



# XM125 I<sup>2</sup>C Presence Detector

User Guide



XM125 I<sup>2</sup>C Presence Detector

User Guide

Author: Acconeer AB

Version:a121-v1.13.0

Acconeer AB March 26, 2026



## Contents

<b>1</b>	<b>Acconeer SDK Documentation Overview</b>	<b>4</b>
<b>2</b>	<b>I<sup>2</sup>C Presence Detector Application</b>	<b>6</b>
2.1	I <sup>2</sup> C Address Configuration	6
2.2	I <sup>2</sup> C Speed	6
2.3	Usage	6
2.3.1	Read Detector Status	6
2.3.2	Writing a command	6
2.3.3	Setup and Start Detector	6
2.3.4	Stop and Restart Detector	7
2.4	Advanced Usage	7
2.4.1	Debug UART logs	7
2.4.2	Reset Module	7
2.4.3	Presence Detection on GPIO	7
<b>3</b>	<b>Register Protocol</b>	<b>8</b>
3.1	I <sup>2</sup> C Slave Address	8
3.2	Protocol Byte Order	8
3.2.1	I <sup>2</sup> C Write Register(s)	8
3.2.2	I <sup>2</sup> C Read Register(s)	8
3.3	Register Protocol - Low Power Mode	10
3.3.1	I <sup>2</sup> C Communication with Low Power Mode	10
<b>4</b>	<b>File Structure</b>	<b>11</b>
<b>5</b>	<b>Embedded Host Example</b>	<b>11</b>
5.1	Register Read/Write functions	11
5.2	Detector setup functions	13
<b>6</b>	<b>Registers</b>	<b>16</b>
6.1	Register Map	16
6.2	Register Descriptions	16
6.2.1	Version	16
6.2.2	Protocol Status	17
6.2.3	Measure Counter	17
6.2.4	Detector Status	17
6.2.5	Presence Result	18
6.2.6	Presence Distance	18
6.2.7	Intra Presence Score	19
6.2.8	Inter Presence Score	19
6.2.9	Presence Actual Frame Rate	19
6.2.10	Sweeps Per Frame	19
6.2.11	Inter Frame Presence Timeout	19
6.2.12	Intra Detection Enabled	19
6.2.13	Inter Detection Enabled	20
6.2.14	Frame Rate	20
6.2.15	Intra Detection Threshold	20
6.2.16	Inter Detection Threshold	20
6.2.17	Inter Frame Deviation Time Const	20
6.2.18	Inter Frame Fast Cutoff	20
6.2.19	Inter Frame Slow Cutoff	21
6.2.20	Intra Frame Time Const	21
6.2.21	Intra Output Time Const	21
6.2.22	Inter Output Time Const	21
6.2.23	Auto Profile Enabled	21
6.2.24	Auto Step Length Enabled	21
6.2.25	Manual Profile	22
6.2.26	Manual Step Length	22
6.2.27	Start	22



6.2.28 End . . . . . 22  
6.2.29 Reset Filters On Prepare . . . . . 22  
6.2.30 Hwaas . . . . . 23  
6.2.31 Automatic Subsweeps . . . . . 23  
6.2.32 Signal Quality . . . . . 23  
6.2.33 Detection On Gpio . . . . . 23  
6.2.34 Command . . . . . 23  
6.2.35 Application Id . . . . . 24

**7 Disclaimer 25**



## 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
<b><i>RSS API documentation (html)</i></b>		
rss_api	The complete C API documentation.	- RSS application implementation - Understanding RSS API functions
<b><i>User guides (PDF)</i></b>		
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 Vibration Example Application	Describes the functionality of the Vibration Example Application.	- Working with the Vibration Example 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
A121 ESP32 User Guide	Describes how to develop with A121 and ESP32 targets.	- Working with ESP32 targets
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
I2C Cargo Example Application	Describes the functionality of the I2C Cargo Example Application.	- Working with the I2C Cargo Example Application
<b><i>A121 Radar Data and Control (PDF)</i></b>		
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
<b><i>Readme (txt)</i></b>		



README	Various target specific information and links	- After SDK download
--------	---	----------------------



## 2 I<sup>2</sup>C Presence Detector Application

The I<sup>2</sup>C Presence Detector is an application that implements the Acconeer Presence Detector with a register based I<sup>2</sup>C interface.

The functionality of the presence detector is described in *A121 Presence Detector User Guide.pdf* or in [Acconeer Docs](#).

**Note:** Some of the registers like **start** and **end** have a different unit in the I<sup>2</sup>C Presence Detector, millimeters instead of meters, to make it easier to handle the register values as integers.

### 2.1 I<sup>2</sup>C Address Configuration

The device has a configurable I<sup>2</sup>C address. The address is selected depending on the state of the **I2C\_ADDR** pin according to the following table:

Connected to GND	0x51
Not Connected	0x52
Connected to VIN	0x53

### 2.2 I2C Speed

The device supports I2C speed up to 100kbps in Standard Mode and up to 400kbps in Fast Mode.

### 2.3 Usage

The module must be ready before the host starts I<sup>2</sup>C communication.

The module will enter ready state by following this procedure.

- Set **WAKE\_UP** pin of the module HIGH.
- Wait for module to be ready, this is indicated by the **MCU\_INT** pin being HIGH.
- Start I<sup>2</sup>C communication.

The module will enter a low power state by following this procedure.

- Wait for module to be ready, this is indicated by the **MCU\_INT** pin being HIGH.
- Set the **WAKE\_UP** pin of the module LOW.
- Wait for ready signal, the **MCU\_INT** pin, to become LOW.

#### 2.3.1 Read Detector Status

The status of the module can be acquired by reading the *Detector Status* register, The most important bits are the **Busy** and **Error** bits.

The **Busy** bit must not be set when a new command is written. If any of the **Error** bits are set the module will not accept any commands except the **RESET\_MODULE** command.

#### 2.3.2 Writing a command

A command is written to the *Command* register. When a command is written the **Busy** bit in the *Detector Status* register is set and it will be cleared automatically when the command has finished.

#### 2.3.3 Setup and Start Detector

Before the module can perform presence detection it must be configured. The following steps is an example of how this can be achieved.

**Note:** The configuration parameters can not be changed after a **APPLY\_CONFIGURATION** command. If reconfiguration is needed the module must be restarted by writing **RESET\_MODULE** to the *Command* register.

- Power on module
- Read *Detector Status* register and verify that neither **Busy** nor **Error** bits are set.
- Write configuration to configuration registers, for example *Start* register and *End* register.



- Write **APPLY\_CONFIGURATION** to *Command* register.
- Poll *Detector Status* until **Busy** bit is cleared.
- Verify that no **Error** bits are set in the *Detector Status* register.
- Write **START\_DETECTOR** to *Command* register.
- Poll *Detector Status* until **Busy** bit is cleared.
- Verify that no **Error** bits are set in the *Detector Status* register.
- Read *Detector Result* register
  - If **PRESENCE\_DETECTED** is set presence is currently detected.
  - If **PRESENCE\_DETECTED\_STICKY** is set presence has been detected since last read.
  - If **DETECTOR\_ERROR** is set an error has occurred, restart module with the **RESET\_MODULE** command.
  - If presence was detected, the presence distance can be read in the *Presence Distance* register.

### 2.3.4 Stop and Restart Detector

The detector can be stopped and restarted.

The following steps is an example of how to stop the detector.

- Read *Detector Status* register and verify that neither **Busy** nor **Error** bits are set.
- Write **STOP\_DETECTOR** to *Command* register.
- Poll *Detector Status* until **Busy** bit is cleared.
- Verify that no **Error** bits are set in the *Detector Status* register.

The following steps is an example of how to re-start the detector.

- Read *Detector Status* register and verify that neither **Busy** nor **Error** bits are set.
- Write **START\_DETECTOR** to *Command* register.
- Poll *Detector Status* until **Busy** bit is cleared.
- Verify that no **Error** bits are set in the *Detector Status* register.

## 2.4 Advanced Usage

### 2.4.1 Debug UART logs

UART logging can be enabled on the DEBUG UART by writing **ENABLE\_UART\_LOGS** to the *Command* register.

The detector configuration can be logged on the UART by writing **LOG\_CONFIGURATION** to the *Command* register.

UART logging can be disabled by writing **DISABLE\_UART\_LOGS** to the *Command* register.

### 2.4.2 Reset Module

The module can be restarted by writing **RESET\_MODULE** to the *Command* register.

After the restart the detector must be configured again.

### 2.4.3 Presence Detection on GPIO

The I<sup>2</sup>C Presence Detector can be configured to set **MISC\_GPIO0** pin HIGH when presence is detected, and LOW when presence is not detected. To enable presence detection on GPIO, write 1 to the *Detection On Gpio* register. To disable presence detection on GPIO, write 0 to the *Detection On Gpio* register.



### 3 Register Protocol

#### 3.1 I<sup>2</sup>C Slave Address

The default slave address is 0x52.

#### 3.2 Protocol Byte Order

Both register address, 16-bit, and register data, 32-bit, are sent in big endian byte order.

##### 3.2.1 I<sup>2</sup>C Write Register(s)

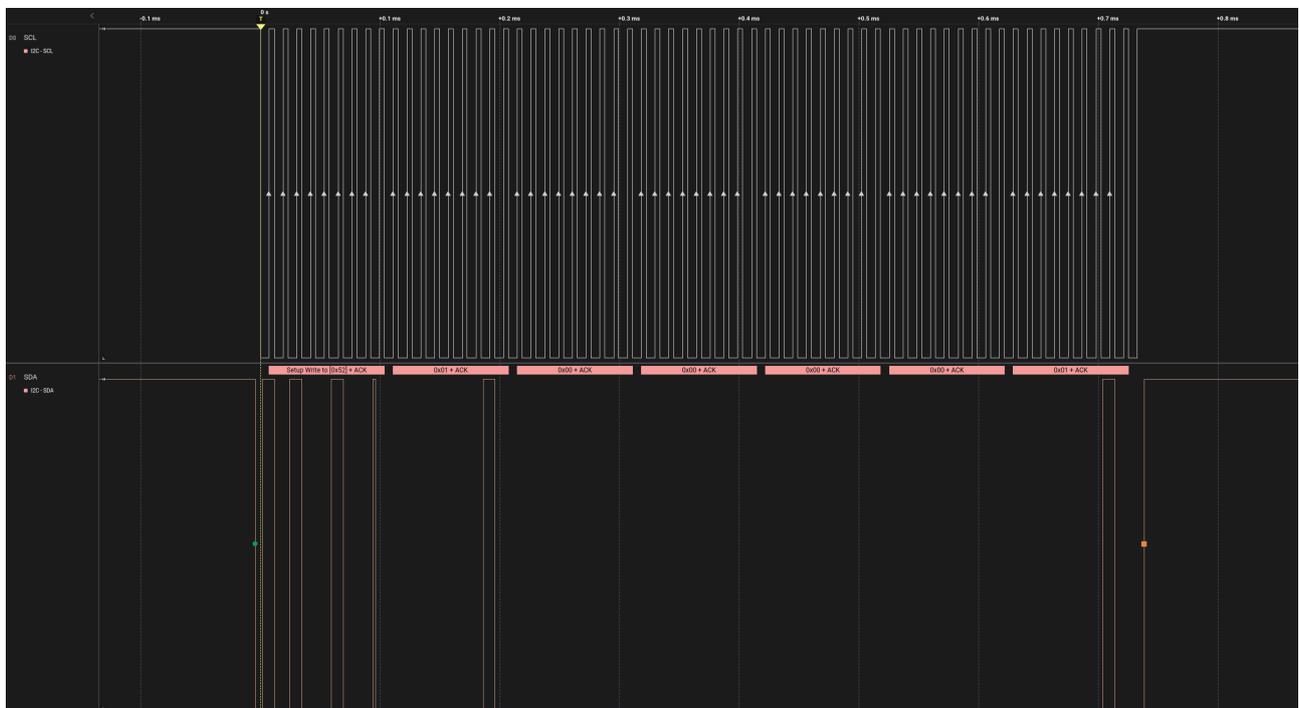
A write register operation consists of an I<sup>2</sup>C write of two address bytes and four data bytes for each register to write. Several registers can be written in the same I<sup>2</sup>C transaction, the register address will be incremented by one for each four data bytes.

*Example 1: Writing six bytes will write one register, two address bytes and four data bytes.*

*Example 2: Writing 18 bytes will write four registers, two address bytes and 16 data bytes.*

**Example operation, write 0x11223344 to address 0x0025.**

Description	Data
I <sup>2</sup> C Start Condition	
Slave Address + Write	0x52 + W
Address to slave [15:8]	0x00
Address to slave [7:0]	0x25
Data to slave [31:24]	0x11
Data to slave [23:16]	0x22
Data to slave [15:8]	0x33
Data to slave [7:0]	0x44
I <sup>2</sup> C Stop Condition	



Example Waveform: Write register with address 0x0100, the data sent from the master to the slave is 0x00000001

##### 3.2.2 I<sup>2</sup>C Read Register(s)

A read register operation consists of an I<sup>2</sup>C write of two address bytes followed by an I<sup>2</sup>C read of four data bytes for each register to read. Several registers can be read in the same I<sup>2</sup>C transaction, the register address will be incremented by one for each four data bytes.

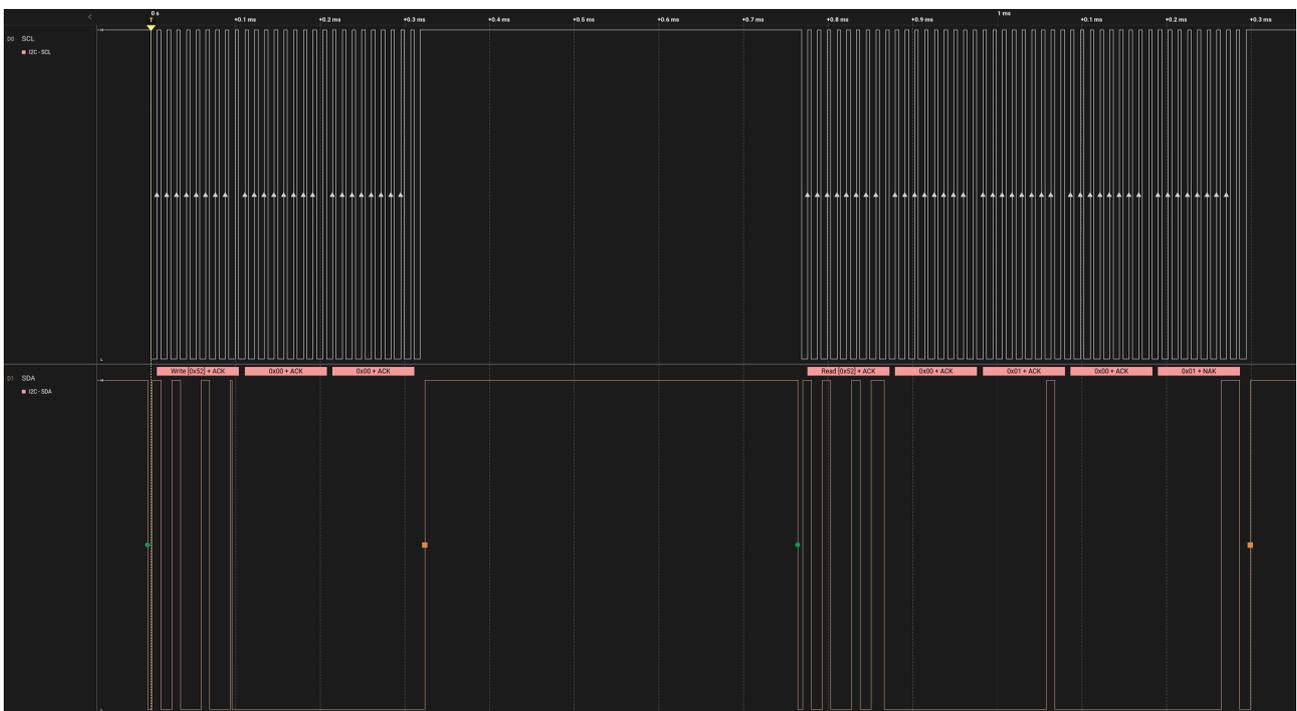
*Example 1: Writing two bytes and reading four bytes will read one register.*



Example 2: Writing two bytes and reading 16 bytes will read four registers.

**Example operation, read 0x12345678 from address 0x0003.**

Description	Data
I <sup>2</sup> C Start Condition	
Slave Address + Write	0x52 + W
Address to slave [15:8]	0x00
Address to slave [7:0]	0x03
I <sup>2</sup> C Stop Condition	
I <sup>2</sup> C Start Condition	
Slave Address + Read	0x52 + R
Data from slave [31:24]	0x12
Data from slave [23:16]	0x34
Data from slave [15:8]	0x56
Data from slave [7:0]	0x78
I <sup>2</sup> C Stop Condition	



Example Waveform: Read register with address 0, the data sent from the slave to the master is 0x00010001



### 3.3 Register Protocol - Low Power Mode

#### 3.3.1 I<sup>2</sup>C Communication with Low Power Mode

##### Low power example

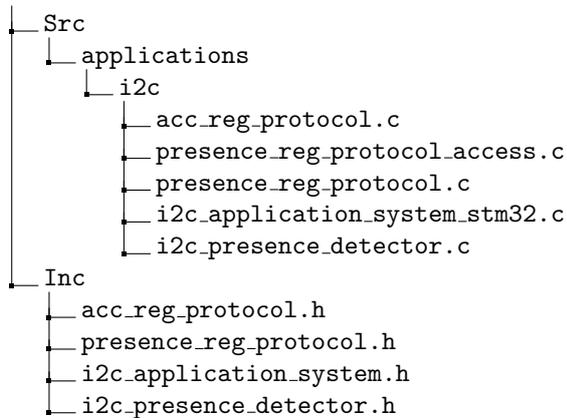


*Low Power Example: Magnification of Wake up, Setup Presence Detector, Power down*



## 4 File Structure

The I<sup>2</sup>C Presence Detector application consists of the following files.



- **acc\_reg\_protocol.c** A generic protocol handler implementation.
- **presence\_reg\_protocol.c** The specific register protocol setup for the I<sup>2</sup>C Presence Detector.
- **presence\_reg\_protocol\_access.c** The register read and write access functions for the I<sup>2</sup>C Presence Detector.
- **i2c\_application\_system\_stm32.c** System functions, such as I<sup>2</sup>C handling, GPIO control and low power state
- **i2c\_presence\_detector.c** The I<sup>2</sup>C Presence Detector application.

## 5 Embedded Host Example

This is an example implementation of the host read and write register functions using the STM32 SDK.

### 5.1 Register Read/Write functions

```
#include <inttypes.h>
#include <stdbool.h>
#include <stdint.h>

#include "presence_reg_protocol.h"

// Use 1000ms timeout
#define I2C_TIMEOUT_MS 1000

// The STM32 uses the i2c address shifted one position
// to the left (0x52 becomes 0xa4)
#define I2C_ADDR 0xa4

// The register address length is two bytes
#define REG_ADDRESS_LENGTH 2

// The register data length is four bytes
#define REG_DATA_LENGTH 4

/**
 * @brief Read register value over I2C
 *
 * @param[in] reg_addr The register address to read
 * @param[out] reg_data The read register data
 * @returns true if successful
 */
bool read_register(uint16_t reg_addr, uint32_t *reg_data)
{
```



```
HAL_StatusTypeDef status = HAL_OK;

uint8_t transmit_data[REG_ADDRESS_LENGTH];

transmit_data[0] = (reg_addr >> 8) & 0xff;
transmit_data[1] = (reg_addr >> 0) & 0xff;

status = HAL_I2C_Master_Transmit(&STM32_I2C_HANDLE, I2C_ADDR,
                                  transmit_data, REG_ADDRESS_LENGTH,
                                  I2C_TIMEOUT_MS);

if (status != HAL_OK)
{
    return false;
}

uint8_t receive_data[REG_DATA_LENGTH];

status = HAL_I2C_Master_Receive(&STM32_I2C_HANDLE, I2C_ADDR,
                                 receive_data, REG_DATA_LENGTH,
                                 I2C_TIMEOUT_MS);

if (status != HAL_OK)
{
    return false;
}

// Convert bytes to uint32_t
uint32_t val = receive_data[0];
val = val << 8;
val |= receive_data[1];
val = val << 8;
val |= receive_data[2];
val = val << 8;
val |= receive_data[3];
*reg_data = val;

return true;
}

/**
 * @brief Write register value over I2C
 *
 * @param[in] reg_addr The register address to write
 * @param[in] reg_data The register data to write
 * @returns true if successful
 */
bool write_register(uint16_t reg_addr, uint32_t reg_data)
{
    HAL_StatusTypeDef status = HAL_OK;

    uint8_t transmit_data[REG_ADDRESS_LENGTH + REG_DATA_LENGTH];

    // Convert uint16_t address to bytes
    transmit_data[0] = (reg_addr >> 8) & 0xff;
    transmit_data[1] = (reg_addr >> 0) & 0xff;
    // Convert uint32_t reg_data to bytes
    transmit_data[2] = (reg_data >> 24) & 0xff;
    transmit_data[3] = (reg_data >> 16) & 0xff;
    transmit_data[4] = (reg_data >> 8) & 0xff;
    transmit_data[5] = (reg_data >> 0) & 0xff;
```



```
    status = HAL_I2C_Master_Transmit(&STM32_I2C_HANDLE, I2C_ADDR,
                                     transmit_data,
                                     REG_ADDRESS_LENGTH + REG_DATA_LENGTH,
                                     I2C_TIMEOUT_MS);

    if (status != HAL_OK)
    {
        return false;
    }

    return true;
}
```

## 5.2 Detector setup functions

```
#include "presence_reg_protocol.h"

/**
 * @brief Test if configuration of detector is OK
 *
 * @returns true if successful
 */
bool configuration_ok(void)
{
    uint32_t status = 0
    if (!read_register(PRESENCE_REG_DETECTOR_STATUS_ADDRESS, &status))
    {
        //ERROR
        return false;
    }

    uint32_t config_ok_mask =
        PRESENCE_REG_DETECTOR_STATUS_FIELD_RSS_REGISTER_OK_MASK |
        PRESENCE_REG_DETECTOR_STATUS_FIELD_CONFIG_CREATE_OK_MASK |
        PRESENCE_REG_DETECTOR_STATUS_FIELD_SENSOR_CREATE_OK_MASK |
        PRESENCE_REG_DETECTOR_STATUS_FIELD_SENSOR_CALIBRATE_OK_MASK |
        PRESENCE_REG_DETECTOR_STATUS_FIELD_DETECTOR_CREATE_OK_MASK |
        PRESENCE_REG_DETECTOR_STATUS_FIELD_DETECTOR_BUFFER_OK_MASK |
        PRESENCE_REG_DETECTOR_STATUS_FIELD_SENSOR_BUFFER_OK_MASK |
        PRESENCE_REG_DETECTOR_STATUS_FIELD_CONFIG_APPLY_OK_MASK;

    if (status != config_ok_mask)
    {
        //ERROR
        return false;
    }

    return true;
}

/**
 * @brief Wait for detector not busy
 *
 * @returns true if successful
 */
bool wait_not_busy(void)
{
    do
    {
        uint32_t status = 0
    }
}
```



```
{
    if (!read_register(PRESENCE_REG_DETECTOR_STATUS_ADDRESS, &status))
    {
        //ERROR
        return false;
    }
} while((status & PRESENCE_REG_DETECTOR_STATUS_FIELD_BUSY_MASK) != 0);

return true;
}

bool example_setup_and_start(void)
{
    // Set start at 1000mm
    if (!write_register(PRESENCE_REG_START_ADDRESS, 1000))
    {
        //ERROR
        return false;
    }
    // Set end at 5000mm
    if (!write_register(PRESENCE_REG_END_ADDRESS, 5000))
    {
        //ERROR
        return false;
    }

    // Apply configuration
    if (!write_register(
        PRESENCE_REG_COMMAND_ADDRESS,
        PRESENCE_REG_COMMAND_ENUM_APPLY_CONFIGURATION))
    {
        //ERROR
        return false;
    }

    // Wait for the configuration to be done
    if (!wait_not_busy())
    {
        //ERROR
        return false;
    }

    // Test if configuration of detector was OK
    if (!configuration_ok())
    {
        //ERROR
        return false;
    }

    // Start detector
    if (!write_register(PRESENCE_REG_COMMAND_ADDRESS,
        PRESENCE_REG_COMMAND_ENUM_START_DETECTOR))
    {
        //ERROR
        return false;
    }

    // Wait for command be done
    if (!wait_not_busy())
    {
```



```
        //ERROR
        return false;
    }

    // Read detector result
    uint32_t result;
    if (!read_register(PRESENCE_REG_PRESENCE_RESULT_ADDRESS, &result))
    {
        //ERROR
        return false;
    }

    // Was presence detected?

    bool presence_detected = (result &
        PRESENCE_REG_PRESENCE_RESULT_FIELD_PRESENCE_DETECTED_MASK) != 0;
    bool presence_detected_sticky = (result &
        PRESENCE_REG_PRESENCE_RESULT_FIELD_PRESENCE_DETECTED_STICKY_MASK) !=
        0;

    // Print peak if found
    if (presence_detected || presence_detected_sticky)
    {
        uint32_t presence_distance_mm;
        if (read_register(PRESENCE_REG_PRESENCE_DISTANCE_ADDRESS, &
            presence_distance_mm))
        {
            printf("Presence detected at distance: %" PRIu32 " mm\n",
                presence_distance_mm);
        }
        else
        {
            //ERROR
            return false;
        }
    }
    else
    {
        printf("No presence detected\n");
    }

    return true;
}
```



## 6 Registers

### 6.1 Register Map

Address	Register Name	Type
0x0000	Version	Read Only
0x0001	Protocol Status	Read Only
0x0002	Measure Counter	Read Only
0x0003	Detector Status	Read Only
0x0010	Presence Result	Read Only
0x0011	Presence Distance	Read Only
0x0012	Intra Presence Score	Read Only
0x0013	Inter Presence Score	Read Only
0x0020	Presence Actual Frame Rate	Read Only
0x0040	Sweeps Per Frame	Read / Write
0x0041	Inter Frame Presence Timeout	Read / Write
0x0043	Intra Detection Enabled	Read / Write
0x0044	Inter Detection Enabled	Read / Write
0x0045	Frame Rate	Read / Write
0x0046	Intra Detection Threshold	Read / Write
0x0047	Inter Detection Threshold	Read / Write
0x0048	Inter Frame Deviation Time Const	Read / Write
0x0049	Inter Frame Fast Cutoff	Read / Write
0x004a	Inter Frame Slow Cutoff	Read / Write
0x004b	Intra Frame Time Const	Read / Write
0x004c	Intra Output Time Const	Read / Write
0x004d	Inter Output Time Const	Read / Write
0x004e	Auto Profile Enabled	Read / Write
0x004f	Auto Step Length Enabled	Read / Write
0x0050	Manual Profile	Read / Write
0x0051	Manual Step Length	Read / Write
0x0052	Start	Read / Write
0x0053	End	Read / Write
0x0054	Reset Filters On Prepare	Read / Write
0x0055	Hwaas	Read / Write
0x0056	Automatic Subsweeps	Read / Write
0x0057	Signal Quality	Read / Write
0x0080	Detection On Gpio	Read / Write
0x0100	Command	Write Only
0xffff	Application Id	Read Only

### 6.2 Register Descriptions

#### 6.2.1 Version

<b>Address</b>	0x0000
<b>Access</b>	Read Only
<b>Register Type</b>	field
<b>Description</b>	Get the RSS version.

Bitfield	Pos	Width	Mask
MAJOR	16	16	0xffff0000
MINOR	8	8	0x0000ff00
PATCH	0	8	0x000000ff

**MAJOR** - Major version number

**MINOR** - Minor version number



**PATCH** - Patch version number

## 6.2.2 Protocol Status

<b>Address</b>	0x0001
<b>Access</b>	Read Only
<b>Register Type</b>	field
<b>Description</b>	Get protocol error flags.

Bitfield	Pos	Width	Mask
PROTOCOL_STATE_ERROR	0	1	0x00000001
PACKET_LENGTH_ERROR	1	1	0x00000002
ADDRESS_ERROR	2	1	0x00000004
WRITE_FAILED	3	1	0x00000008
WRITE_TO_READ_ONLY	4	1	0x00000010

**PROTOCOL\_STATE\_ERROR** - Protocol state error

**PACKET\_LENGTH\_ERROR** - Packet length error

**ADDRESS\_ERROR** - Register address error

**WRITE\_FAILED** - Write register failed

**WRITE\_TO\_READ\_ONLY** - Write to read only register

## 6.2.3 Measure Counter

<b>Address</b>	0x0002
<b>Access</b>	Read Only
<b>Register Type</b>	uint
<b>Description</b>	Get the measure counter, the number of measurements performed since restart.

## 6.2.4 Detector Status

<b>Address</b>	0x0003
<b>Access</b>	Read Only
<b>Register Type</b>	field
<b>Description</b>	Get detector status flags.

Bitfield	Pos	Width	Mask
RSS_REGISTER_OK	0	1	0x00000001
CONFIG_CREATE_OK	1	1	0x00000002
SENSOR_CREATE_OK	2	1	0x00000004
SENSOR_CALIBRATE_OK	3	1	0x00000008
DETECTOR_CREATE_OK	4	1	0x00000010
DETECTOR_BUFFER_OK	5	1	0x00000020
SENSOR_BUFFER_OK	6	1	0x00000040
CONFIG_APPLY_OK	7	1	0x00000080
RSS_REGISTER_ERROR	16	1	0x00010000
CONFIG_CREATE_ERROR	17	1	0x00020000
SENSOR_CREATE_ERROR	18	1	0x00040000
SENSOR_CALIBRATE_ERROR	19	1	0x00080000
DETECTOR_CREATE_ERROR	20	1	0x00100000
DETECTOR_BUFFER_ERROR	21	1	0x00200000
SENSOR_BUFFER_ERROR	22	1	0x00400000
CONFIG_APPLY_ERROR	23	1	0x00800000



DETECTOR_ERROR	28	1	0x10000000
BUSY	31	1	0x80000000

**RSS\_REGISTER\_OK** - RSS register OK

**CONFIG\_CREATE\_OK** - Configuration create OK

**SENSOR\_CREATE\_OK** - Sensor create OK

**SENSOR\_CALIBRATE\_OK** - Sensor calibrate OK

**DETECTOR\_CREATE\_OK** - Detector create OK

**DETECTOR\_BUFFER\_OK** - Detector get buffer size OK

**SENSOR\_BUFFER\_OK** - Memory allocation of sensor buffer OK

**CONFIG\_APPLY\_OK** - Detector configuration apply OK

**RSS\_REGISTER\_ERROR** - RSS register error

**CONFIG\_CREATE\_ERROR** - Configuration create error

**SENSOR\_CREATE\_ERROR** - Sensor create error

**SENSOR\_CALIBRATE\_ERROR** - Sensor calibrate error

**DETECTOR\_CREATE\_ERROR** - Detector create error

**DETECTOR\_BUFFER\_ERROR** - Detector get buffer size error

**SENSOR\_BUFFER\_ERROR** - Memory allocation of sensor buffer error

**CONFIG\_APPLY\_ERROR** - Detector configuration apply error

**DETECTOR\_ERROR** - Detector error occurred, restart necessary

**BUSY** - Detector busy

### 6.2.5 Presence Result

<b>Address</b>	0x0010
<b>Access</b>	Read Only
<b>Register Type</b>	field
<b>Description</b>	The result from the presence detector.

Bitfield	Pos	Width	Mask
PRESENCE_DETECTED	0	1	0x00000001
PRESENCE_DETECTED_STICKY	1	1	0x00000002
DETECTOR_ERROR	15	1	0x00008000
TEMPERATURE	16	16	0xffff0000

**PRESENCE\_DETECTED** - Presence detected

**PRESENCE\_DETECTED\_STICKY** - Presence detected, sticky bit with clear on read

**DETECTOR\_ERROR** - The presence detector failed

**TEMPERATURE** - Temperature in sensor during measurement (in degree Celsius). Note that it has poor absolute accuracy and should only be used for relative temperature measurements.

### 6.2.6 Presence Distance

<b>Address</b>	0x0011
<b>Access</b>	Read Only
<b>Register Type</b>	uint



<b>Unit</b>	mm
<b>Description</b>	The distance, in millimeters, for the detected presence

### 6.2.7 Intra Presence Score

<b>Address</b>	0x0012
<b>Access</b>	Read Only
<b>Register Type</b>	uint
<b>Description</b>	A measure of the amount of fast motion detected. Just like the configuration register <code>intra_detection_threshold</code> , this field is a factor 1000 larger than the RSS value.

### 6.2.8 Inter Presence Score

<b>Address</b>	0x0013
<b>Access</b>	Read Only
<b>Register Type</b>	uint
<b>Description</b>	A measure of the amount of slow motion detected. Just like the configuration register <code>inter_detection_threshold</code> , this field is a factor 1000 larger than the RSS value.

### 6.2.9 Presence Actual Frame Rate

<b>Address</b>	0x0020
<b>Access</b>	Read Only
<b>Register Type</b>	uint
<b>Unit</b>	mHz
<b>Description</b>	The actual frame rate of the presence detector.

### 6.2.10 Sweeps Per Frame

<b>Address</b>	0x0040
<b>Access</b>	Read / Write
<b>Register Type</b>	uint
<b>Description</b>	The number of sweeps that will be captured in each frame (measurement).
<b>Default Value</b>	16

### 6.2.11 Inter Frame Presence Timeout

<b>Address</b>	0x0041
<b>Access</b>	Read / Write
<b>Register Type</b>	uint
<b>Description</b>	Number of seconds the inter-frame presence score needs to decrease before exponential scaling starts for faster decline. Should be between 0 and 30 where 0 means no timeout. Note:
<b>Default Value</b>	3

### 6.2.12 Intra Detection Enabled

<b>Address</b>	0x0043
<b>Access</b>	Read / Write
<b>Register Type</b>	bool
<b>Description</b>	Enable to detect faster movements inside frames.
<b>Default Value</b>	True



### 6.2.13 Inter Detection Enabled

<b>Address</b>	0x0044
<b>Access</b>	Read / Write
<b>Register Type</b>	bool
<b>Description</b>	Enable to detect slower movements between frames.
<b>Default Value</b>	True

### 6.2.14 Frame Rate

<b>Address</b>	0x0045
<b>Access</b>	Read / Write
<b>Register Type</b>	uint
<b>Unit</b>	mHz
<b>Description</b>	The presence detector frame rate. Note: This value is a factor 1000 larger than the RSS value.
<b>Default Value</b>	12000

### 6.2.15 Intra Detection Threshold

<b>Address</b>	0x0046
<b>Access</b>	Read / Write
<b>Register Type</b>	uint
<b>Description</b>	The threshold for detecting faster movements inside frames. Note: This value is a factor 1000 larger than the RSS value.
<b>Default Value</b>	1300

### 6.2.16 Inter Detection Threshold

<b>Address</b>	0x0047
<b>Access</b>	Read / Write
<b>Register Type</b>	uint
<b>Description</b>	This is the threshold for detecting slower movements between frames. Note: This value is a factor 1000 larger than the RSS value.
<b>Default Value</b>	1000

### 6.2.17 Inter Frame Deviation Time Const

<b>Address</b>	0x0048
<b>Access</b>	Read / Write
<b>Register Type</b>	uint
<b>Unit</b>	ms
<b>Description</b>	The time constant of the low pass filter for the inter-frame deviation between fast and slow. Note: This value is a factor 1000 larger than the RSS value.
<b>Default Value</b>	500

### 6.2.18 Inter Frame Fast Cutoff

<b>Address</b>	0x0049
<b>Access</b>	Read / Write
<b>Register Type</b>	uint
<b>Unit</b>	mHz
<b>Description</b>	The cutoff frequency of the low pass filter for the fast filtered absolute sweep mean. Note: This value is a factor 1000 larger than the RSS value.
<b>Default Value</b>	6000



### 6.2.19 Inter Frame Slow Cutoff

<b>Address</b>	0x004a
<b>Access</b>	Read / Write
<b>Register Type</b>	uint
<b>Unit</b>	mHz
<b>Description</b>	The cutoff frequency of the low pass filter for the slow filtered absolute sweep mean. Note: This value is a factor 1000 larger than the RSS value.
<b>Default Value</b>	200

### 6.2.20 Intra Frame Time Const

<b>Address</b>	0x004b
<b>Access</b>	Read / Write
<b>Register Type</b>	uint
<b>Unit</b>	ms
<b>Description</b>	The time constant for the depthwise filtering in the intra-frame part. Note: This value is a factor 1000 larger than the RSS value.
<b>Default Value</b>	150

### 6.2.21 Intra Output Time Const

<b>Address</b>	0x004c
<b>Access</b>	Read / Write
<b>Register Type</b>	uint
<b>Unit</b>	ms
<b>Description</b>	The time constant for the output in the intra-frame part. Note: This value is a factor 1000 larger than the RSS value.
<b>Default Value</b>	300

### 6.2.22 Inter Output Time Const

<b>Address</b>	0x004d
<b>Access</b>	Read / Write
<b>Register Type</b>	uint
<b>Unit</b>	ms
<b>Description</b>	The time constant for the output in the inter-frame part. Note: This value is a factor 1000 larger than the RSS value.
<b>Default Value</b>	2000

### 6.2.23 Auto Profile Enabled

<b>Address</b>	0x004e
<b>Access</b>	Read / Write
<b>Register Type</b>	bool
<b>Description</b>	Enable/Disable automatic selection of profile based on start point of measurement.
<b>Default Value</b>	True

### 6.2.24 Auto Step Length Enabled

<b>Address</b>	0x004f
<b>Access</b>	Read / Write
<b>Register Type</b>	bool
<b>Description</b>	Enable/Disable automatic selection of step length based on the profile.
<b>Default Value</b>	True



### 6.2.25 Manual Profile

<b>Address</b>	0x0050
<b>Access</b>	Read / Write
<b>Register Type</b>	enum
<b>Description</b>	The profile to use. The profile will only be used if profile auto selection was disabled.
<b>Default Value</b>	PROFILE4

Enum	Value
PROFILE1	1
PROFILE2	2
PROFILE3	3
PROFILE4	4
PROFILE5	5

**PROFILE1** - Profile 1

**PROFILE2** - Profile 2

**PROFILE3** - Profile 3

**PROFILE4** - Profile 4

**PROFILE5** - Profile 5

### 6.2.26 Manual Step Length

<b>Address</b>	0x0051
<b>Access</b>	Read / Write
<b>Register Type</b>	uint
<b>Description</b>	The number of steps between each data point. The manual step length will only be used if step length auto selection was disabled.
<b>Default Value</b>	72

### 6.2.27 Start

<b>Address</b>	0x0052
<b>Access</b>	Read / Write
<b>Register Type</b>	uint
<b>Unit</b>	mm
<b>Description</b>	The start point of measurement interval in millimeters. Note: This value is a factor 1000 larger than the RSS value.
<b>Default Value</b>	300

### 6.2.28 End

<b>Address</b>	0x0053
<b>Access</b>	Read / Write
<b>Register Type</b>	uint
<b>Unit</b>	mm
<b>Description</b>	The end point of measurement interval in millimeters. Note: This value is a factor 1000 larger than the RSS value.
<b>Default Value</b>	2500

### 6.2.29 Reset Filters On Prepare

<b>Address</b>	0x0054
----------------	--------



<b>Access</b>	Read / Write
<b>Register Type</b>	bool
<b>Description</b>	Enable/Disable reset of the presence filters during start/restart.
<b>Default Value</b>	True

### 6.2.30 Hwaas

<b>Address</b>	0x0055
<b>Access</b>	Read / Write
<b>Register Type</b>	uint
<b>Description</b>	The hardware accelerated average samples (HWAAS).
<b>Default Value</b>	32

### 6.2.31 Automatic Subsweeps

<b>Address</b>	0x0056
<b>Access</b>	Read / Write
<b>Register Type</b>	bool
<b>Description</b>	Enable/Disable use of subsweeps.
<b>Default Value</b>	True

### 6.2.32 Signal Quality

<b>Address</b>	0x0057
<b>Access</b>	Read / Write
<b>Register Type</b>	int
<b>Description</b>	Signal quality.
<b>Default Value</b>	15000

### 6.2.33 Detection On Gpio

<b>Address</b>	0x0080
<b>Access</b>	Read / Write
<b>Register Type</b>	bool
<b>Description</b>	Output presence detection on generic gpio
<b>Default Value</b>	False

### 6.2.34 Command

<b>Address</b>	0x0100
<b>Access</b>	Write Only
<b>Register Type</b>	enum
<b>Description</b>	Execute command.

Enum	Value
APPLY_CONFIGURATION	1
START_DETECTOR	2
STOP_DETECTOR	3
ENABLE_UART_LOGS	32
DISABLE_UART_LOGS	33
LOG_CONFIGURATION	34
RESET_MODULE	1381192737



**APPLY\_CONFIGURATION** - Apply the configuration

**START\_DETECTOR** - Start the presence detector

**STOP\_DETECTOR** - Stop the presence detector

**ENABLE\_UART\_LOGS** - DEBUG: Enable UART Logs

**DISABLE\_UART\_LOGS** - DEBUG: Disable UART Logs

**LOG\_CONFIGURATION** - DEBUG: Print detector configuration to UART

**RESET\_MODULE** - Reset module, needed to make a new configuration

### 6.2.35 Application Id

<b>Address</b>	0xffff
<b>Access</b>	Read Only
<b>Register Type</b>	enum
<b>Description</b>	The application id register.

<b>Enum</b>	<b>Value</b>
DISTANCE_DETECTOR	1
PRESENCE_DETECTOR	2
REF_APP_BREATHING	3
EXAMPLE_CARGO	4
EXAMPLE_DOOR_OPEN_DETECTION	5

**DISTANCE\_DETECTOR** - Distance Detector Application

**PRESENCE\_DETECTOR** - Presence Detector Application

**REF\_APP\_BREATHING** - Breathing Reference Application

**EXAMPLE\_CARGO** - Cargo Example Application

**EXAMPLE\_DOOR\_OPEN\_DETECTION** - Door Open Detection Example Application



## 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.

