A121 Ripple

User Guide

A121 Ripple

User Guide

Author: Acconeer AB

Version:a121-v1.10.0

Acconeer AB March 26, 2025

© 2025 by Acconeer AB - All rights reserved

Contents

© 2025 by Acconeer AB - All rights reserved

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
	A121 Radar Data and Control (PDF)	
A121 Radar Data and Control	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

Ripple™, hosted by the Consumer Technology Association (CTA)®, is an open-radar API standard to enable hardware and software interoperability while accelerating the growth of applications for general purpose consumer radar.

Experts from across the silicon, sensing, automotive and electronics industries have come together to develop open and standardized API interfaces for radar system development. The standardized API calls for general purpose radar that enables interoperability and the rapid deployment of new applications.

Acconeer has been part of the group defining the API, and is fully compliant with Ripple 2.0 which, unlike previous versions, supports pulsed radar.

3 Ripple Porting Layer

First, let’s highlight some differences in the Ripple API naming compared to the RSS API.

- A ‘burst’ in Ripple is a ‘frame’ in RSS.
- ‘Samples per sweep’ in Ripple is ‘num_points’ in RSS.
- ‘Start offset’ in Ripple is ‘start_point’ in RSS.

3.1 API Mapping

The following table maps the API defined by Ripple to the functions used in RSS.

Ripple API	RSS API	Comment
radarInit	acc_rss_hal_register	-
radarDeinit	-	-
radarCreate	acc_config_create acc_sensor_create acc_sensor_calibrate	-
radarDestroy	acc_config_destroy acc_sensor_destroy acc_processing_destroy	-
radarGetState	-	See details in “Radar State”
radarTurnOn	acc_hal_integration_sensor_supply_on acc_hal_integration_sensor_enable acc_sensor_prepare	-
radarTurnOff	acc_hal_integration_sensor_disable acc_hal_integration_sensor_supply_off	-
radarGoSleep	acc_sensor_hibernate_on acc_hal_integration_sensor_disable	-
radarWakeUp	acc_hal_integration_sensor_enable acc_sensor_hibernate_off	-
radarGetNumConfigSlots	-	-
radarGetMaxActiveConfigSlots	-	Only one config supported currently
radarActivateConfig	acc_processing_create	-
radarDeactivateConfig	acc_processing_destroy	-
radarGetMainParam	-	See “Main Radar Parameters”
radarSetMainParam	-	See “Main Radar Parameters”
radarGetMainParamRange	-	See “Main Radar Parameters”
radarGetRxParam	-	See “RX Radar Parameters”
radarSetRxParam	-	See “RX Radar Parameters”
radarGetRxParamRange	-	See “RX Radar Parameters”
radarGetVendorParam	-	See “Vendor Radar Parameters”
radarSetVendorParam	-	See “Vendor Radar Parameters”
radarGetVendorParamRange	-	See “Vendor Radar Parameters”
radarStartDataStreaming	-	See “Radar Measurement Loop”
radarStopDataStreaming	-	See “Radar Measurement Loop”
radarIsBurstReady	-	See “Radar Measurement Loop”
radarReadBurst	-	See “Radar Measurement Loop”
radarSetBurstReadyCb	-	See “Radar Measurement Loop”
radarSetLogCb	-	-
radarGetSensorInfo	-	-
radarGetRadarApiVersion	-	-
radarLogSensorDetails	acc_config_log	-
radarSetLogLevel	-	-

3.2 Main Radar Parameters

The following main parameters defined in the Ripple API are supported:

Ripple API	RSS API
RADAR_PARAM_AFTERBURST_POWER_MODE	acc_config_inter_frame_idle_state_get acc_config_inter_frame_idle_state_set
RADAR_PARAM_BURST_PERIOD_US	acc_config_frame_rate_get acc_config_frame_rate_set
PULSED_PARAM_INTERSWEEP_POWER_MODE	acc_config_inter_sweep_idle_state_get acc_config_inter_sweep_idle_state_set
PULSED_PARAM_SWEEP_PERIOD_US	acc_config_sweep_rate_get acc_config_sweep_rate_set
PULSED_PARAM_SWEEPS_PER_BURST	acc_config_sweeps_per_frame_get acc_config_sweeps_per_frame_set
PULSED_PARAM_SAMPLES_PER_SWEEP	acc_config_num_points_get acc_config_num_points_set
PULSED_PARAM_START_OFFSET	acc_config_start_point_get acc_config_start_point_set
PULSED_PARAM_PRF_IDX	acc_config_prf_get acc_config_prf_set

3.3 RX Radar Parameters

The following RX parameter defined in the Ripple API is supported:

Ripple API	RSS API
PULSED_RX_PARAM_VGA_IDX	acc_config_receiver_gain_get acc_config_receiver_gain_set

3.4 Vendor Radar Parameters

The following vendor specific parameters defined in the Ripple API are supported:

Ripple API	RSS API
PULSED_PARAM_STEP_LENGTH	acc_config_step_length_get acc_config_step_length_set
PULSED_PARAM_HWAAS	acc_config_hwaas_get acc_config_hwaas_set
PULSED_PARAM_PROFILE	acc_config_profile_get acc_config_profile_set
PULSED_PARAM_ENABLE_TX	acc_config_enable_tx_get acc_config_enable_tx_set

3.5 Radar Measurement Loop

The radar measurement loop is started by invoking **radarStartDataStreaming**.

When new radar data is ready to read, the **RadarBurstReadyCB** callback registered through **radarSetBurstReadyCb** is invoked. It's also possible to poll **radarIsBurstReady**.

Radar data is read using **radarReadBurst**.

The radar measurement loop is stopped by invoking **radarStopDataStreaming**.

The radar measurement loop consists of the following function calls towards RSS

© 2025 by Acconeer AB - All rights reserved

```
    acc_sensor_measure(...)

    while (!not_stopped)
    {
        acc_hal_integration_wait_for_sensor_interrupt(...);

        acc_sensor_read(...);

        acc_processing_execute(...);

        acc_sensor_measure(...);

        if (burst_ready_callback != NULL)
        {
            burst_ready_callback(...)
        }
    }
```

3.5.1 Radar Data Format

The data buffer you get from **radarReadBurst** provides radar data in a complex format. In order to make sense of the data some kind of processing usually needs to be done. For more information about the burst format, see docs.acconeer.com.

3.6 Radar State

The state of the radar can be read out by invoking radarGetState. The state is determined by the following finite state machine.

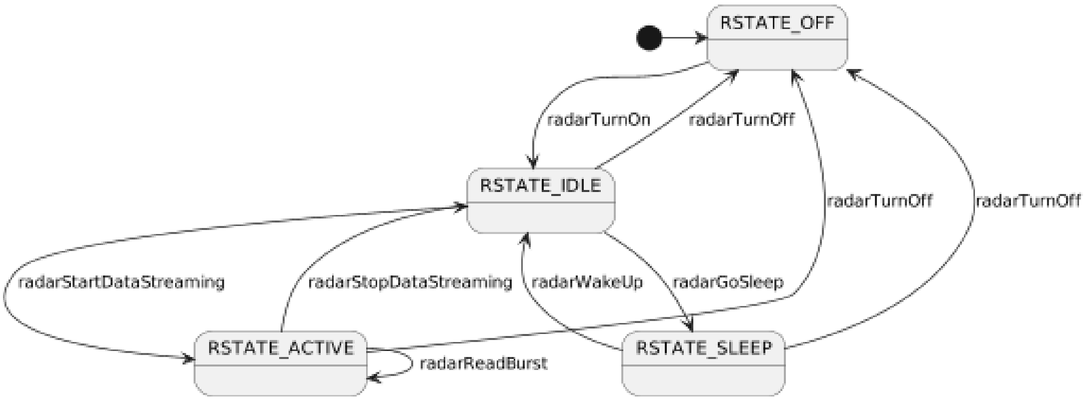

Figure 1: Radar state

3.7 Functions Not Supported

Ripple API
radarIsActiveConfig
radarGetTxParam
radarSetTxParam
radarGetTxParamRange
radarGetVendorTxParam
radarSetVendorTxParam
radarGetVendorTxParamRange
radarGetVendorRxParam
radarSetVendorRxParam
radarGetVendorRxParamRange
radarSetRegisterSetCb
radarCheckCountryCode

radarGetTxPosition
radarGetRxPosition
radarGetAllRegisters
radarGetRegister
radarSetRegister

3.8 Limitations

The Ripple API doesn't support the subsweep concept defined in the RSS API. If this is needed, the RSS API needs to be used directly.

3.9 Example Application

It is recommended to use this Guide together with example_ripple.c located in the Acconeer Ripple package. This example outlines the expected flow of the Ripple API as well as outputting radar data on stdout.

How to build the example is described in the 'A121 Raspberry Pi Software' User Guide also available in the Acconeer Ripple package.

The example application can be executed on RPi + XE121.

© 2025 by Acconeer AB - All rights reserved

4 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.

© 2025 by Acconeer AB - All rights reserved