VL53L3CX Time-of-Flight ranging module with multi object detection

Introduction

VL53L3CX is a Time-of-Flight (ToF) ranging sensor module.

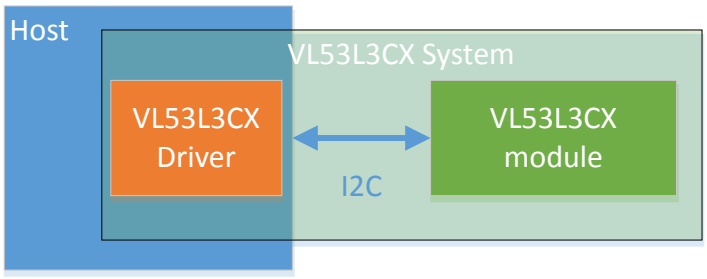
The purpose of this user manual is to describe the integration model and the set of functions to call to get ranging data using the VL53L3CX bare driver.

1 VL53L3CX system overview

VL53L3CX system is composed of the VL53L3CX module and a driver running on the host.

This document describes the driver functions accessible to the Host, to control the device and get the ranging data for integration with non-linux hosts.

Figure 1. VL53L3CX system



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 an implementation of a set of functions required to use the VL53L3CX device. It makes minimal assumptions on the OS integration and services. As such, sequencing of actions, execution/threading model, platform adaptation, and device structures allocation are not part of the bare driver implementation but 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 VL53L3CX ranging device.

2.1 Ranging sequence

The device is running with a handshake mechanism, based on a standard interrupt management scheme.

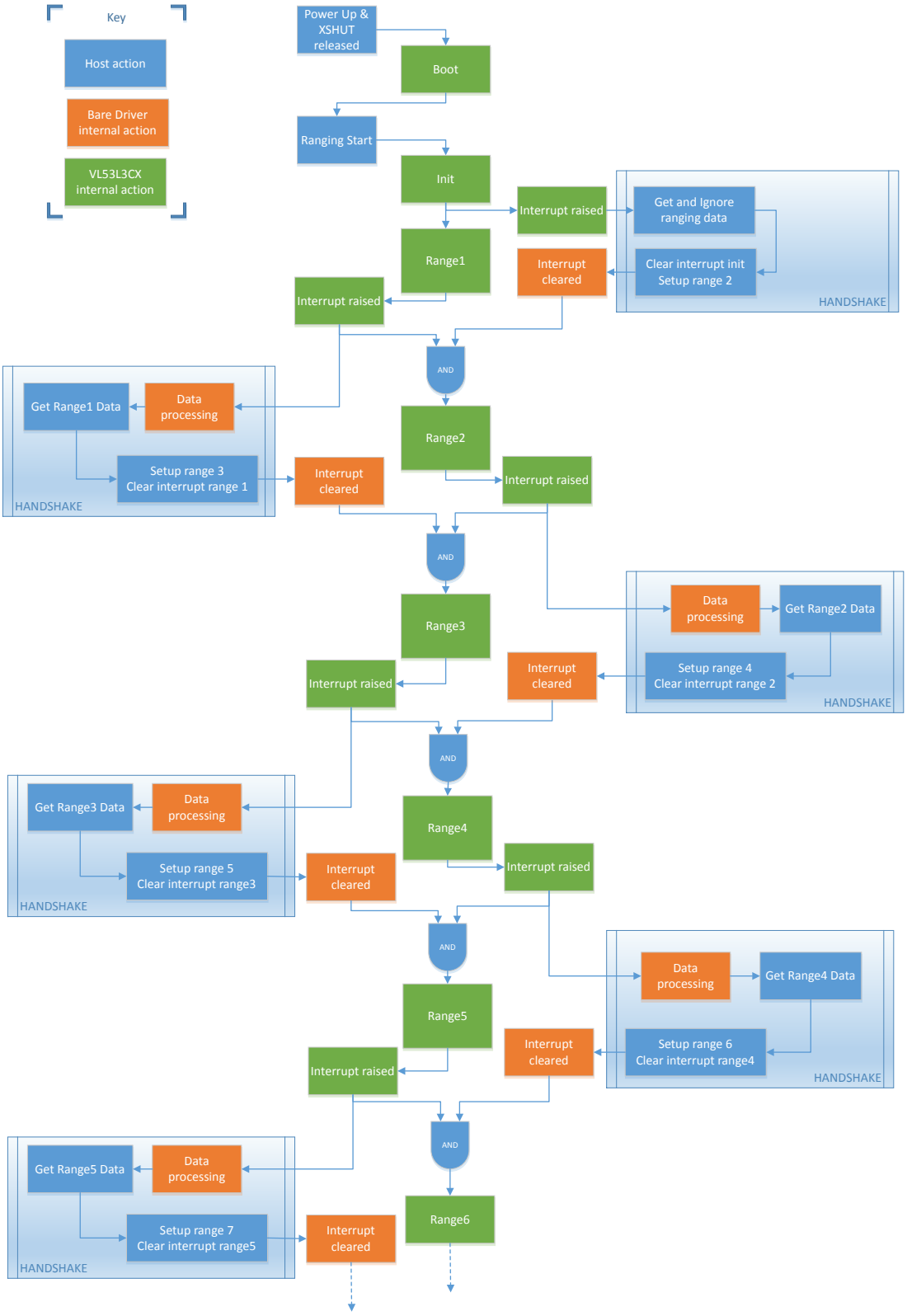
After each ranging, the host acquires the ranging data and enables the next ranging by clearing the interrupt. This process is referred to as the handshake mechanism.

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 faster data transfer, without losing 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 the figure below.

Figure 2. **VL53L3CX ranging sequence overview**



Handshake sequence allows the computation of internal parameters and apply them for next range.

The handshake must be performed by the user of the bare driver. The delay to enable a new ranging after a new measurement has been received is key to overall system measurement rate.

2.2 Timing considerations

Timings are presented in Figure 3. Ranging sequence and timing targets.

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

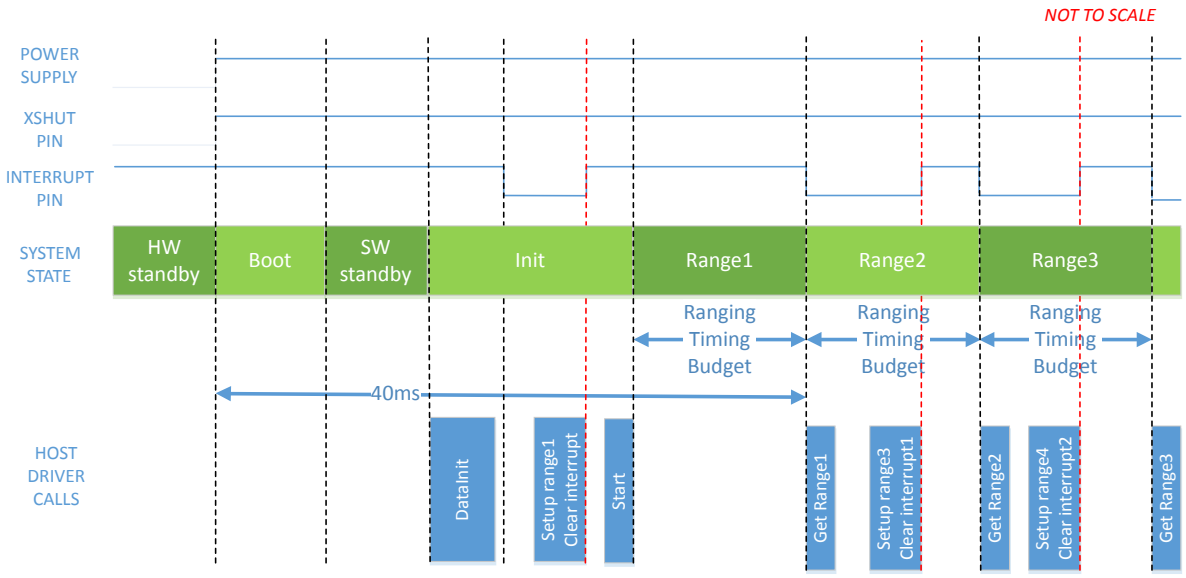
If a delay to clear the interrupt is introduced by the host, the next ranging will be stalled until the pending interrupt is cleared.

Note: Timings indicated in Figure 3. Ranging sequence and timing targets are typical timings.

The host can change the default timing budget by using a dedicated driver function described in Section 5.1 Timing budget. Host can decide to change timing budget either to synchronize on the application or to increase ranging accuracy.

In the following figure, 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 from the platform or the used timing budget. The first range, “Range1”, is not valid, being the wrap-around check not possible. This means that the first valid ranging value is “Range2”, available after 40 ms plus twice the timing budget duration.

Figure 3. Ranging sequence and timing targets



3 Bare driver basic functions description

This section describes the driver functions calls flow that should be followed to perform a ranging measurement using the VL53L3CX.

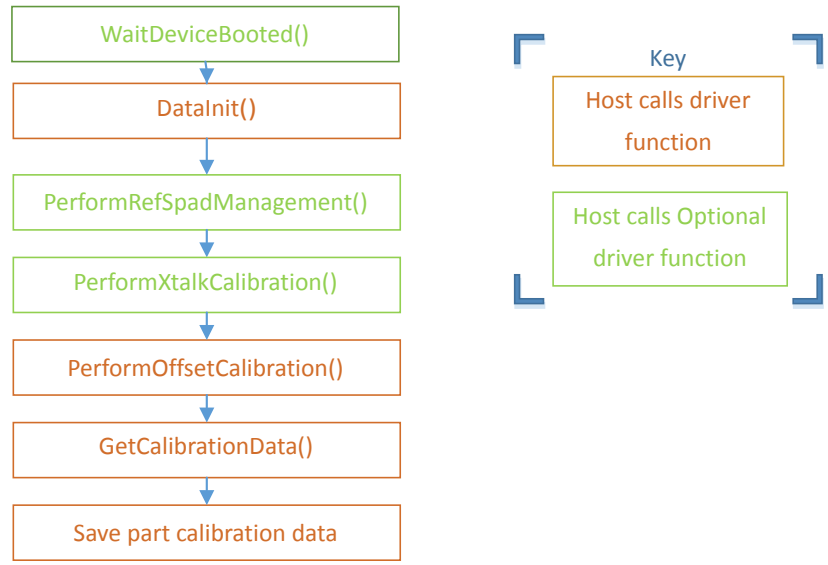
The VL53L3CX driver are used in two classes of applications:

- Factory applications used for device calibration, typically at end product manufacturing test (factory flow)
- Field applications, which gather all end-user applications using the VL53L3CX device (ranging flow)

3.1 Bare driver

Bare driver factory flow is illustrated in the following figure.

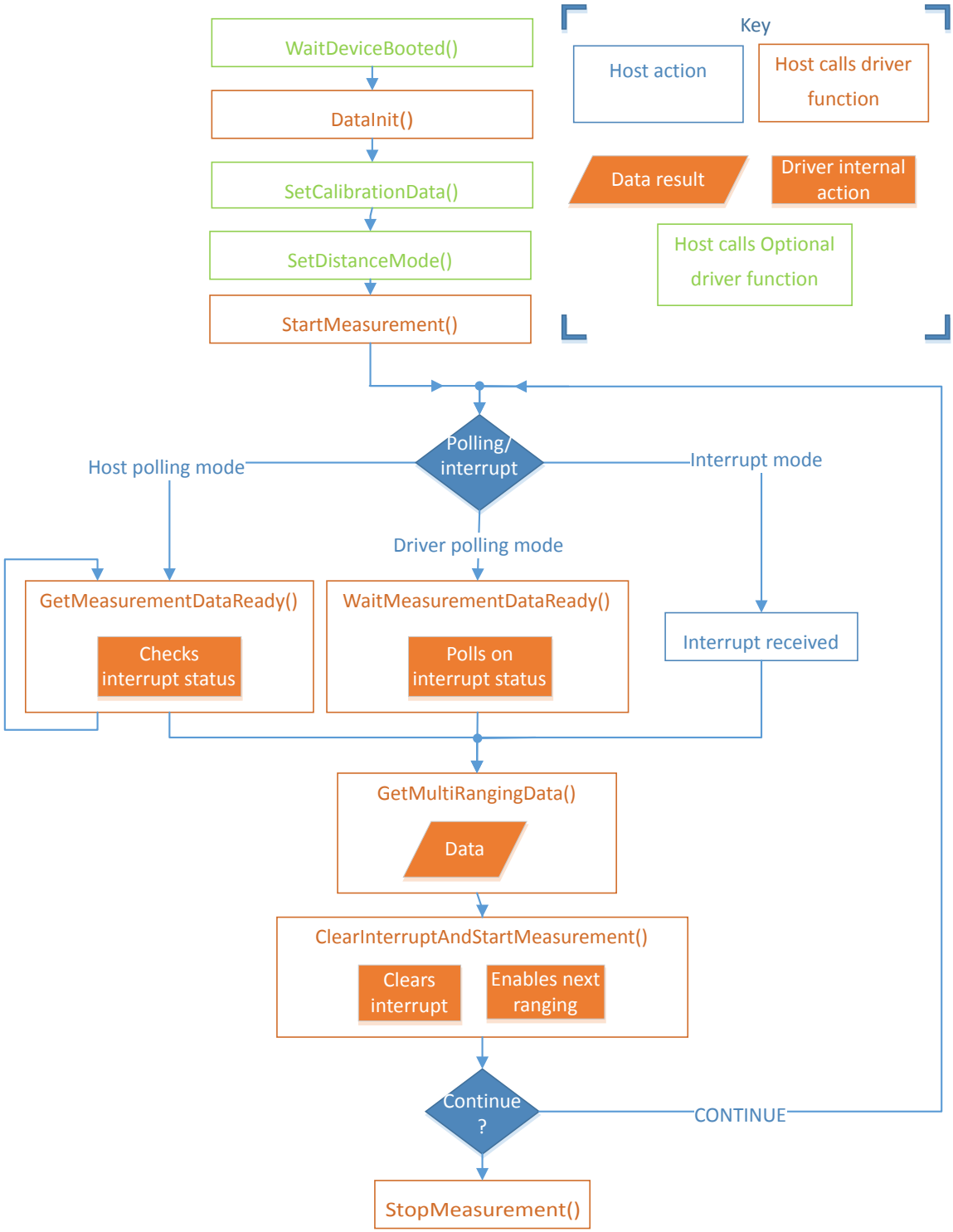
Figure 4. VL53L3CX API ranging flow (factory)



Note: The calibration flow changes the distance mode. It is mandatory to call the SetDistanceMode() function if you want to use the sensor just after a calibration.

Bare driver ranging flow is illustrated in the following figure.

Figure 5. **VL53L3CX API ranging flow (field)**



3.2 System initialization

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

3.2.1 Wait for boot

VL53LX_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 frequency*
- *2 ms latency per transaction*

3.2.2 Data init

The *VL53LX_DataInit()* function must be called each time the device exits from the “initial boot” state. It performs device initialization. After calling the *VL53LX_DataInit()* function the calibration data have to be loaded using the function *VL53LX_SetCalibrationData()*.

4 Ranging with VL53L3CX

On non-Linux hosts, the user of the bare driver sequences calls to the driver in a way that is appropriate to the application needs, the platform capabilities and the bare driver call sequence rules.

4.1 Start a measurement

VL53LX_StartMeasurement() function must be called to start a measurement.

4.2 Wait for a result: polling or interrupt

There are 3 ways to know that a measurement is available. The host can:

1. call a polling function
2. poll on a driver function
3. wait for a physical interrupt

4.2.1 Driver polling to get the result status

The function *VL53LX_WaitMeasurementDataReady()* is polling on an internal status until a measurement is ready.

Note: This function is blocking, as internal polling is performed.

4.2.2 Host polling to get the result status

Host can poll on the function *VL53LX_GetMeasurementDataReady()* to know when a new measurement is ready.

This function is not blocking.

4.2.3 Using physical interrupt

An alternative and preferred way to get the ranging status is to use the physical interrupt output. By default, 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.

Interrupt must be cleared by calling the driver function *VL53LX_ClearInterruptAndStartMeasurement()*.

4.3 Get measurement

Multiple objects can be detected per ranging, and measurement data is reported per object *VL53LX_GetMultiRangingData()* can be used to get ranging data when multiple objects are in the field of view.

When calling this function to get the device multiple ranging results, a structure called *VL53LX_MultiRangingData_t* is returned.

4.4 Stop a measurement

In continuous mode, the host can stop the measurement by calling *VL53LX_StopMeasurement()* function.

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

4.5 Ranging data structures

The structure named **VL53LX_MultiRangingData_t** contains the following data applicable to all targets detected:

- TimeStamp: not implemented.
- StreamCount: this 8-bit integer gives a counter incremented at each range. The value starts at 0, incrementing 1 by 1 up to 255. When it reaches 255, it starts again from 128 to 255.
- NumberOfObjectsFound: 8-bit integer value that gives the number of objects found.
- RangeData[VL53LX_MAX_RANGE_RESULTS]: a table of structure of type *VL53LX_TargetRangeData_t*. The maximum number of targets is given by VL53LX_MAX_RANGE_RESULTS, and is by default equal to 4.
- HasXtalkValueChanged: 8-bit integer value that indicates if the crosstalk value has been changed.
- EffectiveSpadRtnCount: 16-bit integer that returns the effective single photon avalanche diode (SPAD) count for the current ranging. To obtain real value it should be divided by 256.

One structure per target detected (up to 4 by default) called *VL53LX_TargetRangeData_t* which contains the following specific results for each target detected.

- RangeMaxMilliMeter: is a 16-bit integer, indicating the larger detected distance.
- RangeMinMilliMeter: is a 16-bit integer, indicating the smaller detected distance.
- SignalRateRtnMegaCps: this value is the return signal rate in MegaCountPer Second (MCPS), this is a 16.16 fix point value. To obtain the real value it should be divided by 65536.
- AmbientRateRtnMegaCps: this value is the return ambient rate (in MCPS), this is a 16.16 fix point value, which is effectively a measure of the amount of ambient light measured by the sensor. To obtain the real value it should be divided by 65536.
- SigmaMilliMeter: this 16.16 fix point value is an estimation of the standard deviation of the current ranging, expressed in millimeter. To obtain the real value it should be divided by 65536.
- RangeMilliMeter: is a 16-bit integer indicating the range distance in millimeter.
- RangeStatus: this is an 8-bit integer indicating the range status for the current measurement. Value = 0 means ranging is valid. See Table 1. Range status.
- ExtendedRange: this is an 8-bit integer indicating if the range has been unwrapped (only for long distances)

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 in the *VL53LX_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.
- ExtendedRange: forced to 0.

Table 1. Range status

Value	RangeStatus String	Comment
0	VL53LX_RANGESTATUS_RANGE_VALID	Ranging measurement is valid
1	VL53LX_RANGESTATUS_SIGMA_FAIL	Raised if a sigma estimator check is above the internal defined threshold. Sigma estimator gives a qualitative information about the signal.
2	VL53LX_RANGESTATUS_SIGNAL_FAIL	Raised when the signal is too low to detect a target.
4	VL53LX_RANGESTATUS_OUTOFBOUNDS_FAIL	Raised when range result is out of bounds
5	VL53LX_RANGESTATUS_HARDWARE_FAIL	Raised in case of HW or VCSEL failure
6	VL53LX_RANGESTATUS_RANGE_VALID_NO_WRAP_CHECK_FAIL	No wraparound check has been done (this is the very first range)
7	VL53LX_RANGESTATUS_WRAP_TARGET_FAIL	Wraparound occurred
8	VL53LX_RANGESTATUS_PROCESSING_FAIL	Internal processing error
10	VL53LX_RANGESTATUS_SYNCRONISATION_INT	Raised one time after init, ranging value has to be ignored
11	VL53LX_RANGESTATUS_RANGE_VALID_MERGED_PULSE	Ranging is OK, but the distance reported is the result of multiple targets merged.
12	VL53LX_RANGESTATUS_TARGET_PRESENT_LACK_OF_SIGNAL	Indicate that there is a target, but the signal is too low to report ranging
14	VL53LX_RANGESTATUS_RANGE_INVALID	Ranging data is negative and has to be ignored
255	VL53LX_RANGESTATUS_NONE	Target not detected, without warning or error raised

Note: The very first measurement does not include a wraparound check. This ranging measurement can be discarded.

Note: Range status 1 is often caused by noisy measurements. Sigma estimator is 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 Additional driver functions description

5.1 Timing budget

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

VL53LX_SetMeasurementTimingBudgetMicroSeconds() is the function to be used to set the timing budget.

The default timing budget value is 33 ms. Minimum is 8 ms, maximum is 500 ms.

For example, to set the timing budget to 66 ms:

status = VL53LX_SetMeasurementTimingBudgetMicroSeconds(&VL53L3Dev, 66000);

The function *VL53LX_GetMeasurementTimingBudgetMicroSeconds()* returns the programmed timing budget.

5.2 Distance mode

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

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

Table 2. Distance modes

Possible distance mode	Benefit / comments
Short	Better ambient immunity
Medium (Default)	Maximum distance
Long	Lower power consumption

The function to use is called *VL53LX_SetDistanceMode()*.

The driver can help the host to select the optimum distance mode. A specific value is returned at each ranging to indicate the best choice, depending on the ambient conditions.

Possible values are:

- VL53LX_DISTANCE_SHORT
- VL53LX_DISTANCE_MEDIUM
- VL53LX_DISTANCE_LONG

5.3 Tuning parameters

Tuning parameters allow to find the best fit between the sensor and the host use case. For each use case, a set of tuning parameters can be defined and loaded in the driver.

Most of the tuning parameters are tunable thresholds, used in the signal treatment algorithm. Modifying these parameters allows the algorithm to make technical trade offs to the specific customer use case.

5.3.1 Set a tuning parameter

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

The list of tuning parameters and their default values is given in the vl53lx_tuning_parm_defaults.h file. Either change a tuning parameter value in this file and recompile the code, or use the *VL53LX_SetTuningParameter()* function to load this tuning parameter.

Changing a tuning parameter can modify the device performances. ST recommends to use the default values for optimal results.

5.3.2 Improve accuracy

In order to improve the device accuracy, use the tuning parameter called *VL53LX_TUNINGPARM_PHASECAL_PATCH_POWER*. By default this tuning parameter is not applied (value is set to 0).

ST recommends to set the values of calibration and ranging flows to 2 after static_init.

In this case, the time to perform the reference signal measurement is increased and allows better accuracy.

Setting this parameter to 2 increases the duration to get the first measurement by 240 ms.

5.3.3 Improve latency and max ranging distance

When the target is moving, the VL53L3CX may need several ranges to react, depending on the scene.

A way to improve the latency is to tune the *VL53LX_TUNINGPARM_RESET_MERGE_THRESHOLD* parameter.

Default value is 15000. It can be lowered to improve latency, but maximum ranging distance will be impacted.

If the user increases the value, the maximum ranging distance can be improved, but latency is impacted.

5.4 Cover glass smudge detection

The crosstalk can be affected by smudge on the cover glass. VL53L3CX embeds a function able to detect smudge on the fly and apply a new crosstalk correction value.

The user can enable/disable this function by calling *VL53LX_SmudgeCorrectionEnable()*.

The following three options can be set with this function:

- VL53LX_SMUDGE_CORRECTION_NONE to disable the correction
- VL53LX_SMUDGE_CORRECTION_CONTINUOUS to enable a continuous correction
- VL53LX_SMUDGE_CORRECTION_SINGLE to enable a single correction after a start command is received.

Smudge detection is running at each ranging. If some conditions are met (no object below 80 cm, ambient light level below a threshold, and crosstalk value above 1kcps), a new crosstalk value is computed.

If the smudge correction is set, the crosstalk value is corrected and the flag HasXtalkValueChanged is set. This flag is automatically cleared at next range.

Note: The smudge correction is limited to:

- *1.2 m using short distance mode*
- *1.7 m using medium distance mode*
- *3.8 m using long distance mode.*

5.5 I2C address

The default I2C address of the VL53L3CX is 0x52.

Some applications need to set a different I2C device address. This is the case, for example, when several VL53L3CX parts share the same I2C bus.

The customer should apply the following procedure:

- The board mounting the VL53L3CX have to be designed carefully. The Xshut and the GPIO1 (interrupt) pins have to be controlled individually for each VL53L3CX
- The host has to put in HW Standby, setting the Xshut pin low, all the VL53L3CX.
- The host raises the Xshut pin of 1 of the VL53L3CX
- The host calls the function *VL53LX_SetDeviceAddress()*
- The host repeats the latter three points since all the VL53L3CX addresses are correctly set.

For example, by calling the function: *status = VL53LX_SetDeviceAddress(&VL53L3Dev, WantedAddress)*

the value of *WantedAddress* is set as the new I2C address.

6 Customer factory calibration functions

In order to benefit of the full performance of the device, the VL53L3CX driver includes calibration functions to be run once at 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 VL53L3CX at each startup using a dedicated driver function.

Three calibrations are needed: refSPAD, crosstalk and offset.

The order the calibration functions are called as follows:

1. refSPAD
2. crosstalk
3. offset

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

6.1 RefSPAD calibration

The number of single photon avalanche diode (SPAD) is calibrated during final module test at ST. This part-to-part value is stored in NVM and automatically loaded in the device during boot.

This calibration allow to adjust the number of SPADs to optimize the device dynamic.

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 locations: 1 (1x attenuated SPADs), 2 (5 x attenuated SPADs) and 3 (10 x attenuated SPADs). Number of SPADs chosen is done to avoid signal saturation.

6.1.1 RefSPAD calibration function

The following function is available for SPAD calibration: *VL53LX_PerformRefSpadManagement(VL53LX_DEV Dev)*

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

The function can output the following three warning messages :

- VL53LX_WARNING_REF_SPAD_CHAR_NOT_ENOUGH_SPADS
  Less than 5 Good SPAD available, output not valid
- VL53LX_WARNING_REF_SPAD_CHAR_RATE_TOO_HIGH
  At end of search reference rate > 40.0 Mcps Offset stability may be degraded.
- VL53LX_WARNING_REF_SPAD_CHAR_RATE_TOO_LOW
  At end of search reference rate < 10.0 Mcps. Offset stability may be degraded.

6.1.2 RefSPAD calibration procedure

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

Time to perform this calibration is only a few milliseconds.

This function has to be called after *VL53LX_DataInit()* function is called.

6.1.3 Getting refSPAD calibration results

The function *VL53LX_GetCalibrationData()* returns all calibration data. The returned structure VL53LX_CalibrationData_t contains another structure called *VL53LX_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 or 3. It gives the location of the SPADs in the reference 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 *VL53LX_DataInit()* function is called, by using *VL53LX_SetCalibrationData()*.

It is recommended to get the entire calibration structure by calling *VL53LX_GetCalibrationData()*. Modify the eight parameters described in Section 6.1.3 Getting refSPAD calibration results and call *VL53LX_SetCalibrationData()*.

6.2 Crosstalk calibration

Crosstalk (XTalk) 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) added on top of the module for aesthetic reasons.

Depending on the cover glass quality, this parasitic signal can affect the device performances. VL53L3CX has a built in correction that compensates this problem.

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 output of the crosstalk calibration contains many parameters that define the crosstalk model, as described in Section 6.2.3 Getting crosstalk calibration results.

6.2.1 Crosstalk calibration function

The following dedicated function is available for crosstalk calibration:

VL53LX_PerformXTalkCalibration(&VL53L3Dev);

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

6.2.2 Crosstalk calibration procedure

To perform the crosstalk calibration, a target has to be placed at a distance of 600mm from the device.

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

After *VL53LX_DataInit()* and *VL53LX_PerformRefSpadManagement() functions* are called, the dedicated calibration function has to be called, using: *VL53LX_PerformXTalkCalibration(&VL53L3Dev)*.

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

6.2.3 Getting crosstalk calibration results

Calibration results consist, among others, of a histogram and a parameter called “plane offset”. The plane offset represents the amount of correction applied, and the histogram is the repartition of the correction on each bin.

The function *VL53LX_GetCalibrationData()* returns all the calibration data. The returned structure *VL53LX_CalibrationData_t* contains other structures. The plane offset is contained in **VL53LX_customer_nvm_managed_t**:

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.

Two other relevant structures are returned: *VL53LX_xtalk_histogram_data_t* and *algo__xtalk_cpo_HistoMerge_kcps*. It is mandatory to store them.

6.2.4 Setting crosstalk calibration data

Once the *VL53LX_DataInit()* function is called, the customer can load the crosstalk calibration data using: *VL53LX_SetCalibrationData()*

It is better to call *VL53LX_GetCalibrationData()*, modify the parameters described in previous section, xtalk_histogram structure included, and call *VL53LX_SetCalibrationData()*

6.2.5 Enable/disable crosstalk compensation

The function *VL53LX_SetXTalkCompensationEnable()* enables or disables the crosstalk compensation.

Note: Crosstalk compensation is disabled by default.

To enable the crosstalk compensation call *V53LX_SetXTalkCompensationEnable&VL53L3Dev, 1);*

To disable the crosstalk compensation call *VL53LX_SetXTalkCompensationEnable&VL53L3Dev, 0);*

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

Note: Calibration, or loading of calibration data function, has to be called separately from this enable/disable function (see sections above).

6.3 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 during the offset calibration.

Offset calibration also allows to calibrate the dmax value, using the same calibration conditions than the offset calibration.

6.3.1 Offset calibration functions

The following two functions are available for the offset calibration:

- VL53LX_PerformOffsetSimpleCalibration(Dev, CalDistanceMilliMeter)
- VL53LX_PerformOffsetPerVCSELCalibration(Dev, CalDistanceMilliMeter)

The argument of the functions is the target distance in millimeters. Offset calibration has to be performed after crosstalk correction.

VL53LX_PerformOffsetPerVCSELCalibration is the most accurate function, but it takes longer to perform the calibration (time multiplied by 3).

6.3.2 Offset calibration procedure

Customers can select any chart reflectance placed at any distance (using the same setup as the crosstalk calibration). The only point to check is to ensure the signal rate is measured between 2 and 80 MCps with the chosen setup.

Table 3. Offset calibration setup

Chart	Distance	Ambient conditions	Signal Rate
Any	Any	Dark (no IR contribution)	2MCps < Signal Rate <80Mcps

Two warning messages are returned by these functions:

- VL53LX_WARNING_OFFSET_CAL_INSUFFICIENT_MM1_SPADS
  The signal is too low, Accuracy of offset calibration may be degraded.
- VL53LX_WARNING_OFFSET_CAL_PRE_RANGE_RATE_TOO_HIGH
  Signal is too high. Accuracy of offset calibration may be degraded.

6.3.3 Getting offset calibration results

The function *VL53LX_GetCalibrationData()* returns all calibration data. The returned structure *VL53LX_CalibrationData_t* contains another structure called *VL53LX_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

The overall offset applied to the device is the average of the two last values.

If a *perVCSELCalibration* is selected, the output of the function includes the following data:

- short_a_offset_mm
- short_b_offset_mm
- medium_a_offset_mm
- medium_b_offset_mm
- long_a_offset_mm
- long_bb_offset_mm

Depending on the distance mode (VCSEL period) chosen, one of these offsets is applied automatically.

6.3.4 Selecting the offset correction mode

The offset correction mode can be set with two options, using the *VL53LX_SetOffsetCorrectionMode function*.

Note: VL53LX_OFFSETCORRECTIONMODE_PERVCSEL should be used by default. It allows to increase the offset accuracy per VCSEL period.

Table 4. Offset correction options

Offset calibration function called	Correction mode option to be used
PerformSimpleOffsetCalibration	VL53LX_OFFSETCORRECTIONMODE_STANDARD
PerformPerVCSELOffsetCalibration	VL53LX_OFFSETCORRECTIONMODE_PERVCSEL

Note: If only one offset calibration type is available, it is mandatory to set the offset correction mode to the corresponding option. This is not done automatically.

6.3.5 Setting offset calibration data

The customer can load the offset calibration data after *VL53LX_DataInit()* function is called, by using *VL53LX_SetCalibrationData()*.

It is better to call *VL53LX_GetCalibrationData()*, modify the parameters described in previous sections, and call *VL53LX_SetCalibrationData()*.

7 Customer repair shop calibrations

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

The calibration is composed of three steps:

1. RefSpad
2. Crosstalk
3. Offset calibrations

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

A dedicated function is available to perform offset calibration: *VL53LX_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 (with no particular need 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 sections.

8 Bare driver errors and warnings

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

Warnings are there to inform the user that some parameters are not optimized.

The warnings are not blocking for the host.

Table 5. Bare driver errors and warnings description

Error value	API error string	Occurrence
0	VL53LX_ERROR_NONE	No error
-1	VL53LX_ERROR_CALIBRATION_WARNING	Invalid calibration data
-4	VL53LX_ERROR_INVALID_PARAMS	Invalid parameter is set in a function
-5	VL53LX_ERROR_NOT_SUPPORTED	Requested parameter is not supported in the programmed configuration
-6	VL53LX_ERROR_RANGE_ERROR	Interrupt status is incorrect
-7	VL53LX_ERROR_TIME_OUT	Ranging is aborted due to timeout
-8	VL53LX_ERROR_MODE_NOT_SUPPORTED	Requested mode is not supported
-10	VL53LX_ERROR_COMMS_BUFFER_TOO_SMALL	Supplied buffer is larger than I2C supports
-13	VL53LX_ERROR_CONTROL_INTERFACE	Error reported from IO function
-14	VL53LX_ERROR_INVALID_COMMAND	Command is invalid
-16	VL53LX_ERROR_REF_SPAD_INIT	An error occurred during Reference SPAD calibration
-17	VL53LX_ERROR_GPH_SYNC_CHECK_FAIL	Driver out of sync with device. A stop/start or a reboot may be needed
-18	VL53LX_ERROR_STREAM_COUNT_CHECK_FAIL	
-19	VL53LX_ERROR_GPH_ID_CHECK_FAIL	
-20	VL53LX_ERROR_ZONE_STREAM_COUNT_CHECK_FAIL	
-21	VL53LX_ERROR_ZONE_GPH_ID_CHECK_FAIL	
-22	VL53LX_ERROR_XTALK_EXTRACTION_FAIL	No successful samples when using the full array to sample the crosstalk. In this case there is not enough information to generate new crosstalk value. The function will exit and leave the current crosstalk parameters unaltered
-23	VL53LX_ERROR_XTALK_EXTRACTION_SIGMA_LIMIT_FAIL	The avg sigma estimate of the crosstalk sample is > than 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	VL53LX_ERROR_OFFSET_CAL_NO_SAMPLE_FAIL	An error occurred during offset calibration. Check setup is in line with ST recommendations.
-25	VL53LX_ERROR_OFFSET_CAL_NO_SPADS_ENABLED_FAIL	
-28	VL53LX_WARNING_REF_SPAD_CHAR_NOT_ENOUGH_SPADS	Warning: number of spads found is too low to get accurate refSpadManagement calibration. Ensure setup is in line with ST recommendations.
-29	VL53LX_WARNING_REF_SPAD_CHAR_RATE_TOO_HIGH	Warning: signal rate found too low to get accurate refSpadManagement calibration. Ensure setup is in line with ST recommendations.

Error value	API error string	Occurrence
-30	VL53LX_WARNING_REF_SPAD_CHAR_RATE_TOO_LOW	Warning: Number of spads found too low to get accurate offset calibration. Ensure setup is in line with ST recommendations.
-31	VL53LX_WARNING_OFFSET_CAL_MISSING_SAMPLES	Warning occurred during offset calibration. Ensure setup is in line with ST recommendations.
-32	VL53LX_WARNING_OFFSET_CAL_SIGMA_TOO_HIGH	
-33	VL53LX_WARNING_OFFSET_CAL_RATE_TOO_HIGH	
-34	VL53LX_WARNING_OFFSET_CAL_SPAD_COUNT_TOO_LOW	
-38	VL53LX_WARNING_XTALK_MISSING_SAMPLES	Warning occurred during crosstalk calibration. Ensure setup is in line with ST recommendations.
-41	VL53LX_ERROR_NOT_IMPLEMENTED	Function called is not implemented

Revision history

Table 6. **Document revision history**

Date	Version	Changes
28-Sep-2020	1	Initial release
02-Dec-2021	2	Updated the structures returned in Section 6.2.3 Getting crosstalk calibration results
03-Jun-2022	3	Section 3.1 Bare driver: added a note regarding calibration Section 5.4 Cover glass smudge detection: added a note regarding smudge correction

Contents