UM2133

User manual

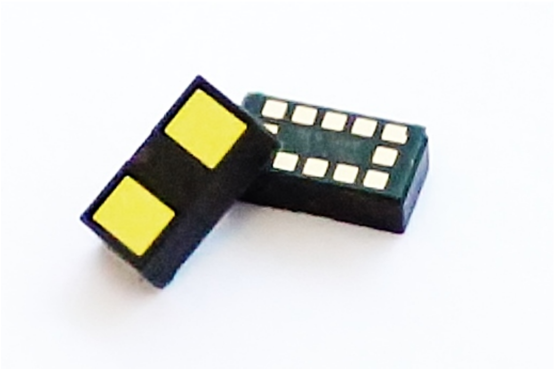
Time-of-Flight long-distance ranging sensor with advanced multi-zone and multi-object detection

Introduction

This user manual is applicable for VL53L1 devices. The VL53L1 is a long distance ranging sensor with multi-zone and multi-object detection (see VL53L1 Datasheet).

The purpose of this User Manual is to describe the possible integration models, and for each of them, the set of functions to call to get ranging data using the VL53L1 bare driver.

Figure 1. VL53L1 ranging sensor module



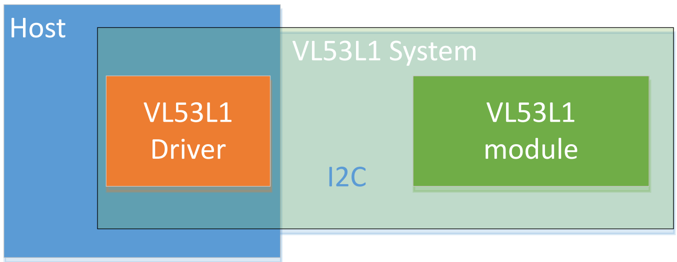
References

- VL53L1 Datasheet (DS11786)
- VL53L1 Application Note to describe performances (AN5573)

1 VL53L1 system overview

The VL53L1 system is composed of the VL53L1 module and a driver running on a host.

Figure 2. VL53L1 system



There are two possible integration models for VL53L1 devices, depending on whether the host is a Linux host or not. ST delivers drivers for both of those integration models:

- For the non-Linux host, ST delivers a bare-metal driver, referred to as “bare driver” in this document.
- For the Linux host, ST delivers a Linux driver

Those drivers allow the user to control the device, to perform processing and output a data structure with ranging values accessible to the host.

This document describes the driver functions accessible to the host, which are used to control the device and get ranging data, for integration with non-Linux hosts.

A separate user manual (UM2326) describes the Linux host integration.

Note: The present document describes the implemented and validated functions.

Any other function available in the drivers should not be used if not described in this document.

The bare driver is provided to be integrated into non-Linux hosts, typically microcontrollers with:

- limited memory and computing resources
- without OS (bare metal)
- or with a small footprint real-time OS

The bare driver is an implementation of a set of functions which uses the VL53L1 device. It makes minimal assumptions regarding OS integration and services. The sequencing of actions, execution/threading model, platform adaptation, and device structure allocation are not part of the bare driver implementation but are left open to the integrator.

The sequencing of bare driver calls must follow a set of rules, defined in this document.

2 Ranging functional description

This section briefly describes the functional capabilities of the VL53L1 ranging device.

2.1 Ranging preset modes

Four ranging preset modes can be used with the VL53L1 bare driver:

1. Ranging mode: This mode aims to get single or multiple (up to 4) object ranging distances.
2. MultiZones scanning mode: This mode allows the multiple region-of-interest (16 maximum per default, and up to 169) to be continuously scanned. It also allows each region-of-interest (ROI) ranging value to be reported one by one. ROI number and sizes are defined by the user before ranging is started.
3. Autonomous mode: This mode allows a programmable inter-measurement period to be periodically scanned. Ranging is done without involvement from the host, allowing the host to go into a low-power state. The host only wakes up upon a measurement interrupt. Programming criteria (distance and/or signal thresholds) allow the device to trigger an interrupt to the host only when the measurement is within the programmed criteria.
4. Lite ranging mode: This mode is obsolete.

Selection of the ranging preset modes is made using the driver function described in Section 3.2.4 Preset mode.

Note: Ranging and MultiZones modes are based on histogram processing

2.2 Ranging sequence

For all modes, except autonomous mode, the device runs with a handshake mechanism, based on a standard interrupt management scheme.

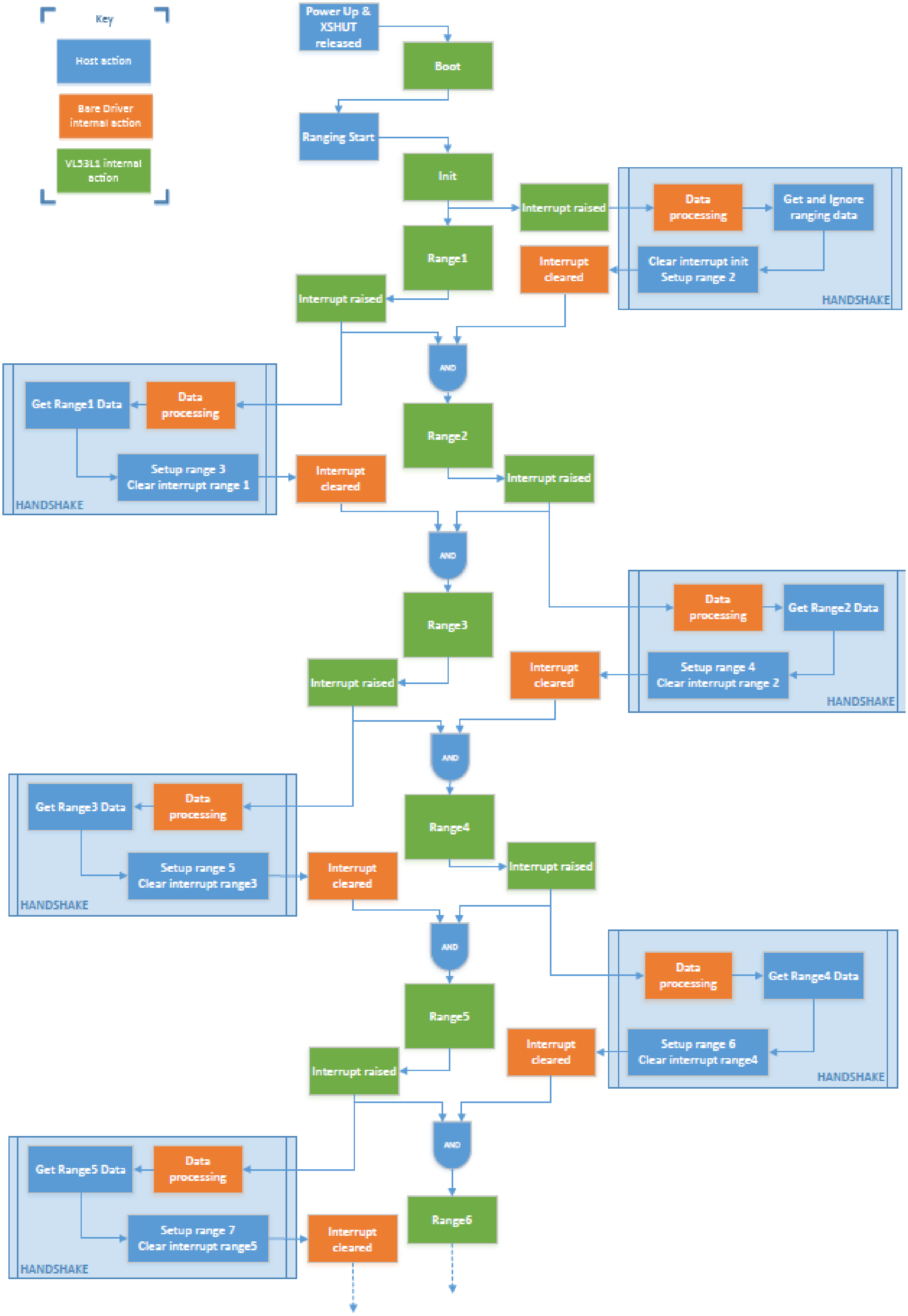
After each ranging, the host retrieves the ranging data and enables the next ranging by clearing the interrupt. This process is what is referred to as the handshake mechanism.

Next ranging is then triggered if the current one is finished and if the host has cleared the previous pending interrupt.

The interrupt mechanism allows the faster data transfer, without loosing any ranging value due to communication or asynchronism issues. During the handshake phase, the host performs some data processing.

The ranging sequence is functionally described in Figure 3. VL53L1 ranging sequence overview below.

Figure 3. **VL53L1 ranging sequence overview**



The handshake sequence allows internal parameters to be computed and applied to the next range.

The handshake must be performed by the user of the bare driver. It is very important to keep the delay as low as possible to enable ranging after a new measurement has been received by the host application. This maintains the desired measurement rate.

Note: Autonomous mode does not require the host to perform any action to continue to range. If the host is too slow to retrieve a measurement, the data are lost.

Note: In autonomous mode, there is no interrupt raised after init.

2.3 Timing considerations

The timings are presented in Figure 4. VL53L1 Ranging sequence and timing targets.

The host can get the latest available ranging during the ranging timing budget of the current range.

In all modes except Autonomous mode, if a delay to clear the interrupt is introduced by the host, the next ranging will be stopped until the pending interrupt is cleared.

Note: The timings in Figure 4. VL53L1 Ranging sequence and timing targets are example timings and can vary depending on the user implementation.

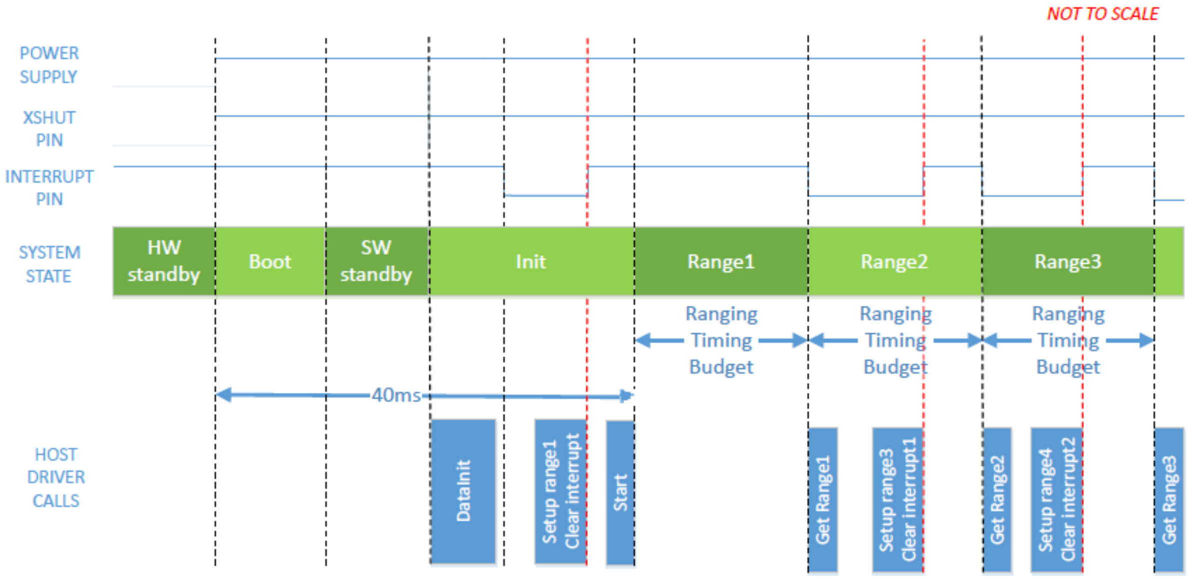
The host can change the default timing budget by using a dedicated driver function described in Section 5.4 Timing budget.

The host can decide to change the timing budget either to synchronize ranging with application, or to increase ranging accuracy.

Note: If tuning parameter to improve accuracy (VL53L1_TUNINGPARM_PHASECAL_PATCH_POWER) is used, the "init" time can go up to 300 ms.

In the Figure 4. VL53L1 Ranging sequence and timing targets, the "Boot", "SW standby" and "Init" lasts 40 ms. This time is needed to perform a correct initialization of the device, and it is independent of the platform or the used timing budget. The first range, "Range1", is not valid, since the wrap-around check is not possible. This means that the first valid ranging value is "Range2", available after 40 ms, plus twice the timing budget duration.

Figure 4. VL53L1 Ranging sequence and timing targets



2.4 Region-of-interest (ROI)

The user can get a ranging result on a specific part of the field of view (FoV), using ROIs.

A driver function is dedicated to this setup (see Section 5.1 Single ROI).

3 Description of basic bare driver functions

This section describes the driver function call flows that should be followed to perform a ranging measurement using the VL53L1.

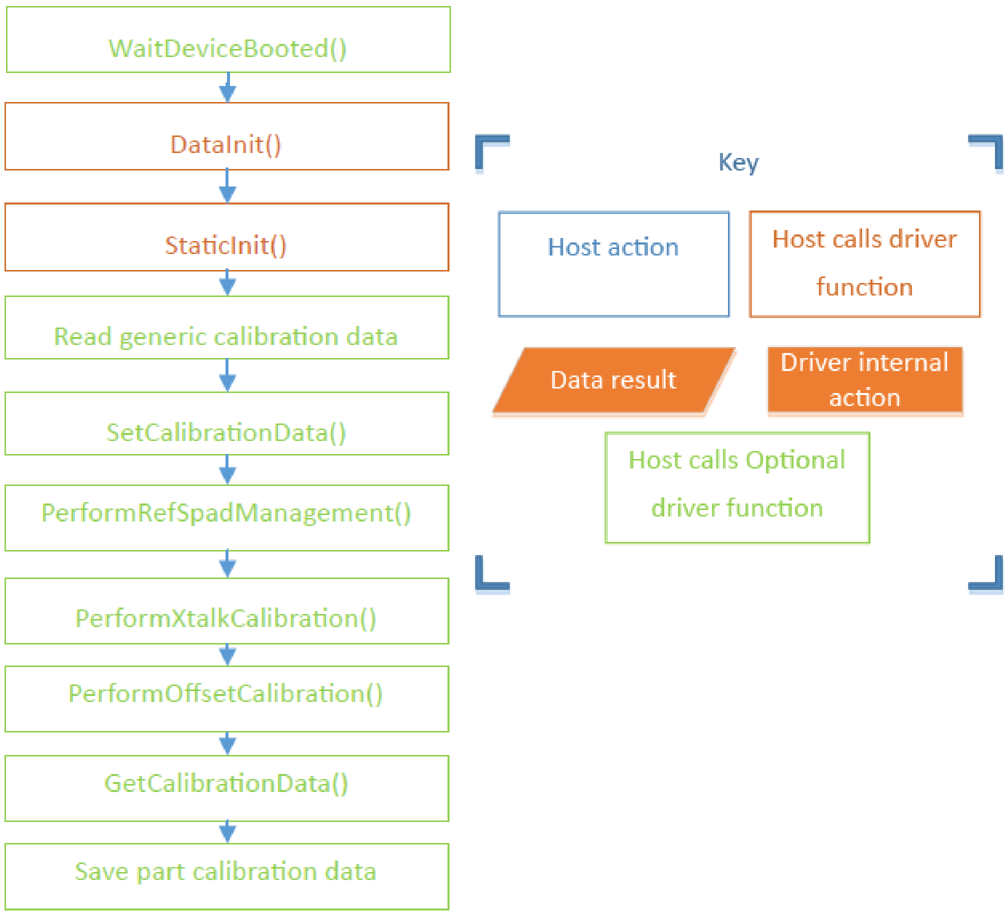
The VL53L1 drivers are used in two classes of applications:

- Factory applications (factory flow): this flow is used for device calibration, typically when a product is tested at the end of manufacturing.
- Field applications (ranging flow): this flow gathers all end user applications using the VL53L1 device.

3.1 Bare driver

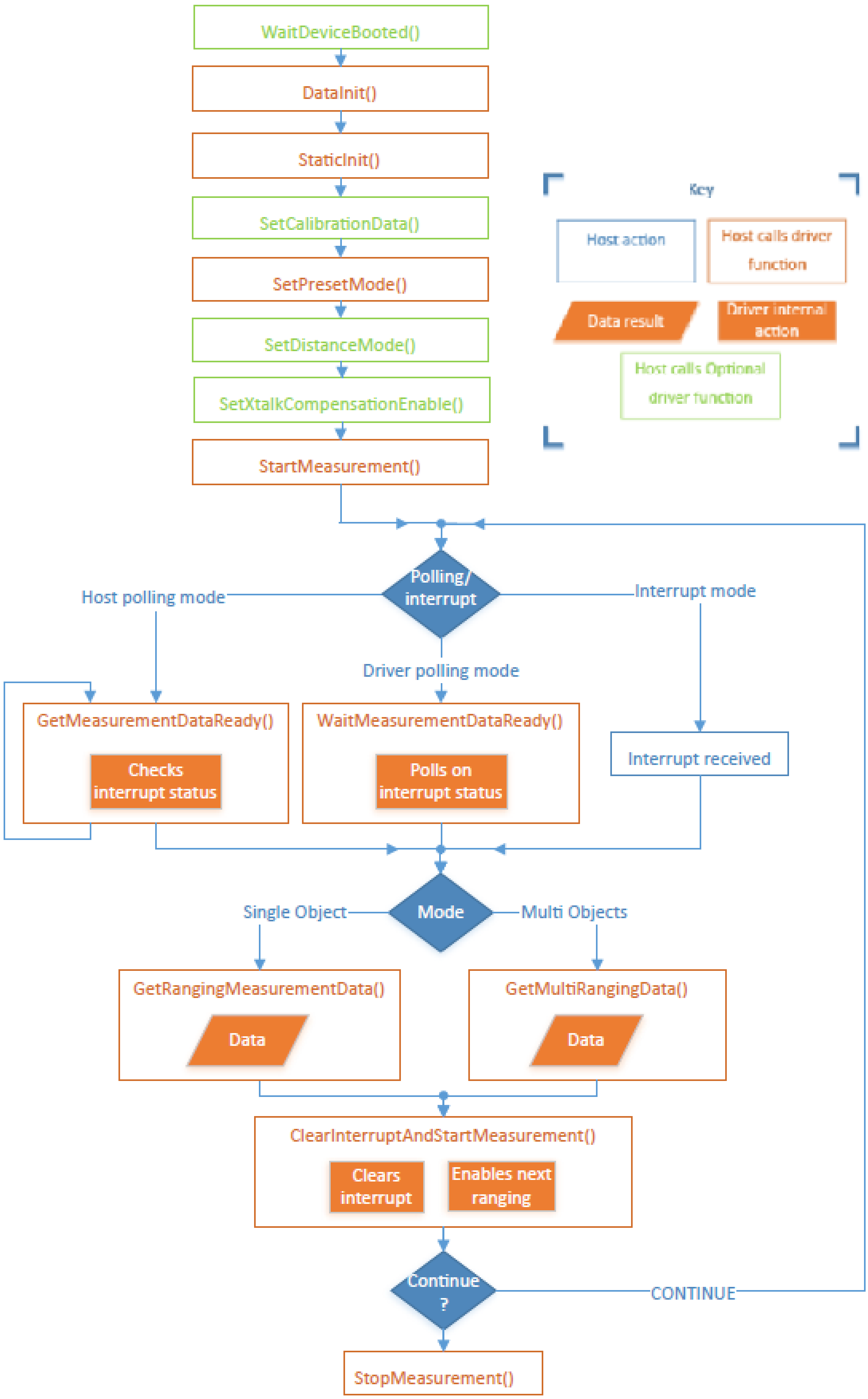
The bare driver factory flow is described in the following figure.

Figure 5. L53L1 API calibration flow (factory)



The bare driver ranging flow is described in Figure 6. VL53L1 API ranging flow (field). Please note that between the *SetCalibrationData()* and *StartMeasurement()*, some important functions are shown as an example, but the list is not complete. Parameters such as timing budget, offset correction mode, or smudge correction, can be set at this time as well.

Figure 6. VL53L1 API ranging flow (field)



3.2 System initialization

The following sections show the API function calls required to perform the system initialization, before starting a measurement.

3.2.1 Wait for boot

The *VL53L1_WaitDeviceBooted()* function ensures that the device is booted and ready.

It is not mandatory to call this function.

Note: This function blocks the host execution. This function should not block for more than 4 ms, assuming 400 kHz I2C and 2 ms latency per transaction.

3.2.2 DataInit

The *VL53L1_DataInit()* function is called only once, to perform the device initialization after the device is reset.

After calling the *VL53L1_DataInit()* function, the calibration data must be loaded using the function *VL53L1_SetCalibrationData()*.

3.2.3 StaticInit

The *VL53L1_StaticInit()* function allows device settings, specific for a given use case, to be loaded.

3.2.4 Preset mode

The *VL53L1_SetPresetMode()* function allows one preset mode to be chosen.

The following preset modes are available using this driver revision:

- VL53L1_PRESETMODE_RANGING
- VL53L1_PRESETMODE_MULTIZONES_SCANNING
- VL53L1_PRESETMODE_LITE_RANGING (Obsolete)
- VL53L1_PRESETMODE_AUTONOMOUS

The default preset mode is VL53L1_PRESETMODE_RANGING.

4 Ranging with the VL53L1

4.1 Starting a measurement

The *VL53L1_StartMeasurement()* function must be called to start a measurement.

4.2 Waiting for a result: polling or using an interrupt

To check if a measurement is available, the host can:
- Call a polling function
- Poll on a driver function
- Use a physical interrupt

4.2.1 Driver polling to get the result status

The function *VL53L1_WaitMeasurementDataReady()* polls on an internal status until a measurement is ready.

Note: This function is a blocking function since internal polling is performed.

4.2.2 Host polling to get the result status

The host can poll on the function *VL53L1_GetMeasurementDataReady()* to know when a new measurement is ready.

Note: This function is not a blocking function.

4.2.3 Using a physical interrupt

An alternative and preferred way to get the ranging status is to use the physical interrupt output. By default, the GPIO1 goes low when a new measurement is ready.

This pin is an output pin only; there is no input interrupt pin on this device.

The interrupt must be cleared by calling the driver function: *VL53L1_ClearInterruptAndStartMeasurement()*.

4.3 Getting a measurement

Depending on the selected mode, the hardware collects a single data measurement or a set of data measurements per ranging loop:
- In autonomous mode, only a single measurement is reported per ranging.
- In ranging mode, multiple objects can be detected per ranging, and data measurements are reported per object
- In multizone scanning mode, multiple zones of the FoV are scanned, and data measurements are reported per zone.

The bare driver provides two functions to select whether single ranging data or multiple ranging data are returned:
- The “Get a single data” function is only available in autonomous mode. This function should not be used in ranging and multizone scanning modes.
- The “Get multiple data” function is only available in ranging and multizone scanning modes. This function should not be used in autonomous mode.

Note: Two consecutive calls to get a measurement function is not allowed if the interrupt is not cleared between the two calls.

4.3.1 Getting a single ranging value

The *VL53L1_GetRangingMeasurementData()* can be used to get a single ranging data.

When calling this function to get the device ranging results, a structure called *VL53L1_RangingMeasurementData_t* is returned.

This structure is described in Section 4.5 Single ranging measurement data structure.

4.3.2 Getting multiple ranging values

The *VL53L1_GetMultiRangingData()* can be used to get multiple ranging data.

When calling this function to get the device multiple ranging results, a structure called *VL53L1_MultipleRangingData_t* is returned.

This structure is described in Section 5 Description of additional driver functions.

4.4 Stopping a measurement

In continuous mode, the host can stop the measurement by calling the *VL53L1_StopMeasurement()* function.

If the stop request occurs during a range measurement, the measurement is aborted immediately.

4.5 Single ranging measurement data structure

The *VL53L1_RangingMeasurementData_t* structure is composed of:

- TimeStamp: not implemented
- StreamCount: this 8-bit integer gives a counter incremented at each range. The value first starts at 0, incrementing to 255, and then incrementing from 128 to 255.
- RangingQualityLevel: not implemented
- SignalRateRtnMegaCps: this value is the return signal rate in MegaCountPer Second (MCPS). This is a 16.16 fixed point value.To obtain a real value, it should be divided by 65536.
- AmbientRateRtnMegaCps: this value is the return ambient rate (in MCPS). It is a 16.16 fixed point value, which is effectively a measure of the infrared light. To obtain a real value, it should be divided by 65536.
- EffectiveSpadRtnCount: this is a16-bit integer that returns the effective SPAD count for the current ranging. To obtain a real value, it should be divided by 256.
- SigmaMilliMeter: this 16.16 fixed point value is an estimation of the standard deviation of the current ranging, expressed in millimeters. To obtain a real value, it should be divided by 65536.
- RangeMilliMeter: this is a 16-bit integer giving the range distance in millimeters.
- RangeFractionalPart: not implemented
- RangeStatus: this is an 8-bit integer giving the range status for the current measurement. Value = 0 means the ranging is valid (refer to Table 1. Range status).
- ExtendedRange: this is a 8-bit integer indicating if the range has been unwrapped (only for long distances, using Ranging Mode)

4.6 MultiRangingData structure

When using ranging or multizone scanning modes, the returned structure is called: *VL53L1_MultiRangingData_t* and contains the following data:

- TimeStamp: not implemented
- StreamCount: this 8-bit integer gives a counter incremented at each range. The value first starts at 0, incrementing to 255, and then incrementing from 128 to 255.
- RoiNumber: this is an 8-bit integer value that gives the ROI the range data is related to.
- NumberOfObjectsFound: this 8-bit integer value gives the number of objects found in the current ROI.
- RoiStatus: it gives the status of the current ROI. The value of this field can be 0, 1 or 2, for “invalid ROI”, “valid ROI”, “valid and last ROI” respectively. This is mainly used to flag the last ROI for synchronization, when the scanning mode is used.
- RangeData[VL53L1_MAX_RANGE_RESULTS]: this is a table structure of type *VL53L1_TargetRangeData_t*, giving the results per ROI (see description below for more details). The maximum number of objects is given by VL53L1_MAX_RANGE_RESULTS and, by default, is equal to four.
- HasXtalkValueChanged: this 8-bit integer value indicates if the crosstalk value has been changed.
- EffectiveSpadRtnCount: this is a16-bit integer that returns the effective SPAD count for the current ranging. To obtain a real value, it should be divided by 256.
- DmaxMilliMeter: this is a 16-bit integer. When the target is not detected, and the ranging is valid (range status 255 only), no object can be detected between zero and dmax. This value can help the host algorithm when no valid ranging data is found.

- RecommendedDistanceMode: this is a value with VL53L1_DistanceModes type. It indicates the distance mode that is recommended based on the ranging conditions, in order to get the best accuracy, focusing on the detected objects.

The *VL53L1_TargetRangeData_t* structure is composed of:

- RangingQualityLevel: not implemented.
- RangeMaxMilliMeter: this a 16-bit integer, giving the larger detected distance.
- RangeMinMilliMeter: this is a 16-bit integer, giving the smaller detected distance.
- SignalRateRtnMegaCps: this value is the return signal rate in MegaCountPer Second (MCPS). This is a 16.16 fixed point value. To obtain a real value, it should be divided by 65536.
- AmbientRateRtnMegaCps: this value is the return ambient rate (in MCPS). It is a 16.16 fixed point value, which is effectively a measure of the amount of light hitting the SPAD matrix. To obtain a real value, it should be divided by 65536.
- SigmaMilliMeter: this 16.16 fixed point value is an estimation of the standard deviation of the current ranging, expressed in millimeters. To obtain a real value, it should be divided by 65536.
- RangeMilliMeter: this is a 16-bit integer giving the range distance in millimeters.
- RangeFractionalPart: not implemented.
- RangeStatus: this is an 8-bit integer giving the range status for the current measurement. Value = 0 means the ranging is valid (refer to Table 1. Range status).

For the multi ranging data structure only (ranging and multizones scanning modes), a particular behavior is implemented when the target is not detected. If the target is not detected, and the measurement is valid, the following values are reported into the *VL53L1_TargetRangeData_t* structure:

- RangeMaxMilliMeter: forced to 8191.
- RangeMinMilliMeter: forced to 8191.
- SignalRateRtnMegaCps: forced to 0.
- AmbientRateRtnMegaCps: the ambient rate value is normally computed.
- SigmaMilliMeter: forced to 0.
- RangeMilliMeter: forced to 8191.
- RangeStatus: forced to 255.

Table 1. Range status

Value	RangeStatus string	Comment
0	VL53L1_RANGESTATUS_RANGE_VALID	Ranging measurement is valid
1	VL53L1_RANGESTATUS_SIGMA_FAIL	Raised if sigma estimator check is above the internal defined threshold. Sigma estimator gives qualitative information about the signal.
2	VL53L1_RANGESTATUS_SIGNAL_FAIL	Raised in autonomous mode only if signal value is below the internally defined threshold.
3	VL53L1_RANGESTATUS_RANGE_VALID_MIN_RANGE_CLIPPED	Raised in autonomous mode only if ranging is below a threshold defined in the tuning parameter.
4	VL53L1_RANGESTATUS_OUTOFBOUNDS_FAIL	Raised when the range result is out of bounds.
5	VL53L1_RANGESTATUS_HARDWARE_FAIL	Raised in case of HW or VCSEL failure
6	VL53L1_RANGESTATUS_RANGE_VALID_NO_WRAP_CHECK_FAIL	No wraparound check has been done
7	VL53L1_RANGESTATUS_WRAP_TARGET_FAIL	Wrapped target not matching phases
8	VL53L1_RANGESTATUS_PROCESSING_FAIL	Internal algorithm underflow or overflow in autonomous ranging.
9	VL53L1_RANGESTATUS_XTALK_SIGNAL_FAIL	Raised in autonomous modes when signal is below the crosstalk threshold.
10	VL53L1_RANGESTATUS_SYNCRONISATION_INT	Raised once after start, ranging value has to be ignored.

Value	RangeStatus string	Comment
11	VL53L1_RANGESTATUS_RANGE_VALID_MERGED_PULSE	Raised in ranging and multizone scanning modes only. Range is OK, but the distance reported is the result of multiple targets merged together.
12	VL53L1_RANGESTATUS_TARGET_PRESENT_LACK_OF_SIGNAL	Indicates that there is a target, but the signal is too low to report ranging
13	VL53L1_RANGESTATUS_MIN_RANGE_FAIL	Programmed ROI is not valid, selected ROI is out of the SPAD array.
14	VL53L1_RANGESTATUS_RANGE_INVALID	The reported range is invalid, ignore it
255	VL53L1_RANGESTATUS_NONE	Single ranging value: ranging is not updated, ignore reported value Multiple ranging values: target not detected.

Note: The very first measurement does not include a wraparound check. This ranging measurement should be discarded. When VL53L1_PRESETMODE_AUTONOMOUS is used, the driver deletes the first ranging value.

Note: Range status 1 is often caused by noisy measurements. sigma estimator is strongly impacted by the SNR of the treated signals.

Note: Range status 4 is raised when some error on the measurement reference occurs. This can cause outliers as negative measurements or extremely high ranging values.

5 Description of additional driver functions

5.1 Single ROI

To set an ROI, so that it is different from the default 16x16 one, the user can call the *VL53L1_SetROI()* function.

The ROI is a square or rectangular area defined by two corners: top left and bottom right.

Four coordinates are used to localize these two corners on the full SPAD array:

- TopLeftX
- TopLeftY
- BotRightX
- BotRightY

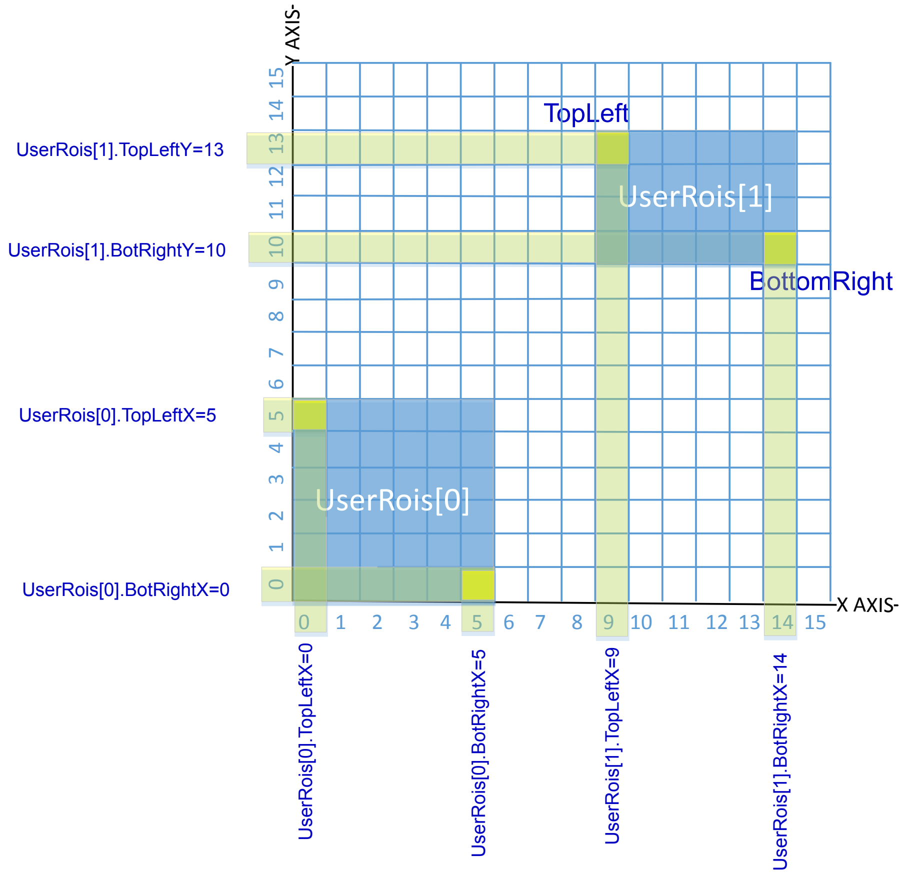
These coordinates are part of a structure of type named *VL53L1_UserROI_t*.

The user has to define the ROI coordinate values in the structure, and call the driver function to apply the ROI change.

The minimum ROI size is 4x4.

An example of a ROI setting is given in the following figure.

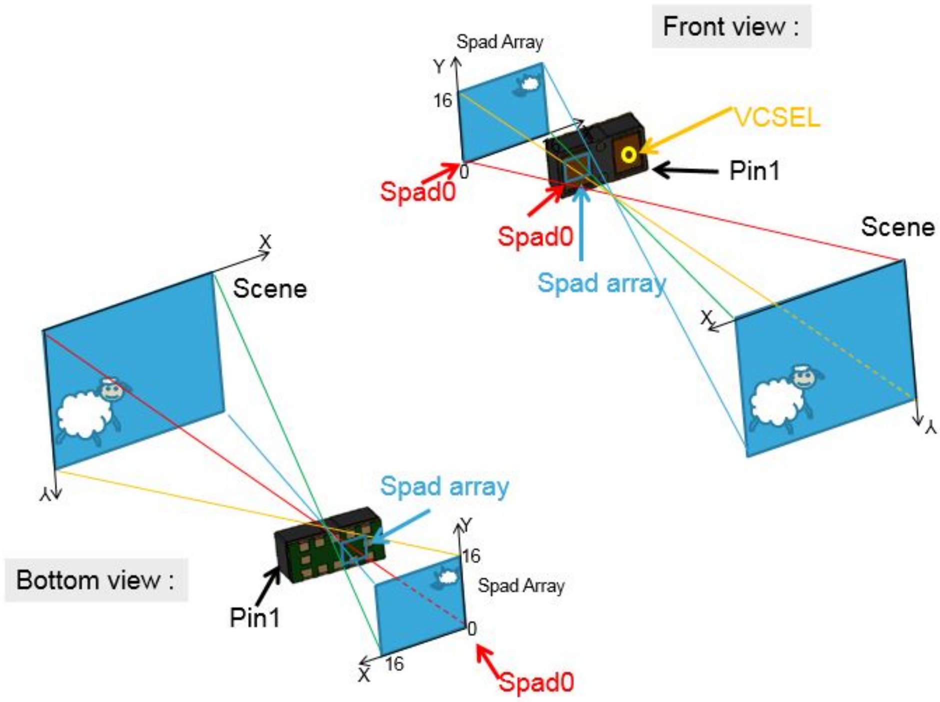
Figure 7. VL53L1 ROI setting example



5.2 SPAD array coordinates versus scene

The following figure shows the coordinates of an object in the SPAD array compared to the location in the FoV.

Figure 8. VL53L1 coordinates vs Scene



5.3 Multiple ROIs for multizone scanning mode

When using multizone scanning mode, the user can set multiple ROIs. The VL53L1 manages this list as follows: it starts ranging for the first ROI, raises an interrupt, moves to next ROI, and restarts the list until the user calls a stop function. When the last ROI is activated, the flag ROIStatus is set to 2, then the next ROI is in the first position of the array (see Section 4.6 MultiRangingData structure).

Note: *Ranging data are available and have to be read by the user as soon as the interrupt is triggered.*

The maximum number of ROIs is defined by the parameter VL53L1_MAX_USER_ZONES. Its default value is 16 (can be set up to 169).

Note: *In order to save memory, the user can decrease the maximum number of ROIs to the actual maximum that is programmed in the final application.*

Note: *The set of ROIs cannot be changed on the fly. The user has to stop ranging, change the ROIs and restart.*

5.3.1 Bare driver structure

The structure *VL53L1_RoiConfig_t* is defined in the bare driver by:

- NumberOfRoi: an 8-bit integer which defines the number of ROIs
- UserRois[VL53L1_MAX_USER_ZONES]: a *VL53L1_UserRoi_t* table structure which contains the ROI coordinates.

The *VL53L1_UserRoi_t* structure contains the coordinates of an ROI:

- TopLeftX: 8-bit integer that gives the top left x coordinate (0-15)
- TopLeftY: 8-bit integer that gives the top left y coordinate (0-15)
- BotRightX: 8-bit integer that gives the bottom right x coordinate (0-15)
- BotRightY: 8-bit integer that gives the bottom right y coordinate (0-15)

Example of *VL53L1_UserRoi_t* to set 2 ROIs (based on Figure 7. VL53L1 ROI setting example):

```
VL53L1_RoiConfig_t roiConfig;
roiConfig.NumberOfRoi = 2;
roiConfig.UserRois[0].TopLeftX = 0;
roiConfig.UserRois[0].TopLeftY = 5;
roiConfig.UserRois[0].BotRightX = 5;
roiConfig.UserRois[0].BotRightY = 0;
roiConfig.UserRois[1].TopLeftX = 9;
roiConfig.UserRois[1].TopLeftY = 13;
roiConfig.UserRois[1].BotRightX = 14;
roiConfig.UserRois[1].BotRightY = 10;
status = VL53L1_SetROI(&VL53L1Dev, &roiConfig);
```

5.4 Timing budget

The timing budget is the time allocated by the user to perform one range measurement.

The *VL53L1_SetMeasurementTimingBudgetMicroSeconds()* is the function to use.

The typical, minimum and maximum timing budgets are described in Table 2. Timing budget values (in ms).

The following example sets the timing budget to 66ms:

status = VL53L1_SetMeasurementTimingBudgetMicroSeconds(&VL53L1Dev, 66000)

The function *VL53L1_GetMeasurementTimingBudgetMicroSeconds()* allows the programmed timing budget to be received.

Table 2. Timing budget values (in ms)

PresetMode	Min. (1)	Typ. (2)	Max.
Ranging	8	16	500
MultiZone scanning	8	16	500
Autonomous	20	40	1000

1. *Warning: Ranging performances are not guaranteed at this timing budget. The device is functional but the detection rate, maximum ranging distance, and accuracy are not in line with those of the datasheet.*
2. *The typical timing budget allows the performances mentioned in the datasheet to be reached.*

Default timing budget (after *VL53L1_DataInit()*) is 33 ms for the ranging preset mode.

When multiple ROIs are used, each zone ranging is an individual ranging operation and the timing budget applies for each zone ranging.

5.5 Distance mode

A function has been added to optimize the internal settings and tunings depending on the ranging distance requested by the user.

The benefit of changing the distance mode is detailed in the following table.

Table 3. Distance modes

Preset mode	Possible distance mode	Benefit/comments
Ranging	Short	Better ambient immunity
	Medium (default)	Maximum distance
	Long	Lower power consumption
Scanning	Short	Better ambient immunity
	Medium (default)	
	Long	Maximum distance
Autonomous	Short	Better ambient immunity
	Medium (default)	
	Long	Maximum distance

The function to use is called *VL53L1_SetDistanceMode()*.

Note: *This function has to be called AFTER the VL53L1_SetPresetMode() function.*

Note: *In Autonomous mode, a ranging offset of 5 mm can be observed in short and medium modes compared to long mode.*

5.5.1 Recommended distance mode

In VL53L1_PRESETMODE_RANGING and VL53L1_PRESETMODE_MULTIZONES_SCANNING, the driver monitors the ranging conditions and provides a recommendation of distance mode, in case it detects that a better distance mode provides better accuracy covering the detected object range. This recommendation is provided unconditionally, as part of the *VL53L1_MultiRangingData_t* structure. It is up to the application to monitor this value, to check if it is different from the programmed distance mode, and decide whether to ignore or follow the recommendation. If the application decides to follow the recommendation and change the distance mode, it has to proceed as follows:

1. *VL53L1_StopMeasurement()*
2. *VL53L1_SetDistanceMode(RecommendedDistanceMode)*
3. *VL53L1_StartMeasurement()*

In VL53L1_PRESETMODE_AUTONOMOUS mode, the driver does not detect whether a better distance mode exists and always reports the programmed mode as recommended distance mode.

5.6 Inter-measurement period

In autonomous mode, the device can be configured so that the ranging rate is reduced: the device ensures that there is the programmed inter-measurement period between two consecutive ranging measurements. In the inter-measurement period, the device is in a low-power inter-measurement state.

It is recommended to set the inter-measurement period greater than timing budget.

This inter-measurement configuration is ignored in ranging, and multizone scanning modes.

By default, ranging data are reported to the host every ranging, so every inter-measurement period. The report can be made using an interrupt.

VL53L1_SetInterMeasurementPeriodMilliSeconds() is the function to be used.

The following example sets the inter-measurement period to 1 second:

status = VL53L1_SetInterMeasurementPeriodMilliSeconds(&VL53L1Dev, 1000)

The function *VL53L1_GetInterMeasurementPeriodMilliSeconds()* allows reception of the programmed inter-measurement period.

The default value is 100 ms.

5.7 Autonomous mode thresholds

In autonomous mode, the device can be configured so that ranging is reported to the host only when the ranging data match a configured set of criteria.

Detection mode

Whether reports to the host are filtered or not depends on the configuration of detection mode.

Report filtering conditions:

- 0: No filter (all ranging generate a report to the host)
- 1: Filter on distance criteria only
- 2: Filter on signal rate criteria only
- 3: Filter on a combination of two criteria
- 4: Filter on one or the other criteria

By default, the filter is disabled.

Distance mode

Distance mode defines how the distance criteria is defined:

- 0: below a certain distance (if the object distance is greater than the configured distance or no object is found, there is no report).
- 1: beyond a certain distance
- 2: within a distance range (min/max)
- 3: out of the distance range (min/max)

Distance low

The minimum distance in millimeters.

Distance high

The maximum distance in millimeters.

Signal rate mode

Similar to distance mode but for the signal rate.

Signal rate low

The minimum signal rate in MCPS

Signal rate high

The maximum signal rate in MCPS

No target

If no target is detected, no interrupt is generated if the “no target” criterion is not set (default). It is recommended to set this value if the user wants to be notified that an object is beyond a maximum distance (for example, an object that is too far to be detected does not otherwise generate an interrupt).

VL53L1_SetThresholdConfig() is the function to use.

Example, to program the device to report ranging only when an object is detected within 10 cm and 1 m:

```
detectionConfig.DetectionMode = 1
detectionConfig.Distance.CrossMode = 2
detectionConfig.Rate.CrossMode = 0
detectionConfig.IntrNoTarget = 0
detectionConfig.Distance.High = 1000
detectionConfig.Distance.Low = 100
detectionConfig.Rate.High = 0
detectionConfig.Rate.Low = 0
status = VL53L1_SetThresholdConfig(&VL53L1Dev, &detectionConfig );
```

The function *VL53L1_GetThresholdConfig()* allows reception of the programmed report filtering/threshold configuration.

5.8 Live crosstalk correction

Crosstalk (xtalk) is defined as the amount of signal received on the SPAD array which is due to VCSEL light reflection inside the protective window (cover glass) added on top of the module.

The VL53L1 embeds a function able to measure the crosstalk value on the fly and apply a new crosstalk correction value.

This functionality can be used to make sure the crosstalk value is always the right one. It is useful particularly when smudge is added to the cover glass.

The user can enable/disable this function by calling: *VL53L1_SmudgeCorrectionEnable()*.

Three options can be set with this function:

- VL53L1_SMUDGE_CORRECTION_NONE to disable the correction
- VL53L1_SMUDGE_CORRECTION_CONTINUOUS to enable a continuous correction
- VL53L1_SMUDGE_CORRECTION_SINGLE to enable a single correction after a start command is received.

Live crosstalk detection runs at each ranging. A new crosstalk value is computed if the following conditions are met:

- no object below 80 cm
- ambient light level below a threshold
- crosstalk value above 1kcps

The threshold of ambient level is a tuning parameter (#32900). This parameter is a 16.16 format, the coded value is by default 57671680 (Encoded Value = 880). If ambient rate is measured above this threshold, the smudge correction will not be applied. This reduces noise on computed smudge value when high ambient signal is present.

Setting this parameter to a higher value (for example 115343360) will allow to get smudge function reporting new crosstalk values under stronger ambient light conditions. The possible drawback is an inaccuracy in the crosstalk value.

Refer to Section 5.11 Tuning parameters to get more details on how to set a tuning parameter.

If the user sets the live crosstalk correction, the crosstalk value is corrected and the flag HasXtalkValueChanged (described in Section 5 Description of additional driver functions) is set. This flag is cleared automatically at the next range.

By default the smudge correction is disabled.

5.9 Ranging gain

In order to reduce possible ranging errors, a ranging gain has to be applied to the device. This gain is a factor directly applied to the final ranging.

Default values are proposed in Table 4. Ranging factor default values. These data are the ones used to get the performances described in the datasheet.

The user can change these default values if needed. The customer can perform a full ranging sweep, monitor the ranging data, and compute the ranging error. Based on the results, the customer decides if the default value has to be changed or not.

This ranging gain factor is available through the set/get calibration data function

The function *VL53L1_GetCalibrationData()* allows the reception of all calibration data. The returned structure *VL53L1_CalibrationData_t* contains another structure called *VL53L1_gain_calibration_data_t* which contains the two possible offset gain values:

- standard_ranging_gain_factor
- histogram_ranging_gain_factor

Table 4. Ranging factor default values

PresetMode	standard_ranging_gain_factor	histogram_ranging_gain_factor
Ranging, multizone scanning	NA	0.971 (default)
Autonomous	0.982 (default)	NA

The maximum gain value is 7. Setting a higher value generates incorrect ranging values.

If the user sets the gain at 0.5, the final ranging reported by the driver is multiplied by 0.5.

The user can set the ranging gain factors by calling *VL53L1_SetCalibrationData()*.

5.10 Optical center coordinates

Due to assembly tolerances, the optical center of the device can vary. The optical center of the device is measured for each part. The optical center coordinates are stored in the device NVM.

The user has access to the optical center coordinates by calling: *VL53L1_GetCalibrationData()*. The returned structure *VL53L1_CalibrationData_t* contains another structure of type *VL53L1_optical_centre_t* which contains the two coordinates:

- x_centre
- y_centre

The host can use these two coordinates to better align the ROIs to the optical center.

5.11 Tuning parameters

An extra function exists to load tuning parameters. For specific use cases, ST can recommend the use of some specific parameters composed of a key and value.

Note: Changing a tuning parameter can have strong effects on the device performances. ST recommends using the default values to benefit of optimal settings.

A dedicated function exists to load this tuning parameter: *VL53L1_SetTuningParameter()*.

5.12 VDDIO configuration

As described in the VL53L1 Datasheet (DS11786), the user can select two modes for the VDDIO value: 1V8 or 2V8 modes.

The selection of the mode is made directly in the code though a compilation key called USE_I2C_2V8.

If this compilation key is defined, the system goes into 2V8 mode, otherwise, it is kept in the default 1V8 mode.

5.13 Dmax

Dmax is computed for each ranging value, and it represents an indication of the maximum distance at which a target can be detected, in the current ambient conditions. This value depends mainly on the ambient light and on the target reflectance. The latter value is indicated by the user, using the function *VL53L1_SetDmaxReflectance()*.

The following example sets the reflectance as 50%:

status = VL53L1_SetDmaxReflectance(&VL53L1Dev, 50);

This means that the dmax indicates the max distance at which a 50% target can be detected.

Note: Dmax needs a calibration, to be operational. The dmax calibration is automatically performed during offset calibration. Refer to Section 6.3 Offset calibration to learn more about dmax calibration.

5.14 I2C address

The default I2C address of the VL53L1 is 0x52.

Some applications need to set a different I2C device address. This is the case, for example, when several VL53L1 parts share the same I2C bus.

The customer should apply the following procedure:

1. The board mounting the VL53L1s has to be designed carefully. The Xshut and the GPIO1 (interrupt) pins have to be controlled individually for each VL53L1.
2. The host has to:
   – Put in Hw Standby, setting the Xshut pin low, all the VL53L1s
   – Raise the Xshut pin of one of the VL53L1s
   – Call the function *VL53L1_SetDeviceAddress()*
3. The host repeats the last three points until all the VL53L1s' addresses are correctly set.

For example, to value of *WantedAddress* sets the new I2C address when calling the following function:

status = VL53L1_SetDeviceAddress(&VL53L1Dev, WantedAddress)

6 Calibration functions

To benefit from device full performance, the VL53L1 driver includes calibration functions to be run once at the customer production line.

Calibration procedures have to be run to compensate part-to-part parameters that may affect the device performances.

Calibration data stored in the host have to be loaded in the VL53L1 at each startup using a dedicated driver function.

Three calibrations are needed: refSPAD, crosstalk, and offset.

Offset calibration also allows calibration of the dmax value.

The order the calibration functions are called is important. They should be called in the following order:

1. refSPAD
2. crosstalk
3. offset

The three calibration functions can be made in sequential mode, or individually. When run individually, the data from the previous step have to be loaded before running the calibration.

The crosstalk part-to-part calibration can be replaced by a live crosstalk calibration. This feature is described in Section 7.1 Live crosstalk calibration.

6.1 RefSPAD calibration

The number of SPADs is calibrated during the final module test at ST. This part-to-part value is stored into the NVM and automatically loaded into the device during bootup.

This calibration allows the number of SPADs to optimize the device dynamic to be adjusted.

However, adding a cover glass on top of the module may affect this calibration. ST recommends that the customer performs this calibration again in the final product application.

The same algorithm running at FMT is applied when this function is called. The algorithm searches through three possible types of SPAD:

- 1 (non attenuated SPAD)
- 2 (SPAD attenuated by a factor of five)
- 3 (SPAD attenuated by a factor of ten)

The number and type of SPAD is selected to avoid internal signal saturation.

6.1.1 RefSPAD calibration function

A dedicated function is available for this operation: *VL53L1_PerformRefSpadManagement(VL53L1_DEV Dev)*

Note: This function must be called first in the calibration procedure.

The function outputs the following three warning messages:

- VL53L1_WARNING_REF_SPAD_CHAR_NOT_ENOUGH_SPADS
  Less than five good SPADs are available, output not valid
- VL53L1_WARNING_REF_SPAD_CHAR_RATE_TOO_HIGH
  At the end of a search reference rate greater than 40.0 Mcps, the range stability may be degraded.
- VL53L1_WARNING_REF_SPAD_CHAR_RATE_TOO_LOW
  At the end of a search reference rate less than 10.0 Mcps, the range stability may be degraded.

6.1.2 RefSPAD calibration procedure

No particular condition has to be followed for this calibration, except that no target should be placed on top of the device.

The time to perform this calibration is a few milliseconds.

This function should be called after the *VL53L1_DataInit()* and *VL53L1_StaticInit()* functions are called.

6.1.3 Getting refSPAD calibration results

The function *VL53L1_GetCalibrationData()* allows reception of all calibration data. The returned structure *VL53L1_CalibrationData_t* contains another structure called *VL53L1_customer_nvm_managed_t* which contains the following eight refSPAD calibration parameters:

- ref_spad_man__num_requested_ref_spads: this value is between 5 and 44. It gives the number of SPADs selected.
- ref_spad_man__ref_location: this value can be 1, 2, or 3. It gives the type of SPAD area.
- Six additional parameters give the correct SPAD maps for the location selected.
  - global_config__spad_enables_ref_0
  - global_config__spad_enables_ref_1
  - global_config__spad_enables_ref_2
  - global_config__spad_enables_ref_3
  - global_config__spad_enables_ref_4
  - global_config__spad_enables_ref_5

6.1.4 Setting refSPAD calibration data

At each startup, after an initial boot, the customer field application can load the refSPAD calibration data after the *VL53L1_DataInit()* and *VL53L1_StaticInit()* functions are called, by using *VL53L1_SetCalibrationData()*.

It is better to get the entire calibration structure by:

1. Calling *VL53L1_GetCalibrationData()*
2. Modifying the eight parameters described in Section 6.1.3 Getting refSPAD calibration results
3. Calling *VL53L1_SetCalibrationData()*

6.2 Crosstalk calibration

Crosstalk is defined as the amount of signal received on the return array which is due to VCSEL light reflection inside the protective window (cover glass) on top of the module.

Depending on the cover glass quality, this signal affects the device performance. In the VL53L1, this phenomenon is compensated for by a built-in correction function.

Depending on the PresetMode used and the object distance, the VL53L1 is immune to crosstalk. The following table gives an overview of the device capabilities.

Table 5. Crosstalk vs Preset mode

Preset mode	Distance	Crosstalk correction
Ranging, MultiZone scanning	< 80cm	Correction needed
	> 80cm	Crosstalk immune, no correction needed
Autonomous	All	Correction needed

Crosstalk calibration is used to estimate the amount of correction needed to compensate the effect of a cover glass added on top of the module.

The calibration procedure is unique and is applicable to all preset modes.

An important factor new to the VL53L1 is the inconsistent crosstalk due to the presence of a lens on top of the SPAD array. This explaine why the plane fit model of crosstalk is required rather than the previous product generation. Consequently, the output of the crosstalk contains many parameters that define the plane model, while previous product generation had only one parameter.

Crosstalk is composed of several parameters:

- x gradient
- y gradient
- offset and crosstalk shape

6.2.1 Crosstalk calibration function

A dedicated function is available for this operation: *VL53L1_PerformXTalkCalibration(&VL53L1Dev, CalibrationType);*

Note: This function must be called in second position in the calibration procedure, after the refSPAD calibration is made, and before the offset calibration.

Two calibration types are available:

- Standard crosstalk calibration. To select this calibration use CalibrationType = VL53L1_XTALKCALIBRATIONMODE_NO_TARGET. This crosstalk calibration is optimized for high crosstalk cover glasses (crosstalk above 5kcps).
- FullROI crosstalk calibration. To select this calibration use CalibrationType = VL53L1_XTALKCALIBRATIONMODE_FULL_ROI. This crosstalk calibration is faster and optimized for low crosstalk cover glasses (crosstalk below 5kcps).

6.2.2 Standard crosstalk calibration procedure

The customer has to ensure that no targets are within the device field of view (FoV) below 80 cm during the crosstalk calibration.

Crosstalk calibration should be conducted in a dark environment with no IR contribution.

Note: If a target or an object is present in the FoV of the device below 80 cm, the calibration result will not be correct and will affect the ranging performance.

Note: The crosstalk will fail if the closest target detected (the cover glass) is measured above 50 mm.

A dedicated calibration function has to be called after the *VL53L1_StaticInit()*, *VL53L1_DataInit()*, and *VL53L1_PerformRefSpadManagement()* functions are called: *VL53L1_PerformXTalkCalibration(&VL53L1Dev, 0);*

When these functions are called, standard crosstalk calibration is performed and the crosstalk correction is applied by default.

6.2.3 FullROI crosstalk calibration procedure

To perform the fullROI crosstalk calibration, a gray target (17% reflectance) has to be placed at a distance of 600 mm from the device.

Crosstalk calibration should be conducted in a dark environment with no IR contribution.

VL53L1_PerformXTalkCalibration(&VL53L1Dev, 2); is a dedicated calibration function which must be called after the *VL53L1_StaticInit()*, *VL53L1_DataInit()*, and *L53L1_PerformRefSpadManagement()* functions are called.

When these functions are called, full ROI crosstalk calibration is performed and the crosstalk correction is applied by default.

6.2.4 Generic shape crosstalk calibration

Generic shape crosstalk calibration is a way to get rid of part to part crosstalk calibration.

This option is only valid for ranging and multizone scanning modes.

The principle is as follows:

- The user characterizes the crosstalk in the final application, using a set of cover glasses. The chosen cover glasses should represent the whole production spread, to include in the analysis all the possible crosstalk values. The crosstalk data are sent to ST, and the generic shape is then computed and sent back to the customer. To use the generic shape crosstalk calibration, the customer has to load it into the crosstalk fields of the calibration structure, using the *VL53L1_SetCalibrationData()* function, as described in the Section 6.2.6 Setting crosstalk calibration data.
- During part to part calibration, the user must:
  - perform a refSPAD calibration
  - load generic shape data
  - perform offset calibration
- During standard operation, the user must:
  - load calibration data, including the generic shape
  - enable live crosstalk detection
  - start ranging

Live crosstalk detection has the same limitations as described in Section 5.8 Live crosstalk correction.

6.2.5 Getting crosstalk calibration results

Calibration results consist of a plan of values that are applied across the SPAD array. This plan is defined by an offset and two coordinates, and a histogram structure.

The function *VL53L1_GetCalibrationData()* allows the reception of all calibration data. The returned structure *VL53L1_CalibrationData_t* contains two other structures. The first one, called *VL53L1_customer_nvm_managed_t*, contains the three crosstalk calibration results:

- algo_crosstalk_compensation_plane_offset_kcps is a fixed point 7.9 coded value. It has to be divided by 512 to get the actual number.
- algo_crosstalk_compensation_x_plane_gradient_kcps is a fixed point 5.11 coded value. It has to be divided by 2048 to get the actual number.
- algo_crosstalk_compensation_y_plane_gradient_kcps is a fixed point 5.11 coded value. It has to be divided by 2048 to get the actual number.

The other structure returned is called *VL53L1_xtalk_histogram_data_t*.

This structure contains important extra data like the shape phase position so the crosstalk correction can correctly realign the shape for different phase calibration results.

This structure, combined with offset, x and y gradients, allows the generation of a crosstalk histogram for ranging and multizone scanning modes.

6.2.6 Setting crosstalk calibration data

The customer can load the crosstalk calibration data after the *VL53L1_DataInit()* and *VL53L1_StaticInit()* functions are called, by using: *VL53L1_SetCalibrationData()*

It is better to call *VL53L1_GetCalibrationData()* and then modify the following parameters:

- algo__crosstalk_compensation_plane_offset_kcps
- algo__crosstalk_compensation_x_plane_gradient_kcps
- algo__crosstalk_compensation_y_plane_gradient_kcps
- xtalkhisto.xtalk_shape.zone_id
- xtalkhisto.xtalk_shape.time_stamp
- xtalkhisto.xtalk_shape.first_bin
- xtalkhisto.xtalk_shape.buffer_size
- xtalkhisto.xtalk_shape.number_of_bins
- xtalkhisto.xtalk_shape.bin_data[0]
- xtalkhisto.xtalk_shape.bin_data[1]
- xtalkhisto.xtalk_shape.bin_data[2]
- xtalkhisto.xtalk_shape.bin_data[3]
- xtalkhisto.xtalk_shape.bin_data[4]
- xtalkhisto.xtalk_shape.bin_data[5]
- xtalkhisto.xtalk_shape.bin_data[6]
- xtalkhisto.xtalk_shape.bin_data[7]
- xtalkhisto.xtalk_shape.bin_data[8]
- xtalkhisto.xtalk_shape.bin_data[9]
- xtalkhisto.xtalk_shape.bin_data[10]
- xtalkhisto.xtalk_shape.bin_data[11]
- xtalkhisto.xtalk_shape.phasecal_result__reference_phase
- xtalkhisto.xtalk_shape.phasecal_result__vcsel_start
- xtalkhisto.xtalk_shape.cal_config__vcsel_start
- xtalkhisto.xtalk_shape.vcsel_width
- xtalkhisto.xtalk_shape.fast_osc__frequency
- xtalkhisto.xtalk_shape.zero_distance_phase

After the values of the structure have been modified, the user has to call *VL53L1_SetCalibrationData()*.

6.2.7 Enable/disable crosstalk compensation

The function to enable or disable the crosstalk compensation is: *VL53L1_SetXTalkCompensationEnable()*.

Calling *VL53L1_SetXTalkCompensationEnable&VL53L1Dev, 0);* disables the crosstalk compensation.

Calling *V53L1_SetXTalkCompensationEnable&VL53L1Dev, 1);* enables the crosstalk compensation.

Note: This function does not perform any calibration or crosstalk data loading, it enables the compensation.

Note: Calibration or loading of calibration data functions have to be called separately from the enable/disable function.

Note: Crosstalk compensation is disabled by default.

6.2.8 Crosstalk calibration error -22

An error -22 may be issued after calibration if the system failed to find a valid crosstalk value. This error is typically raised when the standard crosstalk calibration is coupled with a low crosstalk (belove 5kcps) cover glass. In this case it is strongly suggested to use fullROI crosstalk calibration.

If fullROI crosstalk calibration is not possible, then no crosstalk data should be applied.

In case of error, the crosstalk calibration data structure is not modified and can contain erroneous data.

It is recommended that the user sets the crosstalk calibration data to zero to ensure that spurious calibration data does not affect ranging.

6.3 Offset calibration

Soldering the device onto the customer board or adding a cover glass can introduce an offset in the ranging distance. This part-to-part offset has to be measured during the offset calibration.

During the offset calibration the dmax is also calibrated automatically. The calibration conditions suggested for the offset calibration are optimal for the dmax calibration as well.

6.3.1 Offset calibration functions

Three functions are available for offset calibration:

- *VL53L1_PerformOffsetCalibration(Dev,CalDistanceMilliMeter, CalReflectancePerCent)*
  Applicable to all preset modes
- VL53L1_PerformOffsetSimpleCalibration(Dev,CalDistanceMilliMeter)
  Simplified offset calibration, more accurate but RANGING preset mode ONLY
- VL53L1_PerformOffsetPerVCSELCalibration(Dev,CalDistanceMilliMeter)
  Gives the most accurate result, if several distance modes are used, but RANGING preset mode ONLY

The arguments of the functions are respectively the target distance in millimeters and its reflectance in percent (only for the legacy one).

Note: Offset calibration has to be performed after crosstalk correction

6.3.2 Offset calibration procedure

The customer has to use a calibrated chart, placed at a given distance to allow offset calibration. Details of this recommended setup are given in the following table.

Table 6. Offset calibration setup

Chart	Distance	Ambient conditions
Gray target (17% reflectance at visible light wavelength)	140 mm	Dark (no IR contribution)

6.3.3 Offset calibration options

The user can select the offset calibration options using the function: *VL53L1_SetOffsetCalibrationMode()*.

This function has the following options:

- Standard calibration. This is the default setting.
- Pre-range only. This should be used if standard offset cannot be used (due to a target that is too reflective or a short-distance target).
- Multizone calibration. This allows the ranging accuracy in multizones scanning mode to be increased.

Note that the ROIs must be set before calling the calibration function. A calibration is performed for a given ROI configuration. The same configuration has to be used during ranging.

6.3.4 Offset correction options

In multizone scanning mode, the offset correction can be improved by using a per zone calibration.

If the calibration is made using this option, the user can apply a per zone calibration by calling:

VL53L1_SetOffsetCorrectionMode().

The two possible options are:

- VL53L1_OFFSETCORRECTIONMODE_STANDARD (default)
- VL53L1_OFFSETCORRECTIONMODE_PERZONE

6.3.5 Getting offset calibration results

The function *VL53L1_GetCalibrationData()* allows the reception of all calibration data. The returned structure *VL53L1_CalibrationData_t* contains another structure called *VL53L1_customer_nvm_managed_t* which contains the three offset calibration results:

- algo__part_to_part_range_offset_mm
- mm_config__inner_offset_mm
- mm_config__outer_offset_mm

6.3.6 Setting offset calibration data

Customer can load the offset calibration data after *VL53L1_DataInit()* and *VL53L1_StaticInit()* functions are called, by using *VL53L1_SetCalibrationData()*.

It is better is to

1. Call *VL53L1_GetCalibrationData()*
2. Modify the three parameters described in Section 6.3.4 Offset correction options and the parameters described in Getting dmax calibration results
3. Call *VL53L1_SetCalibrationData()*.

7 Customer repair shop calibrations

If the calibration values are lost, due to component change in a repair shop, customers can apply a dedicated procedure, where no specific setup (targets) are needed.

The calibration is composed of three steps: RefSpad, Xtalk and Offset calibrations.

RefSpad and Xtalk are the same as described in Section 6.1 RefSPAD calibration and Section 6.2 Crosstalk calibration

Note: The customer only needs to check that no target is present below 80 cm during these calibrations.

A dedicated function is available to perform offset calibration: *VL53L1_PerformOffsetZeroDistanceCalibration*.

A target has to be set in front of the device, touching the cover glass. The target can be a simple sheet of paper. There are no particular requirements for the paper reflectance.

The above function has to be called and the results can be retrieved similarly to the process described in the previous chapters.

7.1 Live crosstalk calibration

Live crosstalk calibration is a way to get rid of part to part crosstalk calibration.

This option is only valid for Ranging and Scanning modes.

The principle is as follows:

- User performs a characterization of the crosstalk in the final application, using a typical cover glass. The crosstalk data structure is recorded.
- During part to part calibration, the user must:
  - perform RefSPadManagement calibration
  - load crosstalk characterisation data
  - perform offset calibration
- During standard operation, the user must:
  - load crosstalk characterisation data
  - enable live crosstalk detection
  - start ranging

Live crosstalk detection has the same limitations as described in Section 5.8 Live crosstalk correction.

8 Bare driver errors and warnings

Driver error is reported when any driver function is called. Possible values for driver errors are described in the following table.

Please note that warnings are given to inform the user that some parameters are not optimized. The warnings do not block the host.

Table 7. Bare driver errors and warnings

Error value	API error string	Occurrence
0	VL53L1_ERROR_NONE	No error
-1	VL53L1_ERROR_CALIBRATION_WARNING	Invalid calibration data
-2	VL53L1_ERROR_MIN_CLIPPED	Should not occur, unless using dedicated tuning parameters
-3	VL53L1_ERROR_UNDEFINED	Should not occur
-4	VL53L1_ERROR_INVALID_PARAMS	Invalid parameter is set in a function
-5	VL53L1_ERROR_NOT_SUPPORTED	Requested parameter is not supported by the programmed configuration
-6	VL53L1_ERROR_RANGE_ERROR	Interrupt status is incorrect
-7	VL53L1_ERROR_TIME_OUT	Ranging is aborted due to timeout
-8	VL53L1_ERROR_MODE_NOT_SUPPORTED	Requested mode is not supported
-9	VL53L1_ERROR_BUFFER_TOO_SMALL	Internal error, should not be raised
-10	VL53L1_ERROR_CALIBRATION_WARNING	Supplied buffer is larger than I2C supports
-11	VL53L1_ERROR_GPIO_NOT_EXISTING	Should not be raised
-12	VL53L1_ERROR_GPIO_FUNCTIONALITY_NOT_SUPPORTED	
-13	VL53L1_ERROR_CONTROL_INTERFACE	Error reported from IO function
-14	VL53L1_ERROR_INVALID_COMMAND	Command is invalid in current mode
-15	VL53L1_ERROR_DIVISION_BY_ZERO	Should not be raised
-16	VL53L1_ERROR_REF_SPAD_INIT	An error occurred during reference SPAD calibration
-17	VL53L1_ERROR_GPH_SYNC_CHECK_FAIL	Driver out of sync with device. A stop/start or a reboot may be needed.
-18	VL53L1_ERROR_STREAM_COUNT_CHECK_FAIL	
-19	VL53L1_ERROR_GPH_ID_CHECK_FAIL	
-20	VL53L1_ERROR_ZONE_STREAM_COUNT_CHECK_FAIL	
-21	VL53L1_ERROR_ZONE_GPH_ID_CHECK_FAIL	
-22	VL53L1_ERROR_XTALK_EXTRACTION_FAIL	Thrown when run_xtalk_extraction function has 0 successful samples when using the full array to sample the crosstalk. In this case, there is not enough information to generate new crosstalk parameter information. The function exits and leaves the current crosstalk parameters unaltered.

Error value	API error string	Occurrence
-23	VL53L1_WARNING_OFFSET_CAL_INSUFFICIENT_MM1_SPADS	Thrown when xtalk_extraction function has found that the average sigma estimate of the full array crosstalk sample is greater than the maximum limit allowed. In this case the crosstalk sample is too noisy for measurement. The function exits and leaves the current crosstalk parameters unaltered.
-24	VL53L1_ERROR_OFFSET_CAL_NO_SAMPLE_FAIL	Thrown if one of the stages has no valid offset calibration samples. Fatal error calibration not valid.
-25	VL53L1_ERROR_OFFSET_CAL_NO_SPADS_ENABLED_FAIL	Thrown if one of the stages has zero effective SPADs. Traps the case when MM1 SPADs is zero. Fatal error calibration not valid.
-26	VL53L1_ERROR_ZONE_CAL_NO_SAMPLE_FAIL	Thrown if some of the zones have no valid samples. A fatal error calibration not valid
-27	VL53L1_ERROR_TUNING_PARM_KEY_MISMATCH	Tuning parameter key does not match
-28	VL53L1_WARNING_REF_SPAD_CHAR_NOT_ENOUGH_SPADS	Thrown if less than five good SPADs are available. Ensure setup is in line with ST recommendations.
-29	VL53L1_WARNING_REF_SPAD_CHAR_RATE_TOO_HIGH	Thrown if the final reference rate is greater than the upper reference rate limit. Default is 40 Mcps. Implies a minimum Q3 (x10) SPAD (5) selected. Ensure setup is in line with ST recommendations.
-30	VL53L1_WARNING_REF_SPAD_CHAR_RATE_TOO_LOW	Thrown if the final reference rate is less than the lower reference rate limit. Default is 10 Mcps. Implies maximum Q1 (x1) SPADs selected ensure setup is in line with ST recommendations.
-31	VL53L1_WARNING_OFFSET_CAL_MISSING_SAMPLES	Thrown if there is less than the requested number of valid samples. Ensure setup is in line with ST recommendations.
-32	VL53L1_WARNING_OFFSET_CAL_SIGMA_TOO_HIGH	Thrown if the offset calibration range sigma estimate is greater than 8.0 mm. This is the recommended minimum value to yield a stable offset measurement
-33	VL53L1_WARNING_OFFSET_CAL_RATE_TOO_HIGH	Thrown when peak rate is greater than that 50.0Mcps. This is the recommended maximum rate to avoid pile-up influencing the offset measurement.
-34	VL53L1_WARNING_OFFSET_CAL_SPAD_COUNT_TOO_LOW	Thrown when one of stage’s range has less that 5.0 effective SPADs. This is the recommended minimum value to yield a stable offset.
-35	VL53L1_WARNING_ZONE_CAL_MISSING_SAMPLES	Thrown if one of more of the zones have less than the requested number of valid samples.
-36	VL53L1_WARNING_ZONE_CAL_SIGMA_TOO_HIGH	Thrown if one or more zones have a sigma estimate value greater than 8.0 mm. This is the recommended minimum value to yield a stable offset measurement.
-37	VL53L1_WARNING_ZONE_CAL_RATE_TOO_HIGH	Thrown if one of more zones have a peak rate higher than 50.0Mcps. This is the recommended maximum rate to avoid pile-up influencing the offset measurement.
-38	VL53L1_WARNING_XTALK_MISSING_SAMPLES	Thrown to notify that some of the crosstalk samples did not yield valid ranging pulse data while attempting to measure the crosstalk signal. This can signify any of the zones are missing samples. This warning is for notification only, the crosstalk pulse and shape have still been generated.

Error value	API error string	Occurrence
-39	VL53L1_WARNING_XTALK_NO_SAMPLES_FOR_GRADIENT	Thrown to notify that some of the crosstalk samples used for gradient generation did not yield valid ranging pulse data while attempting to measure the crosstalk signal. This can signify that any one of the zones 0-3 yielded no successful samples. This warning is for notification only, the crosstalk pulse and shape have still been generated.
-40	VL53L1_WARNING_XTALK_SIGMA_LIMIT_FOR_GRADIENT	Thrown to notify that some of the crosstalk samples used for gradient generation did not pass the sigma limit check while attempting to measure the crosstalk signal. This can signify that any one of the zones 0-3 yielded an average sigma_mm value greater than the limit. This warning is for notification only, the crosstalk pulse and shape have still been generated.
-41	VL53L1_ERROR_NOT_IMPLEMENTED	Function called is not implemented.

9 I2C utilization

This section lists the I2C bytes transactions for calls to the VL53L1 bare driver:

One time operations:

- initialization: 535 bytes
- offset calibration: 14512 bytes
- crosstalk calibration: 8423 bytes
- refSPAD calibration: 185 bytes

Frequent operations:

- each ranging mode operation: 155 bytes
- each autonomous ranging mode operation: 206 bytes

Table 8. I2C transactions size

Class of functions	Function call	Total I2C bytes	Comments
Initialization	VL53L1_WaitDeviceBooted()	Writes 2 bytes and Reads 1 byte	Called one time during module load time
	VL53L1_DataInit()	Writes 363 and Reads 169	
System call	VL53L1_SetXTalkCompensationEnable()	Writes 25	
	VL53L1_StartMeasurement()	Writes 137	
Interrupt (Autonomous mode)	VL53L1_GetMultiRangingData()	Writes 2 and Reads 83	Called for every range completion
Interrupt (Autonomous ranging mode)	VL53L1_GetRangingMeasurementData()	Writes 2 and Reads 134	Called for every range completion
Interrupt	VL53L1_GetMeasurementDataReady()	Writes 2 and Reads 1	Checks range ready status
Interrupt/Polling (Ranging mode)	VL53L1_ClearInterruptAndStartMeasurement ()	Writes 70	Start next ranging
Polling loop	VL53L1_WaitMeasurementDataReady()	Writes 2 and Reads 1	Checks range ready status
System call	VL53L1_StopMeasurement()	Writes 7	
System call	VL53L1_PerformRefSpadManagement()	Writes 134 and Reads 51	
System call	VL53L1_PerformXtalkCalibration()	Writes 3875 and Reads 4548 bytes	Called one time during calibration
System call	VL53L1_PerformOffsetCalibration()	Writes 9009 and Reads 5503	Called one time during calibration

10 Memory usage

The bare driver is compiled as a static library, on STM32F401RE, with gcc version 4.9.3, and no debug. It is optimized for size (-Os) and takes 62 kB in flash.

The device structure contains most of the information required by the driver to control the device. The size of the device structure is 8 kB - for the default number of zones (16) and results per zone (4).

Revision history

Table 9. **Document revision history**

Date	Version	Changes
24-Jan-2018	4	Initial corporate release
08-Jan-2019	5	Maintenance 3 update. 2.2: re-phrased 3.2.1 re-phrased 4.3 typo correction 4.6: “target not detected” situation described 5.13: dmax user guide added. modification about dmax in paragraphs 4.3 and 6.3 Proximity and autonomous low power modes deleted from the whole document Figure 4 and Figure 6 modified 5.4: default timing budget 33ms instead of 80ms Table 4: ranging gain modified to 0.971 5.8: live crosstalk correction limit 1kcps 6.2: FullROI crosstalk calibration added 6.3.2: gray17 chart suggested instead of black5 6.4: generic shape rewording
16-Oct-2020	6	Removed references to Lite ranging mode as obsolete Updated Introduction and added Section References Added a Note regarding tuning timing accuracy in Section 2.3 Timing considerations Updated Section 3.2.2 DataInit and Section 3.2.4 Preset mode Added info regarding ExtendedRange in Section 4.5 Single ranging measurement data structure Updated the Autonomous values in Table 2. Timing budget values (in ms) Updated table Table 3. Distance modes Added a note to Section 5.11 Tuning parameters Updated Section 6.3.1 Offset calibration functions and Section 6.3.3 Offset calibration options Removed section regarding dmax calibration results Added Section 7 Customer repair shop calibrations

Contents