HAL Software Integration

User Guide

A111 – HAL Software Integration

User Guide

Author: Acconeer

Version 1.0 DRAFT: 2019-12-04

Acconeer AB

Table of Contents

1 Introduction

This document aims to provide help and support for customers that wish to integrate A111 towards an MCU or processor that is not currently supported by Acconeer. This document covers the SW integration needed to integrate Acconeer's libraries with MCU specific drivers. Hardware integration is covered in the document "Hardware and physical integration guideline PCR Sensor A111" that is available for download at http://developer.acconeer.com.

The A111 pulse coherent radar sensor is dependent on a software stack running on a host MCU. The host CPU software handles low level configuration of the radar sensor and post-processing of radar data. All low-level sensor configuration is uploaded to the sensor at startup meaning that no firmware or configuration parameters is stored in the sensor permanently.

2 Recommended and Supported HW

The number of potential MCU and processors, including variants, that users of A111 might want to use in their products are bordering to endless. Acconeer will not be able to support all of them but have selected a few processors that every SW release are tested against. Some of them have also been included in different versions of our EVK/Reference Designs.

2.1 Supported HW

Acconeer has designed EVK or Modules both for internal and external use based on the following MCU/Processors. All of which can be bought on Digi Key. (www.digikey.com)

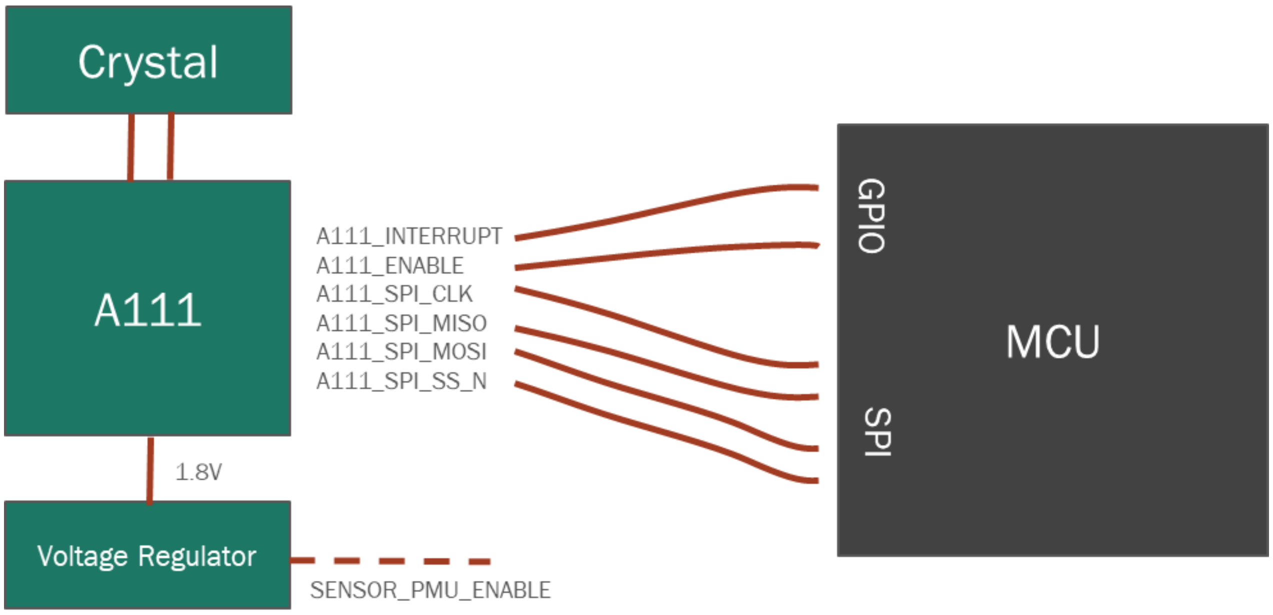
- Raspberry Pi 3, 3+ and 4: Our standard EVK (XC/XR112), running Linux.
- Atmel ATSAME70Q21A-AN: An ARM Cortex M7 MCU that we have chosen to use in our performance module XM112.
- Nordic Semiconductors nRF52840: An ARM Cortex M4 MCU that is featured in our IoT module XM122, designed for Low Cost and Low Power.

Documentation, including schematics and BOM, for above HW is available from Acconeer web site www.acconeer.com/products.

Acconeer also have SW generic support for ARM Cortex M4 and M7 based MCUs. Instructions for how to integrate STM32 MCUs from ST Microelectronics and set up a project in STM32CubeIDE can be found on http://developer.acconeer.com. For other ARM Cortex M4 and M7 based MCUs a Hardware Abstraction Layer (HAL) integration must be written according to the guidelines in this document.

3 HW Integration Overview

Hardware integration is covered in the document “Hardware and physical integration guideline PCR Sensor A111” that is available for download at http://developer.acconeer.com. The purpose of this section is only to define the pin names and connections necessary for the software integration.



3.1 GPIO Control Signals

The Acconeer Sensor SW package uses two GPIO control signals A111_ENABLE and A111_INTERRUPT.

A111_ENABLE signal is used to turn the A111 sensor on and off. The A111_ENABLE signal is active high.

The **A111_INTERRUPT** signal is used to signal the host MCU that the internal buffer memory contains new measurement data ready to be transferred via the SPI interface. The A111_INTERRUPT signal is active high.

3.2 SPI Bus

The A111 communicates with the host MCU via a 4-line SPI interface. The maximum supported SPI clock frequency is 50MHz. The A111 is an SPI slave device. The SPI signals are named **A111_SPI_CLK**, **A111_SPI_MOSI**, **A111_SPI_MISO**, and **A111_SPI_SS_N**. Note that the slave select signal, A111_SPI_SS_N, is active low and often controlled by a GPIO on the MCU.

3.3 Power Control

Some hardware designs include a Power Management Unit (PMU) or a power switch that allows the MCU to completely shut off the power to the A111 sensor. In examples for software integration we use a GPIO pin with the name SENSOR_PMU_ENABLE to control the power to the radar sensor.

3.4 Crystal

The A111 radar sensor needs a crystal for clock reference. The frequency of the crystal has to be specified in the HAL integration.

4 Prototype Integrations

For prototype integrations with an MCU development board we recommend using the "SparkFun Pulsed Radar Breakout - A111". It is a breakout board with an A111 radar sensor, crystal and a level shifter that makes it easy to connect the A111 to an MCU board with I/O-pins that operates at 3.3V. Note that some Sparkfun breakout boards have the Vcc pin labeled 5V. This pin should be connected to 3.3V if you have an MCU with 3.3V logic on the I/O pins.

5 Software Integration

The Acconeer software delivery consists of precompiled libraries and headers together with example programs. The example programs are given in source code, while the Radar System Services, RSS, is part of the precompiled libraries. RSS is the software that will help you interact with the A111 radar sensor. To properly function, however, this software needs help. The reason is that it can't properly know your hardware and what kind of pin configuration you are using. This is solved by a user written Hardware Abstraction Layer (HAL). The HAL is a glue layer between RSS and the MCU drivers.

RSS must be activated before it is used, and a HAL struct is passed as a function argument to the function acc_rss_activate. The HAL struct contains properties and function pointers that RSS use in its communication with the hardware. The HAL struct and the function pointer types are declared in the header file acc_hal_definitions.h.

The implementation of the HAL layer is typically small and can in many cases be fitted into a single integration file. See for example the file acc_hal_integration_single_thread_stm32cube_sparkfun_a111.c that is available in our Cortex M4 package for STM32. It is a good starting point for custom PCB designs.

The following sub-chapters describe how to set the properties and write the functions that are included in the HAL struct.

5.1 The sensor_count Property

The sensor count property should be set the number of sensors supported by the integration.

5.2 The max_spi_transfer_size Property

The max_spi_transfer_size should be set to the maximum buffer size that can be transferred over the SPI bus. If there is no restriction, the limit should be set to 4096.

5.3 The sensor_id Function Parameter

Many of the integration functions, described in the sections below, take a sensor_id as a parameter. The sensor id is only important if several radar sensors are integrated to the same MCU. When designs where only one sensor is used the sensor id can be ignored.

5.4 Power-on Function

The power on function is called when the sensor needs to be turned on and enabled. It takes sensor_id as an argument and it returns void.

The function should do the following:

- Turn on power to the A111 sensor (if needed)
- Set **A111_ENABLE** High
- Set **A111_SPI_SS_N** High
- Wait 2 ms for sensor crystal to stabilize

5.5 Power-off Function

The power off function is called when the sensor should be turned off.

The function should do the following:

- Set **A111_SPI_SS_N** Low
- Set **A111_ENABLE** Low
- Turn off power to sensor (if there is a power switch or PMU)

5.6 Sensor Transfer Function

The sensor transfer function is called by RSS to transfer data to and from the radar sensor over the SPI bus. The maximum buffer size can be limited by setting the property max_spi_transfer_size in the HAL struct.

The function should do the following:

- Set **A111_SPI_SS_N** Low
- Send and receive SPI buffer
- Set **A111_SPI_SS_N** High

5.7 Wait for Interrupt Function

This function shall wait at most timeout_ms for the interrupt to become active and then return true. It may return true immediately if an interrupt has occurred since last call to this function. If a time out occurred and the function waited more than timeout_ms for the interrupt to become active it shall return false.

A simple and portable way of implementing the wait for interrupt function is to use polling. The function should then repeatedly do the following until the interrupt pin is high or a timeout has occurred:

- Check the A111_INTERUPT pin and return 1 if interrupt pin is HIGH
- Sleep a short time
- Return 0 if timeout
- Start over and check the interrupt pin again.

While polling is great for maximum portability an optimized implementation may take advantage of HW interrupt support in the MCU to reduce latency and power consumption.

5.8 Get Time Function

The get time function should return the current time in milliseconds. The value is primarily used in log messages and does not have to return the correct wall clock time. Many implementations return the number of milliseconds since power-on of the MCU.

5.9 Log Function

The purpose of the log function is to format log messages and to print them to e.g. the console or debug UART.

The function below shows an example of how the log formatting can be implemented:

```
void acc_hal_integration_log(acc_log_level_t level,
                             const char *module,
                             const char *buffer)
{
  uint32_t time_ms = acc_hal_integration_get_current_time();
  char     level_ch;

  unsigned int timestamp    = time_ms;
  unsigned int hours        = timestamp / 1000 / 60 / 60;
  unsigned int minutes      = timestamp / 1000 / 60 % 60;
  unsigned int seconds      = timestamp / 1000 % 60;
  unsigned int milliseconds = timestamp % 1000;

  level_ch = (level < ACC_LOG_LEVEL_MAX) ? "EWIVDD"[level] : '?';

  printf(LOG_FORMAT, hours, minutes, seconds, milliseconds,
         0, level_ch, module, buffer);

}
```

5.10 Log Level Configuration

The log.log_level property should be set in HAL struct to restrict the number of log messages generated by RSS.

Supported log levels:

- ACC_LOG_LEVEL_ERROR
- ACC_LOG_LEVEL_WARNING
- ACC_LOG_LEVEL_INFO
- ACC_LOG_LEVEL_VERBOSE
- ACC_LOG_LEVEL_DEBUG

6 Support

Acconeer will be able to provide support on most aspects of integrating our product to an MCU independent if it is already supported or not. Acconeer kindly ask customers to submit technical questions by creating an account on our support portal.

https://acconeer.freshdesk.com/support/login

No specific service level or response time is guaranteed for this free support.

6.1 Service Level Agreement

To ensure support with a specified and committed service level we have the option to offer our customers a Service Level Agreement. This will include, among else, opening hours, committed response times, change requests onsite support etc. For inquiries please send a mail to mailto:info@acconeer.com .

7 References, List of Documentation

1. “Hardware and physical integration guideline PCR Sensor A111”
2. “Integration using STM32CubeIDE”

All Documents referred to can be found on http://developer.acconeer.com

8 Revision History

Date	Version	Change
2019-12-04	1.0	Initial version for SW v 2.0.0

Disclaimer