neonode®

# zForce AIR™ Touch Sensor
# User's Guide

2017-12-21

# Legal Notice

Neonode may make changes to specifications and product descriptions at any time, without notice. Do not finalize a design with this information. Neonode assumes no liability for applications assistance or customer product design. Customers are responsible for their products and applications using Neonode components. To minimize the risks associated with customer products and applications, customers should provide adequate design and operating safeguards.

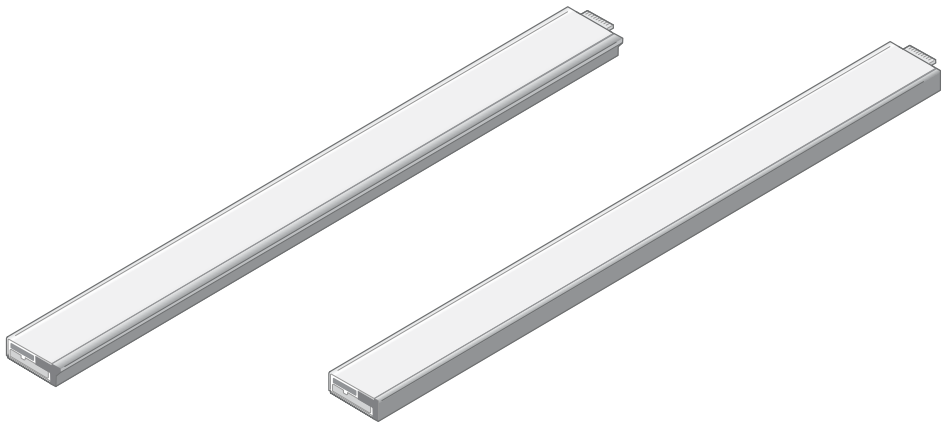
Neonode components are neither designed nor intended for use in FDA Class III applications, or similar life-critical medical equipment. Customers acknowledge and agree that they will not use any Neonode components in FDA Class III applications, or similar life-critical medical equipment, and that Neonode will not be responsible for any failure to meet the requirements of such applications or equipment.

# 1 Table of Contents

# 2 Introduction

## 2.1 Product Overview

The zForce AIR Touch Sensor is a laser light based touch sensor that can be integrated and used in various applications. The sensor characteristics are high scanning frequency, low latency, good touch accuracy and the fact that it can be used on any surface or even in mid air. zForce AIR Touch Sensor can be connected to the host system through a standard connector and communicate through a standard I2C or USB interface.



### 2.1.1 Main Features

- Enables touch on any surface or in mid air
- Dual touch support
- High scanning frequency – up to 200Hz or more depending on sensor length
- Low touch latency
- High touch accuracy
- Idle mode for reduced current power consumption
- Configurable touch active area
- I2C and USB interface
- Standard 5V power supply

## 2.2 Product Variants

In order to fit in a wide range of applications, the zForce AIR Touch Sensor exists in two types and a number of different lengths.

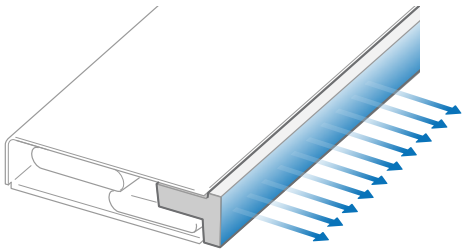
⚠ If the variant you are interested in is not available for purchase from your distributor, please contact the distributor or a Neonode sales representative (refer to www.neonode.com[1]) for more information.

1 http://www.neonode.com/

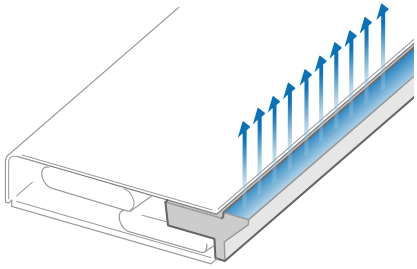
### 2.2.1 Sensor Orientation

The zForce AIR Touch Sensor is available in two types, one where the active area emerges straight out from the sensor (0° type) and one where it emerges out from the sensor at a 90° angle (90° type). This enables both vertical and horizontal integration.

**0° Type**



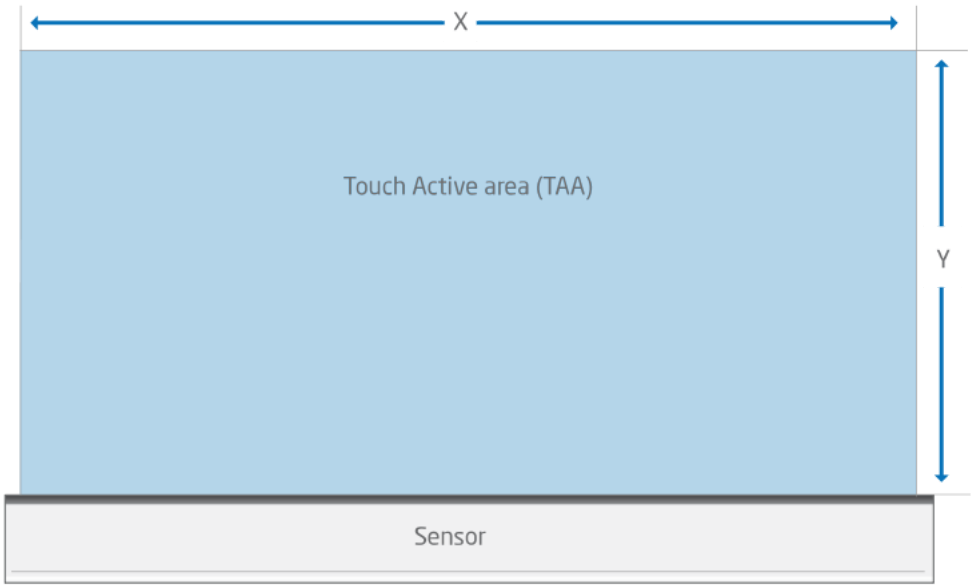
**90° Type**



### 2.2.2 Sensor Length

The Touch Sensor is available in 43 different lengths. The length affects the Touch Active Area (TAA) in both X and Y directions.

## Touch Active Area




The table lists all product variants, the product number, and the TAA for each variant. See also Physical Dimensions and Position of Origin (see page 72).

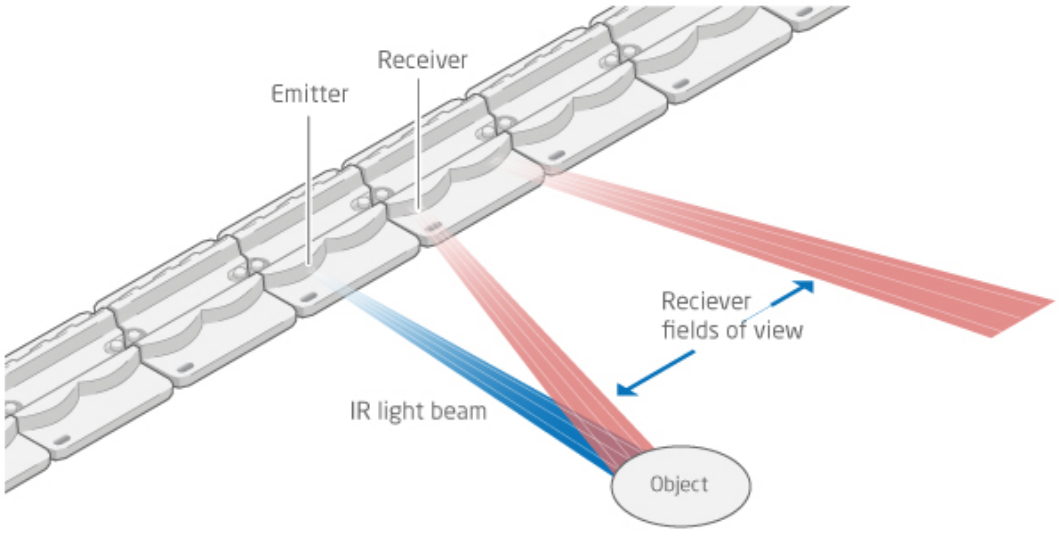
| Product number | | TAA (mm) | |
|---|---|---|---|
| **0° Type** | **90° Type** | **X** | **Y** |
| NNAMC0430PC01 | NNAMC0431PC01 | 43.2 | 14.9 |
| NNAMC0500PC01 | NNAMC0501PC01 | 50.4 | 29.8 |
| NNAMC0580PC01 | NNAMC0581PC01 | 57.6 | 29.8 |
| NNAMC0640PC01 | NNAMC0641PC01 | 64.8 | 44.7 |
| NNAMC0720PC01 | NNAMC0721PC01 | 72 | 44.7 |
| NNAMC0790PC01 | NNAMC0791PC01 | 79.2 | 59.6 |
| NNAMC0860PC01 | NNAMC0861PC01 | 86.4 | 59.6 |
| NNAMC0940PC01 | NNAMC0941PC01 | 93.6 | 74.5 |
| NNAMC1010PC01 | NNAMC1011PC01 | 100.8 | 74.5 |
| NNAMC1080PC01 | NNAMC1081PC01 | 108 | 89.4 |
| NNAMC1150PC01 | NNAMC1151PC01 | 115.2 | 89.4 |

| | | | |
|---|---|---|---|
| NNAMC1220PC01 | NNAMC1221PC01 | 122.4 | 104.3 |
| NNAMC1300PC01 | NNAMC1301PC01 | 129.6 | 104.3 |
| NNAMC1370PC01 | NNAMC1371PC01 | 136.8 | 119.2 |
| NNAMC1440PC01 | NNAMC1441PC01 | 144 | 119.2 |
| NNAMC1510PC01 | NNAMC1511PC01 | 151.2 | 134.0 |
| NNAMC1580PC01 | NNAMC1581PC01 | 158.4 | 134.0 |
| NNAMC1660PC01 | NNAMC1661PC01 | 165.6 | 148.9 |
| NNAMC1730PC01 | NNAMC1731PC01 | 172.8 | 148.9 |
| NNAMC1800PC01 | NNAMC1801PC01 | 180 | 163.8 |
| NNAMC1870PC01 | NNAMC1871PC01 | 187.2 | 163.8 |
| NNAMC1940PC01 | NNAMC1941PC01 | 194.4 | 178.7 |
| NNAMC2020PC01 | NNAMC2021PC01 | 201.6 | 178.7 |
| NNAMC2090PC01 | NNAMC2091PC01 | 208.8 | 193.6 |
| NNAMC2160PC01 | NNAMC2161PC01 | 216 | 193.6 |
| NNAMC2230PC01 | NNAMC2231PC01 | 223.2 | 208.5 |
| NNAMC2300PC01 | NNAMC2301PC01 | 230.4 | 208.5 |
| NNAMC2380PC01 | NNAMC2381PC01 | 237.6 | 208.5 |
| NNAMC2450PC01 | NNAMC2451PC01 | 244.8 | 208.5 |
| NNAMC2520PC01 | NNAMC2521PC01 | 252 | 208.5 |
| NNAMC2590PC01 | NNAMC2591PC01 | 259.2 | 208.5 |
| NNAMC2660PC01 | NNAMC2661PC01 | 266.4 | 208.5 |
| NNAMC2740PC01 | NNAMC2741PC01 | 273.6 | 208.5 |
| NNAMC2810PC01 | NNAMC2811PC01 | 280.8 | 208.5 |
| NNAMC2880PC01 | NNAMC2881PC01 | 288 | 208.5 |
| NNAMC2950PC01 | NNAMC2951PC01 | 295.2 | 208.5 |
| NNAMC3020PC01 | NNAMC3021PC01 | 302.4 | 208.5 |
| NNAMC3100PC01 | NNAMC3101PC01 | 309.6 | 208.5 |

| | | | |
|---|---|---|---|
| NNAMC3170PC01 | NNAMC3171PC01 | 316.8 | 208.5 |
| NNAMC3240PC01 | NNAMC3241PC01 | 324 | 208.5 |
| NNAMC3310PC01 | NNAMC3311PC01 | 331.2 | 208.5 |
| NNAMC3380PC01 | NNAMC3381PC01 | 338.4 | 208.5 |
| NNAMC3460PC01 | NNAMC3461PC01 | 345.6 | 208.5 |

## 2.3 Basic Principles

zForce AIR Touch Sensors detect and trace objects by detecting diffusely reflected infrared light. The sensor comprises an optical system arranged to combine emitted IR beams and receiver fields of view within the same apertures. IR light beams are emitted perpendicular to the output window, while receivers field of view is centered at a certain angle left and right.




Each emitter-receiver combination covers a narrow region on the active area. An object present in the active area will affect several emitter-receiver channels, and the reported coordinates is the outcome of a center of gravity calculation on these signals.

## 2.4 Applications

zForce AIR Touch Sensors can be integrated for a wide range of applications, such as:

- PCs/Tablets
- TVs/Monitors
- Printers
- Mechanical key replacement
- White goods
- Smart furniture
- Interactive mirrors
- Elevator panels
- eReaders

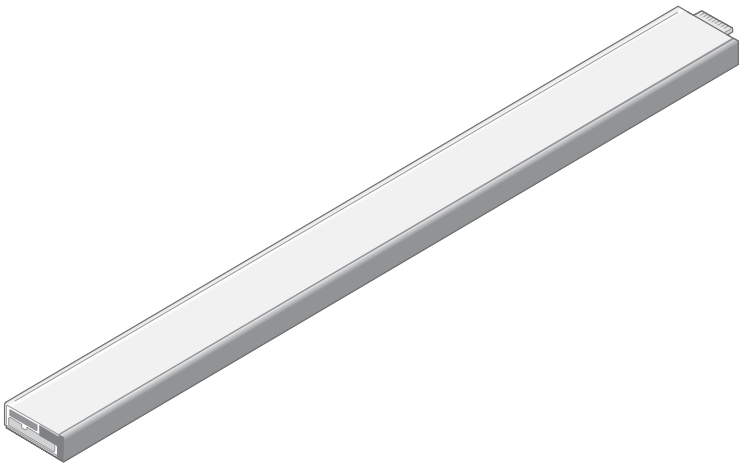
- Instruments
- Vending Machines
- ATM/POS terminals
- Robotics
- Range finders
- Collision detectors
- ... and much more

## 2.5 Product Design and Components

The zForce AIR Touch Sensor is a laser light based touch sensor that can be used for various touch and mid-air detection applications. The sensor is available in varying sizes, see Product Variants (see page 6).

### 2.5.1 Sensor Design

The image below show the sensor design (0° type). The connector is shown to the far right.



#### Exploded view

The image below shows the sensor (0° type) in an exploded view.


| Part | Decription |
|---|---|
| A | Cover |
| B | Adhesive |
| C | Front light pipe – straight shooting or 90 degree shooting depending on sensor type |
| D | Lenses - amount depends on size |
| E | PCBA |

### 2.5.2 Sensor Components

The PCBA is equipped with both active and passive components, for example:

- MCU

- Co-processor, a Neonode proprietary scanning IC
- Optical lenses, made out of polycarbonate
- VCSELs
- Photo diodes
- Other passive components

## 2.6 Product Integration

The zForce AIR Touch Sensor can be integrated to any host system through a physical connector with 8 contact pads with support for both I2C and USB HID. The host system can communicate with the sensor through a communication protocol and an SDK developed by Neonode.

# 3 Getting started with zForce AIR Touch Sensor Evaluation

This section describes how to get started with the evaluation of a zForce AIR Touch Sensor.

## 3.1 Evaluation Kit Contents

The evaluation kit includes the following:

- 1 x zForce AIR Touch Sensor
- 1 x Interface board (with USB and I2C interface)
- 1 x FPC Cable with connector

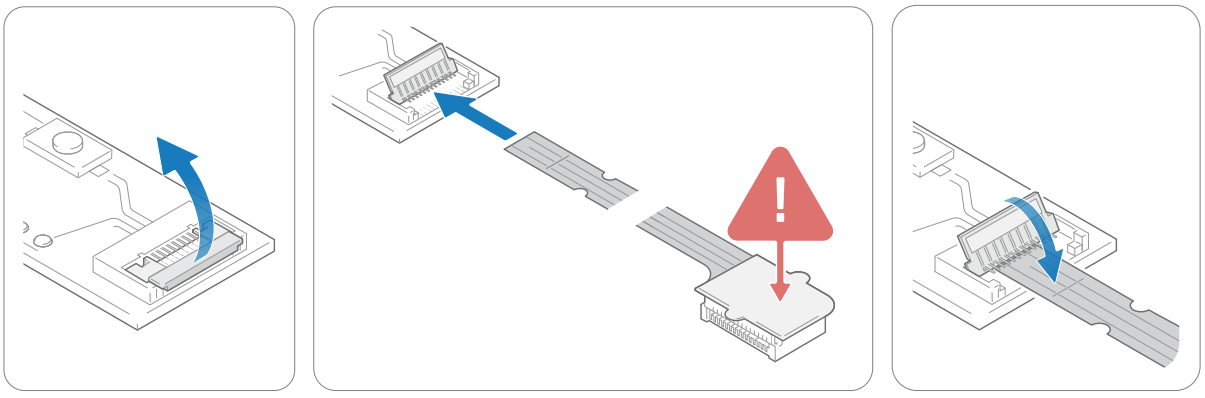
## 3.2 Getting Started

1. Connect the sensor according to Connecting Sensor (see page 14).
2. Start communicating using one of the means listed below:

- Neonode Workbench (see page 16). Use the Neonode Workbench software for Windows to configure a sensor and test and evaluate touch performance.
- USB HID Digitizer Mode (see page 17). This is the easiest and fastest way to try out the Touch Sensor. It only requires connecting the interface board to a Windows or Linux computer via USB, but is limited in functionality.
- USB HID Raw Mode (see page 17). This also uses a USB connection to a Windows or Linux computer, but requires communicating with the sensor using ASN.1 encoded messages.
- I2C Transport (see page 17). This involves sending and receiving ASN.1 encoded messages over I2C.
- SDK (see page 17). Using the zForce SDK function library facilitates communication with sensors without considering ASN.1 encoded messages.

## 3.3 Connecting Sensor

1. Connect the FPC cable to the interface board:

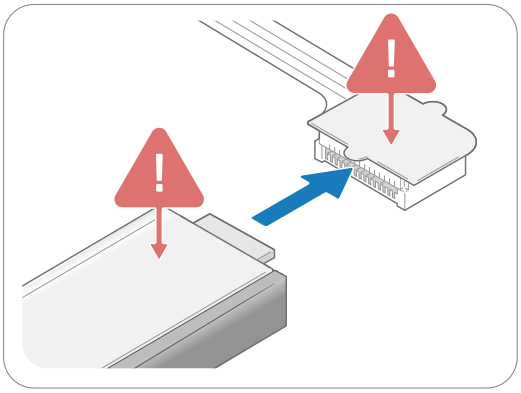


   a. Lift the flip lock on the interface board.
   b. Insert the FPC cable into the end of the connector, with the connector pads facing down, towards interface board. The yellow piece of PCB of the connector on the other side of the cable is facing upwards. Make sure the direction is straight into the connector and the pads have reached the end of the connector.

> Make sure the connector pads of the FPC cable are facing downwards, towards interface board. The sensor risks damage if the FPC cable is connected in wrong direction.

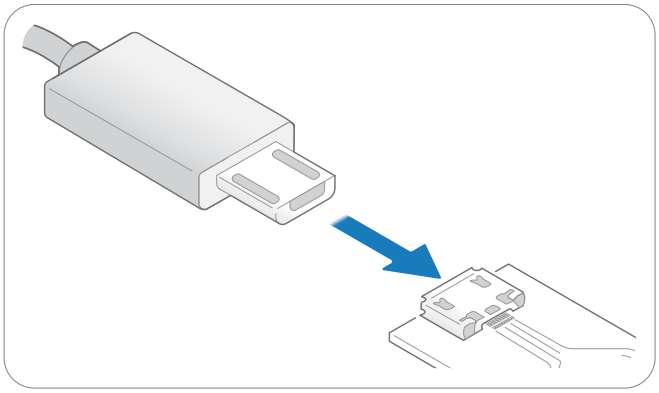
c. Press down the flip lock.

2. Connect the FPC cable to the sensor:



a. Place the sensor so that the sensor connector pads of the sensor are facing downwards (steel surface upwards).
b. Insert sensor into the connector on FPC cable (yellow piece of PCB of the FPC connector still facing upwards).
c. Make sure the direction of the pads is straight into the connector, and the pads have reached the end of the connector.

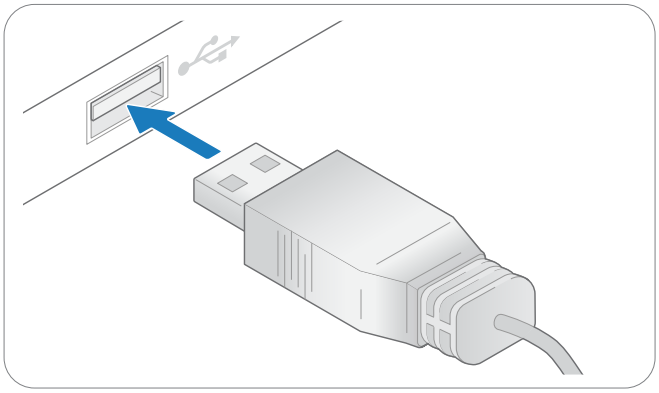
3. To use Neonode Workbench, USB HID Digitizer mode, USB HID Raw mode, or SDK: Connect a USB cable with a Micro USB type B connector to the interface board.



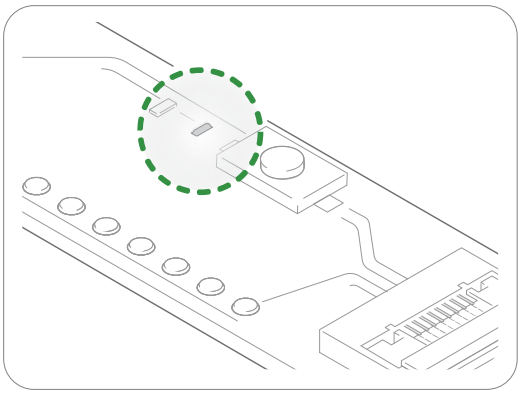
4. To use I2C Transport: Wire the pads of +5V, DR-B0, I2C-D, I2C-C, and GND on the interface board to the host system. For details, refer to Electrical Integration (see page 26). Do not connect power until the following steps have been performed.
5. Place the sensor on a table with the steel surface facing downwards and with the optical surface facing towards you.

Make sure no object is within the touch active area of the sensor before connecting power to the sensor through USB or I2C. The sensor calibrates itself when powered on and an object within the touch active area may interfere with the calibration.

6. To use Neonode Workbench, USB HID Digitizer mode, USB HID Raw mode, or SDK: Insert the USB cable into a computer meeting the requirements of USB HID or SDK, respectively.



7. To use I2C Transport: Connect the power to the sensor through the I2C.
8. The green LED on the interface board lights up when connected.



In case of strong side light, covering the short sides of the sensor with, for example, black tape might improve performace.

## 3.4 Communicating with Sensor

### 3.4.1 Neonode Workbench

Neonode Workbench is a software tool to use with zForce AIR™ sensors. With Neonode Workbench it is possible to:

- Visualize sensor touches.
- Temporarily configure sensor characteristics, such as active area and scanning frequency.
- View the sensor firmware version.
- Conduct Open/Short-circuit tests to identify any damaged photo or laser diodes.

Download Neonode Workbench from Downloads[2] and refer to separate Neonode Workbench documentation.

### 3.4.2 USB HID Digitizer Mode

The easiest way to see the Touch Sensor functionality is to use USB HID Digitizer mode:

1. When the sensor has enumerated, it will act as a touch screen USB HID device.
2. Put an object in the touch active area, touch HID reports will be send to host.

For more information on USB HID Digitizer mode, refer to USB HID Transport (see page 38).

### 3.4.3 USB HID Raw Mode

In USB HID Raw Mode, communication with a Touch Sensor is performed by sending and receiving messages as reports defined by the USB HID standard.

To start communicating, perform the following:

1. When the sensor has enumerated, start communicating as defined in USB HID Transport (see page 38). The messages are encoded with ASN.1.Refer to Presentation Layer (ASN.1) (see page 50).
2. The Touch Sensor starts sending touch notifications once it is initialized. For initialization procedures, refer to Initializing Sensors (see page 33).

### 3.4.4 I2C Transport

To start communicating, perform the following:

1. Send or read data over the I2C bus. Refer to I2C Transport (see page 35). The messages are encoded with ASN. 1. Refer to Presentation Layer (ASN.1) (see page 50).
2. The sensor starts sending touch notifications once it is initialized. For initialization procedures, refer to Initializing Sensors (see page 33).

### 3.4.5 SDK

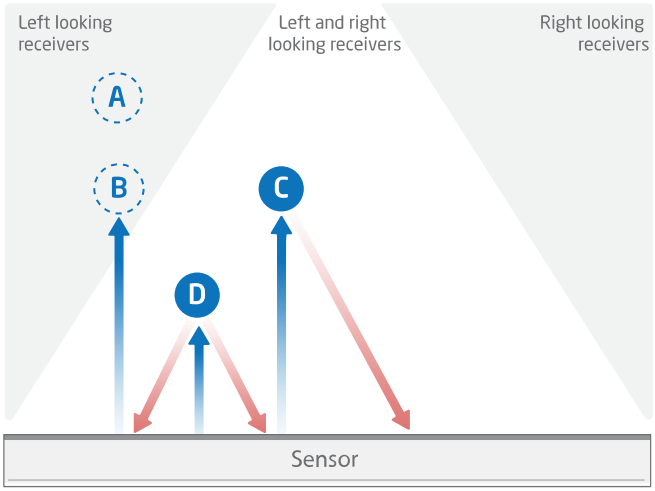
To start communicating, follow the Getting Started instructions in the separate SDK documentation.

2 https://support.neonode.com/docs/display/Downloads

# 4 Multi-Touch Functionality

zForce AIR Touch Sensor determine an object's position by signals derived from emitter-receiver pairs and have the capacity to detect and track several objects at the same time. Both the HW and the SW have been optimized in order to support standard touch gestures like, pinch-to-zoom, rotate, swipe and tap. However, some combinations of two or more objects might need special consideration.

## 4.1 Shadows




- An object directly behind another object cannot be illuminated. In the figure above, object A will not be detected since illumination is blocked by object B.
- The correct receiver must have a clear field of view. Object B is in a region covered only by left looking receivers. Object B will not be detected because its field of view is blocked by object D.
- Object C may be seen by both left and right looking receivers. Although the right looking field of view is blocked by object D, object C is detected by the left looking receiver.
- Object D is detected by both left and right looking receivers.

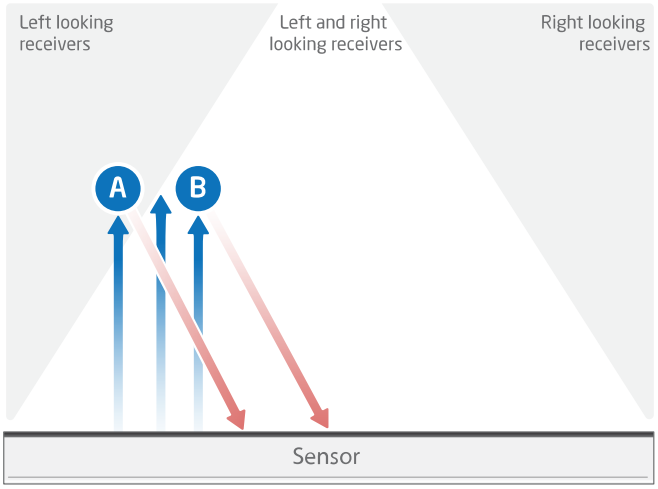
### 4.1.1 Shadow Trick

Note that in most cases, user experience is not affected by the shadow situations mentioned above. This is because of a functionality implemented in the Touch Sensor firmware called "shadow trick", which e.g. generates a smooth "rotate" feeling despite one touch object being shadowed during the rotate gesture . A previously detected object that can no longer be detected is still reported as present if:

- The object was last seen close to a location where it could be shadowed by another object.
- The potentially shadowing object is still detected and hasn't moved away from a shadowing location.

The shadow trick make multi-touch gestures such as "rotate" and "pinch-to-zoom" work better.

## 4.2 Adjacent Objects

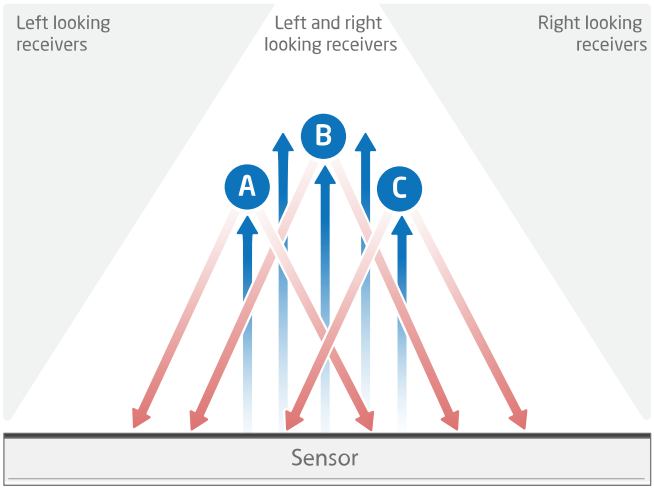



- In order to recognize two objects close to each other (A and B), a separation must allow at least one emitter-receiver channel (~10 mm) to pass freely between them. Otherwise, the two objects will be reported as one large object.

## 4.3 More Than Two Objects

When more than two objects are being tracked the likelihood that an object ends up being in the shadow of another object increases. Therefore, it is only recommended to enable more than two tracked objects if, for example:

- it is not vital to track all detected objects 100% in all possible combinations and locations at all time.
- When all objects are likely to be detected by the sensor, for example when it is expected that all objects will be placed along a line that is parallel to the sensor, as in the example below.

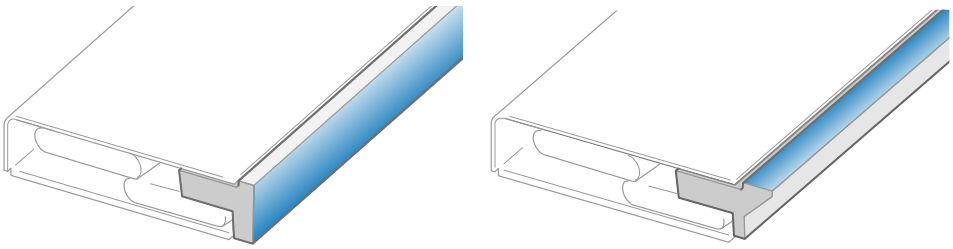


# 5 Mechanical Integration

zForce AIR Touch Sensor can be used for different purposes, such as touch on a surface or motion in mid-air. Assembly requirements differ depending on what purpose the Touch Sensor fulfills. In addition, different industries have different standards and demands to fulfill. Mid-air detection applications generally require lower mounting tolerances.

## 5.1 Means of Integration

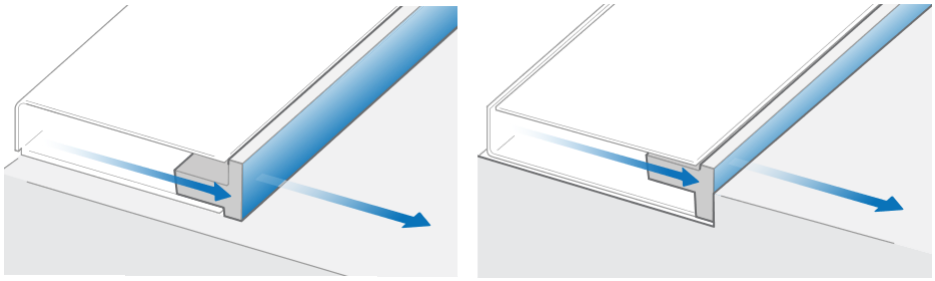
zForce AIR Touch Sensor comes in two types: one is designed to be integrated horizontally and the other vertically. This allows different types of assembly possibilities and better adaptation of the available space in the host system. The two sensor designs are built on the same concept, but use two different front light pipes. One light path is unaffected; the other bent 90°.



The front optical surface is not allowed to be blocked by the host system. In x direction the entire surface is used by the sensors optics.

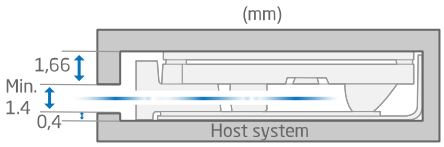
### 5.1.1 Horizontal Integration

Light is sent straight out and enables an active area in front of the module (in the same plane).



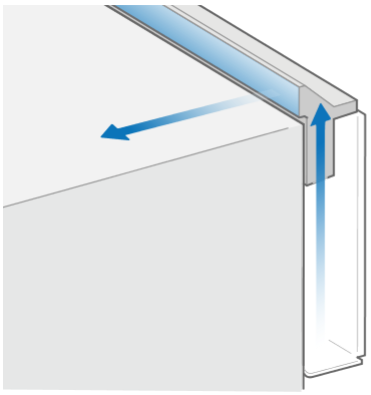
When integrating zForce AIR Touch Sensor into a host system make sure not to interfere with the light path. For horizontal integration, the opening for the sensors light path must be ***minimum 1.4 mm.***

If the host system have large tolerances, opening must be adjusted to always be minimum 1.4 mm.

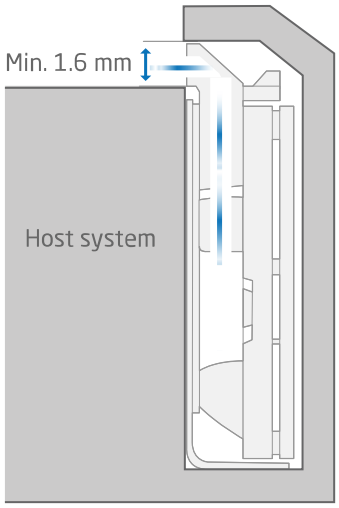


### 5.1.2 Vertical Integration

Light is bent 90 degrees within the Touch Sensor. This allows the sensor to be assembled vertically but still have an active area in the horizontal plane.



To make sure not to interfere with the sensor light path, opening must be ***minimum 1.6 mm.*** If host system have large tolerances, opening must be adjusted to always be minimum 1.6 mm. Also note that it is not allowed to mount, glue or in any other way affect the sensors optical surfaces since it will affect the performance. This applies to both the sensors visible optical surface and also the built-in optical surface A (mirror surface).



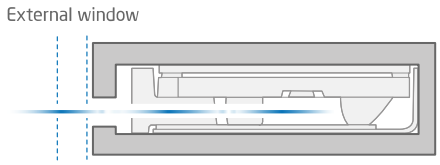

### 5.1.3 Options for Guiding and Fastening

- **Double adhesive tape** – for smaller sizes this can be used alone to hold the zForce AIR Touch Sensor. The host system geometry needs to provide a flat supporting surface.
- **Snaps** – Host system geometry provides some sort of snap features holding the zForce AIR Touch Sensor in place. These must be developed for each case to fit the host System cover and the surrounding.
- **Sandwiched** – the zForce AIR Touch Sensor is mounted by pressing the Touch Sensor between host system exterior cover and display. A structure (ribs, foam gasket or adhesive) is needed to make sure the Touch Sensor cannot move.

The zForce AIR Touch Sensor needs to be protected from outer pressure and forces that can bend the sensor and by that change the direction of the sensor light. The most common cause of bending is when a Touch Sensor is mounted on a non-flat surface, so the host system supporting structure needs to be flat.

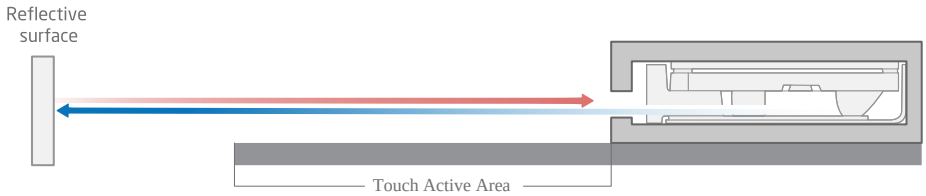
### 5.1.4 External Window

An external window is something placed between the sensor and the desired touch active area, usually in form of a plastic or glass “window”. It is of high importance for the function that these surfaces fulfill the optical demands stated in Optical Requirements on External Window (see page 71). It is important to know that each window the light passes through will reduce the sensors received signal levels, even though the requirements are fulfilled, which in some applications might reduce the maximum detection range.




### 5.1.5 External Reflective Surface

An external reflective surface is a surface located outside the active area, but close enough to be reached by the IR light emitted by the sensor. Depending on the angle and the reflectance of the surface, reflected light might enter the sensor and interfere with touch object detection. If the external reflective surface is close to the touch active area, it is recommended to make sure it has a low reflectance in the direction back towards the sensor.




## 5.2 Touch Applications

The sections below describe integration aspects specific for touch applications and do not concern mid-air applications.

### 5.2.1 Touch Accuracy

Mechanical integration of zForce AIR Touch Sensor and assembly tolerances has a direct impact on touch accuracy. For this reason relaxed assembly tolerances might in some applications have an impact on the perceived touch performance. The best user experience is achieved when the projected touch Active Area from the zForce AIR Touch Sensor perfectly overlaps the intended touch sensitive area on the host device, for example, the active area on a display.

Touch active area of host system and zForce AIR Touch Sensor needs to be well aligned. Translational tolerances in x and y directions and rotational tolerances will affect accuracy. See Translational Tolerances (see page 23) (x and y direction) and Rotational Tolerances (see page 23) (angle "b").

### 5.2.2 Hovering Touches

*Hovering touch* means that the Touch Sensor reports a touch event before the object reaches the surface. The basic principle of the Touch Sensor is that light is sent above the surface. To provide a good user experience the Touch Sensor software adjusts the signal and reports a touch first when the object reaches the surface.

Hovering touches is also direct linked to how the zForce AIR Touch Sensor is integrated in the host system. It's important that the mounting surface has the correct angle compared to the intended touch surface. Twisting and tilting of zForce AIR Touch Sensor should always be avoided. Relaxed tolerances can lead to missed touches and increased hovering. See Translational Tolerances (see page 23) (z direction) and Rotational Tolerances (see page 23) (angle "a").

Furthermore, host system active area surface need to be flat or slightly concave. A convex surface can give false touches.

### 5.2.3 Assembly Tolerances

#### Translational Tolerances

| Direction | Recommended Tolerances for Touch Applications |
|---|---|
| x-direction | ±0.5 mm |
| y-direction | ±0.5 mm |
| z-direction | 0 mm to +0.5 mm |

Translational tolerances affects the overlap between the display active area and the touch active area. For example, if the Touch Sensor is translated 0.5 mm in x-direction there will be a systematic touch offset of 0.5 mm for the complete sensor in x-direction.

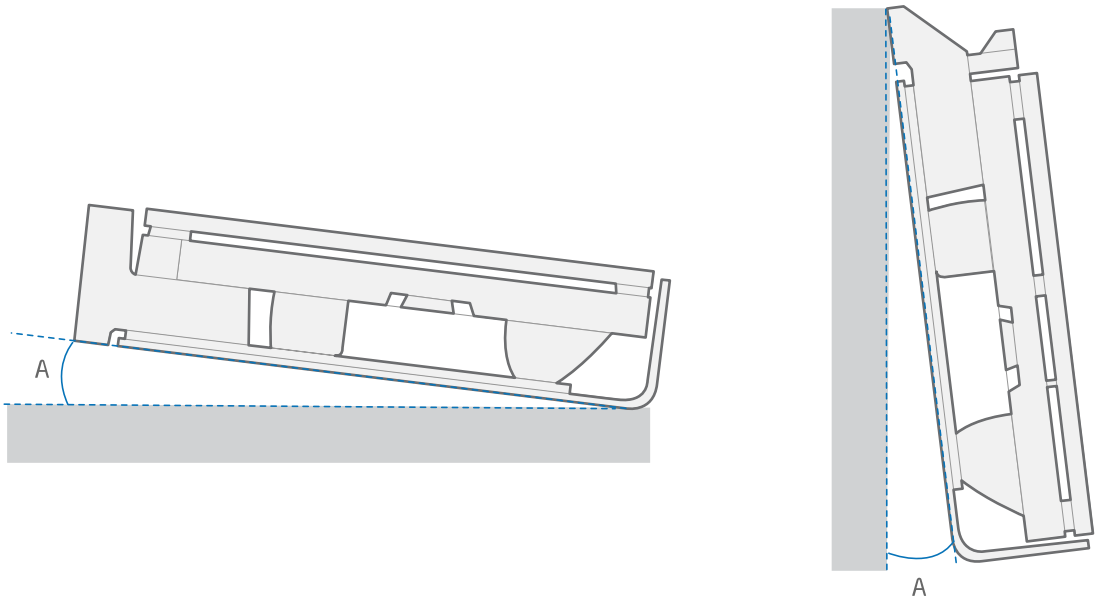
A 0 mm translation in z-direction means that the host system active area surface is positioned exactly at the edge of the light path. A positive translation means that zForce AIR Touch Sensor, and therefore the light path, is translated up from the host system active area surface. This will not affect the touch accuracy in the sensor, but it can affect the perceived touch performance, since it leads to increased hovering. A negative translation in z-direction should be avoided since parts of the light will be blocked which leads to no or reduced touch performance.

#### Rotational Tolerances

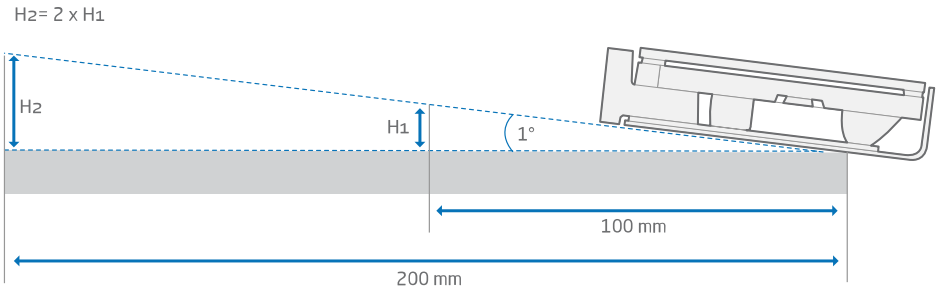
There are two types of rotations that can affect the performance; defined as the angles "a" and "b". Angle "a" affects the floating and angle "b" affects the overlap between the intended active area and the Touch Sensor active area. Both these issues will grow with larger display sizes. The angles are exaggerated in the pictures to better illustrate the problem.

Angle "a"

The angle "a" is defined as shown in the images below.

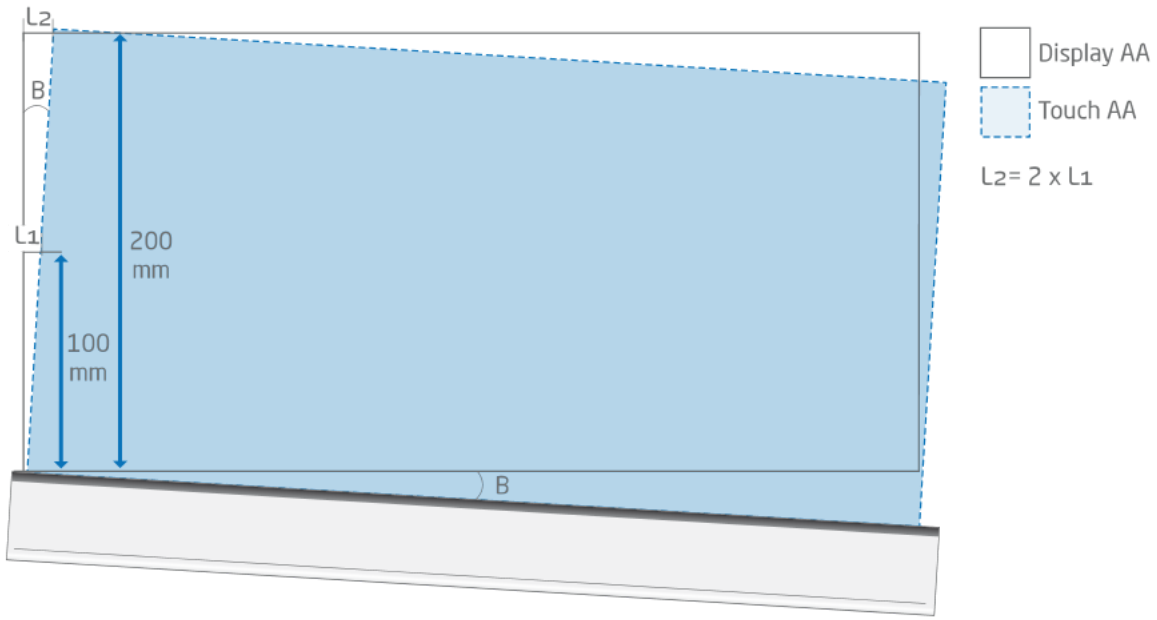



The example below illstrates the problem increasing with larger active areas.




Angle "b"

The angle "b" is defined as shown in the image below. How sensitive zForce AIR Touch Sensor is for assembly rotations is directly linked to the size. At any given angle b, the touch AA will be tilted twice as much at 200 mm compared to at 100 mm.

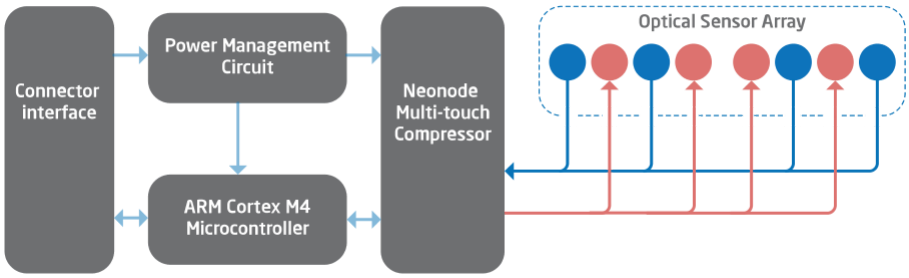

L2
B
L1
200
mm
100
mm
B
Display AA
Touch AA
L2= 2 x L1

# 6 Electrical Integration

**Electrostatic Sensitive Device!**

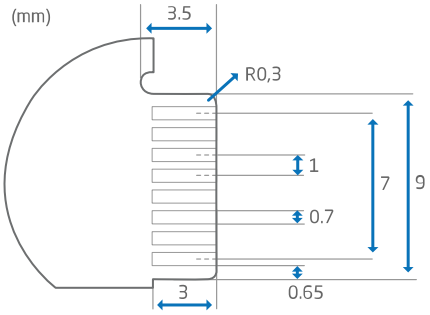
To prevent equipment damage, use proper grounding techniques.

## 6.1 Electrical Block Diagram




## 6.2 Physical Connector

The Touch Sensor has 8 contact pads and a PCB outline that matches that of a standard 0.3-0.33 mm thick FFC/FPC with 1 mm pitch and top mounted connectors:




The contact pads are placed on the backside of the Touch Sensor PCBA.

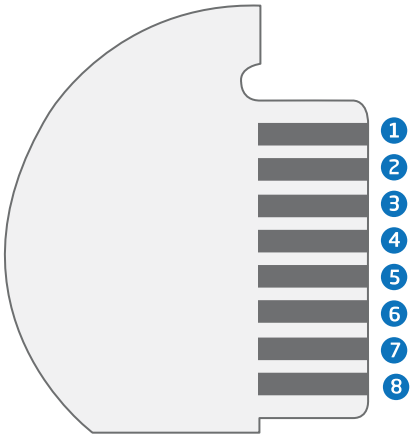
List of supported FFC connectors:

| Supplier | Part number |
|---|---|
| Molex[3] | 2005290080 |
| Omron Electronic Components[4] | XF3M(1)-0815-1B |

3 http://www.molex.com/molex/home
4 https://www.components.omron.com/

| | |
|---|---|
| Wurth Electronics Inc[5]. | 686108148922 |
| Almita Connectors[6] | BL124H-8R-TAND |

## 6.3 Pin-Out




| Function | Pin name | Pin No | Direction | Description | Comment |
|---|---|---|---|---|---|
| Power ground | GND | 1 | - | Ground | |
| Reset | N_RST | 2 | Input | Resets sensor to initial state. Active low. | A minimum of a 300 ns low pulse is required |
| Data Ready | DR | 3 | Output | Indicates that sensor has data to send | Push/pull output. Only used in I2C mode. |
| I2C Data | I2C_DATA | 4 | I/O | I2C data line | Requires external pull-up resistor |
| I2C Clock | I2C_CLK | 5 | Input | I2C clock line | Requires external pull-up resistor |
| USB DM | USB D- | 6 | I/O | USB DM line | USB 2.0 Compliant |
| USB DP | USB D+ | 7 | I/O | USB DP line | USB 2.0 Compliant |

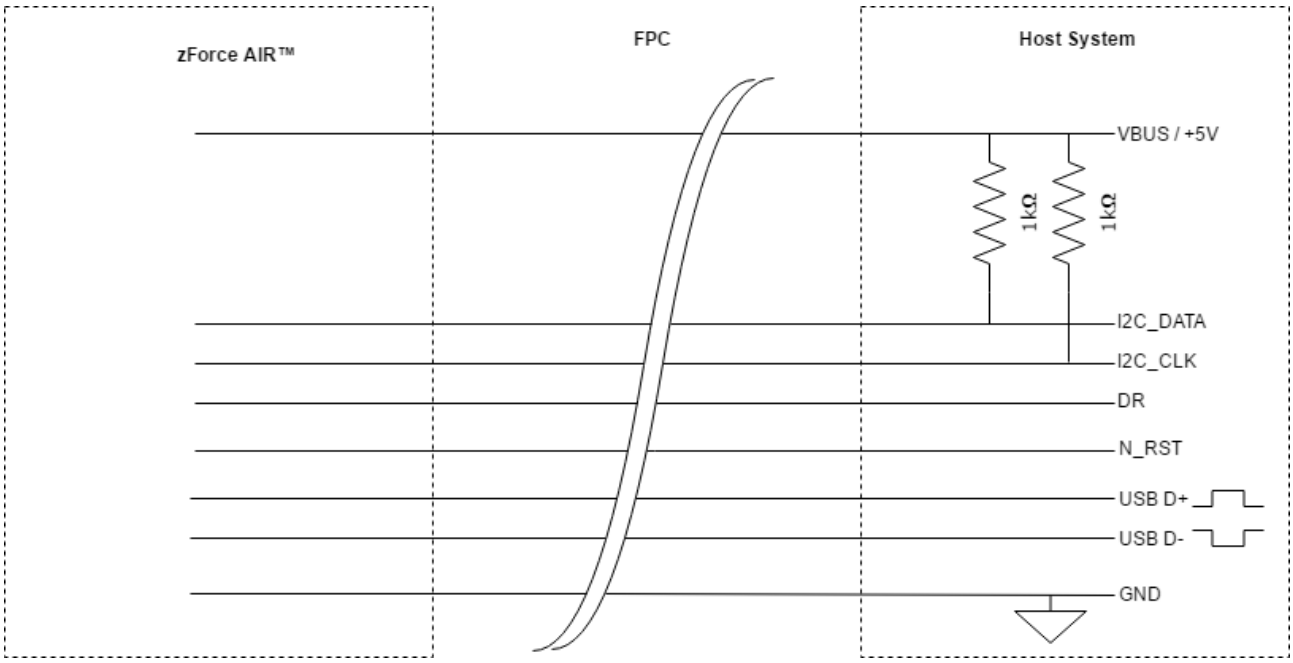
5 http://www.digikey.com/en/supplier-centers/w/wurth-electronics
6 http://www.almita-connectors.com/

| | | | | | |
|---|---|---|---|---|---|
| Power supply | +5V | 8 | - | +5V power supply | USB 2.0 Compliant |

Note: All pins use 3.3V voltage level and have 5V tolerance.

## 6.4 Interface Configuration

The zForce AIR Touch Sensor provides two interfaces for communication with the host system, I2C and USB HID-device. User can choose to connect one of them, or both. The typical Touch Sensor connection to a host system is shown in the diagram below.




### 6.4.1 USB Connection

The zForce AIR Touch Sensor provides a USB full-speed device interface through its 8-pin connector. In this connection, PIN 1, 6, 7, 8 ( GND, USB D-, USB D+, VBUS ) are used. After connecting the sensor to the host system, it could be enumerated as a normal USB HID-device and act as a digitizer for a touch screen.

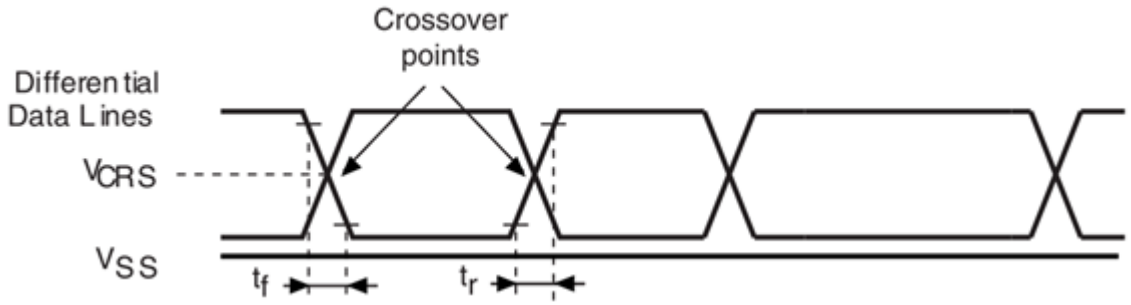
In this connection, only the default touch active area size could be used. Please refer to Product Variants (see page 6) for the actual values.

In this case, it is recommended to use two 1kΩ pull-up resistors to tie up I2C_DATA and I2C_CLK pins to VBUS or 3.3V power supply to avoid noise issue on I2C interface, and leave other pins as not connected. PIN 2 ( N_RST ) could be used to reset or enable/disable the sensor.

#### USB Characteristics

The USB interface meets the requirements of USB specifictions 2.0:

| Symbol | | Parameter | Conditions | Min | Typ | Max | Unit |
|---|---|---|---|---|---|---|---|
| Input levels | $V_{DI}$ | Diff. Input sensitivity | | 0.2 | - | - | V |
| | $V_{CM}$ | Diff. common mode range | | 0.8 | | 2.5 | |
| | $V_{SE}$ | Single ended receiver threshold | | 1.3 | | 2 | |
| Output levels | $V_{OL}$ | Output level low | 1.5kΩ to $V_{DD}$ | - | - | 0.3 | |
| | $V_{OH}$ | Output level high | 15kΩ to $V_{SS}$ | 2.8 | - | 3.6 | |
| $R_{PD}$ | | D+/D- | $V_{IN}=V_{DD}$ | 17 | 21 | 24 | kΩ |
| $R_{PU}$ | | D+/D- | $V_{IN}=V_{SS}$ | 1.5 | 1.8 | 2.1 | |




USB full speed timings: definition of data signal rise and fall time.

| Driver characteristcs | | | | |
|---|---|---|---|---|
| **Symbol** | **Parameter** | **Conditions** | **Min** | **Max** |
| $t_r$ | Rise time | $C_L$ = 50pF | 4 | 20 |
| $t_f$ | Fall time | $C_L$ = 50pF | 4 | 20 |
| $t_{rfm}$ | Rise/fall time matching | $t_r/t_f$ | 90 | 110 |
| $V_{CRS}$ | Output signal crossover | | 1.3 | 2 |

### 6.4.2 I2C Connection

The zForce AIR Touch Sensor provides an I2C interface through its 8-pin connector. The interface runs in fast mode, which means the speed is 400kbps. With this interface, more advanced configuration could be performed. PIN 1, 3, 4, 5, 8 ( GND, DR, I2C_DATA, I2C_CLK, VBUS ) are used for this connection. It is recommended to use two 1kΩ pull-up resistors to tie up I2C_DATA and I2C_CLK pins to VBUS or 3.3V power supply to perform proper I2C communication. If USB connection is not used in parallel, PIN 6, 7 ( USB D-, USB D+ ) could be left unconnected.

After the Touch Sensor is powered on, it will act as a slave device on a I2C bus. For further information about what commands could be send or receive through this interface, please refer to I2C Transport (see page 35).

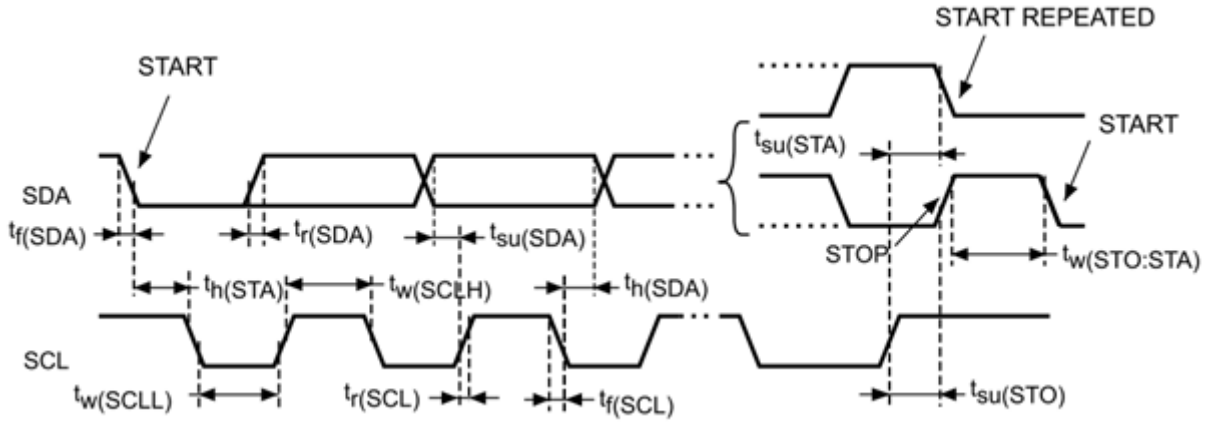
#### I2C Characteristics

The I2C interface meets the requirements of the standard I2C communication protocol with the following restrictions: SDA and SCL are mapped to I/O pins that are not “true” open-drain. When configured as open-drain, the PMOS connected between the I/O pin and VDD is disabled, but is still present.

The I2C characteristics are described in below.

| Symbol | Parameter | Min | Max | Unit |
|---|---|---|---|---|
| $t_{w(SCLL)}$ | SCL clock low time | 4.7 | - | us |
| $t_{w(SCLH)}$ | SCL clock high time | 4.0 | - | |
| $t_{su(SDA)}$ | SDA setup time | 250 | - | ns |
| $t_{h(SDA)}$ | SDA data hold time | 0 | - | |
| $t_{r(SDA)}$ | SDA and SCL rise time | - | 1000 | |
| $t_{r(SCL)}$ | | | | |
| $t_{f(SDA)}$ | SDA and SCL fall time | | 300 | |
| $t_{f(SCL)}$ | | | | |
| $t_{h(STA)}$ | Start condition hold time | 4.0 | - | us |
| $t_{su(STA)}$ | Repeated Start condition setup time | 4.7 | - | |
| $t_{su(STO)}$ | Stop condition setup time | 4.0 | - | |
| $t_{w(STO:STA)}$ | Stop to Start condition time (bus free | 4.7 | - | |

| | | | | |
|---|---|---|---|---|
| $t_{SP}$ | Pulse width of the spikes that are suppressed by the analog filter | 0 | 50 | ns |
| $C_{load}$ | Capacitive load for each bus line | - | 400 | pF |
| $V_{IL}$ | Input low level voltage | | 0.8 | V |
| $V_{IH}$ | Input high level voltage | 2.3 | | V |
| $V_{OL}$ | Output low level voltage | | 0.4 | V |
| $V_{OH}$ | Output high level voltage | 2.9 | | V |



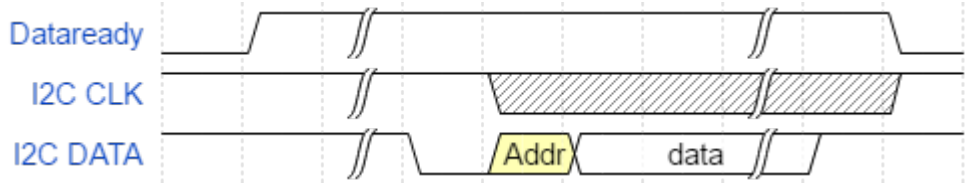

## PIN 3 ( DR ) Characteristics

PIN 3 is used as a 'Dataready' signal output. This signal is only used in I2C communication.

Because the sensor can only act as an I2C slave, this pin is needed to notify host to read out any data in it output buffer.

- This pin will be set to High (‘1’) when there is data in buffer to be sent from sensor.
- This pin will be reset to Low (‘0’) when there is no data in buffer to be sent from sensor.

This pin could be used as an interrupt input or be repeatedly read by host. When this pin is set to high, the FW will keep waiting for the host to read data from I2C bus. When the ‘Read’ transaction finished, this DR will be reset automatically by the zForce AIR Touch Sensor. The figure below shows the timing behavior of a typical I2C transaction.



### I2C Reading Sequence

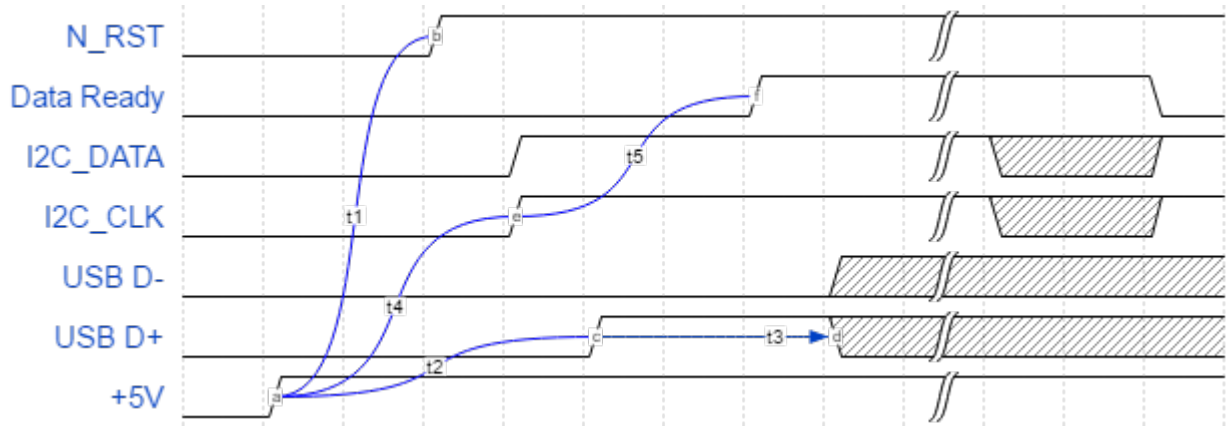
See I2C Transport (see page 35).

## 6.5 Power On and Boot Sequence

Power on timing latency are listed in the table below.

| Name | Min | Typ. | Max | Unit | Comment |
|---|---|---|---|---|---|
| t1 | - | 15 | 20 | ms | Delay time from power on to NRST to high voltage. |
| t2 | - | 60 | 70 | ms | Delay time from power on to USB pins voltage ready. |
| t3 | - | 170 | - | ms | Delay time from USB init ready to sending/receiving data. |
| t4 | 5 | 7 | 10 | ms | Delay time from power on to I2C pins voltage ready. |
| t5 | 65 | 70 | 80 | ms | Delay time from I2C pins voltage ready to triggering boot complete packet request. |

The power on sequence is shown in the figure below.



# 7 Software Integration

## 7.1 Software Integration Overview

### 7.1.1 Communication Protocol

zForce AIR Touch Sensors can communicate with a host system through USB HID or I2C transport. The content of the communication (presentation layer) is encoded in ASN.1 (except for in HID Touch Digitizer mode). Read more under zForce® Communication Protocol (see page 34).

### 7.1.2 zForce SDK

The zForce Software Development Kit (SDK) is a function library that uses the communication protocol and enables communication with a sensor without considering any ASN.1 encoded data. Read more under the separate SDK documentation.

## 7.2 Initializing Sensors

The following are procedures to initialize a sensor for USB HID Raw mode and I2C respectively. An initialization procedure is not required when using USB HID Touch Digitizer mode.

### 7.2.1 USB HID Raw Mode

Do the following procedure to initialize a sensor over USB HID Raw mode.

1. Power on the sensor.
2. Wait for HID enumeration. This signals the sensor is now booted.
3. Set the sensor to the correct Operation Mode by sending an OperationMode command to Feature Report 1. The data to send is:

   ```
   EE 12 40 02 02 00 67 0C 80 01 FF 81 01 00 82 01 00 83 01 00
   ```

   This will put the sensor in Detection Mode (which will produce touch notifications).
4. Wait for the sensor to signal that there is data to read. This comes as an Input Report 2.
5. Read the data from Feature Report 2. The data should be:

   ```
   EF 12 40 02 02 00 67 0C 80 01 FF 81 01 00 82 01 00 83 01 00
   ```

   This means the mode setting has succeeded.

The initialization is now complete. After the initialization, the sensor is enabled by default and will start sending touch notifications. To disable notifications, a Disable request must be sent. Refer to ASN.1 PDU Examples (see page 61) for examples of requests, responses and notifications.

How to communicate with the sensor is described in USB HID Transport (see page 38).

### 7.2.2 I2C

Use the following procedure to initialize the sensor over I2C.

1. Power on the sensor.
2. Wait for sensor to assert Data Ready pin (DR).
3. Initiate 2 byte I2C read operation. Payload of this read should be EE XX where XX is the amount of bytes to read in a second I2C read operation.
4. Read XX amount of bytes (number of bytes to read is indicated by second byte of first I2C Read Operation). Now read a message called BootComplete. The message should be

   ```
   F0 11 40 02 00 00 63 0B 80 01 YY 81 02 03 YY 82 02 00 YY
   ```

   where YY is usually "00" but can have another value. This signals the sensor is now booted.
5. To enable the sensor to start sending touch notifications, do the following:
   a. Sending an Enable command:

      ```
      EE 09 40 02 02 00 65 03 81 01 00
      ```

   b. Read the response. The response should be:

      ```
      EF 09 40 02 02 00 65 03 81 01 00
      ```

The initialization is now complete. When DR is asserted the sensor will send a touch notification or a new BootComplete. A BootComplete indicates that the sensor has restarted for some reason; Enable must then be set again. For more details, refer to I2C Transport (see page 35).

## 7.3 zForce® Communication Protocol

### 7.3.1 Communication Protocol Overview

#### Introduction

Neonode sensors communicate through a transport interface with ASN.1 encoded payload, with the exception of the HID Touch Digitizer mode where the payload follows the HID standard.

#### Transport Layer

Two transport interfaces are supported: USB and I2C.

#### I2C

The sensor's I2C implementation has a certain byte sequence and a signaling system to ensure stability. The sensor takes the role of an I2C slave and has the address 0x50. For more information, refer to I2C Transport (see page 35).

#### USB HID

The sensor has two USB implementations, both over HID:

1. HID Touch Digitizer. The sensor can be used as a standard HID Touch Digitizer to report touch data to the OS.
2. HID raw. This implementation uses HID Get and Set Feature Reports as a pipe to read and write.

For more information, refer to USB HID Transport (see page 38).

#### Presentation Layer

The zForce Communication Protocol is using ASN.1 encoded data to communicate with the host. ASN.1 stands for Abstract Syntax Notation One and provides a dynamic and verbose way of describing the data. For more information, refer to Presentation Layer (ASN.1) (see page 50).

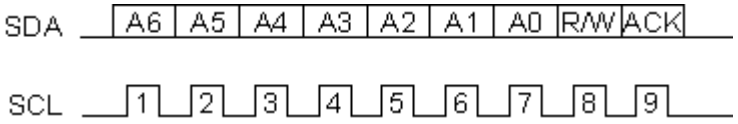
### 7.3.2 Transport Layer

#### I2C Transport

##### Sensor Address

The Slave I2C address of the sensor is 0x50 (7-bit).

When sending the 7-bit address, still 8 bits are used. The extra bit is used to inform the slave if the master is writing to it or reading from it. If the bit is 0 the master is writing to the slave. If the bit is 1 the master is reading from the slave. The 7 bit address is placed in the upper 7 bits of the byte and the Read/Write (R/W) bit is in the LSB (Least Significant Bit).



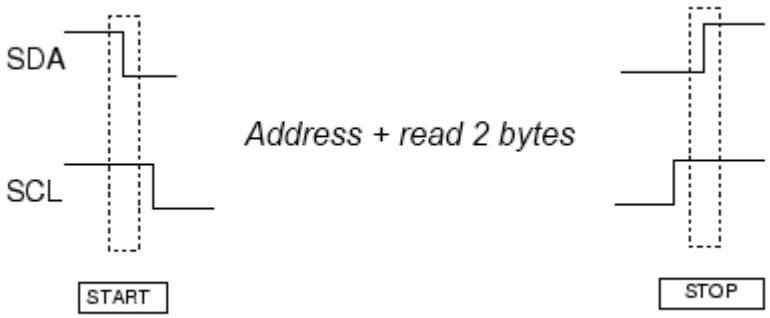

The resulting address bytes are 0xA1 (Read) and 0xA0 (Write).

##### Reading Sequence

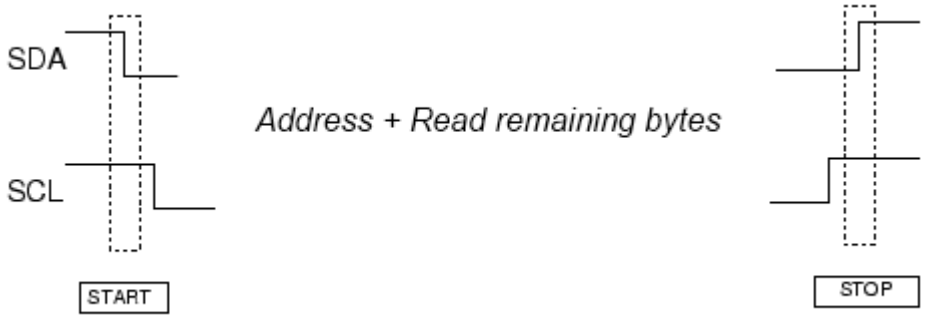
The sensor does not communicate using registers / memory accesses, so the STM32 documentation is not applicable.

To maximize performance and minimize the load on the I2C bus, the host is expected to read data in a certain sequence. This ensures a stable and reliable communication. The I2C read sequence is specified below.

1. Read the first 2 bytes: FrameStart and DataSize (number of bytes in the payload)



2. Read the remaining bytes: Payload (specified by DataSize)




Syntax

The I2C Transport Protocol is very simple and identical in both directions (Read and Write).

- Byte 1: FrameStart (A synchronization byte).
- Byte 2: DataSize (Contains the size of the data that will be transmitted. Please note that the DataSize >= 1. The reason being that :even if the Command Data size is zero, one byte will be transmitted containing the CommandID).
- Bytes 3 - : Payload encoded in ASN.1.

All bytes in this section are written in hexadecimal form, if not indicated otherwise.

The first two bytes are:

```
EE XX [payload]
```

Where EE is a constant and XX is number of bytes of ASN.1 payload.

So, for example, if the ASN.1 payload is 8 bytes long, the I2C Transport Protocol is:

```
EE 08 [payload]
```

If the payload is 25 bytes long the I2C Transport Protocol is:

```
EE 19 [payload]
```

After the first two bytes, the ASN.1 encoded payload is added.

Sending Data

Host initiates an I2C Write operation for I2C 7-bit Address 0x50 and writes the full payload including I2C Transport Protocol bytes.

If DR (Data Ready) is asserted (that is, it is signaling to the host that the host should initiate a Read operation), it is NOT permitted to initiate an I2C Write Operation.

Always wait for the corresponding response from one command before sending another command. Note that not all commands have a response command. If there is no corresponding response command, the host is free to issue another command.

Examples

The following assume DR is NOT asserted.

To send a **Request**, and the first byte of the ASN.1 payload is "EE". Do not confuse this with the EE of the I2C Protocol, even though they are the same they have completely different meaning.

```
EE XX EE [rest of payload]
```

Example: Sending an ENABLE message to the sensor:

```
EE 0B EE 09 40 02 02 00 65 03 81 01 00
```

The first two bytes (EE 0B) is the I2C Transport Protocol and the rest is the ASN.1 Protocol.

Receiving Data

The sensor triggers the DataReady signal when there is data to send. The host processor then triggers an I2C read to read the data.

1. The sensor asserts DR.
2. Host initiates a 2 byte I2C Read operation for I2C 7-bit Address 0x50.
3. The sensor fills in the 2 Bytes. These two bytes are (as described above)

   ```
   EE XX
   ```

   where XX is the length of the following ASN.1 Payload.
4. Host initiates an I2C Read operation for I2C Address and reads XX bytes (as indicated by the second byte of the I2C Transport Protocol).
5. The sensor deasserts DR.

Examples

The data received from the sensor will have the first ASN.1 payload byte be either EF or F0.

If the sensor responds to a request using a **Response**. The first byte of the ASN.1 payload is the "EF", making the whole received message, including the I2C transport protocol:

```
EE XX EF [rest of payload]
```

If a sensor sends something other than a Response, it is called a **Notification**. The first byte of the ASN.1 payload is then "F0".

```
EE XX F0 [rest of payload]
```

#### BootComplete

When the sensor has finished booting, the command BootComplete is put into the send queue. The DataReady signal is triggered and the host needs to read the data to acknowledge that the sensor is ready to operate. If the sensor powers on before the host system does, the host system needs to check if the DataReady signal is active, in which case it needs to read the data.

⚠ Do not send any commands to the sensor before BootComplete has been read. Before BootComplete is put into the send queue, the sensor is configuring hardware such as the I2C module, and therefore it is not possible to communicate with the sensor until BootComplete has been read.

### USB HID Transport

When connected via USB, the sensor communicates in Full Speed (12 Mbit/s) in two modes: Raw HID mode and HID Touch Digitizer mode. HID Touch Digitizer mode is initiated automatically as soon as the sensor is plugged in. When using Raw HID mode, the sensor must be initiated and enabled to start receiving notifications.

#### Raw HID Mode

This mode uses a mix of two HID Feature Reports and an HID Input Report to communicate with the host.

- Send data to the sensor by writing to Feature Report 1.
- Input Report 2 indicates that there is data to read
- Read data from the sensor by reading from Feature Report 2.

Refer to ASN.1 PDU Examples[7] for examples of requests, responses and notifications.

#### HID Touch Digitizer Mode

The sensor acts as a HID Input device and communicates directly with the OS.

#### HID Report Descriptor

| Item | Data |
|---|---|
| Usage Page (Digitizer) | 05 0D |
| Usage (Touch Screen) | 09 04 |
| Collection (Application) | A1 01 |

7 http://confluence.neonode.local/display/ZAMCD/.ASN.1+PDU+Examples+v1.0

| | |
|---|---|
| Report Id (4) | 85 04 |
| Usage (Contact count maximum) | 09 55 |
| Logical minimum (0) | 15 00 |
| Logical maximum (255) | 25 FF |
| Report Size (8) | 75 08 |
| Report Count (1) | 95 01 |
| Feature (Data,Value,Absolute,Non-volatile,Bit Field) | B1 02 |
| Report Id (3) | 85 03 |
| Usage (Contact count) | 09 54 |
| Input (Data,Value,Absolute,Bit Field) | 81 02 |
| Usage (Scan Time) | 09 56 |
| Logical maximum (65 535) | 27 FF FF 00 00 |
| Report Size (16) | 75 10 |
| Unit Exponent (12) | 55 0C |
| Unit (SI Linear: Time [s]) | 66 01 10 |
| Input (Data,Value,Absolute,Bit Field) | 81 02 |
| Usage (Finger) | 09 22 |
| Collection (Logical) | A1 02 |
| Usage (Tip Switch) | 09 42 |
| Logical maximum (1) | 25 01 |
| Report Size (1) | 75 01 |
| Report Count (1) | 95 01 |
| Input (Data,Value,Absolute,Bit Field) | 81 02 |
| Usage (Contact identifier) | 09 51 |
| Logical maximum (127) | 25 7F |
| Report Size (7) | 75 07 |

| | |
|---|---|
| Report Count (1) | 95 01 |
| Input (Data,Value,Absolute,Bit Field) | 81 02 |
| Usage Page (Generic Desktop) | 05 01 |
| Usage (X) | 09 30 |
| Logical maximum (65 535) | 27 FF FF 00 00 |
| Report Size (16) | 75 10 |
| Report Count (1) | 95 01 |
| Input (Data,Value,Absolute,Bit Field) | 81 02 |
| Usage (Y) | 09 31 |
| Input (Data,Value,Absolute,Bit Field) | 81 02 |
| Usage Page (Digitizer) | 05 0D |
| Unit Exponent (14) | 55 0E |
| Unit (SI Linear: Distance [cm]) | 65 11 |
| Usage (Width) | 09 48 |
| Usage (Height) | 09 49 |
| Report Count (2) | 95 02 |
| Input (Data,Value,Absolute,Bit Field) | 81 02 |
| End Collection | C0 |
| Usage (Finger) | 09 22 |
| Collection (Logical) | A1 02 |
| Usage (Tip Switch) | 09 42 |
| Logical maximum (1) | 25 01 |
| Report Size (1) | 75 01 |
| Report Count (1) | 95 01 |
| Input (Data,Value,Absolute,Bit Field) | 81 02 |
| Usage (Contact identifier) | 09 51 |

| | |
|---|---|
| Logical maximum (127) | 25 7F |
| Report Size (7) | 75 07 |
| Report Count (1) | 95 01 |
| Input (Data,Value,Absolute,Bit Field) | 81 02 |
| Usage Page (Generic Desktop) | 05 01 |
| Usage (X) | 09 30 |
| Logical maximum (65 535) | 27 FF FF 00 00 |
| Report Size (16) | 75 10 |
| Report Count (1) | 95 01 |
| Input (Data,Value,Absolute,Bit Field) | 81 02 |
| Usage (Y) | 09 31 |
| Input (Data,Value,Absolute,Bit Field) | 81 02 |
| Usage Page (Digitizer) | 05 0D |
| Unit Exponent (14) | 55 0E |
| Unit (SI Linear: Distance [cm]) | 65 11 |
| Usage (Width) | 09 48 |
| Usage (Height) | 09 49 |
| Report Count (2) | 95 02 |
| Input (Data,Value,Absolute,Bit Field) | 81 02 |
| End Collection | C0 |
| Usage (Finger) | 09 22 |
| Collection (Logical) | A1 02 |
| Usage (Tip Switch) | 09 42 |
| Logical maximum (1) | 25 01 |
| Report Size (1) | 75 01 |
| Report Count (1) | 95 01 |

| | |
|---|---|
| Input (Data,Value,Absolute,Bit Field) | 81 02 |
| Usage (Contact identifier) | 09 51 |
| Logical maximum (127) | 25 7F |
| Report Size (7) | 75 07 |
| Report Count (1) | 95 01 |
| Input (Data,Value,Absolute,Bit Field) | 81 02 |
| Usage Page (Generic Desktop) | 05 01 |
| Usage (X) | 09 30 |
| Logical maximum (65 535) | 27 FF FF 00 00 |
| Report Size (16) | 75 10 |
| Report Count (1) | 95 01 |
| Input (Data,Value,Absolute,Bit Field) | 81 02 |
| Usage (Y) | 09 31 |
| Input (Data,Value,Absolute,Bit Field) | 81 02 |
| Usage Page (Digitizer) | 05 0D |
| Unit Exponent (14) | 55 0E |
| Unit (SI Linear: Distance [cm]) | 65 11 |
| Usage (Width) | 09 48 |
| Usage (Height) | 09 49 |
| Report Count (2) | 95 02 |
| Input (Data,Value,Absolute,Bit Field) | 81 02 |
| End Collection | C0 |
| Usage (Finger) | 09 22 |
| Collection (Logical) | A1 02 |
| Usage (Tip Switch) | 09 42 |
| Logical maximum (1) | 25 01 |
| Report Size (1) | 75 01 |

| | |
|---|---|
| Report Count (1) | 95 01 |
| Input (Data,Value,Absolute,Bit Field) | 81 02 |
| Usage (Contact identifier) | 09 51 |
| Logical maximum (127) | 25 7F |
| Report Size (7) | 75 07 |
| Report Count (1) | 95 01 |
| Input (Data,Value,Absolute,Bit Field) | 81 02 |
| Usage Page (Generic Desktop) | 05 01 |
| Usage (X) | 09 30 |
| Logical maximum (65 535) | 27 FF FF 00 00 |
| Report Size (16) | 75 10 |
| Report Count (1) | 95 01 |
| Input (Data,Value,Absolute,Bit Field) | 81 02 |
| Usage (Y) | 09 31 |
| Input (Data,Value,Absolute,Bit Field) | 81 02 |
| Usage Page (Digitizer) | 05 0D |
| Unit Exponent (14) | 55 0E |
| Unit (SI Linear: Distance [cm]) | 65 11 |
| Usage (Width) | 09 48 |
| Usage (Height) | 09 49 |
| Report Count (2) | 95 02 |
| Input (Data,Value,Absolute,Bit Field) | 81 02 |
| End Collection | C0 |
| Usage (Finger) | 09 22 |
| Collection (Logical) | A1 02 |
| Usage (Tip Switch) | 09 42 |
| Logical maximum (1) | 25 01 |

| | |
|---|---|
| Report Size (1) | 75 01 |
| Report Count (1) | 95 01 |
| Input (Data,Value,Absolute,Bit Field) | 81 02 |
| Usage (Contact identifier) | 09 51 |
| Logical maximum (127) | 25 7F |
| Report Size (7) | 75 07 |
| Report Count (1) | 95 01 |
| Input (Data,Value,Absolute,Bit Field) | 81 02 |
| Usage Page (Generic Desktop) | 05 01 |
| Usage (X) | 09 30 |
| Logical maximum (65 535) | 27 FF FF 00 00 |
| Report Size (16) | 75 10 |
| Report Count (1) | 95 01 |
| Input (Data,Value,Absolute,Bit Field) | 81 02 |
| Usage (Y) | 09 31 |
| Input (Data,Value,Absolute,Bit Field) | 81 02 |
| Usage Page (Digitizer) | 05 0D |
| Unit Exponent (14) | 55 0E |
| Unit (SI Linear: Distance [cm]) | 65 11 |
| Usage (Width) | 09 48 |
| Usage (Height) | 09 49 |
| Report Count (2) | 95 02 |
| Input (Data,Value,Absolute,Bit Field) | 81 02 |
| End Collection | C0 |
| Usage (Finger) | 09 22 |
| Collection (Logical) | A1 02 |
| Usage (Tip Switch) | 09 42 |

| | |
|---|---|
| Logical maximum (1) | 25 01 |
| Report Size (1) | 75 01 |
| Report Count (1) | 95 01 |
| Input (Data,Value,Absolute,Bit Field) | 81 02 |
| Usage (Contact identifier) | 09 51 |
| Logical maximum (127) | 25 7F |
| Report Size (7) | 75 07 |
| Report Count (1) | 95 01 |
| Input (Data,Value,Absolute,Bit Field) | 81 02 |
| Usage Page (Generic Desktop) | 05 01 |
| Usage (X) | 09 30 |
| Logical maximum (65 535) | 27 FF FF 00 00 |
| Report Size (16) | 75 10 |
| Report Count (1) | 95 01 |
| Input (Data,Value,Absolute,Bit Field) | 81 02 |
| Usage (Y) | 09 31 |
| Input (Data,Value,Absolute,Bit Field) | 81 02 |
| Usage Page (Digitizer) | 05 0D |
| Unit Exponent (14) | 55 0E |
| Unit (SI Linear: Distance [cm]) | 65 11 |
| Usage (Width) | 09 48 |
| Usage (Height) | 09 49 |
| Report Count (2) | 95 02 |
| Input (Data,Value,Absolute,Bit Field) | 81 02 |
| End Collection | C0 |
| End Collection | C0 |

| | |
|---|---|
| Usage Page (Vendor-defined 0xFF00) | 06 00 FF |
| Usage (Vendor-defined 0x0001) | 09 01 |
| Collection (Application) | A1 01 |
| Report Id (1) | 85 01 |
| Usage (Vendor-defined 0x0001) | 09 01 |
| Report Size (8) | 75 08 |
| Report Count (1) | 95 01 |
| Logical minimum (0) | 15 00 |
| Logical maximum (255) | 25 FF |
| Feature (Data,Value,Absolute,Non-volatile,Bit Field) | B1 02 |
| Usage (Vendor-defined 0x0002) | 09 02 |
| Report Count (64) | 95 40 |
| Feature (Data,Value,Absolute,Volatile,Buffered Bytes) | B2 82 01 |
| Report Id (2) | 85 02 |
| Usage (Vendor-defined 0x0001) | 09 01 |
| Report Count (1) | 95 01 |
| Feature (Data,Value,Absolute,Non-volatile,Bit Field) | B1 02 |
| Usage (Vendor-defined 0x0002) | 09 02 |
| Report Count (64) | 95 40 |
| Feature (Data,Value,Absolute,Volatile,Buffered Bytes) | B2 82 01 |
| Usage (Vendor-defined 0x0003) | 09 03 |
| Report Size (1) | 75 01 |
| Logical maximum (1) | 25 01 |
| Report Count (1) | 95 01 |
| Input (Data,Value,Absolute,Bit Field) | 81 02 |
| Report Size (7) | 75 07 |

| | |
|---|---|
| Input (Constant,Array,Absolute,Bit Field) | 81 01 |
| Report Size (8) | 75 08 |
| Report Id (128) | 85 80 |
| Usage (Vendor-defined 0x0001) | 09 01 |
| Feature (Constant,Value,Absolute,Non-volatile,Bit Field) | B1 03 |
| Report Id (130) | 85 82 |
| Usage (Vendor-defined 0x0001) | 09 01 |
| Feature (Constant,Value,Absolute,Non-volatile,Bit Field) | B1 03 |
| End Collection | C0 |

Parsed reports by Report ID

| **Input Report 2** | | |
|---|---|---|
| **Bit offset** | **Bit count** | **Description** |
| 0 | 1 | Vendor-defined 0x0003 |
| 1 | 7 | (Not used) |
| **Input Report 3** | | |
| **Bit offset** | **Bit count** | **Description** |
| 0 | 8 | Contact count |
| 8 | 16 | Scan Time |
| 24 | 1 | Tip Switch |
| 25 | 7 | Contact identifier |
| 32 | 16 | X |
| 48 | 16 | Y |
| 64 | 16 | Width |
| 80 | 16 | Height |
| 96 | 1 | Tip Switch |

| | | |
|---|---|---|
| 97 | 7 | Contact identifier |
| 104 | 16 | X |
| 120 | 16 | Y |
| 136 | 16 | Width |
| 152 | 16 | Height |
| 168 | 1 | Tip Switch |
| 169 | 7 | Contact identifier |
| 176 | 16 | X |
| 192 | 16 | Y |
| 208 | 16 | Width |
| 224 | 16 | Height |
| 240 | 1 | Tip Switch |
| 241 | 7 | Contact identifier |
| 248 | 16 | X |
| 264 | 16 | Y |
| 280 | 16 | Width |
| 296 | 16 | Height |
| 312 | 1 | Tip Switch |
| 313 | 7 | Contact identifier |
| 320 | 16 | X |
| 336 | 16 | Y |
| 352 | 16 | Width |
| 368 | 16 | Height |
| 384 | 1 | Tip Switch |
| 385 | 7 | Contact identifier |
| 392 | 16 | X |
| 408 | 16 | Y |

| | | |
|---|---|---|
| 424 | 16 | Width |
| 440 | 16 | Height |
| | | |
| **Feature Report 1** | | |
| **Bit offset** | **Bit count** | **Description** |
| 0 | 8 | Vendor-defined 0x0001 |
| 8 | 512 | Vendor-defined 0x0002 |
| **Feature Report 2** | | |
| **Bit offset** | **Bit count** | **Description** |
| 0 | 8 | Vendor-defined 0x0001 |
| 8 | 512 | Vendor-defined 0x0002 |
| **Feature Report 4** | | |
| **Bit offset** | **Bit count** | **Description** |
| 0 | 8 | Contact count maximum |
| **Feature Report 128** | | |
| **Bit offset** | **Bit count** | **Description** |
| 0 | 8 | Vendor-defined 0x0001 |
| **Feature Report 130** | | |
| **Bit offset** | **Bit count** | **Description** |
| 0 | 8 | Vendor-defined 0x0001 |

Known limitations

A race condition might occur when calling the functions HidD_GetFeature, HidD_GetManufacturerString, HidD_GetProductString or HidD_GetSerialNumberString simultaneously. A Mutex lock might be needed to only call one of them at the same time, otherwise they could fail randomly.

### 7.3.3 Presentation Layer (ASN.1)

#### ASN.1 PDU Description

##### PDU Definition

Download the ASN.1 PDU definition file from https://support.neonode.com/docs/display/downloads.

##### Introduction

This document describes the host interface to a sensor. The host interface is the means for which an outside device can communicate with the sensor. The communication protocol is using ASN.1 (Abstract Syntax Notation One) which is a standard and a notation that is described in ISO/IEC 8824. Rules in the standard enables representation of data that is agnostic to machine specific implementations.

This document is the specification of what corresponds to layers 6 and 7 of the OSI model[8]:

- Layer 6: Presentation Layer: Encoding and decoding of application layer messages
- Layer 7: Application Layer: Communication by sending messages.

##### Message Specification

All communication is done by sending messages in a format specified using ASN.1. The message is either a request, response or notification. The host sends a request to the sensor, and the device responds with a response. The device may send notifications to the host at any time.

All messages contain a virtual device address. Virtual devices are functionally isolated from each other, and communicate separately with the host. The virtual device called "Platform" represents the system. The virtual device called "Air" is a touch sensor. A sensor will always contain one platform virtual device and can contain any number of instances of other virtual device types.

##### Abstract Syntax Notation One (ASN.1)

ASN.1 is a means and method of abstracting the data definition from the message format being used. The data that is to be transferred is expressed as an object. The object definition then has a set of Encoding Rules applied to produce the actual data format. Among the more common Encoding Rules are Basic Encoding Rules (BER), and its subset Distinguished Encoding Rules (DER), and XML Encoding Rules (XER).

The encoding rules are what determines how the data is encoded and decoded, and there are quite a big variance depending on which encoding rules are used. XER produces "fully readable" XML data, whereas BER and DER gives a lot more space preserving binary encoding of the same set of data.

References:

- ITU-T ASN.1:2008 Publications[9]
- Layman's Guide to a subset of ASN.1, BER and DER[10]
- OSS Nokalva's Resource pages[11]

8 https://en.wikipedia.org/wiki/OSI_model
9 http://www.itu.int/ITU-T/studygroups/com17/languages/
10 http://luca.ntop.org/Teaching/Appunti/asn1.html
11 http://www.oss.com/resources/resources.html

- ITU-T Introduction to ASN.1[12]

ASN.1 Definitions
Defining objects in the ASN.1 syntax is done by defining one or more Protocol Data Unit (PDU) on a top level of a definition file. Any data unit that is defined on top level and is not referenced by any other data unit is considered to be a top level unit. The definitions then part by part describe what this PDU can contain.

What can be defined is generally divided into two types of objects; Constructed types and Primitive types. The constructed type contains further definitions defining what the type can contain, whereas the primitive types can be considered as an endpoint, or a leaf node, where they are the types that contains any actual data itself. Examples of constructed types are Sequence, Choice and Set. Examples of primitive types are Integer, Octet String (string of data bytes, binary blob) and IA5String (printable ascii string).

All data types (both constructed and primitive) inherit their traits from an already existing type, which may or may not be defined from another existing type. In the extension all types are however based on a set of basic data types, called the Universal types. As such, the encoding rules only need to cover these universal types to be able to encode any definitions, with a little help of hints in the definition.

The types of hints given in the definition are identifier numbers and definition categories.

In addition to the basic data definitions it is also possible to define cardinality, optionality, extensibility and size as constraints.

zForce PDU
All messages are instances of a Protocol Data Unit (PDU) defined in the zForce definition file.

The top level PDU `ProtocolMessage` contains a choice of either a `request`, `response` or `notification` child. The `request` and `response` are of the same PDU `Message`, and the `notification` is of the PDU `Notification`.

The `Message` PDU contains the `deviceAddress` and a command. The `deviceAddress` specifies a virtual device within the sensor, which is the recipient of the request. In the request the command specifies what is being requested. In the response, the command is of the same PDU as in the reponse, but now contains the result or requested information of the request.

The `Notification` PDU contains the `deviceAddress` and the `notificationMessage`. Some notifications contain the `notificationTimestamp`.

Encoding Rules
When applying a set of Encoding Rules to the ASN.1 definitions, one gets the data format used for the interchange.

zForce uses the Distinguised Encoding Rules (DER).

In this document, examples will be given with the human readable Generic String Encoding Rules (GSER). Here is one such example:

```
request: {
  deviceAddress '0200'H,
  enable {
    enable NULL
  }
}
```

GSER Notation
The examples written down in GSER notation will hopefully be quite easy to read, but some parts are not necessarily obvious. So, as a lead to it here's a list of used notation:

12 http://www.itu.int/en/ITU-T/asn1/Pages/introduction.aspx

Sequences and/or sets (items containing sub items) are shown as curly brackets: **{ <sub elements> }**

Values encoded as octet strings are written as hexadecimal octets enclosed within single quotes and suffixed with H: **'FF9900'H**

Bit strings are also shown as octet strings when the number of bits is a multiple of 8, otherwise each bit is shown as a single 1 or 0, and suffixed with B: **'11010101001'B**

In a choice element, the selected type is denoted by its name followed by a colon: **request:**

Full reference at Generic String Encoding Rules (GSER) for ASN.1 Types[13].

Tools
The tool FFASN1Dump[14] can transcode from GSER to DER:

```
ffasn1dump -I gser -O der zforce_pdu_def.asn ProtocolMessage <input file> <output file>
```

Currently ffasn1dump does not handle identifiers for Integer values. For this reason, they need to be replaced with numerical values.

Application Interface

The application interface specifies what requests can be made and what responses and notifications they activate. Messages will be specified using the templates below, and specifying only:

- address
- request command
- command response
- notification

**request**

```
request: {
  deviceAddress <address>,
  <request command>
}
```

**response**

```
response: {
  deviceAddress <address>,
  <command response>
}
```

13 https://tools.ietf.org/html/rfc3641#page-6
14 http://www.bellard.org/ffasn1/

**notification**

```
notification: {
  deviceAddress <address>,
  <notification>
  notificationTimestamp <timestamp>
}
```

address

Octet string with 2 octets; device type and index: '<device type><index>'H

device types:

- 0: Platform
- 1: Core
- 2: Air
- 3: Plus
- 4: Lighting

Eg. '0000'H for platform.

timestamp

Integer representing int16 counting at 32768 Hz.

Platform

Platform commands relate to generic system information and settings.

These are using the platform address:

**address**

```
'0000'H
```

Device Information

Get product id and FW version.

**request command**

```
  deviceInformation {
  }
```

**command response**

```
  deviceInformation {
    platformInformation {
      platformVersionMajor 7,
```

```
    platformVersionMinor 0,
    protocolVersionMajor 1,
    protocolVersionMinor 5,
    firmwareVersionMajor 1,
    firmwareVersionMinor 0,
    hardwareIdentifier "Demo board",
    hardwareVersion "R2",
    asicType nn1002,
    numberOfAsics 1,
    mcuUniqueIdentifier '340038000E51333439333633'H,
    projectReference "DEMO_1.0_REL",
    platformReference "734cebd",
    buildTime "16:01:14",
    buildDate "2016-07-01"
      }
}
```

The fields have the following meaning:

- versions:
  - platform: FW platform version
  - protocol: communication protocol version
  - firmware: product FW version
- hardware: Product hardware, with configuration and revision
- asic: Which type of the Neonode optical scanner ASICs is used, and count
- mcuUniqueIdentifier: Identifier created at chip manufacturing
- Reference: FW GIT tags or hashes for:
  - project: product specific. Uniquely identifies FW revision
  - platform: Uniquely identifies generic firmware base commit
- build: Date and time of build in Central European Time.

Device Count

Enumerate the available virtual devices.

**request command**

```
deviceCount {
}
```

**command response**

```
deviceCount {
  totalNumberOfDevices 1,
  airDevices 1
}
```

Device type instances are indexed from zero. This response means that the only virtual device available is Air[0].

### Frequency

Change the update frequency of all touch sensors globally.

These update frequencies can be set, if enabled in the product:

- finger: Activated when objects with characteristics matching regular fingers are detected
- stylus: Activated for narrow stylus-like objects
- idle: Activated when no objects are detected, in order to minimize power usage.

The unit is Hz.

**request command**

```
frequency {
  finger 30,
  idle 10
}
```

The response confirms the setting for the frequencies supported on the product:

**command response**

```
frequency {
  finger 30,
  idle 10
}
```

In this example, the touch sensor update frequency will be 30 Hz as long as objects were recently detected. When not, the frequency will drop to 10 Hz.

### Touch Sensor

There are a number of different touch sensor products that can co-exist on the same physical device. They have some product-specific commands, but the ones listed here are general.

Air will be used as example, which means the device address will be that of the first Air virtual device:

**address**

```
'0200'H
```

### Operation Mode

Choose what processing should be done on sensor signals, and what diagnostics should be exposed.

This example sets it to normal object detection:

**request command**

```
operationMode {
  detection TRUE,
  signals FALSE,
  ledLevels FALSE,
  detectionHid FALSE,
  gestures FALSE
}
```

**command response**

```
operationMode {
  detection TRUE,
  signals FALSE,
  ledLevels FALSE,
  detectionHid FALSE
}
```

As can be seen gestures is missing in the response. This is valid response, since the device has been built with a subset of the protocol, or an older forward-compatible version.

Touch Format

Retrieve the binary format of the detected objects.

**request command**

```
touchFormat {
}
```

**command response**

```
touchFormat {
  touchDescriptor { id, event, loc-x-byte1, loc-x-byte2, loc-y-byte1, loc-y-byte2, size-x-byte1, size-y-byte1, confidence }
}
```

The touchDescriptor is a bit string, where each bit signifies one byte of payload being included in the touchNotification octet strings. A touchNotification is the concatenation of those bytes. The following table lists all bits. Bits flagged by the example touchDescriptor are marked green.

| Name | Description | Comment |
|---|---|---|
| id | Touch Identifier | |

| event | Up/Down/Move | 0=Down; 1=Move; 2=Up; 3=Invalid; 4=Ghost |
|---|---|---|
| loc-x-byte1 | X coordinate | |
| loc-x-byte2 | X expanded | for higher precision |
| loc-x-byte3 | X expanded | for higher precision |
| loc-y-byte1 | Y coordinate | |
| loc-y-byte2 | Y expanded | for higher precision |
| loc-y-byte3 | Y expanded | for higher precision |
| loc-z-byte1 | Z coordinate | |
| loc-z-byte2 | Z expanded | for higher precision |
| loc-z-byte3 | Z expanded | for higher precision |
| size-x-byte1 | X size | |
| size-x-byte2 | X size | for higher precision |
| size-x-byte3 | X size | for higher precision |
| size-y-byte1 | Y size | |
| size-y-byte2 | Y size | for higher precision |
| size-y-byte3 | Y size | for higher precision |
| size-z-byte1 | Z size | |
| size-z-byte2 | Z size | for higher precision |
| size-z-byte3 | Z size | for higher precision |
| orientation | Orientation | Hand orientation |
| confidence | Confidence | |
| pressure | Pressure | |

Location and size coordinates can be specified with up to 3 bytes. The byte order in decreasing significance - big-endian. For example:

- 1 byte: location x = loc-x-byte1
- 2 bytes: location x = (loc-x-byte1 << 8) + loc-x-byte2
- 3 bytes: location x = (loc-x-byte1 << 16) + (loc-x-byte2 << 8) + loc-x-byte3

Location is signed, and size is not.

The location coordinate scale is one of two systems, depending on which detector is used:

- Physical: Robair Air and Core detectors: The unit is 0.1 mm. A coordinate value of 463 thus means 46.3 mm from origin.
- Relative: Triangles and Shape Air detectors: Fraction of the largest screen dimension as fixed point with 14 bits after the radix point (q14). On a widescreen display, the horizontal axis ranges [0, $2^{14}$[, and vertical [0, $2^{14}$ * 9/16[ ([0, 16383], [0, 9215]).

The zForce AIR Touch Sensor uses Robair, thus the unit is 0.1 mm.

Size is in mm.

Confidence and pressure is fraction of full in percent.

Enable Execution

This command will activate the touch sensor, and notifications of detections will start streaming.Enable

**request command**

```
enable {
  enable 0
}
```

**command response**

```
enable {
  reset NULL
}
```

This is actually incorrect response. It should respond enable.

To deactivate the touch sensor, send the disable command:

**request command**

```
enable {
  disable NULL
}
```

**command response**

```
enable {
  disable NULL
}
```

Touch Notifications

Objects detection come as the descriptor specified. Every concurrently tracked object will be in its own touchNotification payload.

**notification**

```
notificationMessage touchNotifications: {
    '0001013600730A0A64'H
  },
```

The following table shows its value interpreted with the touch descriptor.

| Name | Description | Comment | Value |
|---|---|---|---|
| id | Touch Identifier | | 0 |
| event | Up/Down/Move | 0=Down; 1=Move; 2=Up; 3=Invalid; 4=Ghost | 1 |
| loc-x-byte1 | X coordinate | | 1 |
| loc-x-byte2 | X expanded | for higher precision | 54 |
| loc-y-byte1 | Y coordinate | | 0 |
| loc-y-byte2 | Y expanded | for higher precision | 115 |
| size-x-byte1 | X size | | 10 |
| size-y-byte1 | Y size | | 10 |
| confidence | Confidence | | 100 |

The touchNotification above (from a Core device) means "Object 0 moved. Location is (31.0, 11.5) mm. Size is 10x10 mm. Confidence in object presence is 100%."

Information

The device information command retrieves some information about this virtual device instance.

**request command**

```
  deviceInformation {
  }
```

**command response**

```
  deviceInformation {
```

```
  deviceInstanceInformation {
    productVersionMajor 1,
    productVersionMinor 38,
    physicalWidth 1584,
    physicalHeight 1341,
    numberOfSignalAxes 0
  }
}
```

The response contains the deviceInstanceInformation structure, with the following fields:

- productVersion: Virtual device type specific version.
- physical: Size in unit 0.1 mm. See section Touch Format for the relationship to location coordinates.
- numberOfSignalAxes: Only applicable for Core devices. The number of sensor arrays, each monitoring one dimensions/axis of a touch sensor. Generally 2.

### Configuration

Some touch sensor device configuration can be changed at run-time. The request command and command response are identical, except some configuration items in the request may be omitted to leave them in their current state.

For instance, to set object size restrictions only, omit all other items:

**request command**

```
deviceConfiguration {
  sizeRestriction {
    maxSizeEnabled TRUE,
    maxSize 100,
    minSizeEnabled FALSE
  }
}
```

The command response contains the state of all configuration items:

**command response**

```
deviceConfiguration {
  subTouchActiveArea {
    lowBoundX 0,
    lowBoundY 0,
    highBoundX 1584,
    highBoundY 1341,
    reverseX FALSE,
    reverseY FALSE,
    flipXY FALSE,
    offsetX 0,
    offsetY 0
  },
  sizeRestriction {
    maxSizeEnabled FALSE,
```

```
      maxSize 0,
      minSizeEnabled FALSE,
      minSize 0
    },
    detectionMode default,
    numberOfReportedTouches 2,
    hidDisplaySize {
      x 1584,
      y 1341
    }
  }
```

The items are:

- subTouchActiveArea: Crop the touch sensor to a rectangle between the specified low and high coordinates in each dimension. Offset can be applied and flip the X and Y axis. Origin of reported locations is set to low coordinates, or if reversed, the high coordinate with increasing coordinates toward low.
- sizeRestriction: Limit detection to objects within this size range. Unit is 0.1 mm.
- detectionMode: One of:
    - default: finger and stylus
    - finger: Finger only
    - mergeTouches: Merges all touch objects into one
    - insensitiveFTIR: Unsupported
- numberOfReportedTouches: Maximum number of reported tracked objects.
- hidDisplaySize: Scaling the coordinate system when using the sensor in HID Touch Digitizer mode.

## ASN.1 PDU Examples

### Encoding Integers

ASN.1 encoded integers, for example values representing scanning frequency or touch active area size, are represented by one or more bytes:

- If the integer is between 0 and 127, it is represented by one byte (`00` to `7F`).
- If the integer is between 128 and 32767, it is represented by two bytes (`00 80` to `7F FF`).

The length of the message therefore varies depending on parameter values.

### Enabling Sensors

Do the following to enable a sensor, that is, tell a sensor to start sending touch notifications.

1. Enable the sensor by sending an Enable command:

    ```
    EE 09 40 02 02 00 65 03 81 01 00
    ```

    This enables the sensor to send touch notifications.
2. Read the response. The response should be:

```
EF 09 40 02 02 00 65 03 81 01 00
```

3. After this, wait for the sensor to indicate it has something to send, which means that the device will send a Touch Notification or a BootComplete. A BootComplete indicates that the device has restarted for some reason, so rerun the initialization and enable the sensor to start receiving touch notifications again.

### Disabling Sensors

Do the following to disable a sensor, that is, tell a sensor to stop sending touch notifications.

1. Disable the sensor by sending the following command:

   ```
   EE 08 40 02 02 00 65 02 80 00
   ```

   This disables the sensor to send touch notifications.
2. Read the response. The response should be:

   ```
   EF 08 40 02 02 00 65 02 80 00
   ```

### Device Configuration

Device configuration is a command that includes different settings for the sensor, for example the Touch Active Area. When sending a device configuration message, all of the settings specified are not required to be sent, however the response from the sensor will include the full device configuration message.

This is an example message that changes the touch active area using the Device Configuration command:

```
EE 1A 40 02 02 00 73 14 A2 12 80 02 00 B5 81 01 43 82 02 06 98 83 02 04 34 85 01 FF
```

The message is explained in the table below:

| Part | Description |
|---|---|
| `EE 1A` | ID for "Request" followed by length of total payload (0x1A = 26 bytes) |
| `40 02 02 00` | Device address (always the same for the zForce AIR Touch Sensor) |
| `73 14` | ID for Device Configuration followed by the length of the total Device Configuration payload (20 bytes) |
| `A2 12` | ID for Sub Touch Active Area followed by the length of Sub Touch Active Area payload |
| `80 02 00 B5` | ID for xMin followed by payload length and an integer value (0x00B5 = 181) |
| `81 01 43` | ID for yMin followed by payload length and an integer value (0x43 = 67) |

| | |
|---|---|
| `82 02 06 98` | ID for xMax followed by payload length and an integer value (0x0698 = 1688) |
| `83 02 04 34` | ID for yMax followed by payload length and an integer value (0x0434 = 1076) |
| `85 01 FF` | ID for "Invert y axis" followed by length of payload and a Boolean (0xFF= True) |

The response from the sensor to the above message will contain the full device configuration message, also the parts not set in the Request.

Setting the Touch Active Area should be done before enabling the sensor with the ENABLE request.

### Setting Frequency

To set the finger frequency to 200 Hz and idle frequency to 63 Hz use the following command:

```
EE 0D 40 02 00 00 68 07 80 02 00 C8 82 01 3F
```

> The zForce AIR Touch Sensor does not support Stylus mode, and setting the stylus frequency does not do anything.

### Decoding Touch Notifications

A packet can contain one, two or three Touches, and optionally a timestamp. On packets where the timestamp is not included, the 58 02 TT TT bytes are missing from the end and the length bytes are adjusted accordingly, For One touch (below), F0 15 in the beginning will be F0 11 and A0 0F in the middle will be A0 0B. The same bytes are decreased by 4 for Two and Three touches.

### One Touch

A packet that contains one touch will look like:

```
F0 15 40 02 02 00 A0 0F 42 09 II VV XX XX YY YY GG HH JJ 58 02 TT TT
```

where the data is defined as follows:

| Syntax | Meaning |
|---|---|
| II | ID of this specific touch object. More than one can be tracked simultaneously. |
| VV | Event:<br>00 = DOWN (new object). 01 = MOVE (existing object has moved). 02 = UP (existing object is no longer being tracked). |
| XX XX | X Coordinate of the object. This is in Network Byte Order / Big Endian / Motorola Endian. |

| | |
|---|---|
| YY YY | Y Coordinate of the object. See above. |
| GG | Size of object on the X Axis. |
| HH | Size of object on the Y Axis. |
| JJ | This value must be ignored. |
| TT TT | Timestamp of the touch. |

Two Touches

A packet that contains two Touches will look like:

```
F0 20 40 02 02 00 A0 1A 42 09 II VV XX XX YY YY GG HH JJ 42 09 II VV XX XX YY YY GG HH JJ 58 02 TT TT
```

where the first "II" and the following bytes up to and including "JJ" are from the first touch, and the second "II" and the following bytes up to and including "JJ" are from the second touch.

Three Touches

A packet that contains three Touches will look like:

```
F0 2B 40 02 02 00 A0 25 42 09 II VV XX XX YY YY GG HH JJ 42 09 II VV XX XX YY YY GG HH JJ 42 09 II VV XX XX
YY YY GG HH JJ 58 02 TT TT
```

# 8 Specifications

## 8.1 Specifications Overview

### 8.1.1 Touch Performance Specification

| Item | Specification |
|---|---|
| Input methods | Finger, hand or glove. |
| Minimum object size (diameter) | 5 mm |
| Number of touch objects | 1, 2, or more, depending on application |
| Touch accuracy | < 5 mm, for sensors ≥ 180 mm<br>< 7.5 mm, for sensors < 180 mm |
| Touch resolution | 0.1 mm |
| Touch activation force | 0 N (no activation force required) |
| Touch Active Area | Up to 345.6 x 208.5 mm. For details, refer to Product Variants (see page 6). |
| Response time | 16-46 ms (initial touch, at 33 Hz in idle mode)<br>10 ms (continuous tracking, at 100 Hz in active mode) |
| Scanning frequency | Configurable up to 900 Hz, depending on product variant. For details, refer to Scanning Frequency (see page 68). |

### 8.1.2 Technical Specification

| Item | Sensor Variant | Specification |
|---|---|---|
| **Module size** (LxHxW) | 0° Type | L x 3.46 x 14.5 mm<br>L depending on product variant. |
| | 90° Type | L x 3.46 x 15.45 mm<br>L depending on product variant. |
| **Power consumption**<br>I2C interface<br>Active mode (100 Hz) | 72 mm sensor | 57 mW |
| | 208.8 mm sensor | 80 mW |

| | | |
|---|---|---|
| | 345.6 mm sensor | 104 mW |
| **Power consumption**<br>I2C interface<br>Idle mode (25 Hz) | 72 mm sensor | 44 mW |
| | 208.8 mm sensor | 45 mW |
| | 345.6 mm sensor | 47 mW |

## 8.2 Touch Performance

### 8.2.1 Touch Object Requirement

zForce AIR Touch Sensors detect and trace objects by detecting diffusely reflected infrared light.

Requirements on the object to detect include:

- A minimum reflectance of 30% in the near IR-spectrum is needed for proper signal levels, that is, the object can not be too dark.
- Object surface must be diffuse. A glossy or mirror-like object may not scatter enough light towards correct receivers in order to generate a reliable detection.
- An object must be ≥ 5 mm to ensure sufficient signal levels. This is closely related to reflectance. A white, diffuse object may be smaller than a dark, glossy one.

### 8.2.2 Touch Accuracy

#### Specification

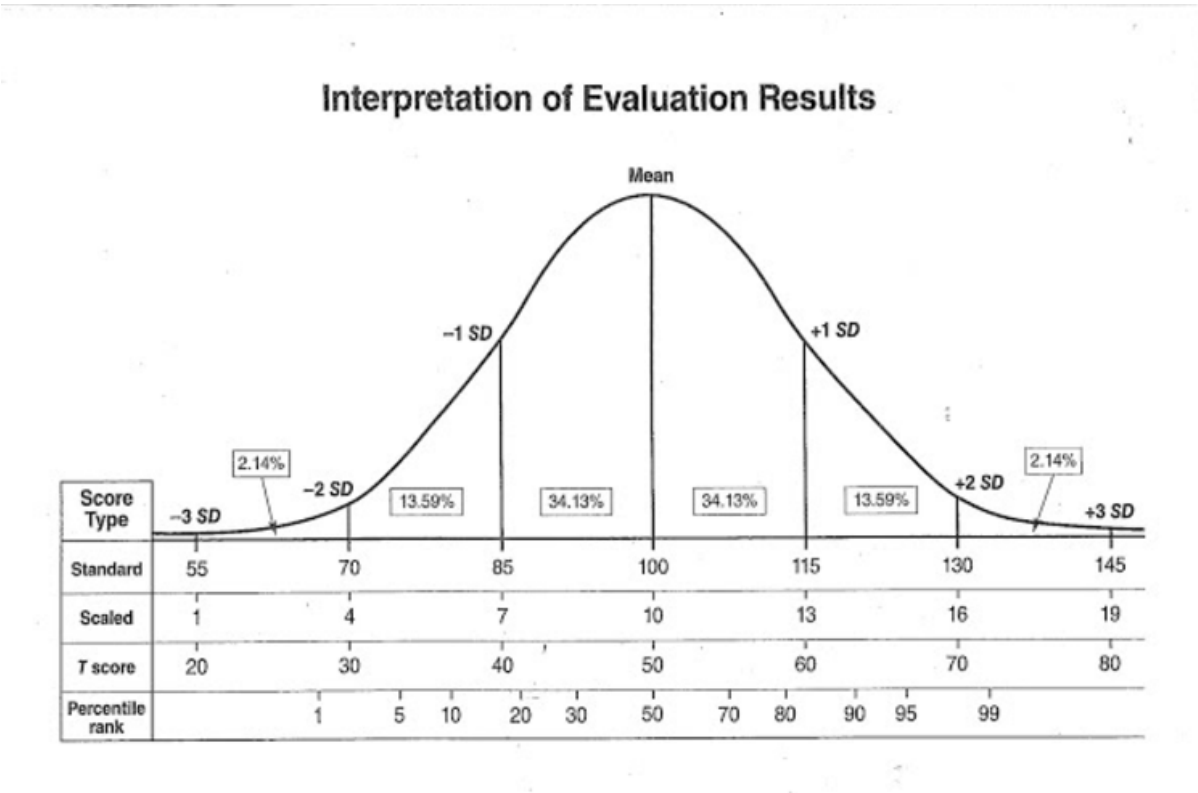
Measured touch coordinate error in X and Y axis is less or equal than the specified value for about 95% of the cases.

Touch coordinate error data is calculated by subtracting the actual position and measured position in X and Y axis.

#### Definition

Touch accuracy is defined statistically through the “Bell curve” describing the normal distribution, and a two-sigma deviation of the data. This means that the touch position reported by the zForce AIR Touch Sensor will deviate less than the specified value in 95% of the cases.

Used "Bell curve" for zForce AIR Touch Sensor statistical analysis is shown below.




Reference: PAR Psychological Assessment Resources Inc.

### 8.2.3 Response Time

The specification of response time reflects the reaction speed of a zForce AIR Touch Sensor.

#### Specification

**Initial touch**: 16-46 ms, at 33 Hz scanning frequency (default frequency in idle mode).

**Continuous tracking**: 10 ms, at 100 Hz scanning frequency (default frequency in active mode).

Increasing the scanning frequency decreases the response time.

#### Definition

Initial Touch

The specified response time for the **initial touch** starts from the instant an object is presented in the sensor's active area and stops when the sensor starts to send the first touch notification for this object. The specified response time consists of two numbers reflecting the best case and the worst case, with the average response time right in the middle.

The response time (t) can be calculated for different idle mode frequencies (f) can be calculated by the formulas below:

**Best case**: *t = 16 ms*

**Worst case**: *t = 1/f + 16 ms*

**Average**: *t = (1/f + 32 ms) / 2*

In touch applications, an object will be detected slightly before it reaches the touch surface, making the perceived response time shorter.

Continuous Tracking

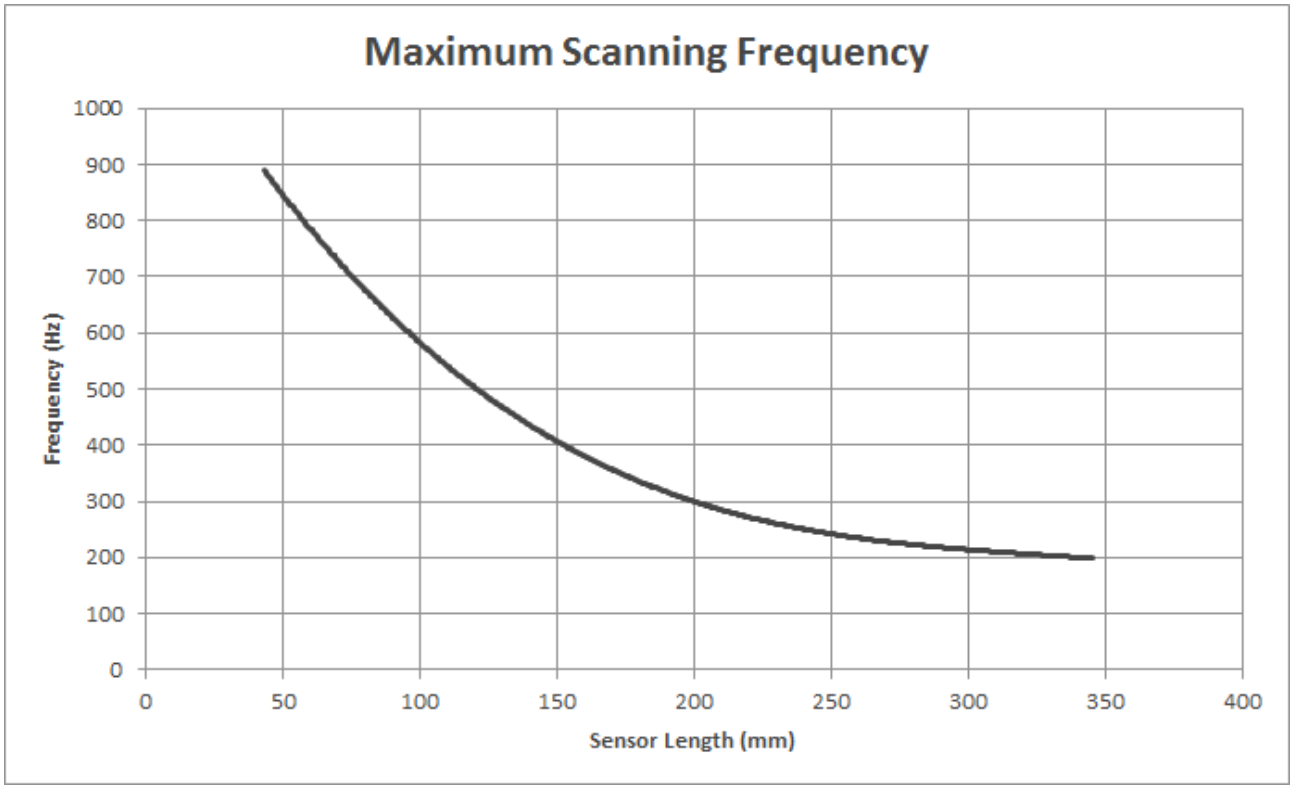
After the first touch notification, the sensor will **continuously track** and send touch notifications to update the object position. The response time is therefore defined as the time between subsequent touch notifications.

The response time (t) can be calculated for different active mode frequencies (f) can be calculated by the formula below:

*t = 1/f*

### 8.2.4 Scanning Frequency

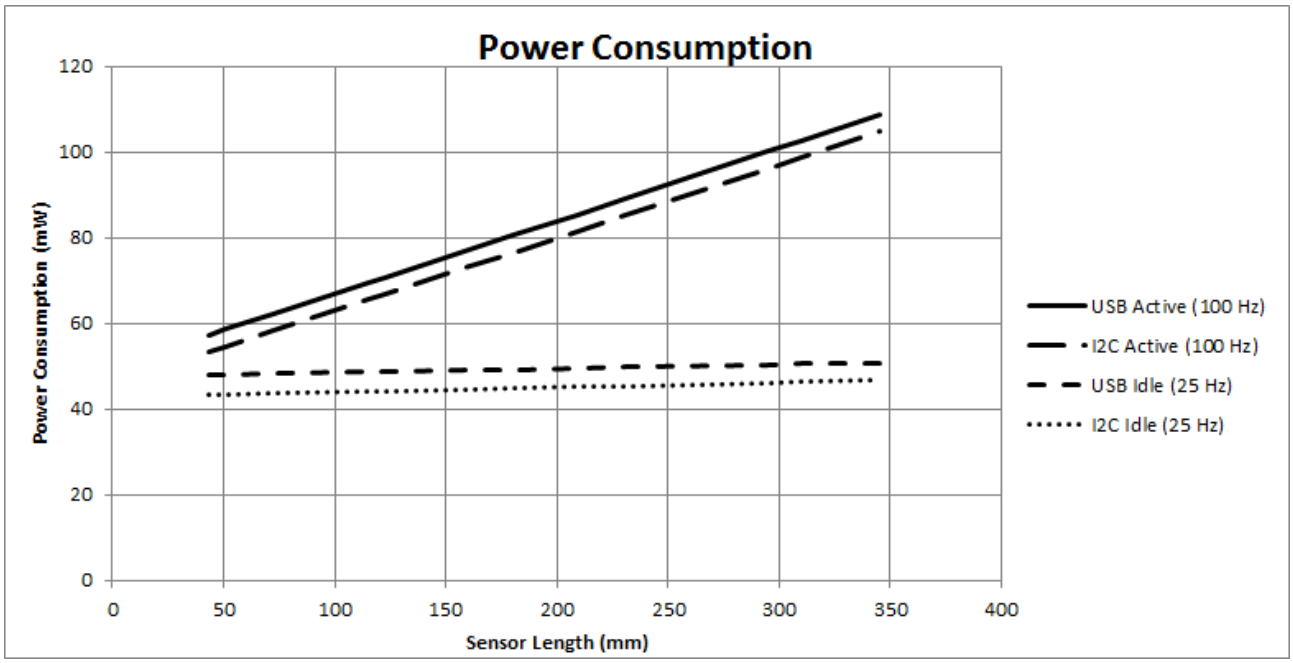
The scanning frequency can be set using the Neonode API. The default value is 100 Hz in active mode, that is, when an object is detected or tracked. The default value in idle mode, that is, when no object is detected or tracked, is 25 Hz. The maximum scanning frequency depends on the product variant (sensor length). See the chart below.



# 8.3 Power Consumption

## 8.3.1 Specification

The graph below shows the power consumption for various sensor lengths, in active and idle mode. In active mode, the scanning frequency is set to 100 Hz, and one object is presented in active area. In idle mode the scanning frequency is set to 25 Hz, with a clean active area. With higher scanning frequency or more detected objects, the power consumption might slightly higher than the values in the graph. The sensor will only be in active mode when a touch object is being detected or tracked.




## 8.3.2 Definition

The power consumption is calculated from the current consumption when supplying the sensor with 5 V.

The current consumption is, in turn, defined as the average current that goes through a sensor. This is measured from the +5V power pin and reflects how much electric energy that is consumed by the whole sensor. In real time, the current is not a stable value. Since the Touch Sensor has a low power consumption design, the processor and some peripheral circuits will switch to sleep mode during the time between two scan periods, to save power. Therefore, the current is frequently changing during run time.

According to the different working modes of the Touch Sensor, the current consumption value also changes between Active mode and Idle mode.

## 8.4 Environmental Requirements

### 8.4.1 Operating and Storage Conditions

| Condition | Operation | Storage |
|---|---|---|
| Temperature | –20°C to +65°C | –40°C to +85°C |
| Humidity | 5% to 95% | 0% to 95% |
| Altitude | ≤5000 m | ≤15 km |

### 8.4.2 ESD rating

EN55024
(61000-4-2)
Direct contact discharge: 4 kV
Indirect contact discharge: 4 kV
Air discharge: 8 kV

### 8.4.3 Agency Approvals

RoHS, IEC60825-1 Class 1

## 8.5 Electrical Requirements

### 8.5.1 Absolute Maximum Ratings

| Parameter | Max Rating | Unit |
|---|---|---|
| Supply voltage | -0.3 to 6.0 | V |
| Input voltage on I/O pins | -0.3 to 4 | V |

### 8.5.2 Recommended Operating Conditions

| Parameter | Min | Typ | Max | Unit |
|---|---|---|---|---|
| Supply voltage | 4.75 | 5.00 | 5.25 | V |

## 8.6 Optical Requirements on External Window

Most applications will require an outer cover window, for design cosmetics and protection against dust and humidity.

The optical properties on cover windows placed in front of the sensor are essential in order to maintain a high touch performance. If light is lost, scattered or diverted it will lead to shorter detection range and lower touch accuracy.

### 8.6.1 Optical Requirements

Window material must be optically clear, without absorption and have optical quality surfaces.

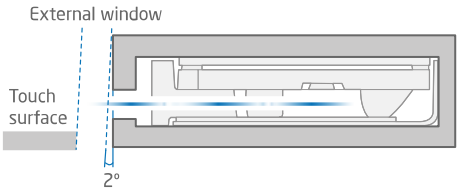
- Transmission: > **88 % at 975nm**
- Haze: **< 3%**
- Surface finish: **SP1-A2 (max Ra 0.05µm).**

Proven plastic materials include optical grade acrylic (PMMA) and polycarbonate. For glass windows, transmission at 975 nm must be verified. Many borosilicate glasses (such as Borofloat) work well, but some common window glasses show substantial absorption due to high iron content.

### 8.6.2 Geometrical Constraints

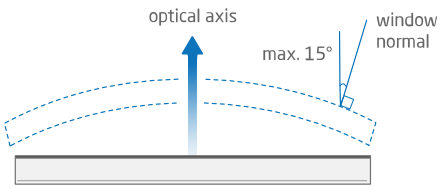
The zForce AIR Touch Sensor is an optical system that both emits and receives IR-light at different incident angles. When the light hits a transparent material, most of the light is transmitted through the material and exit on the other side. But in reality the amount of light being transmitted is angle dependent, why some shape constraints exist on windows placed in front of the sensor:

- Window surfaces must be parallel.
  A wedge, or lens shaped window will shift light beams out of the active area.
- It is a good practice to install the window at a slight angle (~2°) to reduce reflected stray light. See the image below. The angle can be up to approximately 30° without affecting performance.

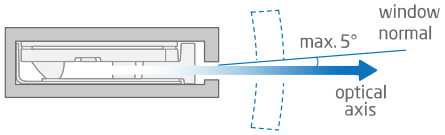



- A slight curvature on the window can be allowed.

- In z-direction, a maximum angle of 15° between window normal and sensors optical axis is recommended.



- In x-direction, the angle should be maximum 5°.



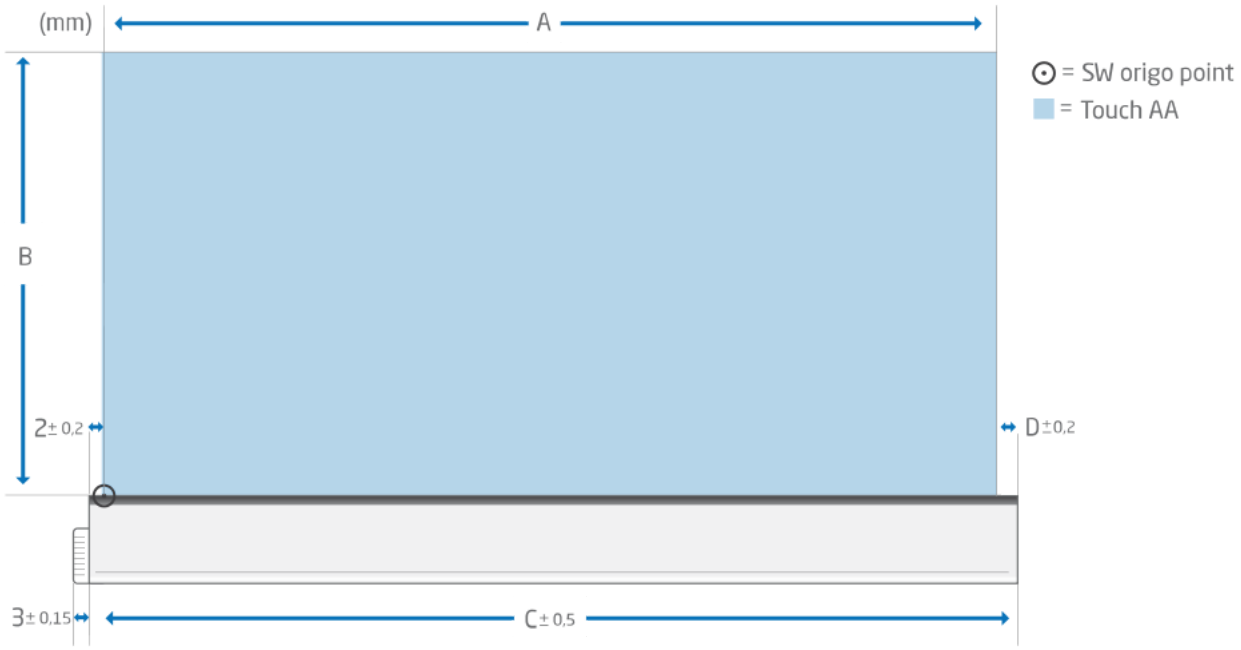

- Keep window thickness as small as mechanically feasible, to reduce absorption losses.

## 8.7 Mechanical Data

### 8.7.1 Physical Dimensions and Position of Origin

#### Top View

Dimensions **C** and **D** varies between the Touch Sensor variants and therefore also the Touch Active Area sizes (**A** and **B**).




| Product number | | Measurements (mm) | | | |
|---|---|---|---|---|---|
| **0°** | **90°** | **A** | **B** | **C** | **D** |
| NNAMC0430PC01 | NNAMC0431PC01 | 43.2 | 14.9 | 47.2 | 2 |
| NNAMC0500PC01 | NNAMC0501PC01 | 50.4 | 29.8 | 55.9 | 3.5 |

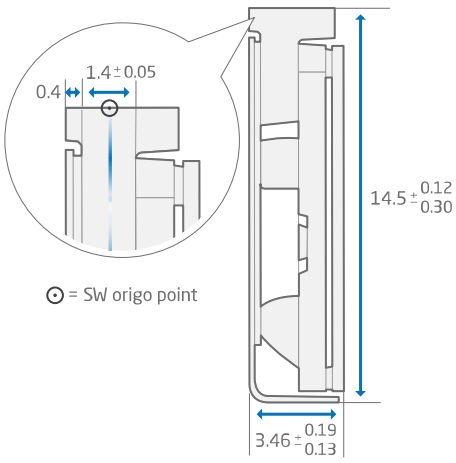
| | | | | | |
|---|---|---|---|---|---|
| NNAMC0580PC01 | NNAMC0581PC01 | 57.6 | 29.8 | 61.6 | 2 |
| NNAMC0640PC01 | NNAMC0641PC01 | 64.8 | 44.7 | 70.3 | 3.5 |
| NNAMC0720PC01 | NNAMC0721PC01 | 72 | 44.7 | 76 | 2 |
| NNAMC0790PC01 | NNAMC0791PC01 | 79.2 | 59.6 | 84.7 | 3.5 |
| NNAMC0860PC01 | NNAMC0861PC01 | 86.4 | 59.6 | 90.4 | 2 |
| NNAMC0940PC01 | NNAMC0941PC01 | 93.6 | 74.5 | 99.1 | 3.5 |
| NNAMC1010PC01 | NNAMC1011PC01 | 100.8 | 74.5 | 104.8 | 2 |
| NNAMC1080PC01 | NNAMC1081PC01 | 108 | 89.4 | 113.5 | 3.5 |
| NNAMC1150PC01 | NNAMC1151PC01 | 115.2 | 89.4 | 119.2 | 2 |
| NNAMC1220PC01 | NNAMC1221PC01 | 122.4 | 104.3 | 127.9 | 3.5 |
| NNAMC1300PC01 | NNAMC1301PC01 | 129.6 | 104.3 | 133.6 | 2 |
| NNAMC1370PC01 | NNAMC1371PC01 | 136.8 | 119.2 | 142.3 | 3.5 |
| NNAMC1440PC01 | NNAMC1441PC01 | 144 | 119.2 | 148 | 2 |
| NNAMC1510PC01 | NNAMC1511PC01 | 151.2 | 134.0 | 156.7 | 3.5 |
| NNAMC1580PC01 | NNAMC1581PC01 | 158.4 | 134.0 | 162.4 | 2 |
| NNAMC1660PC01 | NNAMC1661PC01 | 165.6 | 148.9 | 171.1 | 3.5 |
| NNAMC1730PC01 | NNAMC1731PC01 | 172.8 | 148.9 | 176.8 | 2 |
| NNAMC1800PC01 | NNAMC1801PC01 | 180 | 163.8 | 185.5 | 3.5 |
| NNAMC1870PC01 | NNAMC1871PC01 | 187.2 | 163.8 | 191.2 | 2 |
| NNAMC1940PC01 | NNAMC1941PC01 | 194.4 | 178.7 | 199.9 | 3.5 |
| NNAMC2020PC01 | NNAMC2021PC01 | 201.6 | 178.7 | 205.6 | 2 |
| NNAMC2090PC01 | NNAMC2091PC01 | 208.8 | 193.6 | 214.3 | 3.5 |
| NNAMC2160PC01 | NNAMC2161PC01 | 216 | 193.6 | 220 | 2 |
| NNAMC2230PC01 | NNAMC2231PC01 | 223.2 | 208.5 | 228.7 | 3.5 |
| NNAMC2300PC01 | NNAMC2301PC01 | 230.4 | 208.5 | 234.4 | 2 |
| NNAMC2380PC01 | NNAMC2381PC01 | 237.6 | 208.5 | 243.1 | 3.5 |
| NNAMC2450PC01 | NNAMC2451PC01 | 244.8 | 208.5 | 248.8 | 2 |

| | | | | | |
|---|---|---|---|---|---|
| NNAMC2520PC01 | NNAMC2521PC01 | 252 | 208.5 | 257.5 | 3.5 |
| NNAMC2590PC01 | NNAMC2591PC01 | 259.2 | 208.5 | 263.2 | 2 |
| NNAMC2660PC01 | NNAMC2661PC01 | 266.4 | 208.5 | 271.9 | 3.5 |
| NNAMC2740PC01 | NNAMC2741PC01 | 273.6 | 208.5 | 277.6 | 2 |
| NNAMC2810PC01 | NNAMC2811PC01 | 280.8 | 208.5 | 286.3 | 3.5 |
| NNAMC2880PC01 | NNAMC2881PC01 | 288 | 208.5 | 292 | 2 |
| NNAMC2950PC01 | NNAMC2951PC01 | 295.2 | 208.5 | 300.7 | 3.5 |
| NNAMC3020PC01 | NNAMC3021PC01 | 302.4 | 208.5 | 306.4 | 2 |
| NNAMC3100PC01 | NNAMC3101PC01 | 309.6 | 208.5 | 315.1 | 3.5 |
| NNAMC3170PC01 | NNAMC3171PC01 | 316.8 | 208.5 | 320.8 | 2 |
| NNAMC3240PC01 | NNAMC3241PC01 | 324 | 208.5 | 329.5 | 3.5 |
| NNAMC3310PC01 | NNAMC3311PC01 | 331.2 | 208.5 | 335.2 | 2 |
| NNAMC3380PC01 | NNAMC3381PC01 | 338.4 | 208.5 | 343.9 | 3.5 |
| NNAMC3460PC01 | NNAMC3461PC01 | 345.6 | 208.5 | 349.6 | 2 |

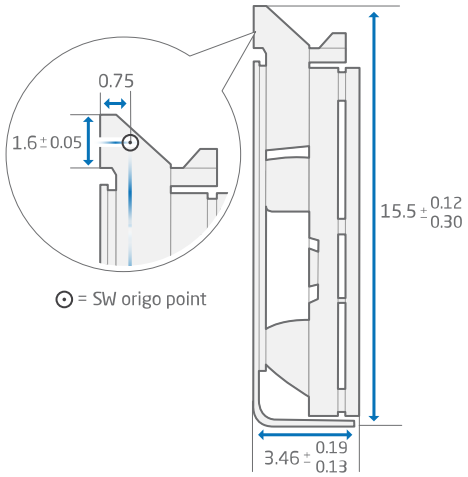
## Side View

These measurements are identical for all sensor lenghts but varies some between the 0° and 90 ° types. The position of origin is marked with "zero software".

### 0° Type



90 ° Type




## 8.8 Test Specifications and Definitions

### 8.8.1 Reliability Test Specification

#### Overview

| TEST ITEM | TEST CONDITION | QUANTITY |
|---|---|---|
| **zForce AIR Touch Sensor:** | | |
| Low temperature operation test | Ta=-20°C, 96 hrs, Powered On | 3 |
| Low temperature storage test | Ta= -40°C, 96 hrs, Power Off | 3 |
| Temperature cycling operation test | Ta: 0°C / 65°C, RH: 60%<br>Cycle time: 60 min, Duration: 20 minutes at each temperature extreme with a 10 minute transition time.<br>240 cycles in total ; Power On,<br>Functionality test every 24 hours. | 3 |

| | | |
|---|---|---|
| Thermal shock | Ta: +85°C to -40°C<br>Humidity: Off<br>Duration: 25 minutes at each temperature extreme with a 5 minute transition time.<br>120 cycles in total<br>Operation mode: The DUT is powered off during testing. | 3 |
| Condensation Test | Ta: +25ºC to +65ºC<br>RH: 95%<br>50 cycles, cycle time:30 + 30 min ; Power Off | 3 |
| Vibration test | Wave form : random<br>Vibration level : 1.0 GRMS<br>Bandwidth : 10-300 Hz<br>Cycle: X,Y,Z, 60 min each<br>Number of cycles: 4<br>DUT on vibration table: Clamped on 3 positions along sensor. | 3 |
| Shock test | Shock level : 50 G<br>Waveform : half sine wave, 2 msec<br>Direction :±X, ±Y, ±Z<br>One time each direction | 3 |
| Pressure Intensity | Pressure on surface with a flat ø 10 mm rod with a force of 10 N. The force should be applied at the center of the PCB. Test for 168 hrs. Power Off | 3 |
| Contact wear test | Wave form : random<br>Vibration level : 1.0 GRMS<br>Bandwidth : 10-300 Hz<br>Cycle: X,Y,Z, 60 min<br>Number of cycles: 1<br>One time each direction<br>DUT on vibration table: Clamped on 3 positions along sensor. The FPC is clamped to the table at one position (center). The FFC-connector will not be clamped to the table. | 3 |
| Cleansing test | Rubbed 5 times using a piece of cotton soaked with 2-5 ml (depending on size of object).<br>Apply the solutions below in following order on the same place of the object:<br>o Denatured alcohol (ethanol 99,5%).<br>o All purpose cleaning solution containing ammonia (e.g. AJAX Tornado).<br>o Water. | 3 |

| | | |
|---|---|---|
| ESD | EN55024<br>(61000-4-2)<br>Direct contact discharge: 2,4,8 kV<br>Indirect contact discharge: 2,4,8 kV<br>Air discharge: 4,8,16 kV | 3 |

### 8.8.2 Reliability Test Report

The reliability test report can be found in Downloads on https://support.neonode.com[15].

15 https://support.neonode.com/