Application note

Connecting multiple VL53L9CX devices on a single host

Introduction

It is possible to connect multiple devices on a single host, sharing the same bus control or not. However, certain issues can be encountered and care must be taken to ensure optimal performance.

This document describes the challenges caused by the presence of multiple sensors connected to the same host, and proposes solutions to solve them.

1 Acronyms and abbreviations

Table 1. Acronyms and abbreviations

Acronym/abbreviation	Definition
AVDD	Analog supply voltage
BCR	Bus characteristics register
CPU	Central processing unit
CSI-2	Camera serial interface 2
DAA	Dynamic address assignment
DCR	Device characteristics register
DSS	Dynamic signal selection
EMI	Electromagnetic interference
FoV	Field of view
GPIO	General-purpose input/output
GPU	Graphics processing unit
I^2C	Inter-integrated circuit
I3C	Improved inter-integrated circuit
IBI	Inbound interrupt
INTR	Interruption output GPIO
IOVDD	I/O supply voltage
LDD	Laser driver
MCU	Microcontroller unit
MIPI	Mobile industry processor interface
MPU	Microprocessor unit
OTP	One-time programmable memory
PID	Provisional ID
ROI	Region of interest
RSTDAA	Reset dynamic address assignment
SCL	Serial clock line
SDA	Serial data line
SSCG	Spread spectrum clock generation
SYNC_IN	Synchronization input
VBAT	Battery voltage
VDD	Supply voltage
VHV	High-voltage supply
VSS	Ground
XSHUT	Shutdown input GPIO

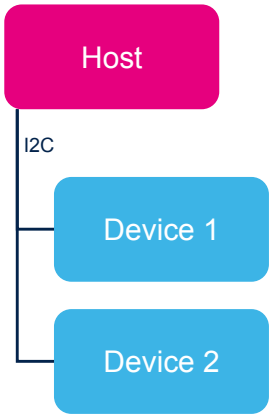
2 Device addressing

Refer to the following document: Introduction to I3C for STM32 MCUs.

2.1 Introduction

The use case described in this application note is for multiple devices connected to the same host via a single control bus. There are other options to connect a multiplexer or to connect sensors to another communication interface; however, this document does not describe these options.

Figure 1. Single host driving two or more sensors

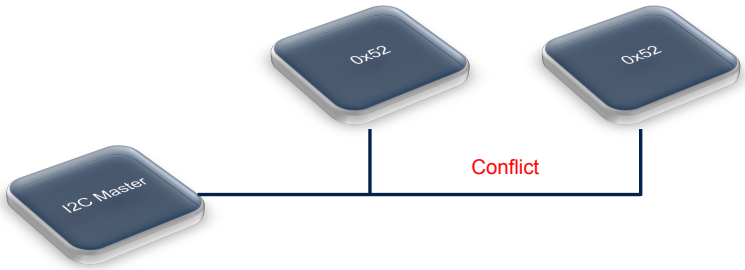


The VL53L9CX default address is configured at 0x52 on the communication bus. By default, this address is used for communication on the control bus.

When multiple devices with the same address are connected to the same I^2C bus, the controller cannot distinguish between them, leading to communication failures or data confusion.

It is therefore necessary to reprogram the sensors' addresses.

Figure 2. Conflict on communication bus



2.2 Options

There are two options to reconfigure an I^2C address on the VL53L9CX products:

- Reconfigure the address on the fly using the product's hardware registers
- Write the new address in the OTP (one-time programmable memory)

Writing in the OTP has a permanent and irreversible effect, whereas changing the address on the fly is more flexible but requires repeating the operation at each startup.

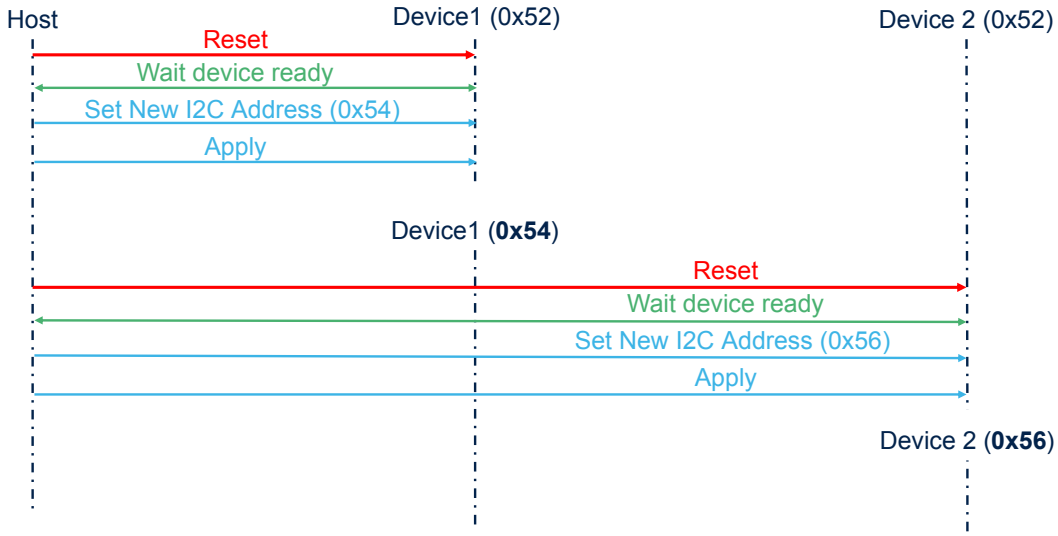
Both options require a dedicated XSHUT pin control signal per device unless the OTP programing is done before the reflow on the PCB.

2.3 I²C software programming procedure

For this type of configuration, execute the boot procedure as follows:

1. Enable the first device (XSHUT high state on the first device while maintaining the XSHUT of the second device low state).
2. Set the new device address (the new I²C address is effective once the configuration is complete):
   Write8 0xD208 0x2A: Sets the new I²C address, for example 0x2A (7 bits)
   Write8 0xD20A 0x01: Applies the configuration.
3. Enable and set the second device (XSHUT of both devices in high state):
   Write8 0xD208 0x2B: Sets the new I²C address, for example 0x2B (7 bits)
   Write8 0xD20A 0x01: Applies the configuration.
4. Both devices can boot using their respective addresses.

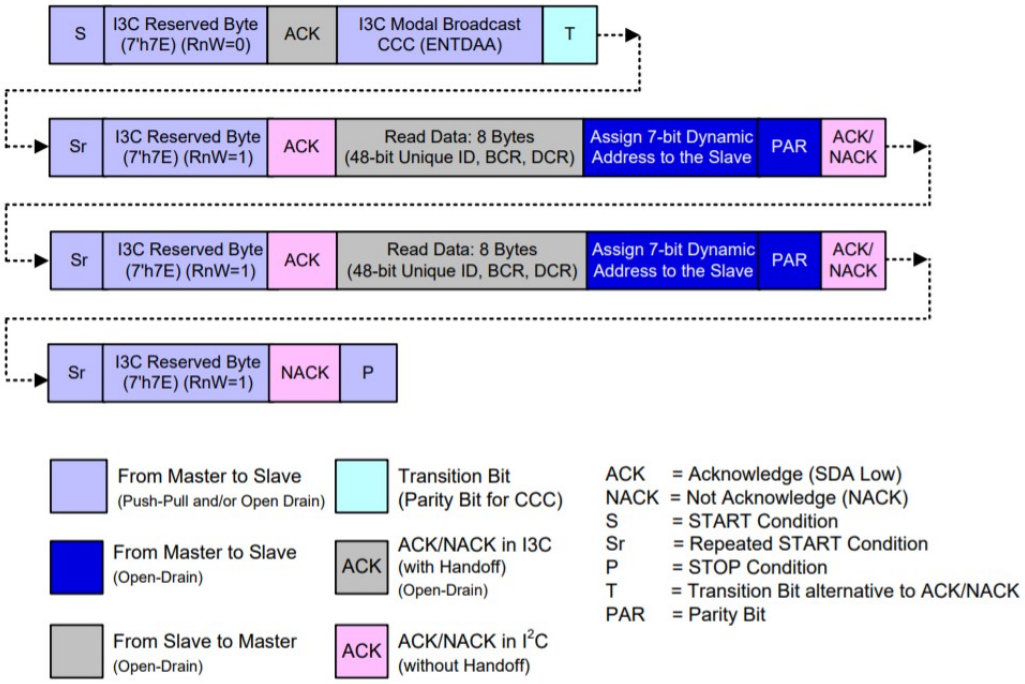
Figure 3. Device address programming sequence



2.4 I3C dynamic address assignment

The I3C host must initiate a dynamic address assignment (DAA) command when the system starts. Some information is exchanged during this configuration, and the characteristics and capabilities of the devices are exchanged.

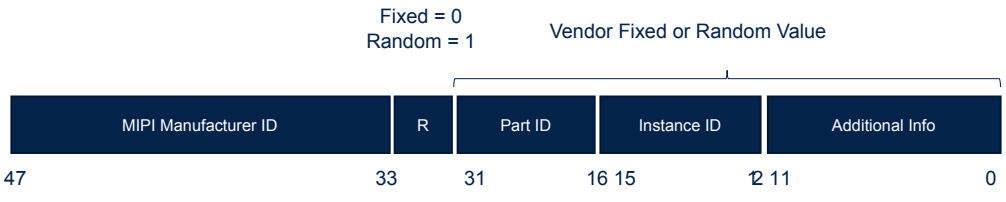
Figure 4. Dynamic address assignment



It might seem that the dynamic addressing mechanism prevents conflicts, but this is not the case.

By default, all VL53L9CX devices use the same BCR (bus characteristics), DCR (device characteristics) and InstanceID. This can lead to an arbitration error on the I3C bus in the same way as when two I^2C devices have the same address. Therefore, a different instance ID value must be assigned to each device. Additionally, I^2C compatibility must be maintained, so the I^2C address must also be changed even for devices operating on an I3C bus.

Figure 5. PID structure



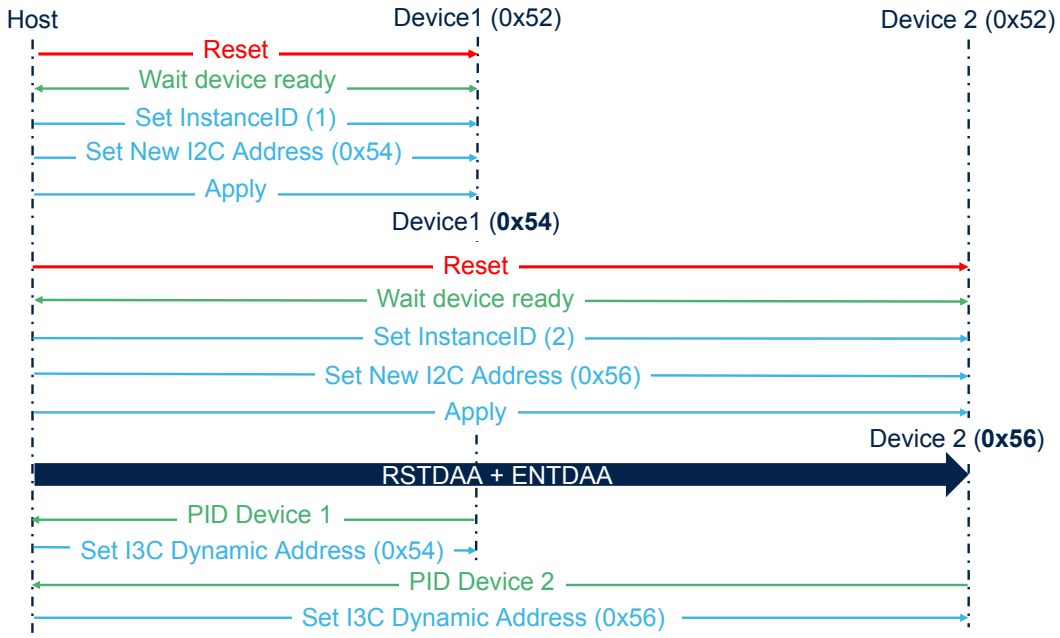
The instance ID is part of the unique ID exchanged during the DAA. To avoid arbitration issues, it is mandatory to program a different InstanceID per device on the same bus.

2.5 I3C software programming procedure

For this type of configuration, execute the boot procedure as follows:

1. Enable the first device (XSHUT high state on the first device while maintaining the XSHUT of the second device low state).
2. Set the I²C/I3C device configuration:
   Write8 0xD238 0x01: Sets the InstanceID (0x1)
   Write8 0xD208 0x2A: Sets the new I²C address, for example 0x2A (7 bits)
   Write8 0xD20A 0x01: Applies the configuration.
3. Enable the second device and configure the second device (XSHUT of both devices in high state):
   Write8 0xD238 0x02: Sets the InstanceID (0x2)
   Write8 0xD208 0x2B: Sets the new I²C address, for example 0x2B (7 bits)
   Write8 0xD20A 0x01: Applies the configuration.
4. Start the DAA, read the PIDs, and assign the addresses.
5. Both devices can boot using their respective addresses.

Figure 6. I3C device addressing programming sequence



2.6 OTP programming procedure

During sensor reset, if the OTP content is different than the default values, the device uses the I²C address programmed in OTP as the new device address. It is also possible to program the I3C instance ID in OTP to be used by the device during the DAA. During the OTP programming procedure, only one device should be enabled. The procedure is described below:

1. Start and boot the device to program.
2. Set the I3C instance ID and I²C address values to be written into the OTP:
   Write8 0x0CEC 0x01: Sets the InstanceID (0x1)
   Write8 0x0CE8 0x2A: Sets the new I²C address, for example 0x2A (7 bits)
3. Configure the OTP write in custom OTP memory:
   Write16 0x0492 0x0000: Configures the custom register address to program.
   Write8 0x0494 0x08: Configures the number of bytes to write.
   Write8 0x0400 0x09: Calls the OTP custom write command.
   Wait 0x0400=0x00: Waits for command completion.

3 Interferences

3.1 Introduction

Sensors operating in sync may interfere with each other. This is especially true when their field of emission and FoV overlap. Even if the distance measured should not be affected, the emission of one device is seen as ambient light by the other device. After processing, the signal rate and reflectance can be incorrectly estimated.

3.2 Mitigation

When multiple VL53L9CX devices operate at the same time with overlapping FoVs, the probability that they interfere with each other is high. Desynchronizing the devices ensures that the exposure does not happen at the same time. However, sometimes it is not possible to control the devices. In this case there are two options to reduce the interference: spread spectrum clock generation (SSCG) and dithering. SSCG reduces EMI and interference, while dithering reduces interference and wraparound occurrences. The dithering side-effect is that it extends the frame integration duration.

3.3 Guidelines

ST recommends enabling both the SSCG and the dithering options (the default configuration in ST official drivers). Disable dithering if the application requires a reduced frame-integration duration and device-to-device interference is not an issue for the application.

Dithering and SSCG have no impact on power consumption. However, dithering increases the frame acquisition duration.

Refer to UM3683 for programming guideline to enable and disable features.

3.4 Cover glass and inter-device crosstalk

ST recommends checking crosstalk generated by other devices. Refer to AN6346 regarding cover glass guidelines.

4 Frame rate

4.1 Introduction

A frame rate is the number of frames the system is able to output per second. A frame consists of acquisition, readout, and postprocessing periods. Each device can stream out several frames per second for a given configuration and exposure duration. Once multiple devices are connected to the system, the maximum frame rate is limited by other parameters.

The following three different successive periods are defined for a given frame.

Figure 7. Frame periods



4.2 Acquisition

The frame exposure is the main contributor but other parameters can influence the duration of the frame acquisition.

The total exposure is defined as the sum of all step exposures for both channels including the blanking periods. When the blanking dithering mode is active it increases the average blanking duration, then the total frame duration.

The sensor has three power modes: regular, low power, and ultralow power. When the device operates in low or ultralow power, it takes longer to enter and exit these modes. This increases the frame acquisition time by 4.5 ms to 11 ms depending on the mode.

4.3 Readout

When multiple devices are connected in a system the streaming interface may have an impact on the maximum frame rate.

Streaming the data over MIPI is faster than reading the frame over I3C. Reading the frame over I3C is still faster than reading it over I^2C.

MIPI data rate can be 1 Gbps while it is 12.5 Mbps on I3C and 1 Mbps on I^2C. Given the data rate, the duration to read a frame is directly proportional to the resolution of the frame.

The table below summarizes the total number of bytes to read for a given resolution.

Table 2. Frame readout with resolution and interface

Resolution	Minimal frame volume (Bytes)	Minimal readout duration		
		MIPI CSI-2	I3C	I^2C
54 x 42	14842	150 µs	11.5 ms	120 ms
24 x 20	3844	30 µs	3 ms	30 ms
18 x 14	1738	15 µs	1.5 ms	15 ms

The table above shows that the limiting factor in MIPI is the frame acquisition while in I^2C this is the readout.

When multiple devices share the same control interface, the bandwidth is shared between all devices. This can drastically reduce the system frame rate.

4.4 Postprocessing

Postprocessing is required to clean up and filter the data streamed out by the sensor. Depending on the host selected, the postprocessing library can run on GPU, CPU, MPU, or MCU. There is no simple method to estimate the time it takes to process a frame. It is strongly dependent on the frame resolution and where it runs.

For low resolutions, a CPU-based solution is faster than a GPU-based solution. On MPU/MCU it depends how fast the host is, as indicated by the example cases shown in the table below.

Table 3. Processing time examples (with all algorithms)

Platform	H563	N6570	Rock chip
Processing time	41 ms	16 ms	8.5 ms

4.5 Optimization options

4.5.1 Parallelism

The total frame duration depends on acquisition, readout, and processing. The frame duration is given by the following equation.

$T_{Frame} = T_{Acquisition} + T_{Readout} + T_{Processing}$

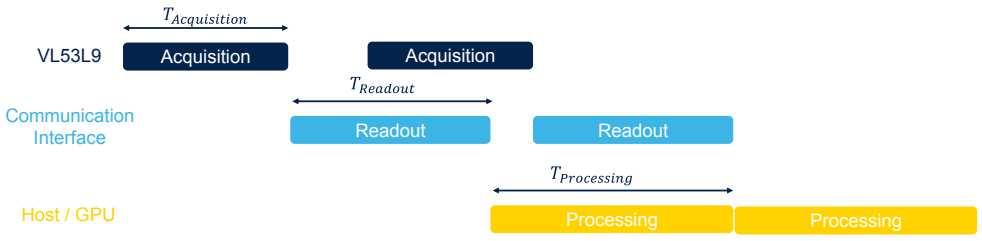
Without parallelism, the maximum frame rate is given by the inverse of the minimum frame duration.

$$FPS = \frac{1}{T_{FRAME}} = \frac{1}{T_{Acquisition} + T_{Readout} + T_{Processing}}$$

In reality, these three functions do not involve the same hardware.

The acquisition runs on the sensor, but during the acquisition the host is free. The processing can be offloaded to a GPU to reduce the load on the CPU or MCU.

Figure 8. Parallelism



4.5.2 Region of interest

If the application uses only a specific part of the sensor FoV, it is strongly recommended to read only the region of interest (ROI). This method optimizes and accelerates the readout duration. It can also reduce the processing time, which increases the system frame rate.

5 Laser safety recommendations

The VL53L9CX contains a laser emitter and associated driver circuit. The laser output is designed to meet Class 1 laser safety limits under all reasonably foreseeable conditions including single faults in compliance with IEC 60825-1:2014.

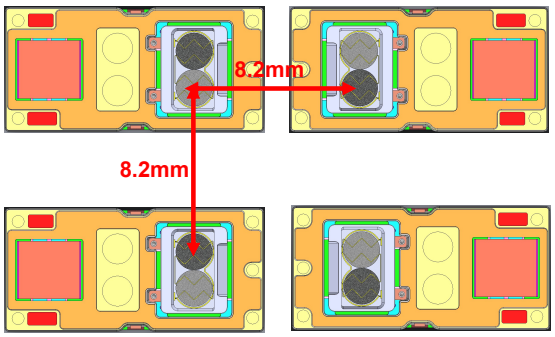
ST cannot guarantee the Class 1 compliance for the full system.

In terms of safety recommendations, depending on the number of devices used, the minimum distance between each device should follow the values indicated in the table below.

Table 4. Recommended distance between devices

Number of devices	Minimum distance between each device (mm)
1	1.1
2	4.6
3	6.7
4	8.2
5	9.3
6	10.2
7	11.0
8	11.7
9	12.3
10	12.8
15	14.9

Figure 9. Device configuration example and spacing



6 Application examples

This section describes various examples with multiple devices connected on the same host. Combining multiple devices extends the field of view of the application.

6.1 Application example code

ST delivers an example in the STM32N6 and STM32H5 release (ranging_multi_i3c). This code example can be reused, and the device configuration can be updated, for example modifying the number of interfaces, devices per interface, and GPIO configurations.

6.2 General recommendations for multiple devices on the same PCB

Power recommendations

- Add a set of decoupling capacitors for each device (as close as possible to each device)
- Check the power supply rails. Ensure the LDOs can deliver the current needed by the application and the operating conditions (check the tuning profile tool to estimate the power consumption per device).
- Check if the overcurrent protection component is adapted to the application.

GPIOs recommendations

- **XSHUT**: ST recommends keeping independent reset GPIOs (XSHUT) by device. It allows the sensors to be powered one by one for I2C/I3C address reprogramming as described in Section 2.3: I²C software programming procedure.
- **SYNC_IN**: The external sync signal (active low) is used to trigger the exposure when the sensor operates in follower mode. To expose all the devices at the same time, connect all devices on the same host GPIO. Otherwise, keep independent GPIOs. Be careful of the power lines and decoupling caps when the sensors are synchronized together. The peak currents, especially on VBAT, is multiplied by the number of devices. This input is not used when the device operates in autonomous or manual modes. Therefore, GPIOs can be saved and the pin left disconnected in applications using these modes.
- **INTR**: The interrupt pin signal (active low) allows the application or the driver to be notified when a new frame is exposed.
  To check whether a new frame is available, use the I3C inbound interrupt (IBI) when the driver is not using the MIPI interface, or poll the firmware status register.
  Using IBI helps to save GPIOs, and the pin can remain disconnected.

I3C/I²C bus recommendation

Use a short trace as much as possible to reduce the line capacitance.

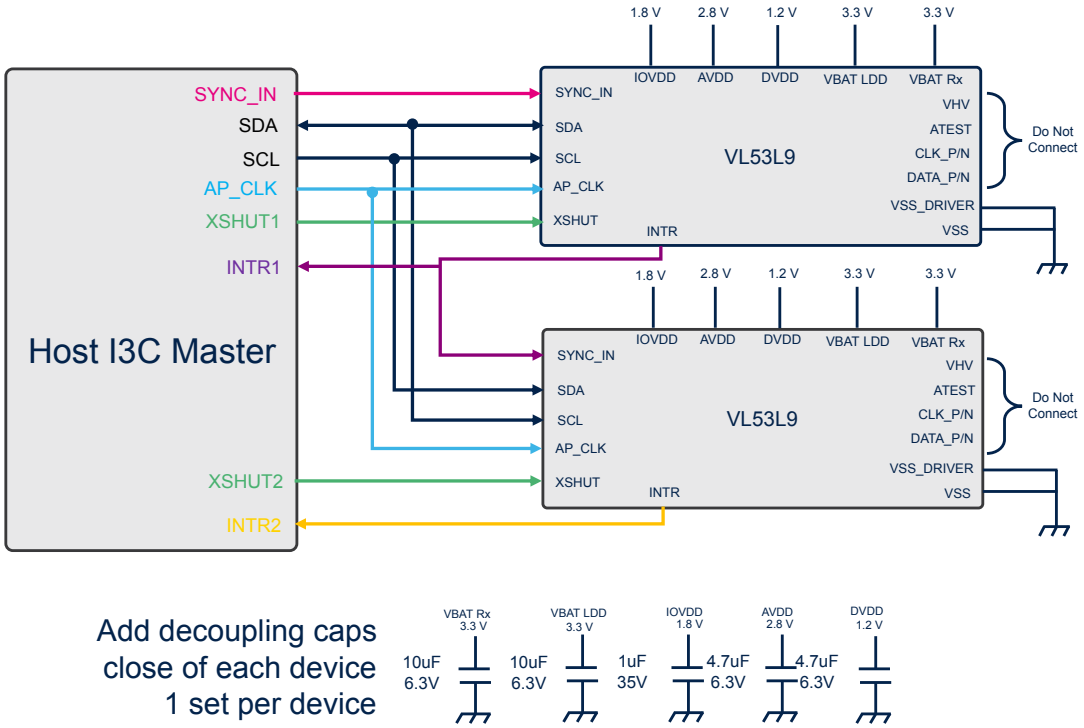
6.3 Application patterns with two devices

6.3.1 Two devices, one I3C

Application schematic

In this configuration, the same I3C interface is shared between the two devices. This configuration is well adapted for platforms with a single I3C interface such as STM32H563.

Figure 10. **Two devices with one I3C interface**



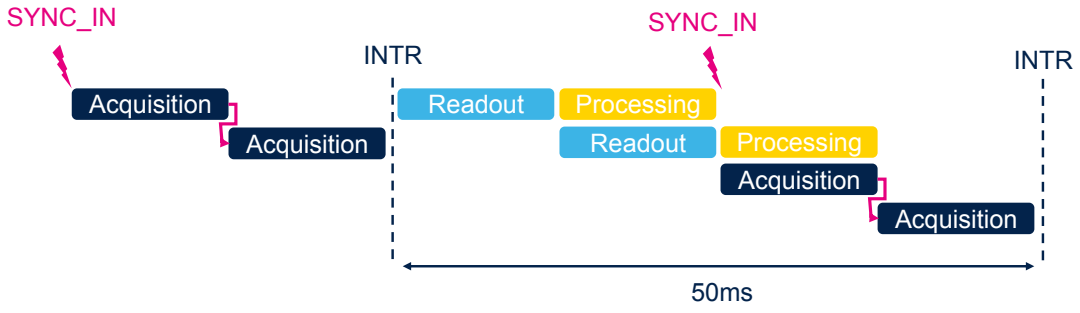
Application example

The main complexity is during the boot operation. The two devices cannot be powered simultaneously unless the device address is reprogrammed in the OTP (refer to Section 2: Device addressing). The application is programming the new address by software before booting the devices.

In this example, the two devices are working in a waterfall mode. The first device integration is triggered by the host, and the second device integration starts automatically after the first device (INTR1 GPIO connected to SYNC_IN GPIO).

The host is responsible for reading the data (using DMA) and processing the data. During the readout of device 1 the host is processing the frame of device 2, and vice versa.

Figure 11. **Device synchronization**

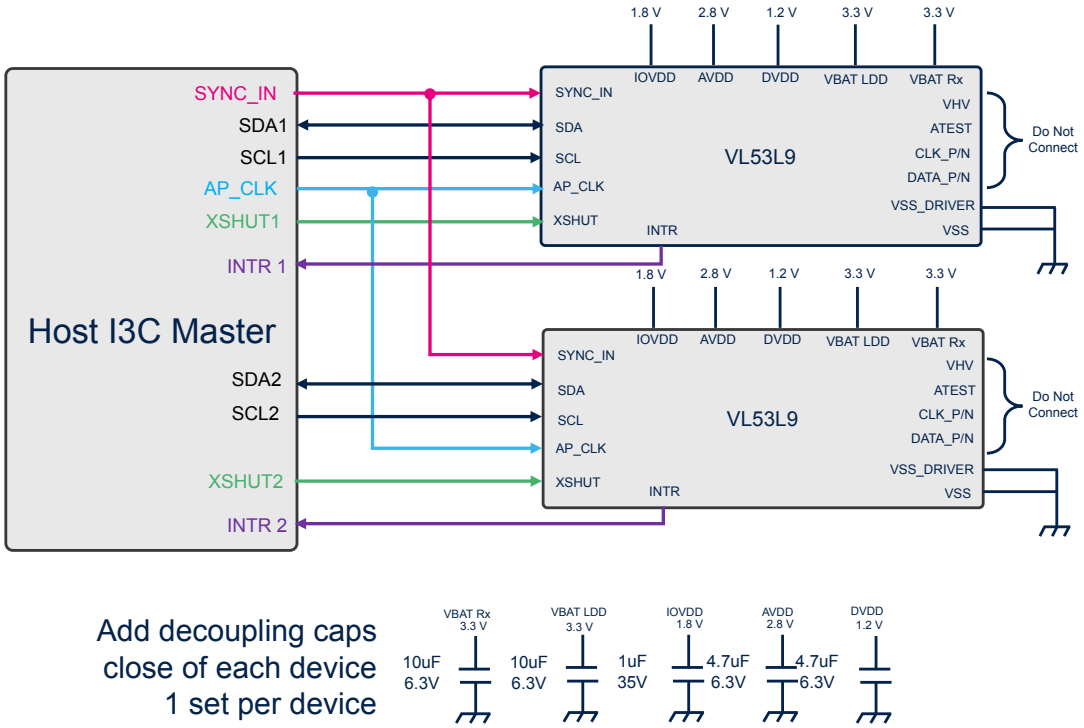


6.3.2 Two devices, two I3C

Application schematic

In this configuration, each device has a dedicated I3C interface. This configuration is well adapted for platforms with two I3C interfaces such as SMT32H5E5 or STM32N6xx.

Figure 12. **Two devices with two I3C interfaces**

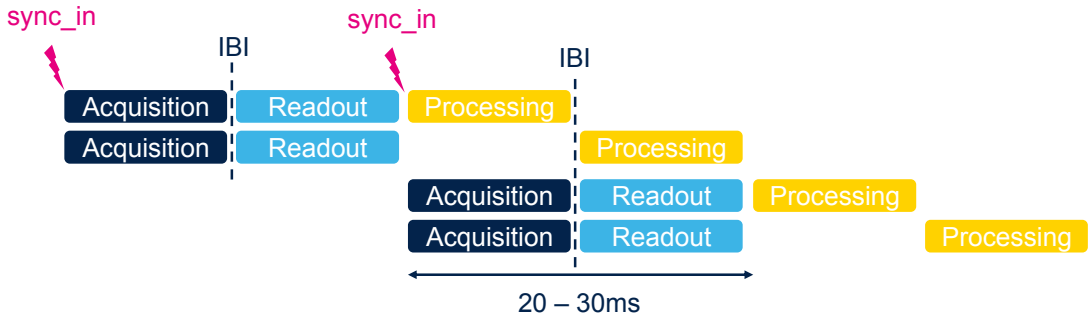


Application example

In this example, the two devices are working in a synchronous acquisition. The integration of the two devices is triggered by the host (SYNC_IN GPIO).

The main advantage of this configuration is to read out the data in parallel on the two devices. This configuration can capture between 30 and 50 frames per second. The processing time is the bottleneck of this architecture; ST recommends using a host with enough RAM and process capability.

Figure 13. **Device synchronization**

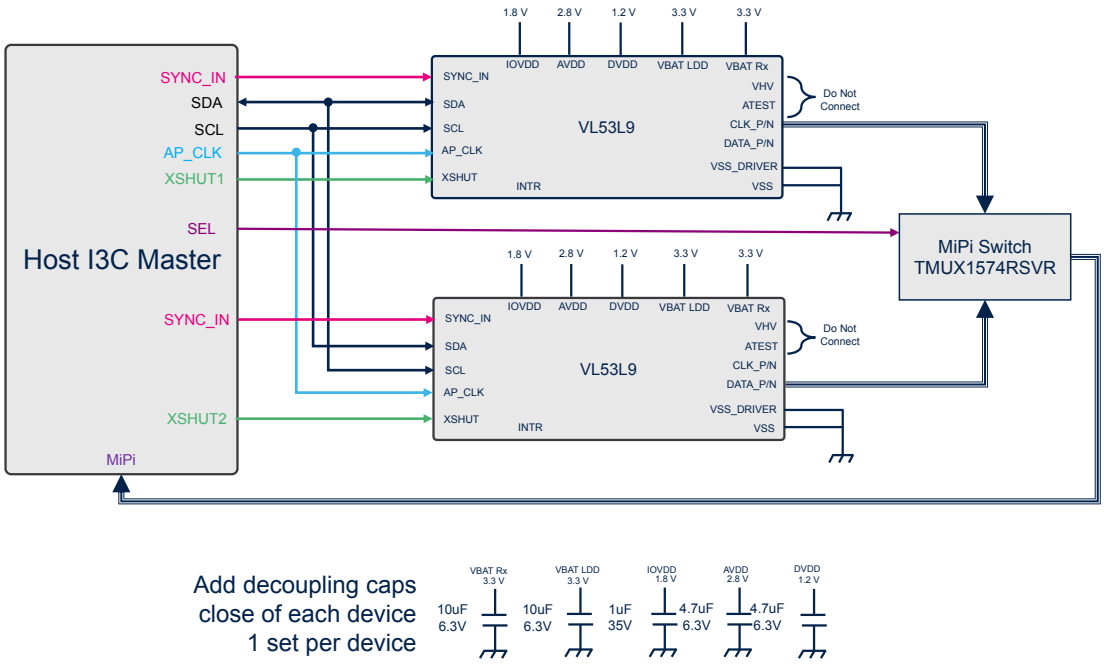


6.3.3 Two devices and a MIPI switch

Application schematic

This configuration uses a single MIPI interface connected to a MIPI switch. Alternately, each device can stream their frames over individual MIPI interfaces.

Figure 14. **Devices with a MIPI switch**



Application example

In this example, the devices are running in ping pong mode. The processing is done during the integration of the second device. The frame rate is maximized thanks to the MIPI CSI-2 interface.

Figure 15. **Device acquisition and processing**



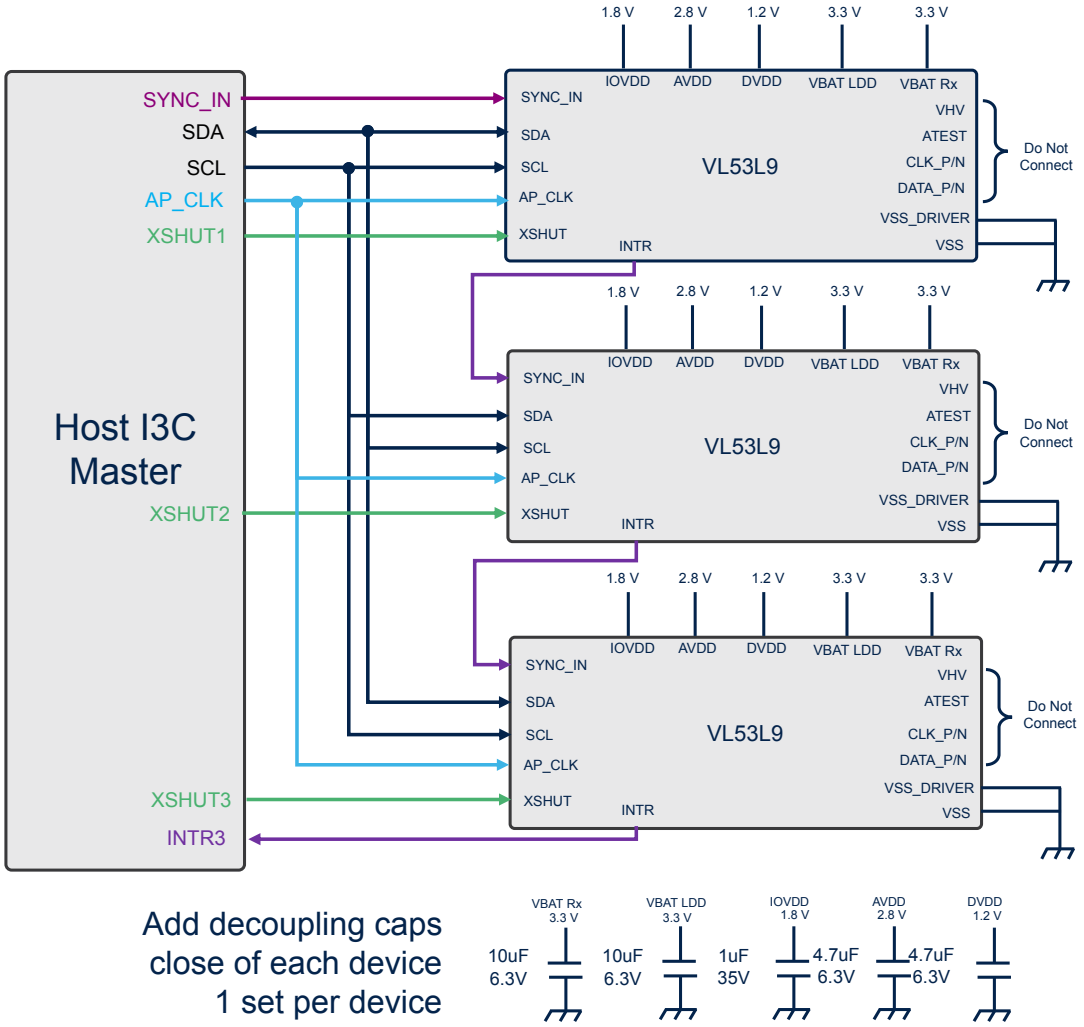
6.4 Application patterns with three or more devices

6.4.1 Three devices, one I3C

Application schematic

In this application, the three devices are connected to the same I3C interface.

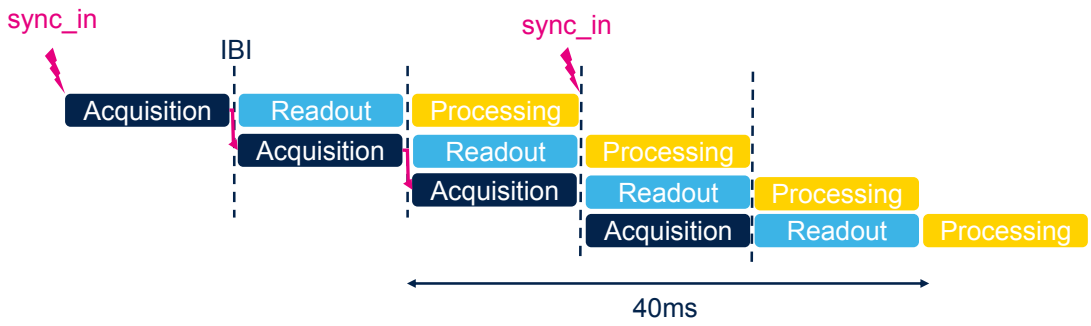
Figure 16. **Three devices on a signal interface**



Application example

In this example, the devices run in sequential mode. During the acquisition of one sensor, the host is programming the DMA to perform the readout and process a frame. Depending on the host platform it could be possible to capture 20 to 25 frames per second per device.

Figure 17. **Frame synchronization**

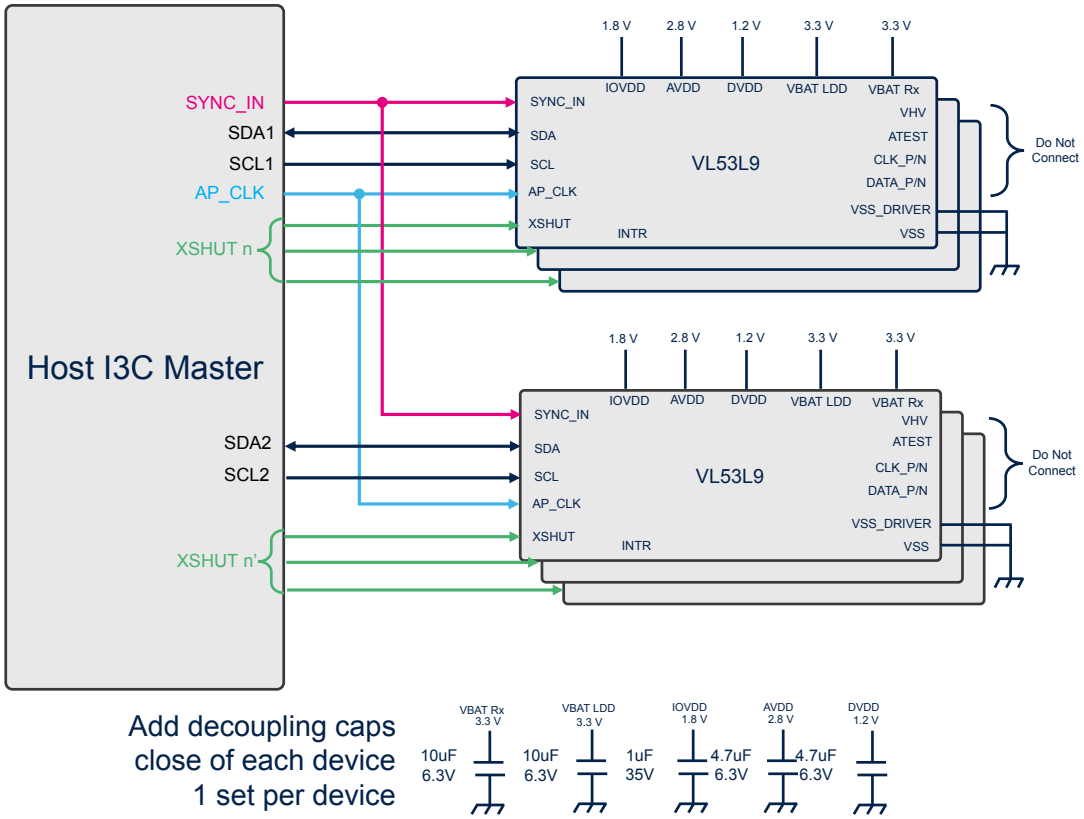


6.4.2 Multiple devices, two I3C

Application schematic

In this configuration multiple devices are connected over two interfaces. All sensors are synchronized and perform a single acquisition (SYNC_IN pin share with all sensors) in order to minimize the artefacts with moving targets during acquisition. In order to save GPIO, ST recommends using I3C IBI instead of physical interrupt.

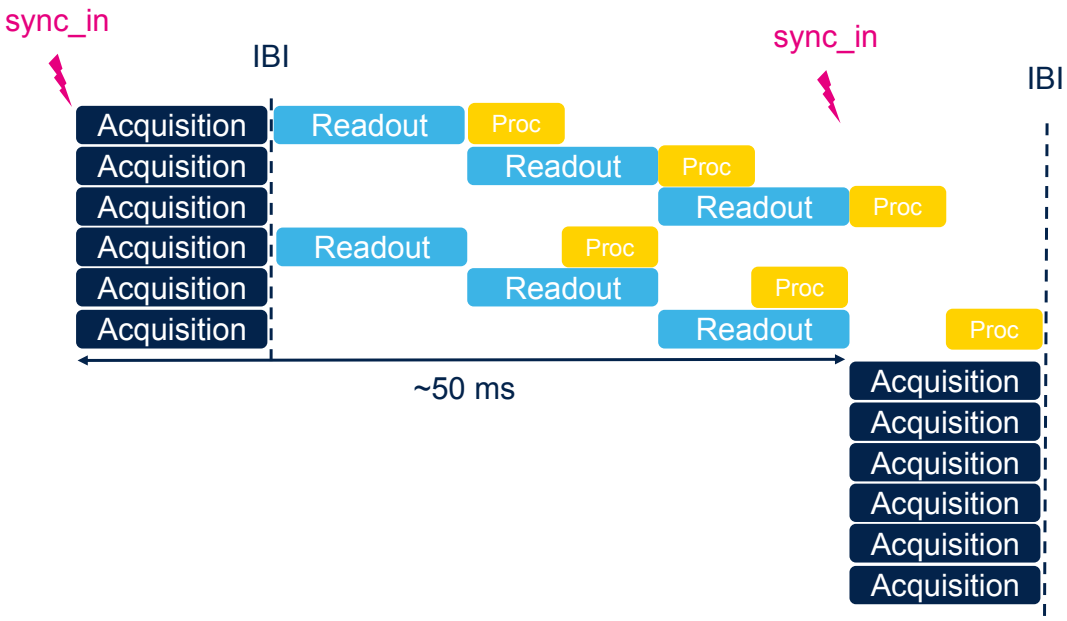
Figure 18. Multiple devices on two interfaces



Application example

In this configuration the limiting factor can be either the processing time if the host is too slow or the readout duration. The total frame period depends on the number of devices per interface. With six devices, the maximum frame rate could be 15 to 20 frames per second and lower with eight devices.

Figure 19. **Multiple device synchronization**



7 Assembly alignment guidelines

Working with multiple Time-of-Flight devices requires a calibration or alignment step during production. The alignment permits the application to merge the device point cloud outputs in a single point cloud dataset.

Intrinsic parameters are characteristics that determine how 3D points in the world are projected onto the 2D image plane. They are determined by the imaging system's hardware and optical design. Extrinsic parameters describe the imaging device's position and orientation within the complete system.

7.1 Intrinsic parameters

Theoretical values are recommended because they are very stable from part to part.

- Focal length: 2428.16 µm
- Distortion with a polynomial model:
  - k1 = -0.00015
  - k2 = 0
  - k3 = 0
- Lens centering is stored in one-time programmable (OTP) memory:
  - The optical offset is coded in **macro SPAD(2x2 SPADs)**. The values are -1, 0, or 1 in both X and Y directions, but the hardware already uses them. **Ignore these values**.
  - The residual offset is **encoded in SPAD** in the range -1.5 to 1.5. **Use the residual offset for the lens centering parameters.**

  Refer to TN1641 for more information about OTP content description.
- SPAD dimension: 10.17 µm

7.2 Extrinsic parameters

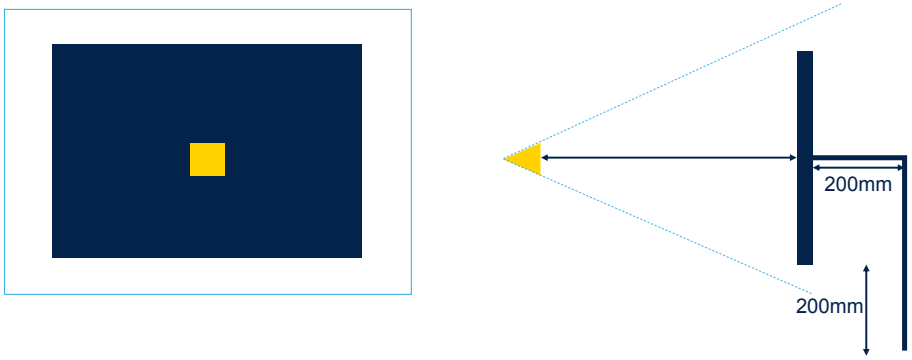
To extract the orientation of the sensor in space, ST recommends designing a specific setup that depends on the number of devices and their orientation.

Customers can reproduce and adapt the procedure in production after assembly to determine the actual sensor orientation.

Each device must have its own target in front of it. Position the target approximately 50 cm from the imaging system, ensuring it occupies about 80% of the FoV and its plane is as perpendicular as possible to the optical axis. Adapt the dimensions to the distance and device orientation. ST recommends using low-reflectance rectangular targets (17% reflectance for example) to reduce the risk of ranging errors caused by a signal that is too high.

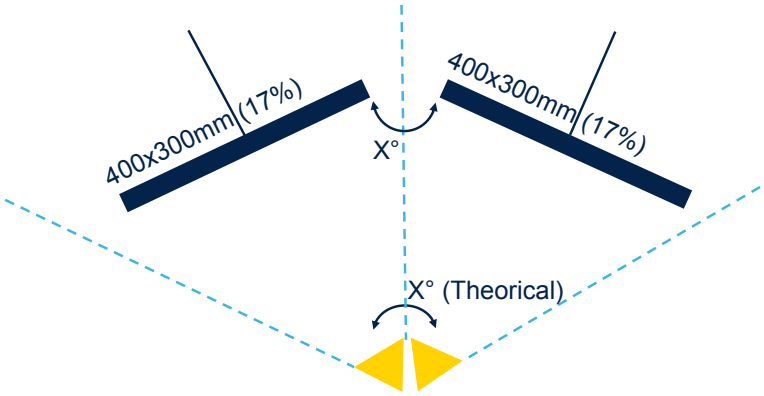
The figure below shows the sensor position in front of the target and the target coverage.

Figure 20. Target dimension and position



The figure below shows an example with two devices side by side. The angle between the two targets must be perfectly adjusted and mastered.

Figure 21. **Dual sensor calibration**



Apply the following alignment procedure for each sensor:

1. Capture one frame for each sensor.
2. Keep the point cloud output.
3. Remove all points that are not at 50 cm ± 10 cm.
4. Run a plane detection algorithm to obtain the plane characteristics and extract the plane tilt angles on the x- and y-axes.
5. Run a rectangle detection algorithm to find the rectangle edges and extract the target tilt angle on the z-axis.
6. Evaluate the center of mass and extract the x, y, and z positions.

The alignment target is designed to obtain values as close as possible to 0 mm or 0°. The deviation from the ideal values and the target characteristics allow the user to compute the extrinsic parameters of each device in a system with a common reference.

7.3 Parameter storage

ST recommends using the OTP to save the extrinsic parameters. 20 bytes are available in the OTP for customers to save application data. The application can read back those values and stream and use it to transform the point cloud output.

Figure 22. **Extrinsic storage example in OTP**

0xCF0	T_X	Translation over X axis (S13.2)
0xCF1		
0xCF2	T_Y	Translation over Y axis (S13.2)
0xCF3		
0xCF4	T_Z	Translation over Z axis (S13.2)
0xCF5		
0xCF6	R_X	Rotation over X axis (U9.2)
0xCF7		
0xCF8	R_Y	Rotation over Y axis (U9.2)
0xCF9		
0xCFA	R_Z	Rotation over Z axis (U9.2)
0xCFB		

S13.2 bit breakdown

- **S (Signed):** Allocates **1 bit** for the sign (typically using two's complement representation).
- **13 (Integer):** Allocates **13 bits** for the integer part of the number.
- **.2 (Fractional):** Allocates **2 bits** for the fractional part (after the binary point).
- **Total width:**16 bits

Dynamic range and resolution

Because it uses two fractional bits, the smallest increment between two consecutive numbers (resolution) is 0.25 mm.

The maximum value is 8119.75 mm and minimum -8192.0 mm, it allows to save the sensor position is a room. The user can change the unit in µm or adapt the fractional part to have more precise translation.

U9.2 bit breakdown

- **U (Signed): Allocates 0 bits** for a sign. The number is strictly non-negative (zero or positive).
- **9 (Integer):** Allocates **9 bits** for the integer part of the number.
- **.2 (Fractional):** Allocates **2 bits** for the fractional part (after the binary point).
- **Total width:**11 bits

Dynamic range and resolution

Because it uses two fractional bits, the smallest increment between two consecutive numbers (resolution) is 0.25°.

The maximum value is 511.75° and the minimum 0°. It allows the sensor orientation angle between 0 and 359.75°.

7.4 OTP writing procedure

OTP writing procedure is like the one described in Section 2: Device addressing:

1. Start the device to program.
2. Write extrinsic information in OTP:
   Write Bytes 0x0CF0 to 0x0CFB
3. Configure the OTP write in custom OTP memory:
   Write16 0x0492 0x0008: Configures the custom register address to program.
   Write8 0x0494 0x0C: Configures the number of bytes to write (12 bytes).
   Write8 0x0400 0x09: Calls the OTP custom write command.
   Wait 0x0400=0x00: Waits for command completion.

Revision history

Table 5. **Document revision history**

Date	Version	Changes
28-May-2026	1	Initial release
10-Nov-2026	2	Modified title. Section 2.2: Options: Added more information regarding the conditions at the end. Section 2.3: I^2C software programming procedure: Added more information to step 1 and 3. Section 2.5: I3C software programming procedure: Added more information to step 1 and 3. Section 3.1: Introduction: Removed reference to DSS configuration. Section 3.3: Guidelines: Reworded when to disable dithering for more clarity. Added Section 3.4: Cover glass and inter-device crosstalk. Section 4.2: Acquisition: Reworded for clarity. Section 4.3: Readout: Modified data rate on I3C. Section 4.5.1: Parallelism: Simplified for clarity. Section 4.5.2: Region of interest: Reworded for clarity. Added Table 3. Processing time examples (with all algorithms). Section 5: Laser safety recommendations: Clarified conditions. Changed title *Application shematics guidelines* to Section 6: Application examples. Rewrote section, adding subsections: • Section 6.1: Application example code • Section 6.2: General recommendations for multiple devices on the same PCB • Section 6.3: Application patterns with two devices • Section 6.4: Application patterns with three or more devices Section 7: Assembly alignment guidelines: Rewrote section, adding subsections: • Section 7.1: Intrinsic parameters • Section 7.2: Extrinsic parameters • Section 7.3: Parameter storage • Section 7.4: OTP writing procedure

Contents

1 **Acronyms and abbreviations** 2
2 **Device addressing** 3
2.1 Introduction 3
2.2 Options 3
2.3 I2C software programming procedure 4
2.4 I3C dynamic address assignment 5
2.5 I3C software programming procedure 6
2.6 OTP programming procedure 6
3 **Interferences** 7
3.1 Introduction 7
3.2 Mitigation 7
3.3 Guidelines 7
3.4 Cover glass and inter-device crosstalk 7
4 **Frame rate** 8
4.1 Introduction 8
4.2 Acquisition 8
4.3 Readout 8
4.4 Postprocessing 8
4.5 Optimization options 9
4.5.1 Parallelism 9
4.5.2 Region of interest 9
5 **Laser safety recommendations** 10
6 **Application examples** 11
6.1 Application example code 11
6.2 General recommendations for multiple devices on the same PCB 11
6.3 Application patterns with two devices 11
6.3.1 Two devices, one I3C 11
6.3.2 Two devices, two I3C 12
6.3.3 Two devices and a MIPI switch 13
6.4 Application patterns with three or more devices 14
6.4.1 Three devices, one I3C 14
6.4.2 Multiple devices, two I3C 16
7 **Assembly alignment guidelines** 18
7.1 Intrinsic parameters 18

7.2 Extrinsic parameters 18
7.3 Parameter storage 19
7.4 OTP writing procedure 20
Revision history 21

IMPORTANT NOTICE – READ CAREFULLY

STMicroelectronics NV and its subsidiaries (“ST”) reserve the right to make changes, corrections, enhancements, modifications, and improvements to ST products and/or to this document at any time without notice.

In the event of any conflict between the provisions of this document and the provisions of any contractual arrangement in force between the purchasers and ST, the provisions of such contractual arrangement shall prevail.

The purchasers should obtain the latest relevant information on ST products before placing orders. ST products are sold pursuant to ST’s terms and conditions of sale in place at the time of order acknowledgment.

The purchasers are solely responsible for the choice, selection, and use of ST products and ST assumes no liability for application assistance or the design of the purchasers’ products.

No license, express or implied, to any intellectual property right is granted by ST herein.

Resale of ST products with provisions different from the information set forth herein shall void any warranty granted by ST for such product.

If the purchasers identify an ST product that meets their functional and performance requirements but that is not designated for the purchasers' market segment, the purchasers shall contact ST for more information.

ST and the ST logo are trademarks of ST. For additional information about ST trademarks, refer to www.st.com/trademarks. All other product or service names are the property of their respective owners.

Information in this document supersedes and replaces information previously supplied in any prior versions of this document.

© 2026 STMicroelectronics – All rights reserved