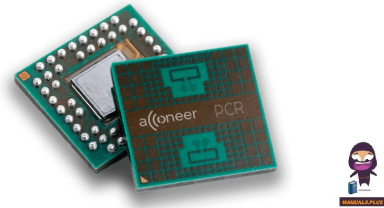



aconeer A121 I2C Distance Detector



aconeer A121 I2C Distance Detector User Guide

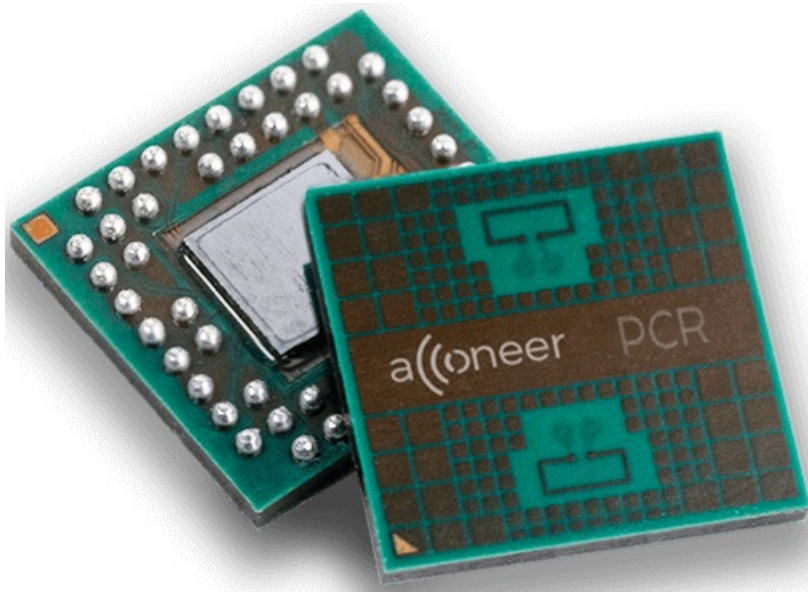
[Home](#) » [aconeer](#) » aconeer A121 I2C Distance Detector User Guide 

Contents

- 1 [aconeer A121 I2C Distance Detector](#)
- 2 [Product Information](#)
- 3 [Product Usage Instructions](#)
- 4 [Documentation Overview](#)
- 5 [Detector Application](#)
- 6 [Register Protocol](#)
- 7 [File Structure](#)
- 8 [Embedded Host Example](#)
- 9 [Registers](#)
- 10 [Disclaimer](#)
- 11 [Documents / Resources](#)
 - 11.1 [References](#)



aconeer A121 I2C Distance Detector



Product Information

- **Specifications:**
 - **Product:** I2C Distance Detector
 - **Manufacturer:** Acconeer AB
 - **Version:** a121-v1.3.0
 - **Release Date:** October 6, 2023

The I2C Distance Detector is a device designed to measure distances using the I2C communication protocol. It provides accurate distance measurements and can be used for various applications.

Product Usage Instructions

- **Read Detector Status:**
 - To read the detector status, follow the instructions provided in the user manual.
- **Writing a Command:**
 - To send commands to the detector, use the specified protocol for writing commands as outlined in the user guide.
- **Setup and Measure:**
 - Follow the setup instructions to configure the detector and initiate distance measurement.
- **Advanced Usage:**
 - **Apply Configuration and Calibration Separately:**
 - For advanced users, it is possible to apply configuration settings and calibration separately for precise measurements.
 - **Re-calibration:**
 - If needed, the detector can be recalibrated following the provided guidelines in the user manual.
 - **Measure on Wake Up Mode:**
 - In wake-up mode, the detector can perform measurements upon activation. Refer to the manual for detailed instructions.
 - **Debug UART Logs:**
 - View debug logs via the UART interface for troubleshooting and analysis purposes.

- **Reset Module:**

- To reset the module, use the recommended procedure to ensure proper functionality.

FAQ:

- **Q: How often should the detector be recalibrated?**

- **A:** The detector should be recalibrated periodically as per the usage and environmental conditions. It is recommended to recalibrate if inconsistencies are observed in measurements.

- **Q: Can the detector measure distances in varying lighting conditions?**

- **A:** The detector's performance may vary in different lighting conditions. For optimal results, use the detector in controlled lighting environments.

Documentation Overview

Acconeer SDK Documentation Overview

To better understand what SDK document to use, a summary of the documents is shown in the table below.

Table 1: SDK document overview.

Name	Description	When to use
<i>RSS API documentation (HTML)</i>		
RSS-API	The complete C API documentation.	<ul style="list-style-type: none">– RSS application implementation– Understanding RSS API functions
<i>User guides (PDF)</i>		
A121 Assembly Test	Describes the Acconeer assembly test functionality.	<ul style="list-style-type: none">– Bring-up of HW/SW– Production test implementation
A121 Breathing Reference Application	Describes the functionality of the Breathing Reference Application.	<ul style="list-style-type: none">– Working with the Breathing Reference Application
A121 Distance Detector	Describes usage and algorithms of the Distance Detector.	<ul style="list-style-type: none">– Working with the Distance Detector
A121 SW Integration	Describes how to implement each integration function needed to use the Acconeer sensor.	<ul style="list-style-type: none">– SW implementation of custom HW integration
A121 Presence Detector	Describes usage and algorithms of the Presence Detector.	<ul style="list-style-type: none">– 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 STM32CubeIDE	Describes the flow of taking an Acconeer SDK and integrating it 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
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
Handbook (PDF)		
Handbook	Describes different aspects of the Acconeer offers, for example, radar principles and how to configure	<ul style="list-style-type: none"> – To understand the Acconeer sensor – Use case evaluation
Readme (txt)		
[README	Various target-specific information and links	– After SDK download

Detector Application

I2C Distance Detector Application

- The I2C Distance Detector is an application that implements the Acconeer Distance Detector with a register-based I2C interface.
- The functionality of the distance detector is described in A121 Distance Detector User Guide.pdf or [Acconeer Docs](#).
- **Note:** Some of the registers like start and end have a different unit in the I2C Distance Detector, millimetres instead of meters, to make it easier to handle the register values as integers.

Usage

- The module must be ready before the host starts I2C communication.
- The module will enter a ready state by following this procedure.

- Set the wake-up pin of the module HIGH.
- Wait for the module to be ready, this is indicated by the MCU INT pin being HIGH.
- Start I2C communication.
- The module will enter a low-power state by following this procedure.
- Wait for the module to be ready, this is indicated by the MCU INT pin being HIGH.
- Set the UP pin of the module LOW.
- Wait for the ready signal, the MCU INT pin, to become LOW.

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.

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.

Setup and Measure

- Before the module can perform distance measurements it must be configured and calibrated. The following steps is an example of how this can be achieved.
- **Note:** The configuration parameters can not be changed after a APPLY CONFIG AND CALIBRATE or 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 the Detector Status register and verify that neither Busy nor Error bits are set.
- Write configuration to configuration registers, for example, the Start register and the End register.
- Write APPLY CONFIG AND CALIBRATE to the Command register.
- Poll Detector Status until Busy bit is cleared.
- Verify that no Error bits are set in the Detector Status register.
- Write MEASURE DISTANCE to the Command register.
- Poll Detector Status until Busy bit is cleared.
- Verify that no Error bits are set in the Detector Status register.
- Read the Detector Result register
 - If MEASURE DISTANCE ERROR is set the measurement failed.
 - If CALIBRATION NEEDED is set the sensor needs to be re-calibrated with the RECALIBRATE command.
 - The number of peak distances detected can be read in the NUM DISTANCES field.
- Read PeakX Distance and PeakX Strength registers depending on how many distances were detected.
- The module is ready for a new MEASURE DISTANCE command.

Advanced Usage

• Apply Configuration and Calibration separately

- Some use-cases require control over when the system is calibrated, therefore the Apply Configuration and Calibrate can be performed as individual steps.
- Power on module
- Read the Detector Status register and verify that neither Busy nor Error bits are set.
- Write configuration to configuration registers, for example, the Start register and End register.
- Write APPLY CONFIGURATION to the Command register
- Poll Detector Status until Busy bit is cleared.
- Verify that no Error bits are set in the Detector Status register.
- Write CALIBRATE to Command Register
- Poll Detector Status until Busy bit is cleared.
- Verify that no Error bits are set in the Detector Status register.
- The module is ready for a MEASURE DISTANCE command.

• Re-calibration

- Re-calibration must be done as soon as the CALIBRATION NEEDED bit is set in the Detector Result register.
- Re-calibration is performed by writing RECALIBRATE to the Command register.

• Measure on Wake Up Mode

- Measure on Wake Up mode can be enabled by writing a non-zero value to the Measure On Wakeup register. When
- Measure on Wake Up is enabled, the module will perform a distance measurement every time it is woken by the
- WAKE UP pin. The measurement will be ready when the MCU INT pin becomes HIGH.

• 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.

• Reset Module

- The module can be restarted by writing RESET MODULE to the Command register.
- After the restart, the detector must be configured again.

Register Protocol

I2C Slave Address

- The default slave address is 0x52.

Protocol Byte Order

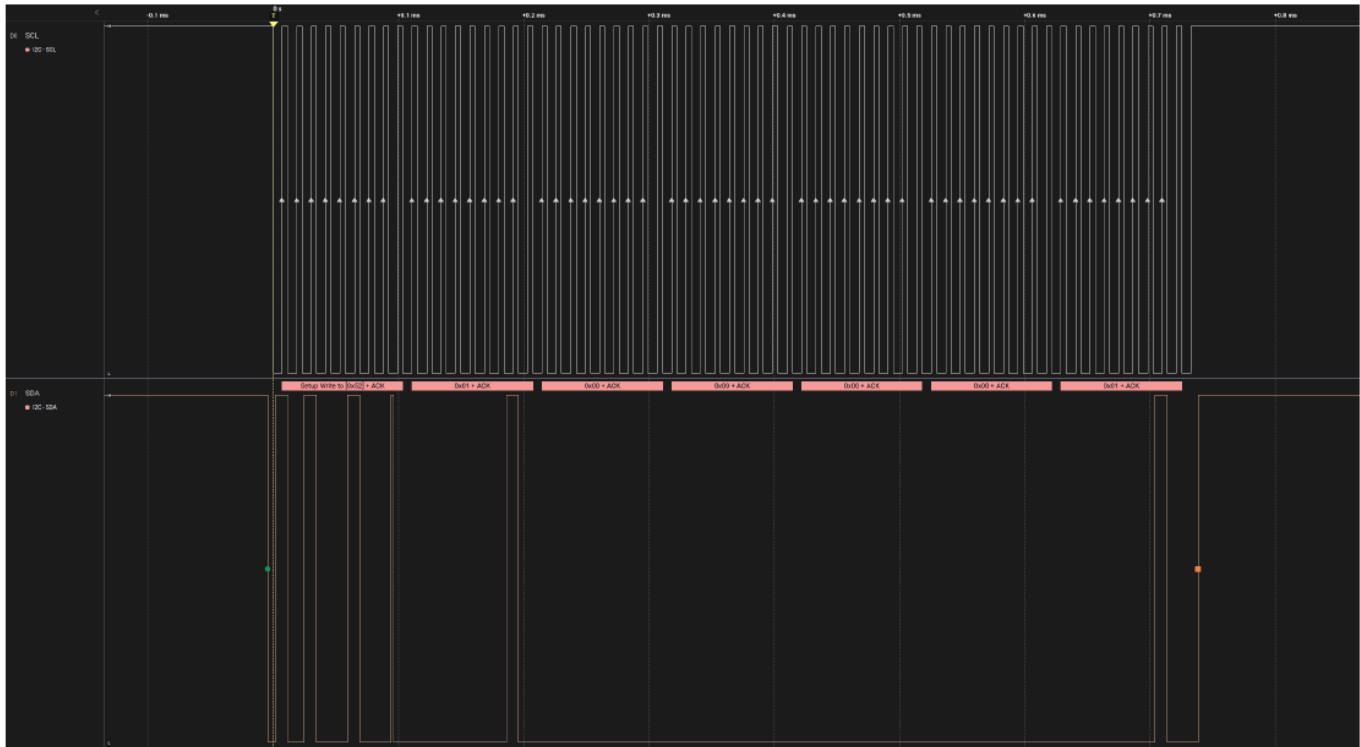
- Both register address, 16-bit, and register data, 32-bit, are sent in big-endian byte order.

I2C Write Register(s)

- A write register operation consists of an I2C write of two address bytes and four data bytes for each register to write.
- Several registers can be written in the same I2C 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.

For example operation, write 0x11223344 to address 0x0025.

Description	Data
I ² 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 ² C Stop Condition	



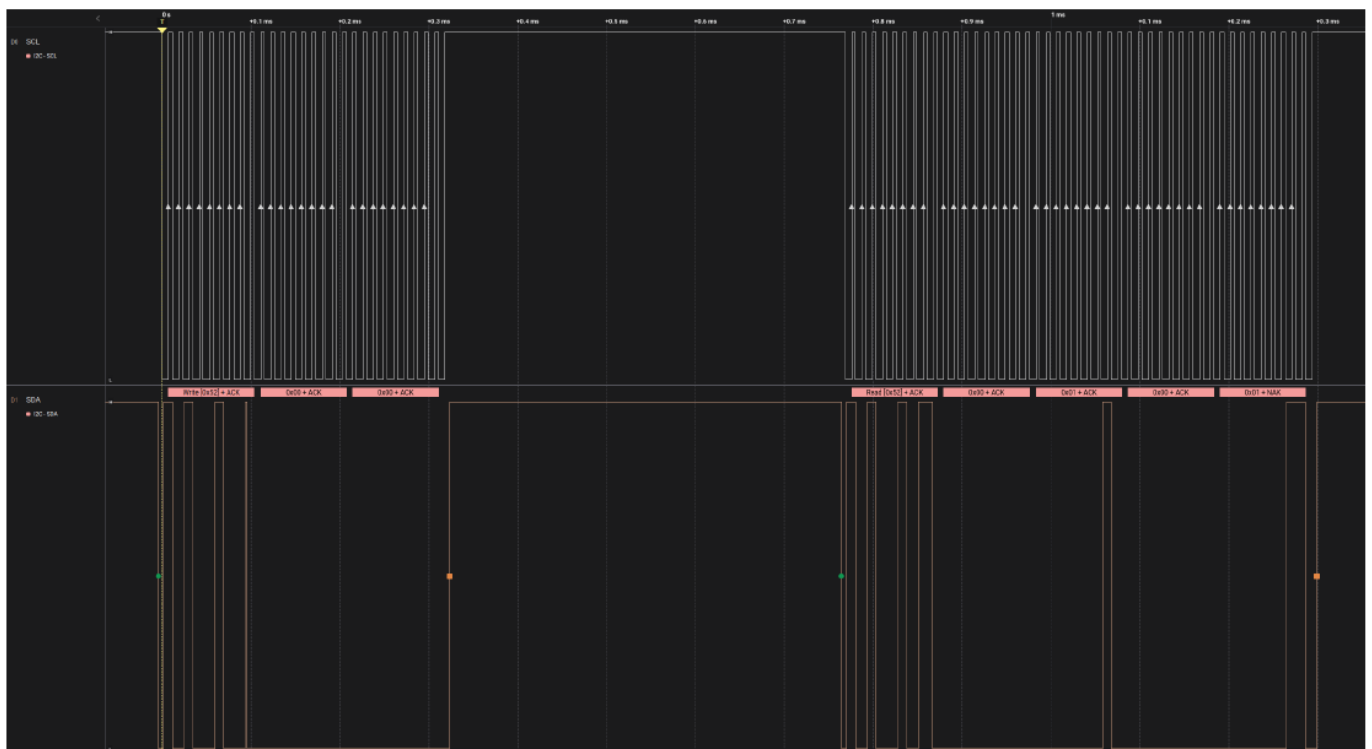
Example Waveform: Write register with address 0x0100, the data sent from the master to the slave is 0x00000001

I2C Read Register(s)

- A read register operation consists of an I2C write of two address bytes followed by an I2C read of four data bytes for each register to read. Several registers can be read in the same I2C 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.

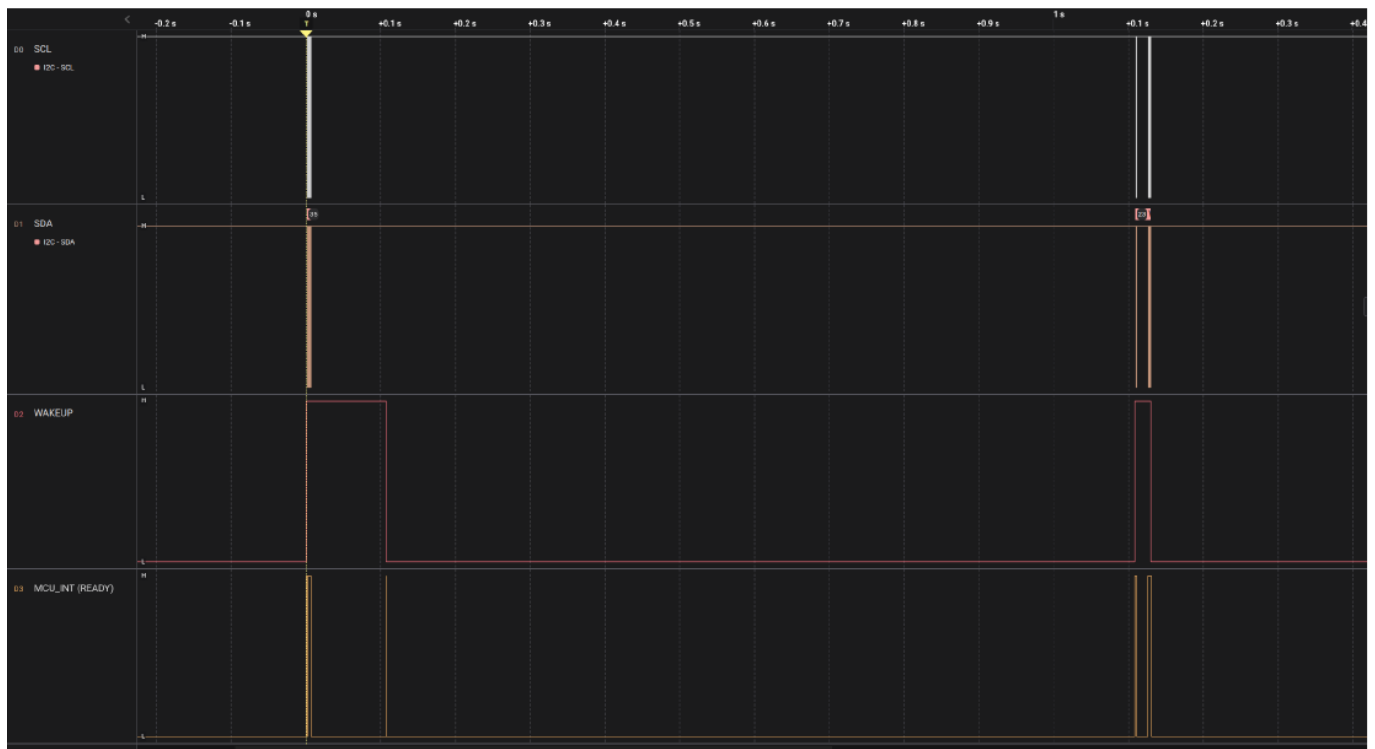
Description	Data
I ² C Start Condition	
Slave Address + Write	0x52 + W
Address to slave [15:8]	0x00
Address to slave [7:0]	0x03
I ² C Stop Condition	
I ² C Start Condition	
Slave Address + Read	0x52 + R
Data from slave [31:24]	0x12
Data from slave [23:16]	0x34
Data from slave [15:8]	0x56
Data from slave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



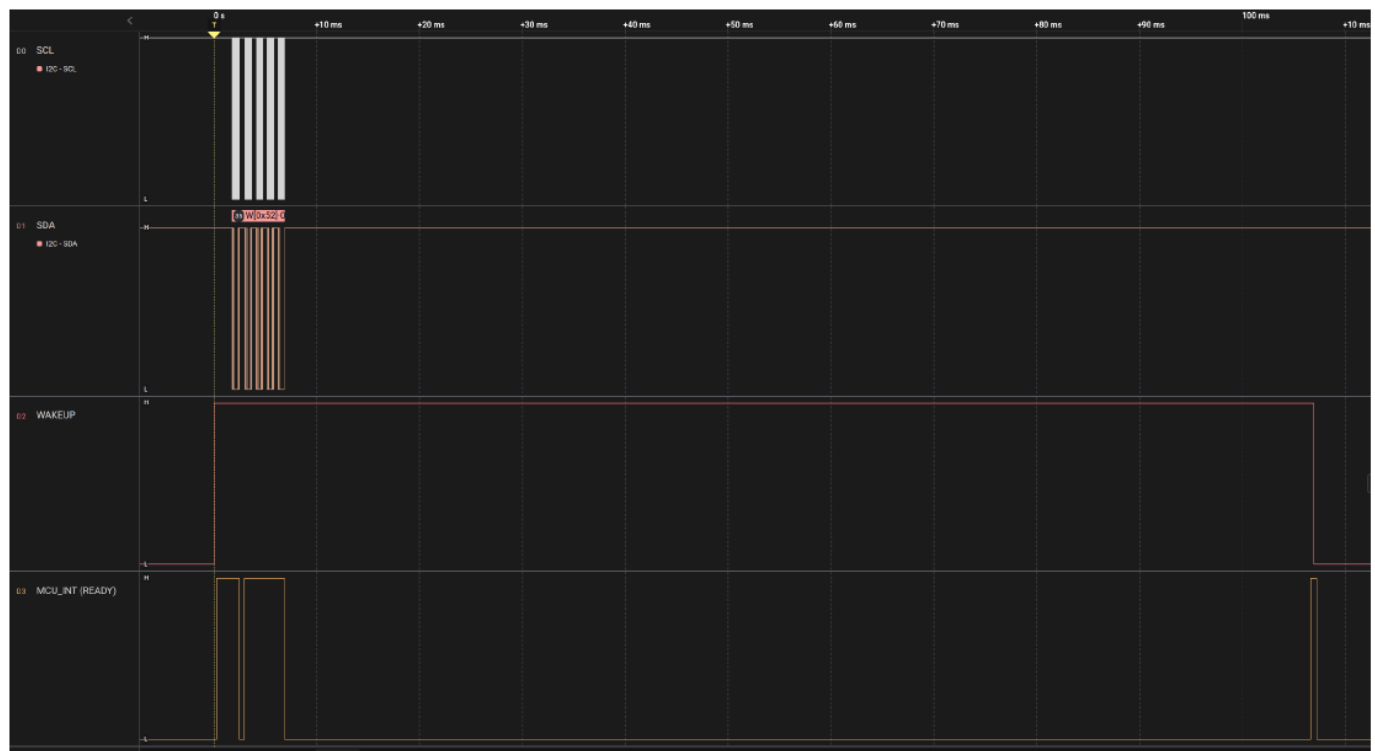
Example Waveform: Read register with address 0, the data sent from the slave to the master is 0x00010001

Register Protocol – Low Power Mode I2C Communication with Low Power Mode

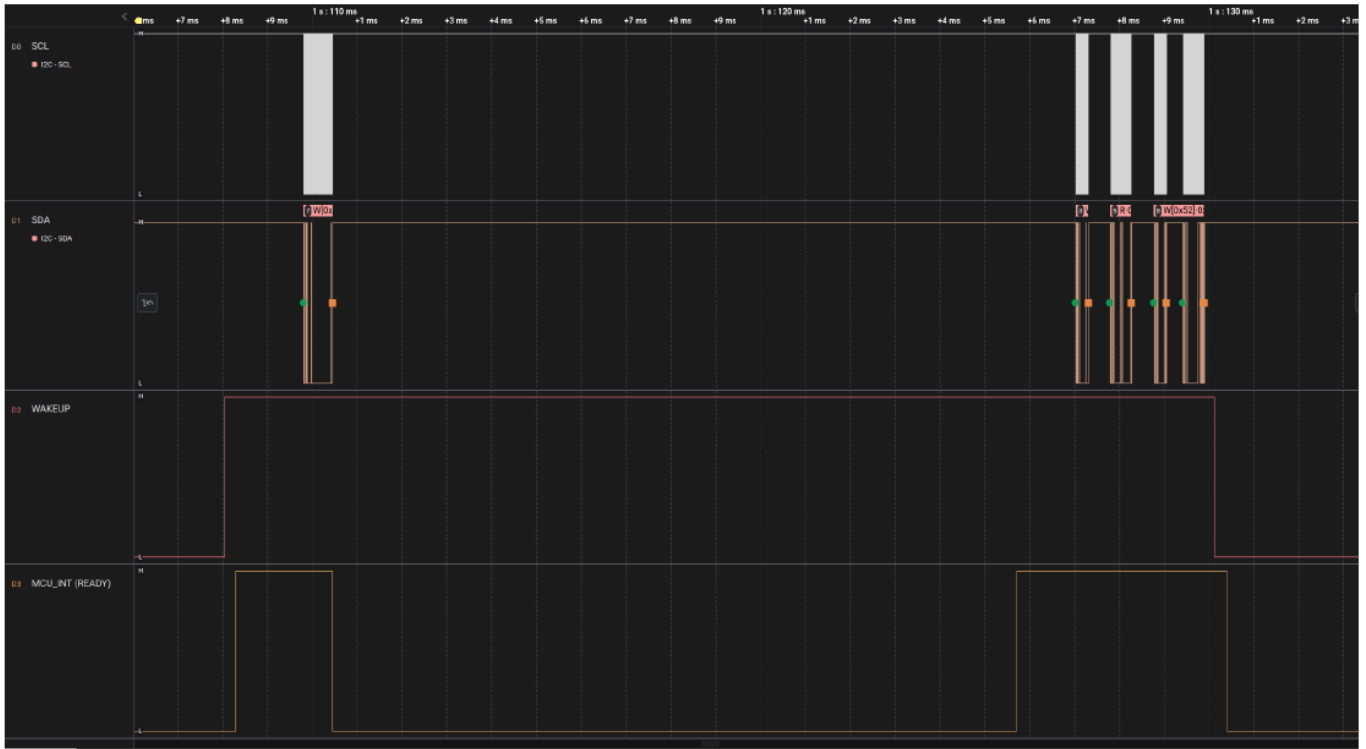
Low power example



Low Power Example: Wake up, Setup Distance Detector, Power down, Wait 1s, Wake up, Measure distance, Power down

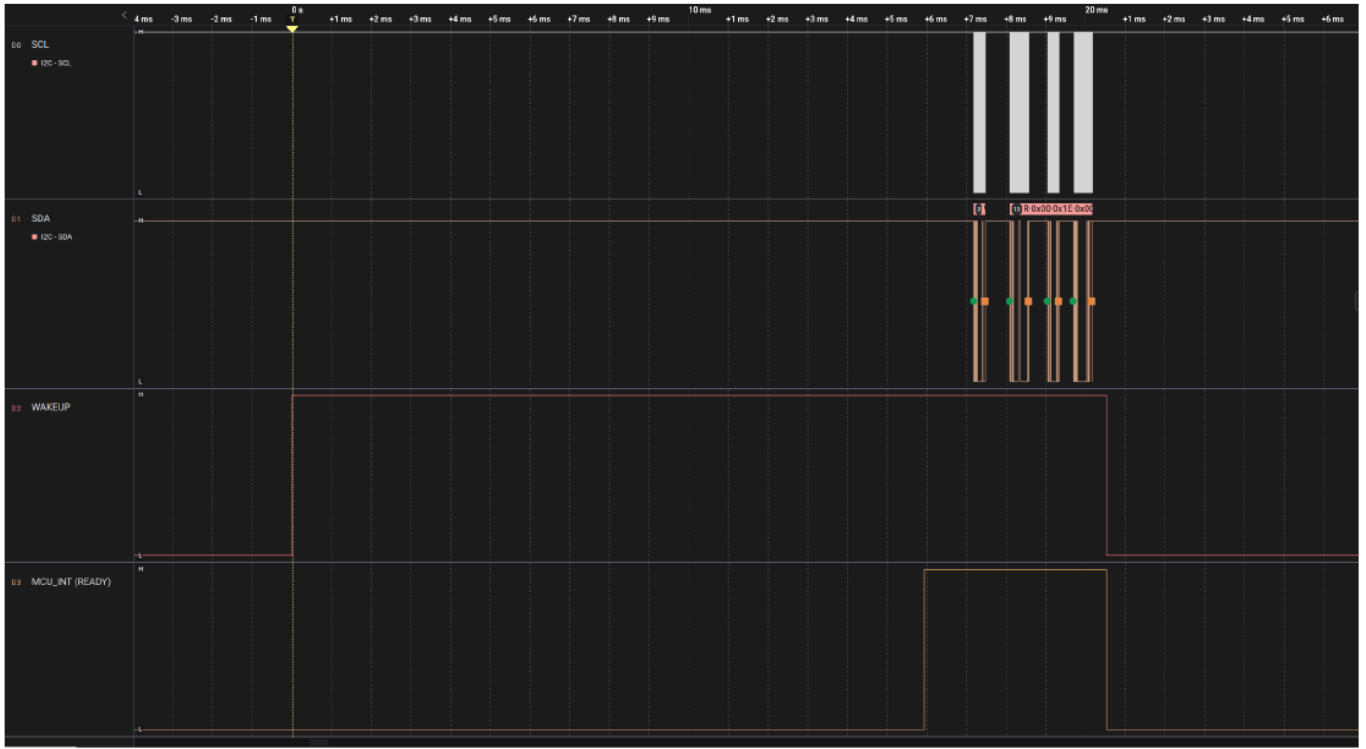


Low Power Example: Magnification of Wake up, Setup Distance Detector, Power down



Low Power Example: Magnification of Wake up, Measure distance, Power down

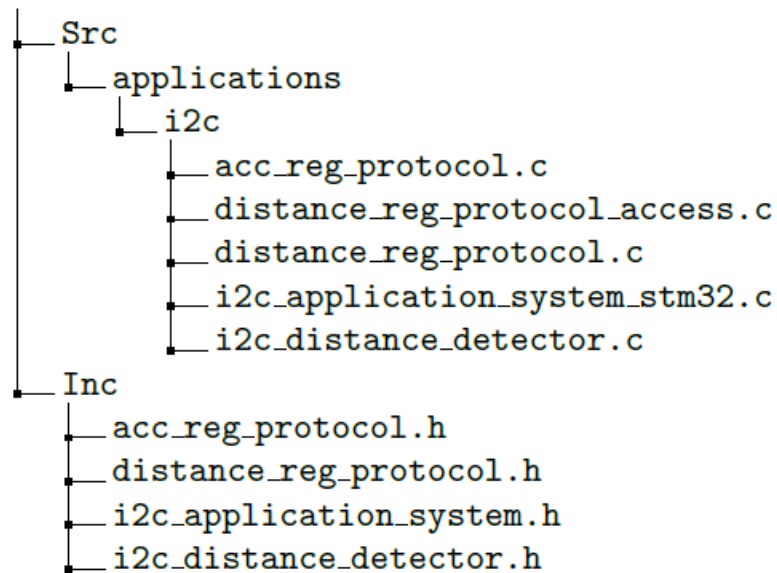
Low power example with 'Measure on wake up'



Measure on Wake Up Example: Magnification of Wake up, Measure on wake up, Power down

File Structure

The I2C Distance Detector application consists of the following files.



- `acc_reg_protocol.c` A generic protocol handler implementation.
- `distance_reg_protocol.c` The specific register protocol setup for the I2C Distance Detector.
- `distance_reg_protocol_access.c` The register read and write access functions for the I2C Distance Detector.
- `i2c_application_system_stm32.c` System functions, such as I2C handling, GPIO control and low power state
- `i2c_distance_detector.c` The I2C Distance Detector application.

Embedded Host Example

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

Register Read/Write functions

```

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

#include "distance_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;
}

```

Detector setup functions

```

#include "distance_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(DISTANCE_REG_DETECTOR_STATUS_ADDRESS, &status))
    {
        //ERROR
        return false;
    }

    uint32_t config_ok_mask =
        DISTANCE_REG_DETECTOR_STATUS_FIELD_RSS_REGISTER_OK_MASK |
        DISTANCE_REG_DETECTOR_STATUS_FIELD_CONFIG_CREATE_OK_MASK |
        DISTANCE_REG_DETECTOR_STATUS_FIELD_SENSOR_CREATE_OK_MASK |
        DISTANCE_REG_DETECTOR_STATUS_FIELD_DETECTOR_CREATE_OK_MASK |
        DISTANCE_REG_DETECTOR_STATUS_FIELD_DETECTOR_BUFFER_OK_MASK |
        DISTANCE_REG_DETECTOR_STATUS_FIELD_SENSOR_BUFFER_OK_MASK |
        DISTANCE_REG_DETECTOR_STATUS_FIELD_CALIBRATION_BUFFER_OK_MASK |
        DISTANCE_REG_DETECTOR_STATUS_FIELD_CONFIG_APPLY_OK_MASK |
        DISTANCE_REG_DETECTOR_STATUS_FIELD_SENSOR_CALIBRATE_OK_MASK |
        DISTANCE_REG_DETECTOR_STATUS_FIELD_DETECTOR_CALIBRATE_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)
{

```

```

uint32_t status = 0
do
{
    if (!read_register(DISTANCE_REG_DETECTOR_STATUS_ADDRESS, &status))
    {
        //ERROR
        return false;
    }
} while((status & DISTANCE_REG_DETECTOR_STATUS_FIELD_BUSY_MASK) != 0);

return true;
}

bool example_setup_and_measure(void)
{
    // Set start at 1000mm
    if (!write_register(DISTANCE_REG_START_ADDRESS, 1000))
    {
        //ERROR
        return false;
    }
    // Set end at 5000mm
    if (!write_register(DISTANCE_REG_END_ADDRESS, 5000))
    {
        //ERROR
        return false;
    }

    // Apply configuration
    if (!write_register(
        DISTANCE_REG_COMMAND_ADDRESS,
        DISTANCE_REG_COMMAND_ENUM_APPLY_CONFIG_AND_CALIBRATE))
    {
        //ERROR
        return false;
    }

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

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

    // Measure
    if (!write_register(DISTANCE_REG_COMMAND_ADDRESS,
        DISTANCE_REG_COMMAND_ENUM_MEASURE_DISTANCE))
    {
        //ERROR
        return false;
    }

    // Wait for measure distance to be done

```

```

if (!wait_not_busy())
{
    //ERROR
    return false;
}

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

// Did we detect a peak?
uint32_t num_distances =
    (result & DISTANCE_REG_DISTANCE_RESULT_FIELD_NUM_DISTANCES_MASK) >>
    DISTANCE_REG_DISTANCE_RESULT_FIELD_NUM_DISTANCES_POS;

// Print peak if found
if (num_distances > 0)
{
    uint32_t peak_distance_mm;
    if (read_register(DISTANCE_REG_PEAK0_DISTANCE_ADDRESS, &
        peak_distance_mm))
    {
        printf("Peak distance: %" PRIu32 " mm\n", peak_distance_mm);
    }
    else
    {
        //ERROR
        return false;
    }
}
else
{
    printf("No peak detected\n");
}

return true;
}

```

Registers

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	Distance Result	Read Only
0x0011	Peak0 Distance	Read Only
0x0012	Peak1 Distance	Read Only
0x0013	Peak2 Distance	Read Only

0x0014	Peak3 Distance	Read Only
0x0015	Peak4 Distance	Read Only
0x0016	Peak5 Distance	Read Only
0x0017	Peak6 Distance	Read Only
0x0018	Peak7 Distance	Read Only
0x0019	Peak8 Distance	Read Only
0x001a	Peak9 Distance	Read Only
0x001b	Peak0 Strength	Read Only
0x001c	Peak1 Strength	Read Only
0x001d	Peak2 Strength	Read Only
0x001e	Peak3 Strength	Read Only
0x001f	Peak4 Strength	Read Only
0x0020	Peak5 Strength	Read Only
0x0021	Peak6 Strength	Read Only
0x0022	Peak7 Strength	Read Only
0x0023	Peak8 Strength	Read Only
0x0024	Peak9 Strength	Read Only
0x0040	Start	Read / Write
0x0041	End	Read / Write
0x0042	Max Step Length	Read / Write
0x0043	Close Range Leakage Cancellation	Read / Write

0x0044	Signal Quality	Read / Write
0x0045	Max Profile	Read / Write
0x0046	Threshold Method	Read / Write
0x0047	Peak Sorting	Read / Write
0x0048	Num Frames Recorded Threshold	Read / Write
0x0049	Fixed Amplitude Threshold Value	Read / Write
0x004a	Threshold Sensitivity	Read / Write
0x004b	Reflector Shape	Read / Write
0x004c	Fixed Strength Threshold Value	Read / Write
0x0080	Measure On Wakeup	Read / Write
0x0100	Command	Write Only

Register Descriptions Version

Address	0x0000
Access	Read Only
Register Type	field
Description	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

Protocol Status

Address	0x0001
Access	Read Only
Register Type	field
Description	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

Measure Counter

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

Detector Status

Address	0x0003
Access	Read Only
Register Type	field
Description	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
DETECTOR CREATE OK	3	1	0x00000008
DETECTOR BUFFER OK	4	1	0x00000010
SENSOR BUFFER OK	5	1	0x00000020
CALIBRATION BUFFER OK	6	1	0x00000040
CONFIG APPLY OK	7	1	0x00000080
SENSOR CALIBRATE OK	8	1	0x00000100
DETECTOR CALIBRATE OK	9	1	0x00000200

RSS REGISTER ERROR	16	1	0x00010000
CONFIG CREATE ERROR	17	1	0x00020000
SENSOR CREATE ERROR	18	1	0x00040000
DETECTOR CREATE ERROR	19	1	0x00080000
DETECTOR BUFFER ERROR	20	1	0x00100000
SENSOR BUFFER ERROR	21	1	0x00200000
CALIBRATION BUFFER ERROR	22	1	0x00400000
CONFIG APPLY ERROR	23	1	0x00800000
SENSOR CALIBRATE ERROR	24	1	0x01000000
DETECTOR CALIBRATE ERROR	25	1	0x02000000
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
- **DETECTOR CREATE OK** – Detector create OK
- **DETECTOR BUFFER OK** – Detector get buffer size OK
- **SENSOR BUFFER OK** – Memory allocation of sensor buffer OK
- **CALIBRATION BUFFER OK** – Memory allocation of calibration buffer OK
- **CONFIG APPLY OK** – Detector configuration apply OK
- **SENSOR CALIBRATE OK** – Sensor calibrate OK
- **DETECTOR CALIBRATE OK** – Detector calibrate OK

- **RSS REGISTER ERROR** – RSS register error
- **CONFIG CREATE ERROR** – Configuration creates an error
- **SENSOR CREATE ERROR** – The sensor creates an error
- **DETECTOR CREATE ERROR** – The detector creates an error
- **DETECTOR BUFFER ERROR** – Detector get buffer size error
- **SENSOR BUFFER ERROR** – Memory allocation of sensor buffer error
- **CALIBRATION BUFFER ERROR** – Memory allocation of calibration buffer error
- **CONFIG APPLY ERROR** – Detector configuration applies error
- **SENSOR CALIBRATE ERROR** – Sensor calibrate error
- **DETECTOR CALIBRATE ERROR** – Detector calibrate error
- **DETECTOR ERROR** – Detector error occurred, restart necessary
- **BUSY** – Detector busy

Distance Result

Address	0x0010
Access	Read Only
Register Type	field
Description	The result from the distance detector.

Bitfield	Pos	Width	Mask
NUM DISTANCES	0	4	0x0000000f
NEAR START EDGE	8	1	0x00000100
CALIBRATION NEEDED	9	1	0x00000200

MEASURE DISTANCE ERROR	10	1	0x00000400
TEMPERATURE	16	16	0xffff0000

- **NUM DISTANCES** – The number of detected distances
- **NEAR START EDGE** – Indicating that there might be an object near the start point of the measured range
- **CALIBRATION NEEDED** – Indication of sensor calibration needed. The sensor calibration needs to be redone
- **MEASURE DISTANCE ERROR** – The measure command failed
- **TEMPERATURE** – Temperature in sensor during measurement (in degrees Celsius). Note that it has poor absolute accuracy and should only be used for relative temperature measurements.

Peak0 Distance

Address	0x0011
Access	Read Only
Register Type	uint
Unit	mm
Description	The distance to the peak is 0. Note: This value is a factor of 1000 larger than the RSS value.

Peak1 Distance

Address	0x0012
Access	Read Only
Register Type	uint
Unit	mm
Description	The distance to peak 1. Note: This value is a factor 1000 larger than the RSS value.

Peak2 Distance

Address	0x0013
Access	Read Only
Register Type	uint
Unit	mm
Description	The distance to peak 2. Note: This value is a factor 1000 larger than the RSS value.

Peak3 Distance

Address	0x0014
Access	Read Only
Register Type	uint
Unit	mm
Description	The distance to peak 3. Note: This value is a factor 1000 larger than the RSS value.

Peak4 Distance

Address	0x0015
Access	Read Only
Register Type	uint
Unit	mm
Description	The distance to peak 4. Note: This value is a factor 1000 larger than the RSS value.

Peak5 Distance

Address	0x0016
Access	Read Only
Register Type	uint
Unit	mm
Description	The distance to peak 5. Note: This value is a factor of 1000 larger than the RSS value.

Peak6 Distance

Address	0x0017
Access	Read Only
Register Type	uint
Unit	mm
Description	The distance to peak 6. Note: This value is a factor of 1000 larger than the RSS value.

Peak7 Distance

Address	0x0018
Access	Read Only
Register Type	uint
Unit	mm
Description	The distance to peak 7. Note: This value is a factor of 1000 larger than the RSS value.

Peak8 Distance

Address	0x0019
Access	Read Only
Register Type	uint
Unit	mm
Description	The distance to peak 8. Note: This value is a factor of 1000 larger than the RSS value.

Peak9 Distance

Address	0x001a
Access	Read Only
Register Type	uint
Unit	mm
Description	The distance to peak 9. Note: This value is a factor 1000 larger than the RSS value.

Peak0 Strength

Address	0x001b
Access	Read Only
Register Type	int
Description	The reflective strength of peak 0. Note: This value is a factor of 1000 larger than the RSS value.

Peak1 Strength

Address	0x001c
Access	Read Only
Register Type	int
Description	The reflective strength of peak 1. Note: This value is a factor 1000 larger than the RSS value.

Peak2 Strength

Address	0x001d
Access	Read Only
Register Type	int
Description	The reflective strength of peak 2. Note: This value is a factor of 1000 larger than the RSS value.

Peak3 Strength

Address	0x001e
Access	Read Only
Register Type	int
Description	The reflective strength of peak 3. Note: This value is a factor of 1000 larger than the RSS value.

Peak4 Strength

Address	0x001f
Access	Read Only
Register Type	int
Description	The reflective strength of peak 4. Note: This value is a factor of 1000 larger than the RSS value.

Peak5 Strength

Address	0x0020
Access	Read Only
Register Type	int
Description	The reflective strength of peak 5. Note: This value is a factor of 1000 larger than the RSS value.

Peak6 Strength

Address	0x0021
Access	Read Only
Register Type	int
Description	The reflective strength of peak 6. Note: This value is a factor of 1000 larger than the RSS value.

Peak7 Strength

Address	0x0022
Access	Read Only
Register Type	int
Description	The reflective strength of peak 7. Note: This value is a factor of 1000 larger than the RSS value.

Peak8 Strength

Address	0x0023
Access	Read Only
Register Type	int
Description	The reflective strength of peak 8. Note: This value is a factor of 1000 larger than the RSS value.

Peak9 Strength

Address	0x0024
Access	Read Only
Register Type	int
Description	The reflective strength of peak 9. Note: This value is a factor of 1000 larger than the RSS value.

Start

Address	0x0040
Access	Read / Write
Register Type	uint
Unit	mm
Description	The start of the measured interval is in millimetres. Note: This value is a factor of 1000 larger than the RSS value.
Default Value	250

End

Address	0x0041
Access	Read / Write
Register Type	uint
Unit	mm
Description	The end of the measured interval is in millimetres. Note: This value is a factor of 1000 larger than the RSS value.
Default Value	3000

Max Step Length

Address	0x0042
Access	Read / Write
Register Type	uint
Description	Used to limit step length. If set to 0 (default), the step length is calculated based on profile.
Default Value	0

Close-Range Leakage Cancellation

Address	0x0043
Access	Read / Write
Register Type	bool
Description	Enable the close-range leakage cancellation logic.
Default Value	True

Signal Quality

Address	0x0044
Access	Read / Write
Register Type	int
Description	High signal quality results in a better SNR (because of higher HWAAS) and higher power consumption. Note: This value is a factor of 1000 larger than the RSS value.
Default Value	15000

Max Profile

Address	0x0045
Access	Read / Write
Register Type	enum
Description	Max profile.
Default Value	PROFILE5

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

Threshold Method

Address	0x0046
Access	Read / Write
Register Type	enum
Description	Threshold method.
Default Value	CAR

Enum	Value
FIXED AMPLITUDE	1
RECORDED	2
CAR	3
FIXED STRENGTH	4

- **FIXED AMPLITUDE** – Fixed amplitude threshold
- **RECORDED** – Recorded threshold

- **CFAR** – CFAR threshold
- **FIXED STRENGTH** – Fixed strength threshold

Peak Sorting

Address	0x0047
Access	Read / Write
Register Type	enum
Description	Peak sorting method.
Default Value	STRONGEST

Enum	Value
CLOSEST	1
STRONGEST	2

- **CLOSEST** – Sort peaks by range, closest first
- **STRONGEST** – Sort peaks by amplitude, strongest first

Num Frames Recorded Threshold

Address	0x0048
Access	Read / Write
Register Type	uint
Description	The number of frames to use for the recorded threshold.
Default Value	100

Fixed Amplitude Threshold Value

Address	0x0049
Access	Read / Write
Register Type	uint
Description	Fixed amplitude threshold value Note: This value is a factor 1000 larger than the RSS value.
Default Value	100000

Threshold Sensitivity

Address	0x004a
Access	Read / Write
Register Type	uint
Description	Threshold sensitivity (0 <= sensitivity <= 1000) Note: This value is a factor of 1000 larger than the RSS value.
Default Value	500

Reflector Shape

Address	0x004b
Access	Read / Write
Register Type	enum
Description	Reflector shape.
Default Value	GENERIC

Enum	Value
GENERIC	1
PLANAR	2

- **GENERIC** – Generic reflector shape
- **PLANAR** – Planar reflector shape

Fixed Strength Threshold Value

Address	0x004c
Access	Read / Write
Register Type	int
Description	Fixed strength threshold value Note: This value is a factor of 1000 larger than the RSS value.
Default Value	0

Measure On Wakeup

Address	0x0080
Access	Read / Write
Register Type	bool
Description	Perform measure on wake up.
Default Value	False

Command

Address	0x0100
Access	Write Only
Register Type	enum
Description	Execute command.

Enum	Value
APPLY CONFIG AND CALIBRATE	1
MEASURE DISTANCE	2
APPLY CONFIGURATION	3
CALIBRATE	4
RECALIBRATE	5
ENABLE UART LOGS	32
DISABLE UART LOGS	33
LOG CONFIGURATION	34
RESET MODULE	1381192737

- **APPLY CONFIG AND CALIBRATE** – Apply configuration, calibrate sensor and detector
- **MEASURE DISTANCE** – Measure the distance
- **APPLY CONFIGURATION** – Apply the configuration
- **CALIBRATE** – Calibrate sensor and detector
- **RECALIBRATE** – Re-calibrate sensor and 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


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 in this document Acconeer has not performed any regulatory conformity test. It is the user's responsibility to ensure 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 before the publication hereof.

© 2023 by Acconeer AB – All rights reserved

Documents / Resources

 <small>PC Distance Detector</small>	acconeer A121 I2C Distance Detector [pdf] User Guide A121 I2C Distance Detector, A121, I2C Distance Detector, Distance Detector
--	--

References

- [Distance Detection — Acconeer docs](#)
- [User Manual](#)

Manuals+ Privacy Policy

This website is an independent publication and is neither affiliated with nor endorsed by any of the trademark owners. The "Bluetooth®" word mark and logos are registered trademarks owned by Bluetooth SIG, Inc. The "Wi-Fi®" word mark and logos are registered trademarks owned by the Wi-Fi Alliance. Any use of these marks on this website does not imply any affiliation with or endorsement.