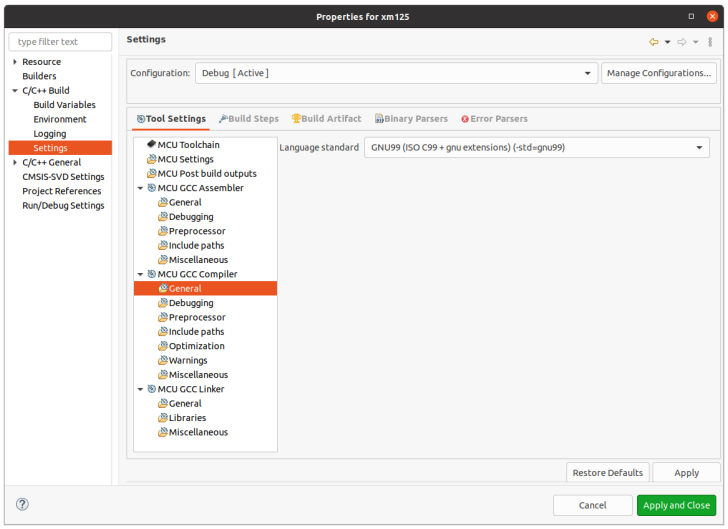
Once the path is set, you can add the specific libraries by the following:

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.

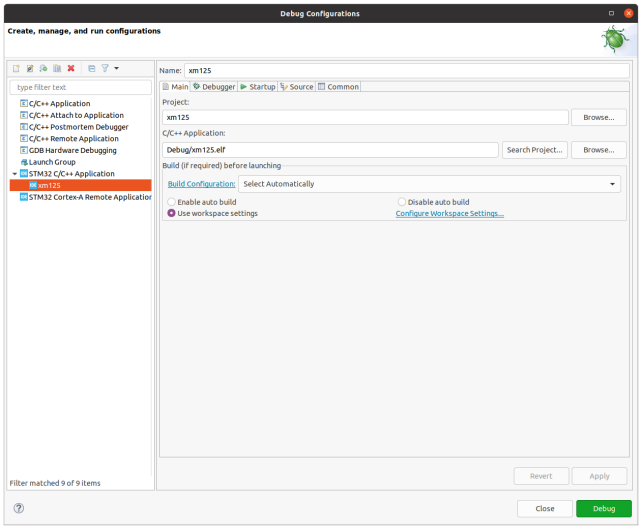
4.3.2 Project Settings

Select GNU99 as language standard in “Project → Properties → C/C++ Build → Settings → Tool Settings → MCU GCC Compiler → General”.



4.3.3 Running the Program

Build the software by pressing “Ctrl-B” and then start debugging by right-clicking on the project “xm125 → Debug As → STM32 Cortex-M C/C++ Application”. This will open the “Debug Configurations” dialog and there you can choose which debugger to use, “Debugger → Debug Probe”, either ST-LINK or SEGGER J-LINK. Click “Debug”, this will automatically flash the XM125 and execute the program until the “main()” function.



4.3.4 Debug Output

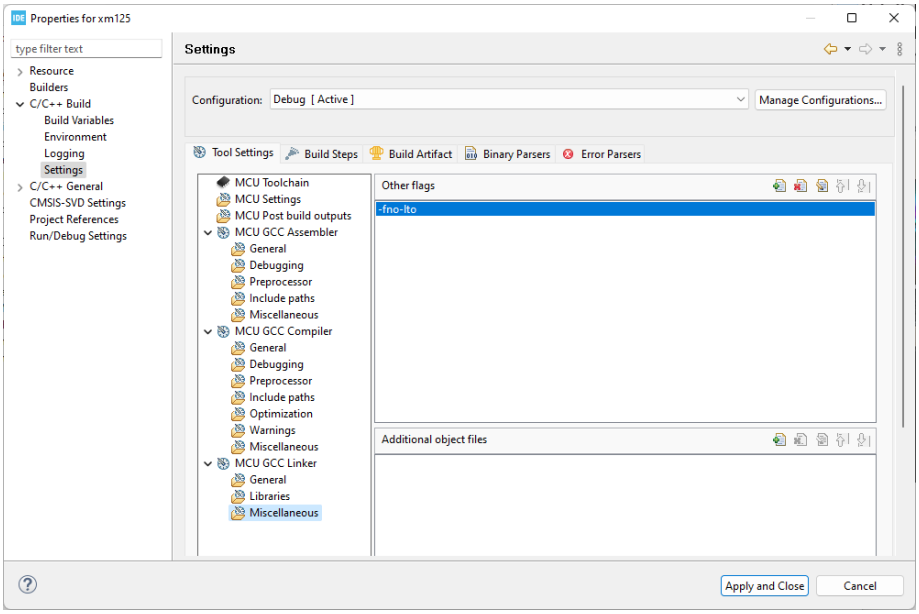
Debug logs will be outputted on UART2 using a baud rate of 921600.

5 Troubleshooting and FAQ

5.1 LTO wrapper fails

When using STM32CubeIDE for Windows there is a problem with the LTO wrapper. Therefore you need to explicitly disable LTO (link-time optimizations):

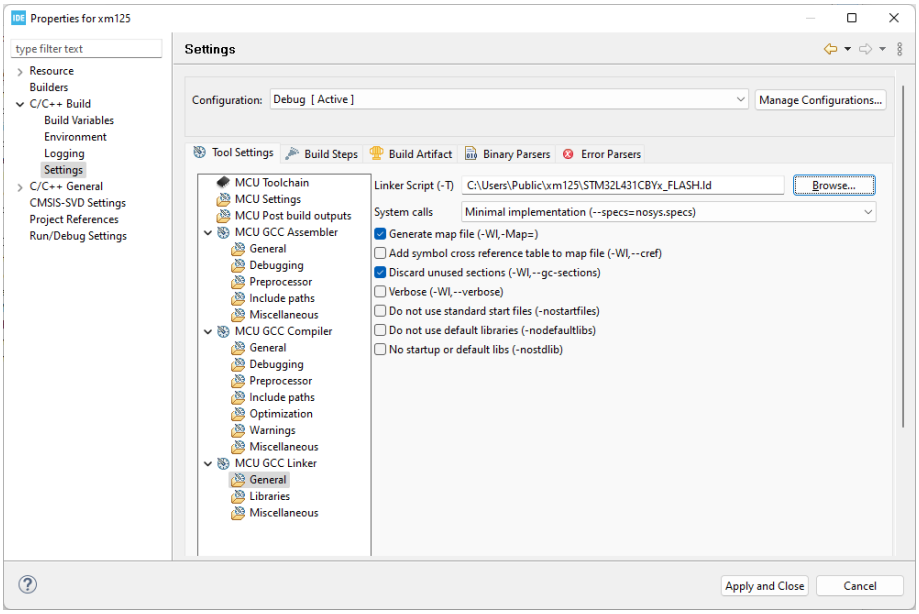
1. Go to“Project → Properties → C/C++ Build → Settings → Tool Settings → MCU GCC Linker → Miscellaneous → Other flags”.
2. Add “-fno-lto”



5.2 Link errors in sysmem.c

When using STM32CubeIDE for Windows there might be a problem with the tool not finding the linker script which will lead to linker errors in sysmem.c. Therefore you need to change the path to the Linker script:

1. Go into “Project → Properties → C/C++ Build → Settings → Tool Settings → MCU GCC Linker → General”.
2. Press “Linker Script (-T) → Browse” and find the file “STM32L431CBYx_FLASH.ld” from the project



6 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.