UM2356
User manual

VL53L1X API user manual

Introduction

The VL53L1X is a long distance ranging Time-of-Flight sensor.

The purpose of this user manual is to describe the set of functions to call to get ranging data using the VL53L1X driver. Please refer to the VL53L1X datasheet.

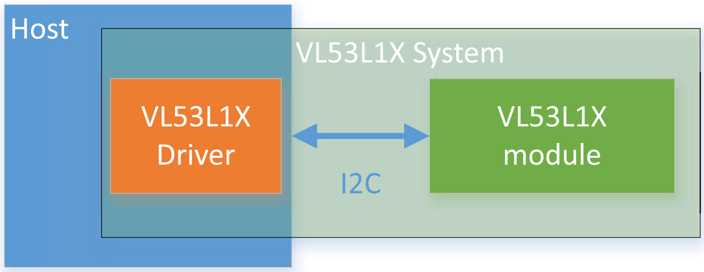
Figure 1. VL53L1X ranging sensor module

Contents

1 VL53L1X system overview

The VL53L1X system is composed of the VL53L1X module and a driver running on the host.

Figure 2. VL53L1X system



ST delivers a software driver, referred to as the “driver” in this document.

This document describes the driver functions accessible to the host, to control the device and get the ranging data.

The driver is an implementation of a set of functions for using the VL53L1X device. It makes minimal assumptions on the OS integration and services. As such, sequencing of actions, executing/threading model, platform adaptation, and device structure allocation are not part of the driver implementation but are left open to the software integrator.

The sequencing of the function calls must follow a set of rules, defined in this document.

2 Ranging API function descriptions

This section give a functional description of the ranging and describes the API call flow that should be followed to perform a ranging measurement using the VL53L1X.

2.1 Autonomous ranging description

The sensor performs the ranging continuously and autonomously with a programmable inter-measurement period.

Ranging is done without involvement from the host which allows the host to be in a low-power state. The host is only woken up upon measurement interrupts when a ranging is available.

It is possible to set a threshold of distance and/or signal detection criteria, then an interrupt is raised when the criteria is met.

2.2 Timing considerations

The timing budget is defined as the programmed time needed by the sensor to perform and report ranging measurement data. During this time, the VCSEL is pulsed. An interrupt is raised or the date ready register is updated at the end of the timing budget.

The inter-measurement period is defined as the programmed time between two consecutive measurements.

Figure 3 shows the timing budget and inter-measurement period.

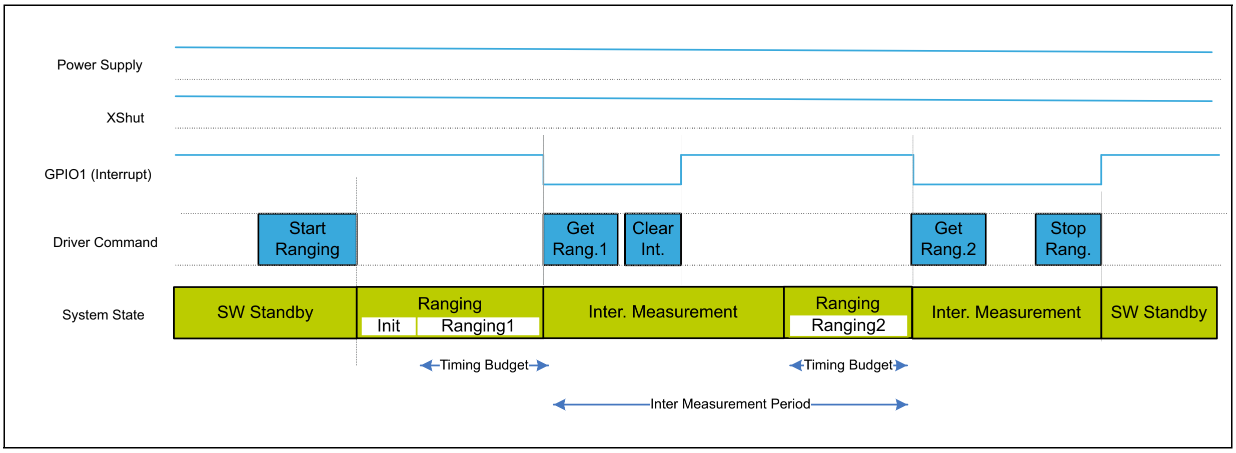
The host can change the default timing budget and inter-measurement period by using a dedicated driver function described in *Section 2.5.2: Timing budget and inter-measurement period*.

The minimum inter-measurement period must be longer than the timing budget + 4 ms.

If this condition is not respected, the VL53L1_StartMeasurement function will returns an error code (VL53L1_ERROR_INVALID_PARAMS).

The host can decide to change the timing budget to improve ranging accuracy or maximum distance limits.

Figure 3. VL53L1X autonomous ranging sequence and timings



2.3 API function call flows

The VL53L1X driver is used in two use cases:

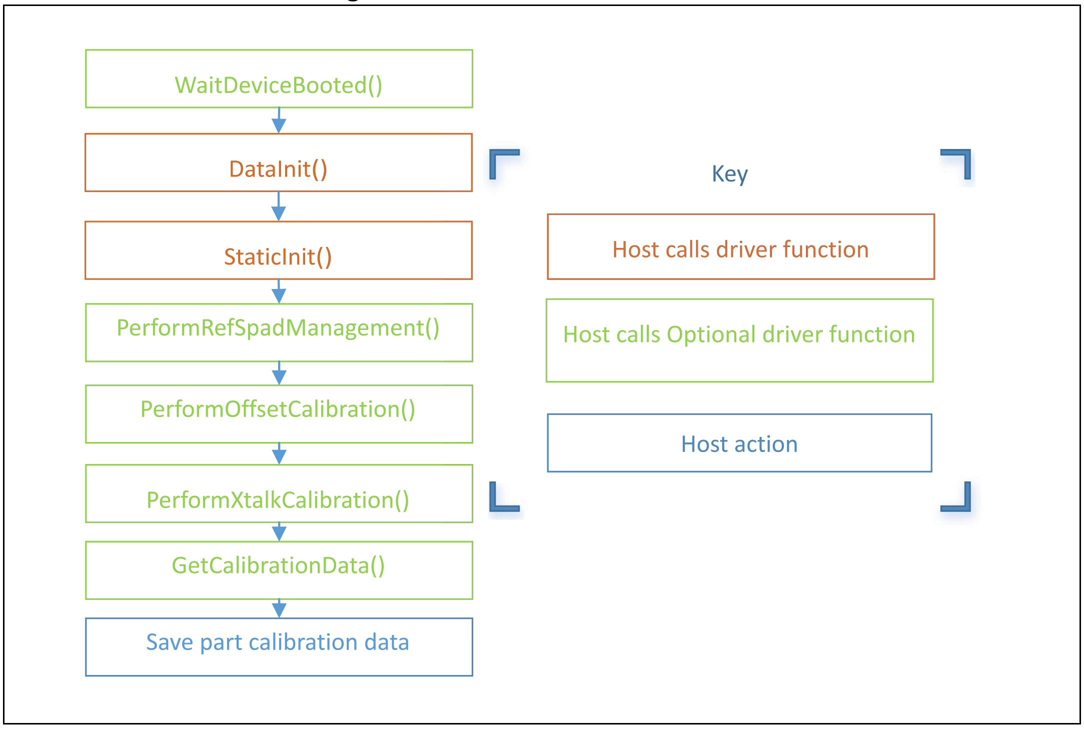
- Calibration flow for device calibration
- Ranging flow used at user applications level

2.3.1 Calibration flow

Calibration flow is described in *Figure 4*.

All API functions for calibration are described in *Section 3: Calibration functions*.

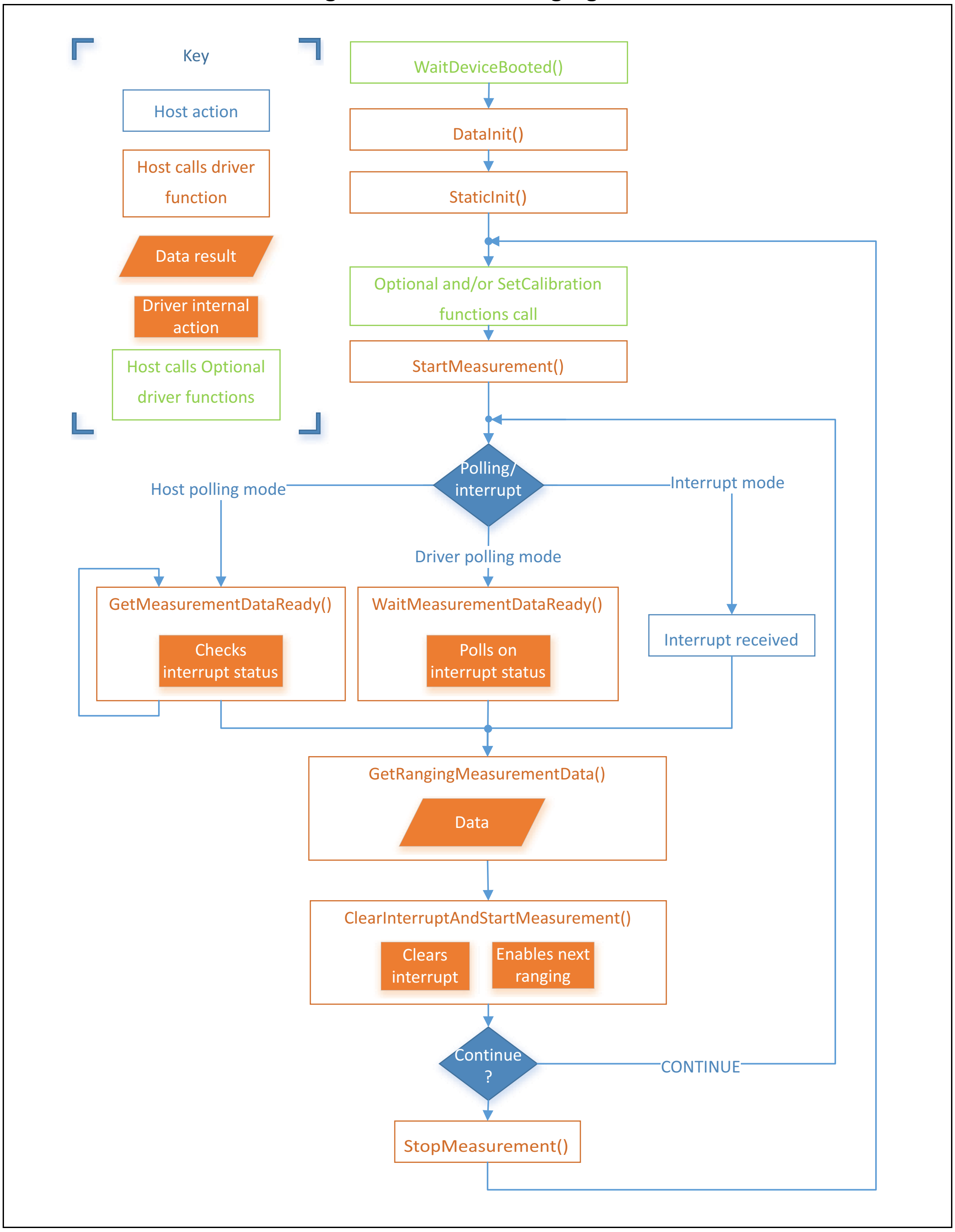
Figure 4. VL53L1X calibration flow



2.3.2 Ranging flow

Ranging flow is described in *Figure 5*.

Figure 5. VL53L1X ranging flow



2.4 Mandatory ranging functions

The following sections shows the API functions required to perform system initialization, before starting a measurement.

2.4.1 Data init

The `VL53L1_DataInit()` function is called once. It performs device initialization.

It is called **once and only once** after the device is brought out of reset.

2.4.2 Static Init

The `VL53L1_StaticInit()` function allows loading of the device settings that are specific for a given use case.

2.4.3 Start a measurement

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

2.4.4 Waiting for a result: polling or interrupt

There are three ways to know that ranging data are available:

1. **The host can call a polling function to wait until ranging data are available**

The function `VL53L1_WaitMeasurementDataReady()` polls on the device interrupt status until ranging data are ready.

This function blocks all other operations on the host as long as the function is not completed, because an internal polling is performed.

2. **The host can poll on a function to ask if the ranging data are available**

The host can poll on the function `VL53L1_GetMeasurementDataReady()` to know if new ranging data are ready.

This function does not block other operations. It is the preferred and recommended method if the sensor is used in polling mode.

3. **The host can wait for a physical interrupt**

An alternative and preferred way to get the ranging status is to use the physical interrupt output: by default, GPIO1 is pulled down when new ranging data are ready.

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

2.4.5 Get measurement

The `Vl53L1_GetRangingMeasurementData()` can be used to get 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 2.6: RangingMeasurementData structure*.

2.4.6 Clear source of interrupt

The interrupt must be cleared by calling the driver function: `VL53L1_ClearInterruptAndStartMeasurement()` after reading the ranging data

To get consistent results, it is mandatory to call this function after getting the ranging measurement.

If this function is not called, the next ranging will start and the results will be updated. But, the data ready status flag will not be updated, and the physical interrupt pin will not be cleared.

2.4.7 Stop a measurement

The host can decide to stop the measurement by calling the `VL53L1_StopMeasurement()` function.

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

2.5 Optional driver functions

2.5.1 Wait for boot

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

This function is optional. It is a blocking function because there is an internal polling. This function should not be blocking for more than 4 ms, assuming 400 kHz I2C and 2 ms latency per transaction.

2.5.2 Timing budget and inter-measurement period

Timing budget is the time required by the sensor to perform one range measurement.

The `VL53L1_SetMeasurementTimingBudgetMicroSeconds()` is the function to be used.

The minimum and maximum timing budgets are [20 ms, 1000 ms]

Warning: The maximum timing budget must not be exceeded to ensure the laser output remains within VL53L1 Class 1 limits in case of a single fault.

Example:

`Status = VL53L1_SetMeasurementTimingBudgetMicroSeconds(&VL53L1Dev, 66000 );` *sets the timing budget to 66 ms.*

The function `VL53L1_GetMeasurementTimingBudgetMicroSeconds()` gets the programmed timing budget.

The inter-measurement period is the delay between two ranging operations.

An inter-measurement period can be programmed. When a ranging completes, the device waits for the end of the programmed inter-measurement period before resuming the next ranging. In the inter-measurement period, the sensor is in a low-power state.

The `VL53L1_SetInterMeasurementPeriodMilliSeconds()` is the function to be used.

Example:

`Status = VL53L1_SetInterMeasurementPeriodMilliSeconds(&VL53L1Dev, 1000 );` sets the inter-measurement period to 1 s.

The function `VL53L1_GetInterMeasurementPeriodMilliSeconds()` gets the programmed inter-measurement period.

The minimum inter-measurement period must be longer than the timing budget + 4 ms. If the condition is not respected the VL53L1_StartMeasurment function returns an error code (VL53L1_ERROR_INVALID_PARAMS).

Note: The timing budget and inter-measurement period should not be called when the sensor is ranging. The user has to stop the ranging, change these parameters, and restart ranging.

2.5.3 Distance mode

Distance mode is a parameter provided to optimize the internal settings and tunings to get the best ranging performances depending on the ranging distance required by the application and the ambient light conditions.

The benefit of changing the distance mode is detailed in *Table 1: Distance modes*

Table 1. Distance modes

Possible distance modes	Maximum distance	Benefit / comments
Short	Up to 1.3 m	Better ambient immunity
Medium	Up to 3 m	
Long (default)	Up to 4 m	Maximum distance

The function to use is `VL53L1_SetDistanceMode()`.

The user can call `VL53L1_GetDistanceMode()` to get the programmed distance mode.

2.5.4 Limit check settings

The driver uses two parameters to qualify the ranging measurement: signal and sigma.

If signal or sigma are outside the limits, the ranging is flagged as invalid (note that RangeStatus is different than zero).

Applicable limits are:

- Sigma: VL53L1X_CHECKENABLE_SIGMA_FINAL_RANGE

Sigma is expressed in mm and is the estimation of the standard deviation of the measurement.

- Signal: *VL53L1X_CHECKENABLE_SIGNAL_RATE_FINAL_RANGE*

The signal rate measurement, expressed in MCPS, represents the amplitude of the signal reflected from the target and detected by the device.

Table 2 gives the default limit states and values.

Table 2. Default limit states and values

Limit ID	Default limit state	Default limit value	Associated *RangeStatus*
Sigma	Enabled	15 mm	1
Signal	Enabled	1 Mcps	2

If the user disables the limit checks, the ranging values will no longer be filtered and an incorrect measurement could be returned by the sensor. In this case, RangeStatus 1 and 2 will never be reported.

Changing limit default settings should be done with care, as the side effects can be important.

The limit change effects on the standard deviation and maximum ranging distances are shown in *Table 3*.

Table 3. Signal and sigma limit change effects

Limit ID	Action	Effect on standard deviation	Effect on maximum ranging distance
Sigma	Increase limit	-	+
	Decrease limit	+	-
Signal	Increase limit	+	-
	Decrease limit	-	+

`VL53L1X_SetLimitCheckEnable()` and `VL53L1X_GetLimitCheckEnable()` are used to enable/disable a limit.

The limit value is set using `VL53L1X_SetLimitCheckValue()` and `VL53L1X_GetLimitCheckValue()`.

One example of the signal limit setting is to enable a signal check and set the limit to 0.4 MCps:

```
Status = VL53L1X_SetLimitCheckEnable(&VL53L1Dev,
VL53L1X_CHECKENABLE_SIGNAL_RATE_FINAL_RANGE, 1);
Status = VL53L1X_SetLimitCheckValue(&VL53L1Dev,
VL53L1X_CHECKENABLE_SIGNAL_RATE_FINAL_RANGE, 0.40*65536);
```

2.5.5 Thresholds

The device can be configured to operate in distance and/or signal threshold detection mode. The ranging data is reported to the host when the pre-configured criteria are matched.

Detection mode

Detection mode allows selection of the filtering conditions:

- 0: no filter (default value, standard ranging mode)
- 1: filter on distance criteria only

Distance detection mode, based on thresholds:

Distance detection mode (through the CrossMode parameter) defines the distance criteria:

- 0: below a certain distance: "threshold low"
  - If object distance > distance low or no object found: no report
  - If object distance < distance low and object found: report
- 1: beyond a certain distance: "threshold high"
  - If object distance < distance high or no object found: no report
  - If object distance > distance high and object found: report
- 2: out of a distance range (min/max), "out of window"
  - Distance low < detected distance < distance high: no report
  - Distance low > detected distance > distance high: report
- 3: Within a distance range (min/max), "inside window"
  - Distance low > detected distance > distance high: no report
  - Distance low < detected distance < distance high: report

Distance low is the minimum configured distance in millimeters.

Distance high is the maximum configured distance in millimeters.

No target

This is an alternate detection mode. In the standard use case, if no target is detected, no ranging is reported. Using no target detection mode (setting IntrNoTarget to 1) allows an interrupt to be generated when no target is present.

API function

`VL53L1_SetThresholdConfig()` is the function to use.

The VL53L1_DetectionConfig_t structure contains all parameters to be set.

Example:

Detectionconfig.DetectionMode = 1

Detectionconfig.Distance.CrossMode = 3

Detectionconfig.IntrNoTarget = 0

Detectionconfig.Distance.High = 1000

Detectionconfig.Distance.Low = 100

Status = VL53L1_SetThresholdConfig(&VL53L1Dev, &detectionConfig);

This function is used to program the device to report ranging only when an object is detected within 10 cm and 1 m (as in this example).

The function *VL53L1_GetThresholdConfig()* allows the programmed report threshold configuration to be obatined.

2.5.6 Region of interest (ROI) setting

The receiving SPAD array of the sensor includes 16x16 SPADs which cover the full field of view (FoV). It is possible to program a smaller region of interest (ROI), with a smaller number of SPADs, to reduce the FoV.

To set a ROI different than the default 16x16 one, the user can call the *VL53L1_SetUserROI()* function.

The ROI is a square or rectangle 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

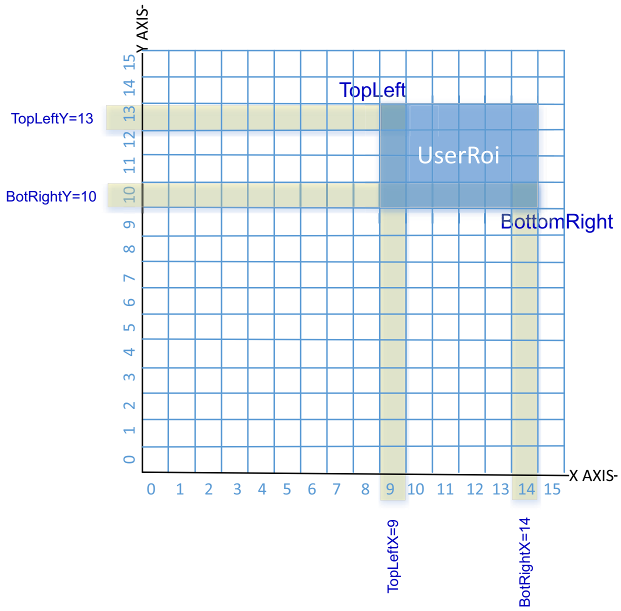
These coordinates are part of the VL53L1_UserRoi_t structure.

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 an ROI setting is given in *Figure 6*.

Figure 6. VL53L1X ROI setting example



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 x coordinate [0;15]

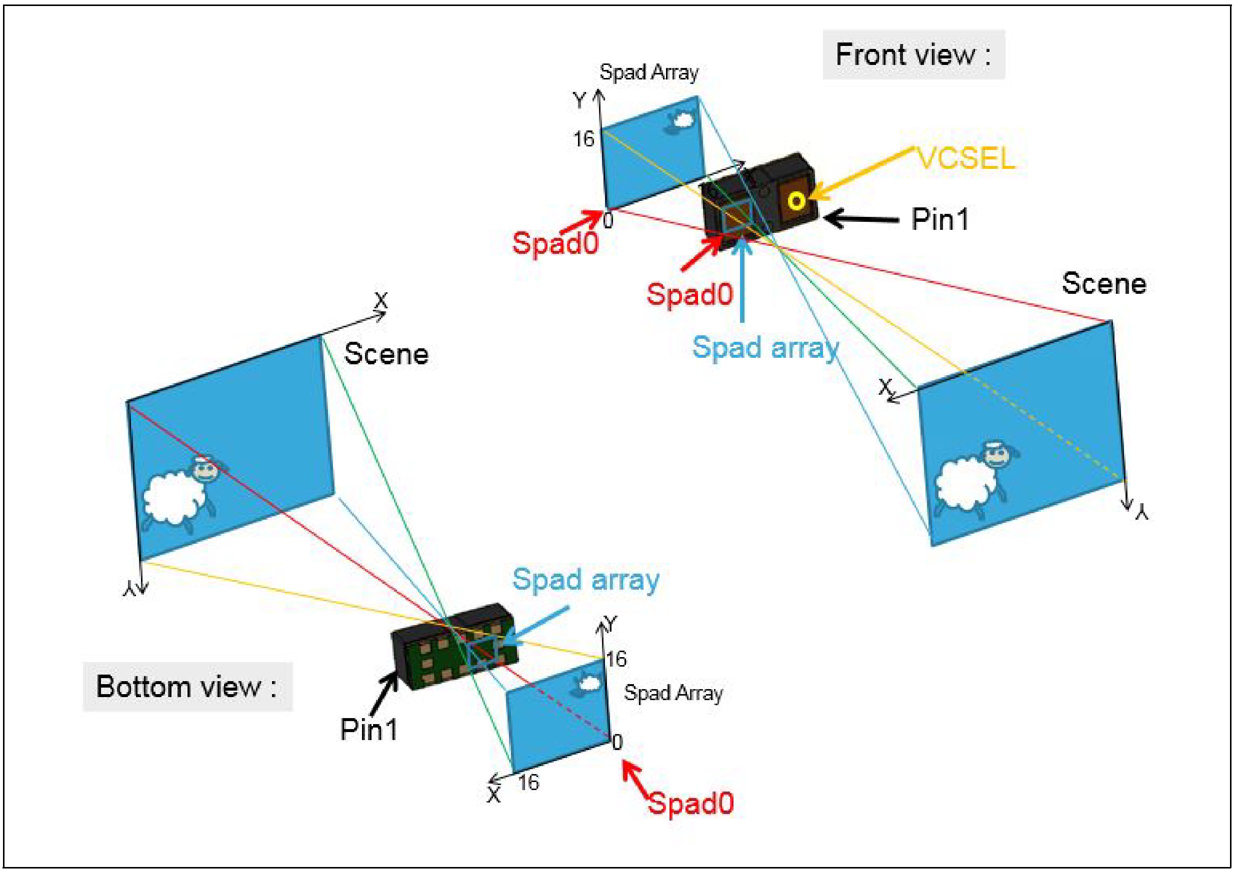
Example to set one ROI (based on *Figure 6*):

```
VL53L1_UserRoi_t roiConfig;
roiConfig.TopLeftX  = 9;
roiConfig.TopLeftY  = 13;
roiConfig.BotRightX = 14;
roiConfig.BotRightY = 10;
status = VL53L1_SetUserROI(&VL53L1Dev, &roiConfig);
```

2.5.7 Spad array coordinates versus scene

Figure 7 shows the coordinates of an object in the SPAD array compared to the location in the FoV.

Figure 7. VL53L1X coordinates vs scene



2.5.8 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 a substructure VL53L1_optical_centre_t which contains the two coordinates (expressed in the SPAD number):

- x_centre
- y_centre

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

2.5.9 VDDIO configuration

As described in the datasheet, the user can select two modes for a VDDIO value of 1V8 or 2V8 modes.

The selection of the mode is made directly in the code though a compilation key called `USE_I2C_2V8k`.

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

2.6 RangingMeasurementData structure

The **VL53L1_RangingMeasurementData_t** structure is composed of:

- ***TimeStamp***: not implemented, please ignore it.
- ***StreamCount***: this 8-bit integer counter increments at each range. The value starts at 0, increments to 255, and then increments from 128 to 255.
- ***RangingQualityLevel***: not implemented, please ignore it.
- ***SignalRateRtnMegaCps***: this value is the return signal rate in MegaCountPer Second (MCPS). It is a 16.16 fix 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 fix point value, which is effectively a measure of the infrared light. To obtain a real value it should be divided by 65536.
- ***EffectiveSpadRtnCount***: this 16 bit integer returns the effective SPAD count for the current ranging. To obtain a real value it should be divided by 256.
- ***SigmaMilliMeter***: this 16.16 fix 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 16 bit integer give the range distance in millimeters.
- ***RangeFractionalPart***: not implemented, please ignore it.
- ***RangeStatus***: this 8 bit integer gives the range status for the current measurement. A value of 0 means the ranging is valid (refer to *Table 4*).

Table 4. 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
2	VL53L1_RANGESTATUS_SIGNAL_FAIL	Raised if signal value is below the internal defined threshold
4	VL53L1_RANGESTATUS_OUTOFBOUNDS_FAIL	Raised when phase is out of bounds
5	VL53L1_RANGESTATUS_HARDWARE_FAIL	Raised in case of HW or VCSEL failure
7	VL53L1_RANGESTATUS_WRAP_TARGET_FAIL	Wrapped target, not matching phases
8	VL53L1_RANGESTATUS_PROCESSING_FAIL	Internal algorithm underflow or overflow
14	VL53L1_RANGESTATUS_RANGE_INVALID	The reported range is invalid

3 Calibration functions

To benefit from the full performance of the device, the VL53L1X driver includes calibration functions that should be run once at the customer production line.

Calibration procedures have to be run to compensate the part-to-part parameters and the presence of the cover glass that may affect the device performance.

Calibration data stored in the host have to be loaded into theVL53L1X at each startup using a dedicated driver function.

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

The order the calibration functions are called is important: RefSPAD should be called first, offset second, and crosstalk third.

The three calibration functions can be done sequentially one after another, or individually. When run individually, the previous step data have to be loaded before running the current calibration.

3.1 RefSPAD calibration

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

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

However, adding a cover glass on top of the module may affect this calibration. We recommend 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 the three possible types of SPAD:

1. Non attenuated SPAD
2. SPAD attenuated by a factor of 5
3. SPAD attenuated by a factor of 10

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

3.1.1 RefSPAD calibration function

A dedicated function is available for this operation:
`VL53L1_PerformRefSpadManagement(&VL53L1Dev)`

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

3.1.2 RefSPAD calibration procedure

The user has to ensure that there are no targets closer than 5 cm from the sensor during the calibration.

It is better to perform this calibration in low IR light conditions (indoor).

The time to perform this calibration is only few milliseconds.

The `VL53L1_PerformRefSpadManagement` function has to be called after the `VL53L1_DataInit()` and `VL53L1_StaticInit()` functions are called. Refer to *Figure 4: VL53L1X calibration flow*.

When the calibration function is called, the RefSPAD calibration is performed and the new RefSPAD parameters are applied at the end.

3.1.3 Getting RefSPAD calibration results

The function `VL53L1_GetCalibrationData()` allows to read all calibration data. The returned structure *VL53L1_CalibrationData_t* also contains a substructure called *VL53L1_customer_nvm_managed_t* which contains the 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 [1,2,3]
  – value 1: non attenuated SPAD
  – value 2: SPAD attenuated by a factor of 5
  – value 3: SPAD attenuated by a factor of 10
- 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

After factory calibration, these calibration data have to be stored in the host memory and loaded at each device start up to avoid redoing the calibration. Either the user stores the entire structure *VL53L1_CalibrationData_t* or stores the eight parameters (to save memory space).

3.1.4 Setting RefSPAD calibration data

At each start up, after a hard reset, the user can load the RefSPAD calibration data from the host memory. The `VL53L1_SetCalibrationData()` has to be called after the `VL53L1_DataInit()` and `VL53L1_StaticInit()` functions are called. Refer to *Figure 5: VL53L1X ranging flow*

If the user has optimized the calibration data storage during the calibration, it is recommended to get the entire calibration structure by calling `VL53L1_GetCalibrationData()`, modifying the eight parameters described in *Section 3.1.3: Getting RefSPAD calibration results*, and calling `VL53L1_SetCalibrationData()`.

3.2 Offset calibration

Soldering the device on 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 and compensated during the offset calibration.

After performing the RefSPAD calibration, the timing budget and inter measurement period have been set to 41 ms and 1 s respectively by the RefSPAD calibration function. To avoid a long offset and crosstalk calibration execution time, it is recommended to set the new timing budget to 30 ms and the inter measurement period to 40 ms before performing the offset and/or crosstalk calibration.

3.2.1 Offset calibration function

A dedicated function is available for this operation:

```
VL53L1_PerformOffsetSimpleCalibration(&VL53L1Dev,
CalDistanceMilliMeter)
```

The argument of the function is the offset calibration distance in millimeters.

Note: Offset calibration has to be performed before crosstalk calibration and after RefSPAD optimization is done (calibration done or RefSPAD parameters loaded).

3.2.2 Offset calibration procedure

The customer has to use a calibrated chart, placed at a given distance (`CalDistanceMilliMeter`) to perform the offset calibration.

Details of the recommended setup are given in *Table 5*.

Table 5. Offset calibration set up

Chart	Chart distance (`CalDistanceMilliMeter`)	Ambient conditions
Gray target (17 % reflectance at 940 nm)	Recommended value: 140 mm	Dark (no IR contribution)

When the calibration function is called, the offset calibration is performed and the offset correction is applied at the end.

3.2.3 Getting offset calibration results

The function `VL53L1_GetCalibrationData()` allows all calibration data to be obtained. The returned structure VL53L1_CalibrationData_t also contains a substructure called VL53L1_customer_nvm_managed_t which contains the main offset calibration result in mm_config__inner_offset_mm and in mm_config__outer_offset_mm. Both have the same value: mm_config__inner_offset_mm = mm_config__outer_offset_mm.

3.2.4 Setting offset calibration data

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

However, it is better to call `VL53L1_GetCalibrationData()`, modify the parameter described in *Section 3.2.3: Getting offset calibration results* (mm_config__inner_offset_mm and mm_config__outer_offset_mm) and call `VL53L1_SetCalibrationData()`.

3.3 Crosstalk calibration

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

Depending on the cover glass quality, the amount of the return signal can be significant and can affect the sensor performance. The VL53L1X has a built-in correction that allows for this crosstalk phenomenon to be compensated.

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.

3.3.1 Cross talk calibration function

A dedicated function is available for this operation:
`VL53L1_PerformSingleTargetXTalkCalibration(&VL53L1Dev, XtalkCalDistance);`

One argument of the function is the crosstalk calibration distance in millimeters.

Note: This function must be called in third place in the calibration flow, after the offset compensation is done (calibration done or offset parameters loaded).

3.3.2 Cross talk calibration procedure

The cross talk calibration should be conducted in a dark environment, with no IR contribution.

When the calibration function is called, the crosstalk calibration is performed and the crosstalk correction is applied at the end.

Table 6. Crosstalk calibration set up

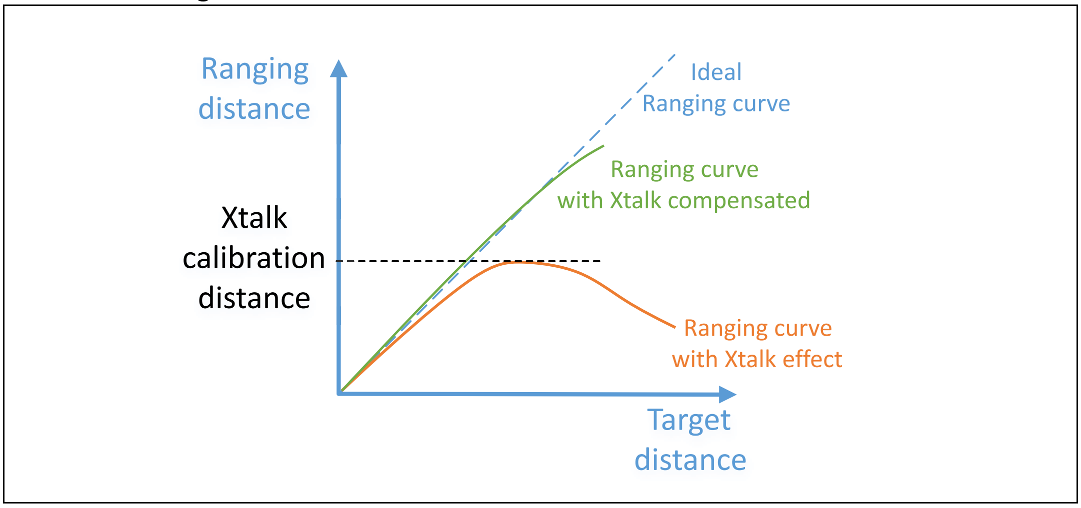
Chart	Chart distance (`XtalkCalDistance`)	Ambient conditions
Gray target (17 % reflectance at 940 nm)	As defined in *Section 3.3.3: Crosstalk calibration distance characterization*	Dark (no IR contribution)

3.3.3 Crosstalk calibration distance characterization

The crosstalk calibration distance needs to be characterized by the user, as it depends on the system environment, mainly the cover glass material and optical properties, and the air gap value (distance between the sensor and the cover glass).

Figure 8: VL53L1X crosstalk calibration distance definition shows the crosstalk effect on the ranging curve. From a given distance, the effect of the crosstalk is predominant, and the sensor starts to under-range.

Figure 8. VL53L1X crosstalk calibration distance definition



The crosstalk calibration distance corresponds to the maximum ranging distance achievable reported by the sensor when the cover glass is present (see *Figure 8*).

This maximum ranging distance is one argument of the crosstalk calibration driver function.

The ranging curve with crosstalk corrected is the ranging result when the crosstalk compensation is applied (when crosstalk calibration is completed or after crosstalk calibration data are loaded).

3.3.4 Getting crosstalk calibration results

The function `VL53L1_GetCalibrationData()` allows all calibration data to be obtained. The returned structure VL53L1_CalibrationData_t also contains a substructure called VL53L1_customer_nvm_managed_t which contains the crosstalk calibration result: algo__crosstalk_compensation_plane_offset_kcps. The crosstalk calibration value is coded in 7.9 format and thus required to be divided by 512 to get the actual crosstalk value in kilo count per second per SPAD (kcps/SPAD).

Example: algo__crosstalk_compensation_plane_offset_kcps = 1500, the actual crosstalk value is 1500/512 = 2.92 kcps/SPAD.

3.3.5 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 recommended to call *VL53L1_GetCalibrationData()*, modify the algo__crosstalk_compensation_plane_offset_kcps parameter in the VL53L1_customer_nvm_managed_t substructure, and then call *VL53L1_SetCalibrationData()* to apply the crosstalk compensation.

The actual crosstalk value to apply must be multiplied by 512 because of the 7.9 format.

Example: The crosstalk value to apply to correct the crosstalk effect is 3 kcps/SPAD, algo__crosstalk_compensation_plane_offset_kcps = 3*512.

3.3.6 Enable/Disable crosstalk compensation

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

VL53L1_SetXTalkCompensationEnable(&VL53L1Dev, 0); // disables the crosstalk compensation.

V53L1_SetXTalkCompensationEnable(&VL53L1Dev, 1); // enables the crosstalk compensation.

Note: This function does not perform any crosstalk calibration or data loading, it just enables the crosstalk compensation. It has to be called before starting ranging, with the optional function calls.

4 Driver errors and warnings

The driver error is reported when any driver function is called. Possible values for driver errors are described in *Table 7*.

Please note that warnings exist to inform the user that some parameters are not optimized. The warnings are not blocking points for the host.

Table 7. Bare driver errors and warnings descriptions

Error value	API error string	Occurrence
0	VL53L1_ERROR_NONE	No error
-1	VL53L1_ERROR_CALIBRATION_WARNING	Invalid calibration data
-4	VL53L1_ERROR_INVALID_PARAMS	Invalid parameter is set in a function
-5	VL53L1_ERROR_NOT_SUPPORTED	Requested parameter is not supported in 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
-10	VL53L1_ERROR_CALIBRATION_WARNING	Supplied buffer is larger than I2C supports
-14	VL53L1_ERROR_INVALID_COMMAND	Command is invalid in current mode
-16	VL53L1_ERROR_REF_SPAD_INIT	An error occurred during reference SPAD calibration
-22	VL53L1_ERROR_XTALK_EXTRACTION_FAIL	Thrown when crosstalk calibration function has no successful samples to compute the crosstalk. In this case there is not enough information to generate new crosstalk parameter information. The function will exit and leave the current crosstalk parameters unaltered.
-23	VL53L1_ERROR_XTALK_EXTRACTION_SIGMA_ LIMIT_FAIL	Thrown when crosstalk calibration function has found that the sigma estimate is above the maximal limit allowed. In this case the crosstalk sample is too noisy for measurement. The function will exit and leave the current crosstalk parameters unaltered.
-24	VL53L1_ERROR_OFFSET_CAL_NO_SAMPLE_ FAIL	Thrown when offset calibration function has found no valid ranging.
-28	VL53L1_WARNING_REF_SPAD_CHAR_NOT_ ENOUGH_SPADS	Thrown if there are less than five good SPADs available. Ensure calibration 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. Ensure calibration 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. Ensure calibration setup is in line with ST recommendations.

Table 7. Bare driver errors and warnings descriptions (continued)

Error value	API error string	Occurrence
-31	VL53L1_WARNING_OFFSET_CAL_MISSING_SAMPLES	Thrown if there is less than the requested number of valid samples. Ensure offset calibration setup is in line with ST recommendations.
-32	VL53L1_WARNING_OFFSET_CAL_SIGMA_TOO_HIGH	Thrown if the offset calibration range sigma estimate is too high. Ensure offset calibration setup is in line with ST recommendations.
-33	VL53L1_WARNING_OFFSET_CAL_RATE_TOO_HIGH	Thrown when signal rate is greater than a limit and sensor is saturating. Ensure offset calibration setup is in line with ST recommendations.
-34	VL53L1_WARNING_OFFSET_CAL_SPAD_COUNT_TOO_LOW	Thrown when not enough SPADS can be used. Ensure offset calibration setup is in line with ST recommendations.
-41	VL53L1_ERROR_NOT_IMPLEMENTED	Function called is not implemented

5 Acronyms and abbreviations

Table 8. Acronyms and abbreviations

Acronym/ abbreviation	Definition
I2C	Inter-integrated circuit (serial bus)
NVM	Non volatile memory
SPAD	Single photon avalanche diode
VCSEL	Vertical cavity surface emitting laser
FMT	Final Module Test
API	Application Programming Interface

6 Revision history

Table 9. Document revision history

Date	Revision	Changes
07-Mar-2018	1	Initial release
19-Jun-2018	2	Updated the following sections relating to timing budget, inter-measurement period, and calibration functions: *Section 2.2: Timing considerations*, *Section 2.5.2: Timing budget and inter-measurement period*, *Section 3.1.3: Getting RefSPAD calibration results*, *Section 3.2.3: Getting offset calibration results*, *Section 3.2.4: Setting offset calibration data*, *Section 3.3.4: Getting crosstalk calibration results*, and *Section 3.3.5: Setting crosstalk calibration data*.
27-Oct-2022	3	*Section 2.5.2: Timing budget and inter-measurement period*: added a warning about the maximum timing budget. *Section 3.3.4: Getting crosstalk calibration results* and *Section 3.3.5: Setting crosstalk calibration data*: corrected the crosstalk calibration value coded in 7.9 format.