

# **VL53L0X API Specification**

Version 1.0.1.4606  
6/9/2016 5:43:00 PM



# Table of Contents

Documentation .....	1
Introduction .....	1
Overview .....	1
Device Info from API .....	1
Coding Standards.....	1
Platform .....	1
RangeStatus .....	2
Strings.....	3
Disclaimer.....	4
Module Index.....	4
Data Structure Index .....	5
File Index.....	5
Module Documentation .....	6
VL53L0X Platform Functions .....	6
PAL Register Access Functions .....	8
Basic type definition .....	11
VL53L0X cut1.1 Function Definition .....	11
VL53L0X General Functions .....	12
VL53L0X Init Functions .....	19
VL53L0X Parameters Functions .....	22
VL53L0X Measurement Functions .....	40
VL53L0X Interrupt Functions .....	47
VL53L0X SPAD Functions.....	51
VL53L0X Defines .....	55
Error and Warning code returned by API .....	58
Defines Device modes .....	61
Defines Histogram modes.....	62
List of available Power Modes .....	63
Defines the current status of the device .....	64
Defines the Polarity .....	65
Vcsel Period Defines .....	66
Defines the steps .....	67
Defines the Polarity .....	67
General Macro Defines.....	68
VL53L0X cut1.1 Device Specific Defines .....	71
Device Error .....	71
Check Enable list .....	74
Gpio Functionality.....	75
Define Registers .....	76
Data Structure Documentation .....	84
VL53L0X_Dev_t.....	84
VL53L0X_DevData_t .....	85
VL53L0X_DeviceInfo_t.....	88
VL53L0X_DeviceParameters_t.....	89
VL53L0X_DeviceSpecificParameters_t.....	91
VL53L0X_DMaxData_t.....	93
VL53L0X_HistogramData_t .....	94
VL53L0X_HistogramMeasurementData_t.....	94
VL53L0X_RangeData_t.....	95
VL53L0X_RangingMeasurementData_t.....	96
VL53L0X_SchedulerSequenceSteps_t.....	97
VL53L0X_SpadData_t.....	98
VL53L0X_Version_t.....	99
File Documentation .....	100
PAL_disclaimer.c .....	100

vl53l0x_api.h .....	100
vl53l0x_api_calibration.h .....	106
vl53l0x_api_core.h .....	107
vl53l0x_api_ranging.h .....	110
vl53l0x_api_strings.h .....	110
vl53l0x_def.h .....	118
vl53l0x_device.h .....	121
vl53l0x_doxydoc.c .....	123
vl53l0x_i2c_platform.h .....	123
vl53l0x_interrupt_threshold_settings.h .....	130
vl53l0x_platform.h .....	131
vl53l0x_platform_log.h .....	132
vl53l0x_tuning.h .....	133
vl53l0x_types.h .....	134
Index .....	135

---

# Documentation

## Introduction

The Photonics Abstraction Layer (PAL) is intended to provide an API functions to aid the development of applications.

## Overview

This document is intended to aid in the development of applications around PAL sensor family and describes the various API functions provided by the API delivered by ST as open source C code.

Some of the API files are hardware and platform dependent (specially I2C access) so need to be adapted to the platform used by the customer.

## Device Info from API

The API provide a function that can be used to obtain information of the device used like the cut version. This function is [VL53L0X\\_GetDeviceInfo\(\)](#).

## Coding Standards

The implementation of this API will follow Linux Kernel rules as defined in <https://www.kernel.org/doc/Documentation/CodingStyle>

---

# Platform

All API settings that are platform-dependent must be adapted to the platform on which API is compiled/running.

This is done in [VL53L0X\\_platform.h](#) file. Platform settings are described in the [VL53L0 Platform Functions](#) module.

### 1. PAL device type definition

User must provide [VL53L0X\\_Dev\\_t](#) type (in [VL53L0X\\_platform.h](#) file) as all API functions and macros rely on [VL53L0X\\_Dev\\_t dev](#) (given as first argument). This **dev** object does the link between API and platform abstraction layer and is passed from function to function down to final platform abstraction layer that handles final access to the device :

```
int VL53L0X_XXXX(VL53L0X_Dev_t dev, ... )
```

In single device case, **dev** can be as simple as an integer being the i2c device address

For more elaborated platform, **dev** can be a pointer to a structure containing all necessary items for the platform.

## 2. Read & Write access

API low-level functions rely on a few set of read & write functions which perform the access to the device. These functions must be implemented with respect to the platform on which API is compiled and running. Internal PAL register access functions should be used :

- [VL53L0X WriteMulti\(\)](#)
- [VL53L0X ReadMulti\(\)](#)
- [VL53L0X WrByte\(\)](#)
- [VL53L0X WrWord\(\)](#)
- [VL53L0X WrDWord\(\)](#)
- [VL53L0X UpdateByte\(\)](#)
- [VL53L0X RdByte\(\)](#)
- [VL53L0X RdWord\(\)](#)
- [VL53L0X RdDWord\(\)](#)
- 

## 3. Data Types declaration

API functions rely on data types which are defined in [VL53L0X\\_types.h](#) file (under **platform/template** directory). This file may require user attention and porting in case of warning messages.

## 4. Delay for polling operations

API polling high level functions do call the function [VL53L0X PollingDelay\(\)](#) inside their while loop. A default implementation of the [VL53L0X PollingDelay\(\)](#) function is provided. You may decide to change and implement your own [VL53L0X PollingDelay\(\)](#) function.

## 5. API logging

All API functions entry and leave can be logged to help debugging issues. By default logging is disabled please define VL53L0X\_LOG\_ENABLE at compilation level. If logging is enabled, a small set of macros must be implemented to adapt logging operation to the platform : [LOG\\_FUNCTION\\_START](#) , [LOG\\_FUNCTION\\_END](#) and [LOG\\_FUNCTION\\_END\\_FMT](#)

---

# RangeStatus

The Range Status is contained in the [VL53L0X RangingMeasurementData\\_t](#) and give the quality of the latest ranging.

This is a 8 bit data which contains the following fields:

### Value 0 = Range Valid

This value indicate that the ranging is valid.

### Value 1 = Sigma Fail

This value indicate that the sigma limit check has failed. Use the function [VL53L0X SetLimitCheckEnable\(\)](#) and [VL53L0X SetLimitCheckValue\(\)](#) to manage the limit.

### Value 2 = Signal Fail

This value indicate that the signal check has failed. This can happens when there is no target or when the Range Ignore threshold check has failed. Use the function [VL53L0X\\_SetLimitCheckEnable\(\)](#) and [VL53L0X\\_SetLimitCheckValue\(\)](#) to manage the limit.

### Value 3 = Min Range Fail

This value indicate that the min range check has failed. Use the function [VL53L0X\\_SetLimitCheckEnable\(\)](#) and [VL53L0X\\_SetLimitCheckValue\(\)](#) to manage the limit.

### Value 4 = Phase Fail

This value indicate that the Phase check has failed.

### Value 5 = HardWare Fail

This value indicate that the Hardware check has failed.

### Value 255 = None

No Update

## Strings

The API uses character strings to inform the user about the state of the API, the meaning of the error or about the name of a particular mode.

### 1. String can be removed

At compilation stage a DEFINE can be used to remove all the strings to save some space on device. Strings will be replaced with empty string.

The Define to be used is USE\_EMPTY\_STRING:

- if USE\_EMPTY\_STRING is defined: all the strings are replaced with empty string.
- if USE\_EMPTY\_STRING is NOT defined: all the strings are well defined and not empty.

### 2. Max Lenght String

The API uses the macro VL53L0X\_COPYSTRING to copy strings. For example the following code from get device info

```
VL53L0X_COPYSTRING(pVL53L0X_DeviceInfo->Type,
    VL53L0X_STRING_DEVICE_INFO_TYPE);
```

This MACRO is defined inside platform code. This means that is the responsibility of the customer to use the right function to copy the string. In the Platform gives as example this is:

```
#define VL53L0X_COPYSTRING(str, ...) strcpy(str, ## __VA_ARGS__)
```

In previous example we copy the string defined in VL53L0X\_STRING\_DEVICE\_INFO\_TYPE in a field in a structure pVL53L0X\_DeviceInfo->Type. This is defined with a max lenght:

```
char Type[VL53L0X_MAX_STRING_LENGTH];
```

In that case by construction the Define:

```
len(VL53L0X_STRING_DEVICE_INFO_TYPE) < VL53L0X_MAX_STRING_LENGTH.
```

In the API the max lenght is defined in the VL53L0X\_api\_def.h as follow:

```
#define VL53L0X_MAX_STRING_LENGTH 32
```

In the API there are some functions which output directly the string like the following:

```
VL53L0X_Error VL53L0X_GetRangeStatusString(uint8_t RangeStatus,
char *pRangeStatusString)
```

Even in that case a copy string is done. To avoid overflow problem when the copy is done, the string which will contains the one is copied, should be greather or equal to the max lenght described before.

```
void print_range_status(VL53L0X_RangingMeasurementData_t* pRangingMeasurementData){
    char buf[VL53L0X_MAX_STRING_LENGTH];
    uint8_t RangeStatus;

    RangeStatus = pRangingMeasurementData->RangeStatus;

    VL53L0X_GetRangeStatusString(RangeStatus, buf);
    printf("Range Status: %i : %s\n", RangeStatus, buf);
}
```

## Disclaimer

Copyright (C) 2015 STMicroelectronics Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.3 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts.

A copy of the license is included in the section entitled "GNU Free Documentation License".

## Module Index

### Modules

Here is a list of all modules:

VL53L0 Platform Functions.....	6
PAL Register Access Functions .....	8
Basic type definition.....	11
VL53L0X cut1.1 Function Definition .....	11
VL53L0X General Functions .....	12
VL53L0X Init Functions .....	19
VL53L0X Parameters Functions .....	22
VL53L0X Measurement Functions .....	40
VL53L0X Interrupt Functions.....	47
VL53L0X SPAD Functions .....	51
VL53L0X Defines .....	55



Error and Warning code returned by API.....	58
Defines Device modes .....	61
Defines Histogram modes .....	62
List of available Power Modes .....	63
Defines the current status of the device.....	64
Defines the Polarity .....	65
Vcsel Period Defines .....	66
Defines the steps.....	67
Defines the Polarity .....	67
General Macro Defines .....	68
 VL53L0X cut1.1 Device Specific Defines .....	 71
Device Error .....	71
Check Enable list.....	74
Gpio Functionality.....	75
Define Registers .....	76

## Data Structure Index

### Data Structures

Here are the data structures with brief descriptions:

<a href="#"><u>VL53L0X_Dev_t</u></a> (Generic PAL device type that does link between API and platform abstraction layer ) .....	84
<a href="#"><u>VL53L0X_DevData_t</u></a> (VL53L0X PAL device ST private data structure End user should never access any of these field directly ) .....	85
<a href="#"><u>VL53L0X_DeviceInfo_t</u></a> (Defines the parameters of the Get Device Info Functions ) .....	88
<a href="#"><u>VL53L0X_DeviceParameters_t</u></a> (Defines all parameters for the device ) .....	89
<a href="#"><u>VL53L0X_DeviceSpecificParameters_t</u></a> .....	91
<a href="#"><u>VL53L0X_DMaxData_t</u></a> (Structure containing the Dmax computation parameters and data ) .....	93
<a href="#"><u>VL53L0X_HistogramData_t</u></a> (Histogram measurement data ) .....	94
<a href="#"><u>VL53L0X_HistogramMeasurementData_t</u></a> .....	94
<a href="#"><u>VL53L0X_RangeData_t</u></a> (Range measurement data ) .....	95
<a href="#"><u>VL53L0X_RangingMeasurementData_t</u></a> .....	96
<a href="#"><u>VL53L0X_SchedulerSequenceSteps_t</u></a> .....	97
<a href="#"><u>VL53L0X_SpadData_t</u></a> (Spad Configuration Data ) .....	98
<a href="#"><u>VL53L0X_Version_t</u></a> (Defines the parameters of the Get Version Functions ) .....	99

## File Index

### File List

Here is a list of all files with brief descriptions:

<a href="#"><u>PAL_disclaimer.c</u></a> (No code doxygen doc only ) .....	100
<a href="#"><u>vl53l0x_api.h</u></a> .....	100

<a href="#">vl53l0x_api_calibration.h</a>	106
<a href="#">vl53l0x_api_core.h</a>	107
<a href="#">vl53l0x_api_ranging.h</a>	110
<a href="#">vl53l0x_api_strings.h</a>	110
<a href="#">vl53l0x_def.h</a> (Type definitions for VL53L0X API )	118
<a href="#">vl53l0x_device.h</a>	121
<a href="#">vl53l0x_doxydoc.c</a>	123
<a href="#">vl53l0x_i2c_platform.h</a>	123
<a href="#">vl53l0x_interrupt_threshold_settings.h</a>	130
<a href="#">vl53l0x_platform.h</a> (Function prototype definitions for Ewok Platform layer )	131
<a href="#">vl53l0x_platform_log.h</a>	132
<a href="#">vl53l0x_tuning.h</a>	133
<a href="#">vl53l0x_types.h</a>	134

## Module Documentation

### VL53L0 Platform Functions

VL53L0 Platform Functions.

#### Modules

- [PAL Register Access Functions](#)
- *PAL Register Access Functions.* [Basic type definition](#)

***file vl53l0\_types.h files hold basic type definition that may requires porting***  
**Data Structures**

- struct [VL53L0X\\_Dev\\_t](#)

***Generic PAL device type that does link between API and platform abstraction layer. Macros***

- #define [PALDevDataGet](#)(Dev, field) (Dev->Data.field)  
Get ST private structure [VL53L0X\\_DevData\\_t](#) data access.
- #define [PALDevDataSet](#)(Dev, field, data) (Dev->Data.field)=(data)  
Set ST private structure [VL53L0X\\_DevData\\_t](#) data field.

#### Typedefs

- typedef [VL53L0X\\_Dev\\_t](#) \* [VL53L0X\\_DEV](#)  
Declare the device Handle as a pointer of the structure [VL53L0X\\_Dev\\_t](#).

#### Functions

- [VL53L0X\\_Error](#) [VL53L0X\\_PollingDelay](#) ([VL53L0X\\_DEV](#) Dev)  
execute delay in all polling API call

### Detailed Description

VL53L0 Platform Functions.

## Macro Definition Documentation

### **#define PALDevDataGet( Dev, field) (Dev->Data.field)**

Get ST private structure [VL53L0X\\_DevData\\_t](#) data access.

#### Parameters:

<i>Dev</i>	Device Handle
<i>field</i>	ST structure field name It maybe used and as real data "ref" not just as "get" for sub-structure item like PALDevDataGet(FilterData.field)[i] or PALDevDataGet(FilterData.MeasurementIndex)++

Definition at line 84 of file vl53l0x\_platform.h.

### **#define PALDevDataSet( Dev, field, data) (Dev->Data.field)=(data)**

Set ST private structure [VL53L0X\\_DevData\\_t](#) data field.

#### Parameters:

<i>Dev</i>	Device Handle
<i>field</i>	ST structure field name
<i>data</i>	Data to be set

Definition at line 93 of file vl53l0x\_platform.h.

## Typedef Documentation

### **typedef [VL53L0X\\_Dev\\_t](#)\* [VL53L0X\\_DEV](#)**

Declare the device Handle as a pointer of the structure [VL53L0X\\_Dev\\_t](#).

Definition at line 73 of file vl53l0x\_platform.h.

## Function Documentation

### **[VL53L0X\\_Error](#) [VL53L0X\\_PollingDelay](#) ([VL53L0X\\_DEV](#) Dev)**

execute delay in all polling API call

A typical multi-thread or RTOs implementation is to sleep the task for some 5ms (with 100Hz max rate faster polling is not needed) if nothing specific is need you can define it as an empty/void macro

```
1 #define VL53L0X_PollingDelay(...) (void)0
```

#### Parameters:

<i>Dev</i>	Device Handle
------------	---------------

#### Returns:

VL53L0X\_ERROR\_NONE Success  
 "Other error code" See [VL53L0X\\_Error](#)

## PAL Register Access Functions

PAL Register Access Functions.

### Functions

- [VL53L0X\\_Error VL53L0X\\_LockSequenceAccess](#) ([VL53L0X\\_DEV](#) Dev)  
*Lock comms interface to serialize all commands to a shared I2C interface for a specific device.*
- [VL53L0X\\_Error VL53L0X\\_UnlockSequenceAccess](#) ([VL53L0X\\_DEV](#) Dev)  
*Unlock comms interface to serialize all commands to a shared I2C interface for a specific device.*
- [VL53L0X\\_Error VL53L0X\\_WriteMulti](#) ([VL53L0X\\_DEV](#) Dev, [uint8\\_t](#) index, [uint8\\_t](#) \*pdata, [uint32\\_t](#) count)  
*Writes the supplied byte buffer to the device.*
- [VL53L0X\\_Error VL53L0X\\_ReadMulti](#) ([VL53L0X\\_DEV](#) Dev, [uint8\\_t](#) index, [uint8\\_t](#) \*pdata, [uint32\\_t](#) count)  
*Reads the requested number of bytes from the device.*
- [VL53L0X\\_Error VL53L0X\\_WrByte](#) ([VL53L0X\\_DEV](#) Dev, [uint8\\_t](#) index, [uint8\\_t](#) data)  
*Write single byte register.*
- [VL53L0X\\_Error VL53L0X\\_WrWord](#) ([VL53L0X\\_DEV](#) Dev, [uint8\\_t](#) index, [uint16\\_t](#) data)  
*Write word register.*
- [VL53L0X\\_Error VL53L0X\\_WrDWord](#) ([VL53L0X\\_DEV](#) Dev, [uint8\\_t](#) index, [uint32\\_t](#) data)  
*Write double word (4 byte) register.*
- [VL53L0X\\_Error VL53L0X\\_RdByte](#) ([VL53L0X\\_DEV](#) Dev, [uint8\\_t](#) index, [uint8\\_t](#) \*data)  
*Read single byte register.*
- [VL53L0X\\_Error VL53L0X\\_RdWord](#) ([VL53L0X\\_DEV](#) Dev, [uint8\\_t](#) index, [uint16\\_t](#) \*data)  
*Read word (2byte) register.*
- [VL53L0X\\_Error VL53L0X\\_RdDWord](#) ([VL53L0X\\_DEV](#) Dev, [uint8\\_t](#) index, [uint32\\_t](#) \*data)  
*Read dword (4byte) register.*
- [VL53L0X\\_Error VL53L0X\\_UpdateByte](#) ([VL53L0X\\_DEV](#) Dev, [uint8\\_t](#) index, [uint8\\_t](#) AndData, [uint8\\_t](#) OrData)  
*Threat safe Update (read/modify/write) single byte register.*

---

### Detailed Description

PAL Register Access Functions.

---

### Function Documentation

#### [VL53L0X\\_Error VL53L0X\\_LockSequenceAccess](#) ([VL53L0X\\_DEV](#) Dev)

Lock comms interface to serialize all commands to a shared I2C interface for a specific device.

#### Parameters:

<i>Dev</i>	Device Handle
------------	---------------

#### Returns:

VL53L0X\_ERROR\_NONE Success  
 "Other error code" See [VL53L0X\\_Error](#)

### [VL53L0X\\_Error](#) VL53L0X\_UnlockSequenceAccess ([VL53L0X\\_DEV](#) *Dev*)

Unlock comms interface to serialize all commands to a shared I2C interface for a specific device.

#### Parameters:

<i>Dev</i>	Device Handle
------------	---------------

#### Returns:

VL53L0X\_ERROR\_NONE Success  
 "Other error code" See [VL53L0X\\_Error](#)

### [VL53L0X\\_Error](#) VL53L0X\_WriteMulti ([VL53L0X\\_DEV](#) *Dev*, [uint8\\_t](#) *index*, [uint8\\_t](#) \* *pdata*, [uint32\\_t](#) *count*)

Writes the supplied byte buffer to the device.

#### Parameters:

<i>Dev</i>	Device Handle
<i>index</i>	The register index
<i>pdata</i>	Pointer to uint8_t buffer containing the data to be written
<i>count</i>	Number of bytes in the supplied byte buffer

#### Returns:

VL53L0X\_ERROR\_NONE Success  
 "Other error code" See [VL53L0X\\_Error](#)

### [VL53L0X\\_Error](#) VL53L0X\_ReadMulti ([VL53L0X\\_DEV](#) *Dev*, [uint8\\_t](#) *index*, [uint8\\_t](#) \* *pdata*, [uint32\\_t](#) *count*)

Reads the requested number of bytes from the device.

#### Parameters:

<i>Dev</i>	Device Handle
<i>index</i>	The register index
<i>pdata</i>	Pointer to the uint8_t buffer to store read data
<i>count</i>	Number of uint8_t's to read

#### Returns:

VL53L0X\_ERROR\_NONE Success  
 "Other error code" See [VL53L0X\\_Error](#)

### [VL53L0X\\_Error](#) VL53L0X\_WrByte ([VL53L0X\\_DEV](#) *Dev*, [uint8\\_t](#) *index*, [uint8\\_t](#) *data*)

Write single byte register.

#### Parameters:

<i>Dev</i>	Device Handle
<i>index</i>	The register index
<i>data</i>	8 bit register data

#### Returns:

VL53L0X\_ERROR\_NONE Success  
 "Other error code" See [VL53L0X\\_Error](#)

[VL53L0X\\_Error](#) VL53L0X\_WrWord ([VL53L0X\\_DEV](#) *Dev*, [uint8\\_t](#) *index*, [uint16\\_t](#) *data*)

Write word register.

#### Parameters:

<i>Dev</i>	Device Handle
<i>index</i>	The register index
<i>data</i>	16 bit register data

#### Returns:

VL53L0X\_ERROR\_NONE Success  
 "Other error code" See [VL53L0X\\_Error](#)

[VL53L0X\\_Error](#) VL53L0X\_WrDWord ([VL53L0X\\_DEV](#) *Dev*, [uint8\\_t](#) *index*, [uint32\\_t](#) *data*)

Write double word (4 byte) register.

#### Parameters:

<i>Dev</i>	Device Handle
<i>index</i>	The register index
<i>data</i>	32 bit register data

#### Returns:

VL53L0X\_ERROR\_NONE Success  
 "Other error code" See [VL53L0X\\_Error](#)

[VL53L0X\\_Error](#) VL53L0X\_RdByte ([VL53L0X\\_DEV](#) *Dev*, [uint8\\_t](#) *index*, [uint8\\_t](#) \* *data*)

Read single byte register.

#### Parameters:

<i>Dev</i>	Device Handle
<i>index</i>	The register index
<i>data</i>	pointer to 8 bit data

#### Returns:

VL53L0X\_ERROR\_NONE Success  
 "Other error code" See [VL53L0X\\_Error](#)

[VL53L0X\\_Error](#) VL53L0X\_RdWord ([VL53L0X\\_DEV](#) *Dev*, [uint8\\_t](#) *index*, [uint16\\_t](#) \* *data*)

Read word (2byte) register.

#### Parameters:

<i>Dev</i>	Device Handle
<i>index</i>	The register index
<i>data</i>	pointer to 16 bit data

#### Returns:

VL53L0X\_ERROR\_NONE Success  
 "Other error code" See [VL53L0X\\_Error](#)

[VL53L0X\\_Error](#) VL53L0X\_RdDWord ([VL53L0X\\_DEV](#) *Dev*, [uint8\\_t](#) *index*, [uint32\\_t](#) \* *data*)

Read dword (4byte) register.

#### Parameters:

<i>Dev</i>	Device Handle
<i>index</i>	The register index
<i>data</i>	pointer to 32 bit data

#### Returns:

VL53L0X\_ERROR\_NONE Success  
 "Other error code" See [VL53L0X\\_Error](#)

[VL53L0X\\_Error](#) VL53L0X\_UpdateByte ([VL53L0X\\_DEV](#) *Dev*, [uint8\\_t](#) *index*, [uint8\\_t](#) *AndData*, [uint8\\_t](#) *OrData*)

Threat safe Update (read/modify/write) single byte register.

Final\_reg = (Initial\_reg & and\_data) | or\_data

#### Parameters:

<i>Dev</i>	Device Handle
<i>index</i>	The register index
<i>AndData</i>	8 bit and data
<i>OrData</i>	8 bit or data

#### Returns:

VL53L0X\_ERROR\_NONE Success  
 "Other error code" See [VL53L0X\\_Error](#)

## Basic type definition

file vl53l0\_types.h files hold basic type definition that may requires porting

file vl53l0\_types.h files hold basic type definition that may requires porting

contains type that must be defined for the platform

when target platform and compiler providestdint.h and stddef.h it is enough to include it.

Ifstdint.h is not available review and adapt all signed and unsigned 8/16/32 bits basic types.

If stddef.h is not available review and adapt NULL definition .

## VL53L0X cut1.1 Function Definition

VL53L0X cut1.1 Function Definition.

## Modules

- [VL53L0X General Functions](#)
- General functions and definitions. [VL53L0X Init Functions](#)
- VL53L0X Init Functions. [VL53L0X Parameters Functions](#)
- Functions used to prepare and setup the device. [VL53L0X Measurement Functions](#)
- Functions used for the measurements. [VL53L0X Interrupt Functions](#)
- Functions used for interrupt managements. [VL53L0X SPAD Functions](#)

Functions used for SPAD managements.

---

## Detailed Description

VL53L0X cut1.1 Function Definition.

## VL53L0X General Functions

General functions and definitions.

### Functions

- [VL53L0X\\_API](#) [VL53L0X\\_Error](#) [VL53L0X\\_GetVersion](#) ([VL53L0X\\_Version\\_t](#) \*pVersion)  
*Return the VL53L0X PAL Implementation Version.*
- [VL53L0X\\_API](#) [VL53L0X\\_Error](#) [VL53L0X\\_GetPalSpecVersion](#) ([VL53L0X\\_Version\\_t](#) \*pPalSpecVersion)  
*Return the PAL Specification Version used for the current implementation.*
- [VL53L0X\\_API](#) [VL53L0X\\_Error](#) [VL53L0X\\_GetProductRevision](#) ([VL53L0X\\_DEV](#) Dev, [uint8\\_t](#) \*pProductRevisionMajor, [uint8\\_t](#) \*pProductRevisionMinor)  
*Reads the Product Revision for a for given Device This function can be used to distinguish cut1.0 from cut1.1.*
- [VL53L0X\\_API](#) [VL53L0X\\_Error](#) [VL53L0X\\_GetDeviceInfo](#) ([VL53L0X\\_DEV](#) Dev, [VL53L0X\\_DeviceInfo\\_t](#) \*pVL53L0X\_DeviceInfo)  
*Reads the Device information for given Device.*
- [VL53L0X\\_API](#) [VL53L0X\\_Error](#) [VL53L0X\\_GetDeviceErrorStatus](#) ([VL53L0X\\_DEV](#) Dev, [VL53L0X\\_DeviceError](#) \*pDeviceErrorStatus)  
*Read current status of the error register for the selected device.*
- [VL53L0X\\_API](#) [VL53L0X\\_Error](#) [VL53L0X\\_GetRangeStatusString](#) ([uint8\\_t](#) RangeStatus, char \*pRangeStatusString)  
*Human readable Range Status string for a given RangeStatus.*
- [VL53L0X\\_API](#) [VL53L0X\\_Error](#) [VL53L0X\\_GetDeviceErrorString](#) ([VL53L0X\\_DeviceError](#) ErrorCode, char \*pDeviceErrorString)  
*Human readable error string for a given Error Code.*
- [VL53L0X\\_API](#) [VL53L0X\\_Error](#) [VL53L0X\\_GetPalErrorString](#) ([VL53L0X\\_Error](#) PalErrorCode, char \*pPalErrorString)  
*Human readable error string for current PAL error status.*
- [VL53L0X\\_API](#) [VL53L0X\\_Error](#) [VL53L0X\\_GetPalStateString](#) ([VL53L0X\\_State](#) PalStateCode, char \*pPalStateString)  
*Human readable PAL State string.*
- [VL53L0X\\_API](#) [VL53L0X\\_Error](#) [VL53L0X\\_GetPalState](#) ([VL53L0X\\_DEV](#) Dev, [VL53L0X\\_State](#) \*pPalState)  
*Reads the internal state of the PAL for a given Device.*
- [VL53L0X\\_API](#) [VL53L0X\\_Error](#) [VL53L0X\\_SetPowerMode](#) ([VL53L0X\\_DEV](#) Dev, [VL53L0X\\_PowerModes](#) PowerMode)  
*Set the power mode for a given Device The power mode can be Standby or Idle.*



- [VL53L0X\\_API\\_VL53L0X\\_Error\\_VL53L0X\\_GetPowerMode](#) ([VL53L0X\\_DEV](#) Dev, [VL53L0X\\_PowerModes](#) \*pPowerMode)  
*Get the power mode for a given Device.*
- [VL53L0X\\_API\\_VL53L0X\\_Error\\_VL53L0X\\_SetOffsetCalibrationDataMicroMeter](#) ([VL53L0X\\_DEV](#) Dev, [int32\\_t](#) OffsetCalibrationDataMicroMeter)  
*Set or over-hide part to part calibration offset.*
- [VL53L0X\\_API\\_VL53L0X\\_Error\\_VL53L0X\\_GetOffsetCalibrationDataMicroMeter](#) ([VL53L0X\\_DEV](#) Dev, [int32\\_t](#) \*pOffsetCalibrationDataMicroMeter)  
*Get part to part calibration offset.*
- [VL53L0X\\_API\\_VL53L0X\\_Error\\_VL53L0X\\_SetLinearityCorrectiveGain](#) ([VL53L0X\\_DEV](#) Dev, [int16\\_t](#) LinearityCorrectiveGain)  
*Set the linearity corrective gain.*
- [VL53L0X\\_API\\_VL53L0X\\_Error\\_VL53L0X\\_GetLinearityCorrectiveGain](#) ([VL53L0X\\_DEV](#) Dev, [uint16\\_t](#) \*pLinearityCorrectiveGain)  
*Get the linearity corrective gain.*
- [VL53L0X\\_API\\_VL53L0X\\_Error\\_VL53L0X\\_SetGroupParamHold](#) ([VL53L0X\\_DEV](#) Dev, [uint8\\_t](#) GroupParamHold)  
*Set Group parameter Hold state.*
- [VL53L0X\\_API\\_VL53L0X\\_Error\\_VL53L0X\\_GetUpperLimitMilliMeter](#) ([VL53L0X\\_DEV](#) Dev, [uint16\\_t](#) \*pUpperLimitMilliMeter)  
*Get the maximal distance for actual setup.*
- [VL53L0X\\_Error\\_VL53L0X\\_GetTotalSignalRate](#) ([VL53L0X\\_DEV](#) Dev, [FixPoint1616\\_t](#) \*pTotalSignalRate)  
*Get the Total Signal Rate.*

## Detailed Description

General functions and definitions.

## Function Documentation

### [VL53L0X\\_API\\_VL53L0X\\_Error\\_VL53L0X\\_GetVersion](#) ([VL53L0X\\_Version\\_t](#) \* pVersion)

Return the VL53L0X PAL Implementation Version.

#### Note:

This function doesn't access to the device

#### Parameters:

<a href="#">pVersion</a>	Pointer to current PAL Implementation Version
--------------------------	---

#### Returns:

VL53L0X\_ERROR\_NONE Success  
"Other error code" See [VL53L0X\\_Error](#)

### [VL53L0X\\_API\\_VL53L0X\\_Error\\_VL53L0X\\_GetPalSpecVersion](#) ([VL53L0X\\_Version\\_t](#) \* pPalSpecVersion)

Return the PAL Specification Version used for the current implementation.

**Note:**

This function doesn't access to the device

**Parameters:**

<i>pPalSpecVersion</i>	Pointer to current PAL Specification Version
------------------------	--

**Returns:**

VL53L0X\_ERROR\_NONE Success  
 "Other error code" See [VL53L0X\\_Error](#)

[VL53L0X\\_API](#) [VL53L0X\\_Error](#) VL53L0X\_GetProductRevision ([VL53L0X\\_DEV](#) Dev, [uint8\\_t](#) \*  
[pProductRevisionMajor](#), [uint8\\_t](#) \* [pProductRevisionMinor](#))

Reads the Product Revision for a for given Device This function can be used to distinguish cut1.0 from cut1.1.

**Note:**

This function Access to the device

**Parameters:**

<i>Dev</i>	Device Handle
<i>pProductRevisionMajor</i>	Pointer to Product Revision Major for a given Device
<i>pProductRevisionMinor</i>	Pointer to Product Revision Minor for a given Device

**Returns:**

VL53L0X\_ERROR\_NONE Success  
 "Other error code" See [VL53L0X\\_Error](#)

[VL53L0X\\_API](#) [VL53L0X\\_Error](#) VL53L0X\_GetDeviceInfo ([VL53L0X\\_DEV](#) Dev,  
[VL53L0X\\_DeviceInfo\\_t](#) \* [pVL53L0X\\_DeviceInfo](#))

Reads the Device information for given Device.

**Note:**

This function Access to the device

**Parameters:**

<i>Dev</i>	Device Handle
<i>pVL53L0X_DeviceInfo</i>	Pointer to current device info for a given Device

**Returns:**

VL53L0X\_ERROR\_NONE Success  
 "Other error code" See [VL53L0X\\_Error](#)

[VL53L0X\\_API](#) [VL53L0X\\_Error](#) VL53L0X\_GetDeviceErrorStatus ([VL53L0X\\_DEV](#) Dev,  
[VL53L0X\\_DeviceError](#) \* [pDeviceErrorStatus](#))

Read current status of the error register for the selected device.

**Note:**

This function Access to the device

#### Parameters:

<i>Dev</i>	Device Handle
<i>pDeviceErrorStatus</i>	Pointer to current error code of the device

#### Returns:

VL53L0X\_ERROR\_NONE Success  
 "Other error code" See [VL53L0X\\_Error](#)

[VL53L0X\\_API](#) [VL53L0X\\_Error](#) VL53L0X\_GetRangeStatusString ([uint8\\_t](#) RangeStatus, char \* pRangeStatusString)

Human readable Range Status string for a given RangeStatus.

#### Note:

This function doesn't access to the device

#### Parameters:

<i>RangeStatus</i>	The RangeStatus code as stored on <a href="#">VL53L0X_RangingMeasurementData_t</a>
<i>pRangeStatusString</i>	The returned RangeStatus string.

#### Returns:

VL53L0X\_ERROR\_NONE Success  
 "Other error code" See [VL53L0X\\_Error](#)

[VL53L0X\\_API](#) [VL53L0X\\_Error](#) VL53L0X\_GetDeviceErrorString ([VL53L0X\\_DeviceError](#) ErrorCode, char \* pDeviceErrorString)

Human readable error string for a given Error Code.

#### Note:

This function doesn't access to the device

#### Parameters:

<i>ErrorCode</i>	The error code as stored on <a href="#">VL53L0X_DeviceError</a>
<i>pDeviceErrorString</i>	The error string corresponding to the ErrorCode

#### Returns:

VL53L0X\_ERROR\_NONE Success  
 "Other error code" See [VL53L0X\\_Error](#)

[VL53L0X\\_API](#) [VL53L0X\\_Error](#) VL53L0X\_GetPalErrorString ([VL53L0X\\_Error](#) PalErrorCode, char \* pPalErrorString)

Human readable error string for current PAL error status.

#### Note:

This function doesn't access to the device

#### Parameters:

<i>PalErrorCode</i>	The error code as stored on <a href="#">VL53L0X_Error</a>
<i>pPalErrorString</i>	The error string corresponding to the PalErrorCode

#### Returns:

VL53L0X\_ERROR\_NONE Success

"Other error code" See [VL53L0X\\_Error](#)

**[VL53L0X\\_API](#) [VL53L0X\\_Error](#) VL53L0X\_GetPalStateString ([VL53L0X\\_State](#) *PalStateCode*,  
char \* *pPalStateString*)**

Human readable PAL State string.

**Note:**

This function doesn't access to the device

**Parameters:**

<i>PalStateCode</i>	The State code as stored on <i>VL53L0X_State</i>
<i>pPalStateString</i>	The State string corresponding to the PalStateCode

**Returns:**

VL53L0X\_ERROR\_NONE Success

"Other error code" See [VL53L0X\\_Error](#)

**[VL53L0X\\_API](#) [VL53L0X\\_Error](#) VL53L0X\_GetPalState ([VL53L0X\\_DEV](#) *Dev*, [VL53L0X\\_State](#) \*  
*pPalState*)**

Reads the internal state of the PAL for a given Device.

**Note:**

This function doesn't access to the device

**Parameters:**

<i>Dev</i>	Device Handle
<i>pPalState</i>	Pointer to current state of the PAL for a given Device

**Returns:**

VL53L0X\_ERROR\_NONE Success

"Other error code" See [VL53L0X\\_Error](#)

**[VL53L0X\\_API](#) [VL53L0X\\_Error](#) VL53L0X\_SetPowerMode ([VL53L0X\\_DEV](#) *Dev*,  
[VL53L0X\\_PowerModes](#) *PowerMode*)**

Set the power mode for a given Device The power mode can be Standby or Idle.

Different level of both Standby and Idle can exists. This function should not be used when device is in Ranging state.

**Note:**

This function Access to the device

**Parameters:**

<i>Dev</i>	Device Handle
<i>PowerMode</i>	The value of the power mode to set. see <a href="#">VL53L0X_PowerModes</a> Valid values are: VL53L0X_POWERMODE_STANDBY_LEVEL1, VL53L0X_POWERMODE_IDLE_LEVEL1

**Returns:**

VL53L0X\_ERROR\_NONE Success

VL53L0X\_ERROR\_MODE\_NOT\_SUPPORTED This error occurs when PowerMode is not in the supported list

"Other error code" See [VL53L0X\\_Error](#)

[VL53L0X API](#) [VL53L0X Error](#) [VL53L0X\\_GetPowerMode \(VL53L0X\\_DEV Dev, VL53L0X\\_PowerModes \\* pPowerMode\)](#)

Get the power mode for a given Device.

#### Note:

This function Access to the device

#### Parameters:

<i>Dev</i>	Device Handle
<i>pPowerMode</i>	Pointer to the current value of the power mode. see <a href="#">VL53L0X_PowerModes</a> Valid values are: VL53L0X_POWERMODE_STANDBY_LEVEL1, VL53L0X_POWERMODE_IDLE_LEVEL1

#### Returns:

VL53L0X\_ERROR\_NONE Success  
"Other error code" See [VL53L0X Error](#)

[VL53L0X API](#) [VL53L0X Error](#) [VL53L0X\\_SetOffsetCalibrationDataMicroMeter \(VL53L0X\\_DEV Dev, int32\\_t OffsetCalibrationDataMicroMeter\)](#)

Set or over-hide part to part calibration offset.

#### See also:

[VL53L0X\\_DataInit\(\)](#) [VL53L0X\\_GetOffsetCalibrationDataMicroMeter\(\)](#)

#### Note:

This function Access to the device

#### Parameters:

<i>Dev</i>	Device Handle
<i>OffsetCalibrationDataMicroMeter</i>	Offset (microns)

#### Returns:

VL53L0X\_ERROR\_NONE Success  
"Other error code" See [VL53L0X Error](#)

[VL53L0X API](#) [VL53L0X Error](#) [VL53L0X\\_GetOffsetCalibrationDataMicroMeter \(VL53L0X\\_DEV Dev, int32\\_t \\* pOffsetCalibrationDataMicroMeter\)](#)

Get part to part calibration offset.

#### Function Description

Should only be used after a successful call to *VL53L0X\_DataInit* to backup device NVM value

#### Note:

This function Access to the device

#### Parameters:

<i>Dev</i>	Device Handle
<i>pOffsetCalibrationDataMicroMeter</i>	Return part to part calibration offset from device (microns)

#### Returns:

VL53L0X\_ERROR\_NONE Success  
"Other error code" See [VL53L0X Error](#)

## [VL53L0X\\_API](#) [VL53L0X\\_Error](#) VL53L0X\_SetLinearityCorrectiveGain ([VL53L0X\\_DEV](#) Dev, [int16\\_t](#) LinearityCorrectiveGain)

Set the linearity corrective gain.

### Note:

This function Access to the device

### Parameters:

<i>Dev</i>	Device Handle
<i>LinearityCorrectiveGain</i>	Linearity corrective gain in x1000 if value is 1000 then no modification is applied.

### Returns:

VL53L0X\_ERROR\_NONE Success  
"Other error code" See [VL53L0X\\_Error](#)

## [VL53L0X\\_API](#) [VL53L0X\\_Error](#) VL53L0X\_GetLinearityCorrectiveGain ([VL53L0X\\_DEV](#) Dev, [uint16\\_t](#) \* pLinearityCorrectiveGain)

Get the linearity corrective gain.

### Function Description

Should only be used after a successful call to *VL53L0X\_DataInit* to backup device NVM value

### Note:

This function Access to the device

### Parameters:

<i>Dev</i>	Device Handle
<i>pLinearityCorrectiveGain</i>	Pointer to the linearity corrective gain in x1000 if value is 1000 then no modification is applied.

### Returns:

VL53L0X\_ERROR\_NONE Success  
"Other error code" See [VL53L0X\\_Error](#)

## [VL53L0X\\_API](#) [VL53L0X\\_Error](#) VL53L0X\_SetGroupParamHold ([VL53L0X\\_DEV](#) Dev, [uint8\\_t](#) GroupParamHold)

Set Group parameter Hold state.

### Function Description

Set or remove device internal group parameter hold

### Note:

This function is not Implemented

### Parameters:

<i>Dev</i>	Device Handle
<i>GroupParamHold</i>	Group parameter Hold state to be set (on/off)

### Returns:

VL53L0X\_ERROR\_NOT\_IMPLEMENTED Not implemented

[VL53L0X API](#) [VL53L0X Error](#) [VL53L0X\\_GetUpperLimitMilliMeter](#) ([VL53L0X\\_DEV](#) Dev, [uint16\\_t](#) \* [pUpperLimitMilliMeter](#))

Get the maximal distance for actual setup.

### Function Description

Device must be initialized through [VL53L0X\\_SetParameters\(\)](#) prior calling this function.  
Any range value more than the value returned is to be considered as "no target detected" or "no target in detectable range"

### Warning:

The maximal distance depends on the setup

### Note:

This function is not Implemented

### Parameters:

<i>Dev</i>	Device Handle
<i>pUpperLimitMilliMeter</i>	The maximal range limit for actual setup (in millimeter)

### Returns:

VL53L0X\_ERROR\_NOT\_IMPLEMENTED Not implemented

[VL53L0X Error](#) [VL53L0X\\_GetTotalSignalRate](#) ([VL53L0X\\_DEV](#) Dev, [FixPoint1616\\_t](#) \* [pTotalSignalRate](#))

Get the Total Signal Rate.

### Function Description

This function will return the Total Signal Rate after a good ranging is done.

### Note:

This function access to Device

### Parameters:

<i>Dev</i>	Device Handle
<i>pTotalSignalRate</i>	Total Signal Rate value in Mega count per second

### Returns:

VL53L0X\_ERROR\_NONE Success  
"Other error code" See [VL53L0X Error](#)

## VL53L0X Init Functions

VL53L0X Init Functions.

### Functions

- [VL53L0X API](#) [VL53L0X Error](#) [VL53L0X\\_SetDeviceAddress](#) ([VL53L0X\\_DEV](#) Dev, [uint8\\_t](#) DeviceAddress)  
*Set new device address.*
- [VL53L0X API](#) [VL53L0X Error](#) [VL53L0X\\_DataInit](#) ([VL53L0X\\_DEV](#) Dev)  
*One time device initialization.*

- [VL53L0X\\_API VL53L0X\\_Error VL53L0X\\_SetTuningSettingBuffer](#) ([VL53L0X\\_DEV](#) Dev, [uint8\\_t](#) \*pTuningSettingBuffer, [uint8\\_t](#) UseInternalTuningSettings)  
Set the tuning settings pointer.
- [VL53L0X\\_API VL53L0X\\_Error VL53L0X\\_GetTuningSettingBuffer](#) ([VL53L0X\\_DEV](#) Dev, [uint8\\_t](#) \*\*ppTuningSettingBuffer, [uint8\\_t](#) \*pUseInternalTuningSettings)  
Get the tuning settings pointer and the internal external switch value.
- [VL53L0X\\_API VL53L0X\\_Error VL53L0X\\_StaticInit](#) ([VL53L0X\\_DEV](#) Dev)  
Do basic device init (and eventually patch loading) This function will change the VL53L0X\_State from VL53L0X\_STATE\_WAIT\_STATICINIT to VL53L0X\_STATE\_IDLE.
- [VL53L0X\\_API VL53L0X\\_Error VL53L0X\\_WaitDeviceBooted](#) ([VL53L0X\\_DEV](#) Dev)  
Wait for device booted after chip enable (hardware standby) This function can be run only when VL53L0X\_State is VL53L0X\_STATE\_POWERDOWN.
- [VL53L0X\\_API VL53L0X\\_Error VL53L0X\\_ResetDevice](#) ([VL53L0X\\_DEV](#) Dev)  
Do an hard reset or soft reset (depending on implementation) of the device call of this function, device must be in same state as right after a power-up sequence. This function will change the VL53L0X\_State to VL53L0X\_STATE\_POWERDOWN.

## Detailed Description

VL53L0X Init Functions.

## Function Documentation

### [VL53L0X\\_API VL53L0X\\_Error VL53L0X\\_SetDeviceAddress](#) ([VL53L0X\\_DEV](#) Dev, [uint8\\_t](#) DeviceAddress)

Set new device address.

After completion the device will answer to the new address programmed. This function should be called when several devices are used in parallel before start programming the sensor. When a single device is used, there is no need to call this function.

#### Note:

This function Access to the device

#### Parameters:

<i>Dev</i>	Device Handle
<i>DeviceAddress</i>	The new Device address

#### Returns:

VL53L0X\_ERROR\_NONE Success  
"Other error code" See [VL53L0X\\_Error](#)

### [VL53L0X\\_API VL53L0X\\_Error VL53L0X\\_DataInit](#) ([VL53L0X\\_DEV](#) Dev)

One time device initialization.

To be called once and only once after device is brought out of reset (Chip enable) and booted see [VL53L0X\\_WaitDeviceBooted\(\)](#)

#### Function Description

When not used after a fresh device "power up" or reset, it may return [VL53L0X\\_ERROR\\_CALIBRATION\\_WARNING](#) meaning wrong calibration data may have been fetched from device that can result in ranging offset error



If application cannot execute device reset or need to run VL53L0X\_DataInit multiple time then it must ensure proper offset calibration saving and restore on its own by using VL53L0X\_GetOffsetCalibrationData() on first power up and then VL53L0X\_SetOffsetCalibrationData() in all subsequent init This function will change the VL53L0X\_State from VL53L0X\_STATE\_POWERDOWN to VL53L0X\_STATE\_WAIT\_STATICINIT.

**Note:**

This function Access to the device

**Parameters:**

<i>Dev</i>	Device Handle
------------	---------------

**Returns:**

VL53L0X\_ERROR\_NONE Success  
 "Other error code" See [VL53L0X\\_Error](#)

**[VL53L0X\\_API](#) [VL53L0X\\_Error](#) VL53L0X\_SetTuningSettingBuffer ([VL53L0X\\_DEV](#) *Dev*, [uint8\\_t](#) \* *pTuningSettingBuffer*, [uint8\\_t](#) *UseInternalTuningSettings*)**

Set the tuning settings pointer.

This function is used to specify the Tuning settings buffer to be used for a given device. The buffer contains all the necessary data to permit the API to write tuning settings. This function permit to force the usage of either external or internal tuning settings.

**Note:**

This function Access to the device

**Parameters:**

<i>Dev</i>	Device Handle
<i>pTuningSettingBuffer</i>	Pointer to tuning settings buffer.
<i>UseInternalTuningSettings</i>	Use internal tuning settings value.

**Returns:**

VL53L0X\_ERROR\_NONE Success  
 "Other error code" See [VL53L0X\\_Error](#)

**[VL53L0X\\_API](#) [VL53L0X\\_Error](#) VL53L0X\_GetTuningSettingBuffer ([VL53L0X\\_DEV](#) *Dev*, [uint8\\_t](#) \*\* *ppTuningSettingBuffer*, [uint8\\_t](#) \* *pUseInternalTuningSettings*)**

Get the tuning settings pointer and the internal external switch value.

This function is used to get the Tuning settings buffer pointer and the value. of the switch to select either external or internal tuning settings.

**Note:**

This function Access to the device

**Parameters:**

<i>Dev</i>	Device Handle
<i>ppTuningSettingBuffer</i>	Pointer to tuning settings buffer.
<i>pUseInternalTuningSettings</i>	Pointer to store Use internal tuning settings value.

**Returns:**

VL53L0X\_ERROR\_NONE Success  
 "Other error code" See [VL53L0X\\_Error](#)

## [VL53L0X\\_API](#) [VL53L0X\\_Error](#) VL53L0X\_StaticInit ([VL53L0X\\_DEV](#) Dev)

Do basic device init (and eventually patch loading) This function will change the VL53L0X\_State from VL53L0X\_STATE\_WAIT\_STATICINIT to VL53L0X\_STATE\_IDLE.

In this stage all default setting will be applied.

### Note:

This function Access to the device

### Parameters:

Dev	Device Handle
-----	---------------

### Returns:

VL53L0X\_ERROR\_NONE Success  
"Other error code" See [VL53L0X\\_Error](#)

## [VL53L0X\\_API](#) [VL53L0X\\_Error](#) VL53L0X\_WaitDeviceBooted ([VL53L0X\\_DEV](#) Dev)

Wait for device booted after chip enable (hardware standby) This function can be run only when VL53L0X\_State is VL53L0X\_STATE\_POWERDOWN.

### Note:

This function is not Implemented

### Parameters:

Dev	Device Handle
-----	---------------

### Returns:

VL53L0X\_ERROR\_NOT\_IMPLEMENTED Not implemented

## [VL53L0X\\_API](#) [VL53L0X\\_Error](#) VL53L0X\_ResetDevice ([VL53L0X\\_DEV](#) Dev)

Do an hard reset or soft reset (depending on implementation) of the device call of this function, device must be in same state as right after a power-up sequence. This function will change the VL53L0X\_State to VL53L0X\_STATE\_POWERDOWN.

### Note:

This function Access to the device

### Parameters:

Dev	Device Handle
-----	---------------

### Returns:

VL53L0X\_ERROR\_NONE Success  
"Other error code" See [VL53L0X\\_Error](#)

## VL53L0X Parameters Functions

Functions used to prepare and setup the device.

### Functions

- [VL53L0X\\_API](#) [VL53L0X\\_Error](#) [VL53L0X\\_SetDeviceParameters](#) ([VL53L0X\\_DEV](#) Dev, const [VL53L0X\\_DeviceParameters\\_t](#) \*pDeviceParameters)  
*Prepare device for operation.*

- [VL53L0X\\_API\\_VL53L0X\\_Error\\_VL53L0X\\_GetDeviceParameters](#) ([VL53L0X\\_DEV](#) Dev, [VL53L0X\\_DeviceParameters\\_t](#) \*pDeviceParameters)  
*Retrieve current device parameters.*
- [VL53L0X\\_API\\_VL53L0X\\_Error\\_VL53L0X\\_SetDeviceMode](#) ([VL53L0X\\_DEV](#) Dev, [VL53L0X\\_DeviceModes](#) DeviceMode)  
*Set a new device mode.*
- [VL53L0X\\_API\\_VL53L0X\\_Error\\_VL53L0X\\_GetDeviceMode](#) ([VL53L0X\\_DEV](#) Dev, [VL53L0X\\_DeviceModes](#) \*pDeviceMode)  
*Get current new device mode.*
- [VL53L0X\\_API\\_VL53L0X\\_Error\\_VL53L0X\\_SetRangeFractionEnable](#) ([VL53L0X\\_DEV](#) Dev, [uint8\\_t](#) Enable)  
*Sets the resolution of range measurements.*
- [VL53L0X\\_API\\_VL53L0X\\_Error\\_VL53L0X\\_GetFractionEnable](#) ([VL53L0X\\_DEV](#) Dev, [uint8\\_t](#) \*pEnable)  
*Gets the fraction enable parameter indicating the resolution of range measurements.*
- [VL53L0X\\_API\\_VL53L0X\\_Error\\_VL53L0X\\_SetHistogramMode](#) ([VL53L0X\\_DEV](#) Dev, [VL53L0X\\_HistogramModes](#) HistogramMode)  
*Set a new Histogram mode.*
- [VL53L0X\\_API\\_VL53L0X\\_Error\\_VL53L0X\\_GetHistogramMode](#) ([VL53L0X\\_DEV](#) Dev, [VL53L0X\\_HistogramModes](#) \*pHistogramMode)  
*Get current new device mode.*
- [VL53L0X\\_API\\_VL53L0X\\_Error\\_VL53L0X\\_SetMeasurementTimingBudgetMicroSeconds](#) ([VL53L0X\\_DEV](#) Dev, [uint32\\_t](#) MeasurementTimingBudgetMicroSeconds)  
*Set Ranging Timing Budget in microseconds.*
- [VL53L0X\\_API\\_VL53L0X\\_Error\\_VL53L0X\\_GetMeasurementTimingBudgetMicroSeconds](#) ([VL53L0X\\_DEV](#) Dev, [uint32\\_t](#) \*pMeasurementTimingBudgetMicroSeconds)  
*Get Ranging Timing Budget in microseconds.*
- [VL53L0X\\_API\\_VL53L0X\\_Error\\_VL53L0X\\_GetVcselPulsePeriod](#) ([VL53L0X\\_DEV](#) Dev, [VL53L0X\\_VcselPeriod](#) VcselPeriodType, [uint8\\_t](#) \*pVCSELPulsePeriod)  
*Gets the VCSEL pulse period.*
- [VL53L0X\\_API\\_VL53L0X\\_Error\\_VL53L0X\\_SetVcselPulsePeriod](#) ([VL53L0X\\_DEV](#) Dev, [VL53L0X\\_VcselPeriod](#) VcselPeriodType, [uint8\\_t](#) VCSELPulsePeriod)  
*Sets the VCSEL pulse period.*
- [VL53L0X\\_API\\_VL53L0X\\_Error\\_VL53L0X\\_SetSequenceStepEnable](#) ([VL53L0X\\_DEV](#) Dev, [VL53L0X\\_SequenceStepId](#) SequenceStepId, [uint8\\_t](#) SequenceStepEnabled)  
*Sets the (on/off) state of a requested sequence step.*
- [VL53L0X\\_API\\_VL53L0X\\_Error\\_VL53L0X\\_GetSequenceStepEnable](#) ([VL53L0X\\_DEV](#) Dev, [VL53L0X\\_SequenceStepId](#) SequenceStepId, [uint8\\_t](#) \*pSequenceStepEnabled)  
*Gets the (on/off) state of a requested sequence step.*
- [VL53L0X\\_API\\_VL53L0X\\_Error\\_VL53L0X\\_GetSequenceStepEnables](#) ([VL53L0X\\_DEV](#) Dev, [VL53L0X\\_SchedulerSequenceSteps\\_t](#) \*pSchedulerSequenceSteps)  
*Gets the (on/off) state of all sequence steps.*
- [VL53L0X\\_API\\_VL53L0X\\_Error\\_VL53L0X\\_SetSequenceStepTimeout](#) ([VL53L0X\\_DEV](#) Dev, [VL53L0X\\_SequenceStepId](#) SequenceStepId, [FixPoint1616\\_t](#) TimeOutMilliSecs)  
*Sets the timeout of a requested sequence step.*
- [VL53L0X\\_API\\_VL53L0X\\_Error\\_VL53L0X\\_GetSequenceStepTimeout](#) ([VL53L0X\\_DEV](#) Dev, [VL53L0X\\_SequenceStepId](#) SequenceStepId, [FixPoint1616\\_t](#) \*pTimeOutMilliSecs)  
*Gets the timeout of a requested sequence step.*
- [VL53L0X\\_API\\_VL53L0X\\_Error\\_VL53L0X\\_GetNumberOfSequenceSteps](#) ([VL53L0X\\_DEV](#) Dev, [uint8\\_t](#) \*pNumberOfSequenceSteps)  
*Gets number of sequence steps managed by the API.*
- [VL53L0X\\_API\\_VL53L0X\\_Error\\_VL53L0X\\_GetSequenceStepsInfo](#) ([VL53L0X\\_SequenceStepId](#) SequenceStepId, char \*pSequenceStepsString)

*Gets the name of a given sequence step.*

- [VL53L0X\\_API\\_VL53L0X\\_Error\\_VL53L0X\\_SetInterMeasurementPeriodMilliseconds](#) ([VL53L0X\\_DEV](#) Dev, [uint32\\_t](#) InterMeasurementPeriodMilliseconds)  
*Program continuous mode Inter-Measurement period in milliseconds.*
- [VL53L0X\\_API\\_VL53L0X\\_Error\\_VL53L0X\\_GetInterMeasurementPeriodMilliseconds](#) ([VL53L0X\\_DEV](#) Dev, [uint32\\_t](#) \*pInterMeasurementPeriodMilliseconds)  
*Get continuous mode Inter-Measurement period in milliseconds.*
- [VL53L0X\\_API\\_VL53L0X\\_Error\\_VL53L0X\\_SetXTalkCompensationEnable](#) ([VL53L0X\\_DEV](#) Dev, [uint8\\_t](#) XTalkCompensationEnable)  
*Enable/Disable Cross talk compensation feature.*
- [VL53L0X\\_API\\_VL53L0X\\_Error\\_VL53L0X\\_GetXTalkCompensationEnable](#) ([VL53L0X\\_DEV](#) Dev, [uint8\\_t](#) \*pXTalkCompensationEnable)  
*Get Cross talk compensation rate.*
- [VL53L0X\\_API\\_VL53L0X\\_Error\\_VL53L0X\\_SetXTalkCompensationRateMegaCps](#) ([VL53L0X\\_DEV](#) Dev, [FixPoint1616\\_t](#) XTalkCompensationRateMegaCps)  
*Set Cross talk compensation rate.*
- [VL53L0X\\_API\\_VL53L0X\\_Error\\_VL53L0X\\_GetXTalkCompensationRateMegaCps](#) ([VL53L0X\\_DEV](#) Dev, [FixPoint1616\\_t](#) \*pXTalkCompensationRateMegaCps)  
*Get Cross talk compensation rate.*
- [VL53L0X\\_API\\_VL53L0X\\_Error\\_VL53L0X\\_SetRefCalibration](#) ([VL53L0X\\_DEV](#) Dev, [uint8\\_t](#) VhvSettings, [uint8\\_t](#) PhaseCal)  
*Set Reference Calibration Parameters.*
- [VL53L0X\\_API\\_VL53L0X\\_Error\\_VL53L0X\\_GetRefCalibration](#) ([VL53L0X\\_DEV](#) Dev, [uint8\\_t](#) \*pVhvSettings, [uint8\\_t](#) \*pPhaseCal)  
*Get Reference Calibration Parameters.*
- [VL53L0X\\_API\\_VL53L0X\\_Error\\_VL53L0X\\_GetNumberOfLimitCheck](#) ([uint16\\_t](#) \*pNumberOfLimitCheck)  
*Get the number of the check limit managed by a given Device.*
- [VL53L0X\\_API\\_VL53L0X\\_Error\\_VL53L0X\\_GetLimitCheckInfo](#) ([VL53L0X\\_DEV](#) Dev, [uint16\\_t](#) LimitCheckId, char \*pLimitCheckString)  
*Return a description string for a given limit check number.*
- [VL53L0X\\_API\\_VL53L0X\\_Error\\_VL53L0X\\_GetLimitCheckStatus](#) ([VL53L0X\\_DEV](#) Dev, [uint16\\_t](#) LimitCheckId, [uint8\\_t](#) \*pLimitCheckStatus)  
*Return a the Status of the specified check limit.*
- [VL53L0X\\_API\\_VL53L0X\\_Error\\_VL53L0X\\_SetLimitCheckEnable](#) ([VL53L0X\\_DEV](#) Dev, [uint16\\_t](#) LimitCheckId, [uint8\\_t](#) LimitCheckEnable)  
*Enable/Disable a specific limit check.*
- [VL53L0X\\_API\\_VL53L0X\\_Error\\_VL53L0X\\_GetLimitCheckEnable](#) ([VL53L0X\\_DEV](#) Dev, [uint16\\_t](#) LimitCheckId, [uint8\\_t](#) \*pLimitCheckEnable)  
*Get specific limit check enable state.*
- [VL53L0X\\_API\\_VL53L0X\\_Error\\_VL53L0X\\_SetLimitCheckValue](#) ([VL53L0X\\_DEV](#) Dev, [uint16\\_t](#) LimitCheckId, [FixPoint1616\\_t](#) LimitCheckValue)  
*Set a specific limit check value.*
- [VL53L0X\\_API\\_VL53L0X\\_Error\\_VL53L0X\\_GetLimitCheckValue](#) ([VL53L0X\\_DEV](#) Dev, [uint16\\_t](#) LimitCheckId, [FixPoint1616\\_t](#) \*pLimitCheckValue)  
*Get a specific limit check value.*
- [VL53L0X\\_API\\_VL53L0X\\_Error\\_VL53L0X\\_GetLimitCheckCurrent](#) ([VL53L0X\\_DEV](#) Dev, [uint16\\_t](#) LimitCheckId, [FixPoint1616\\_t](#) \*pLimitCheckCurrent)  
*Get the current value of the signal used for the limit check.*
- [VL53L0X\\_API\\_VL53L0X\\_Error\\_VL53L0X\\_SetWrapAroundCheckEnable](#) ([VL53L0X\\_DEV](#) Dev, [uint8\\_t](#) WrapAroundCheckEnable)  
*Enable (or disable) Wrap around Check.*

- [VL53L0X\\_API\\_VL53L0X\\_Error\\_VL53L0X\\_GetWrapAroundCheckEnable](#) ([VL53L0X\\_DEV](#) Dev, [uint8\\_t](#) \*pWrapAroundCheckEnable)  
*Get setup of Wrap around Check.*
- [VL53L0X\\_API\\_VL53L0X\\_Error\\_VL53L0X\\_SetDmaxCalParameters](#) ([VL53L0X\\_DEV](#) Dev, [uint16\\_t](#) RangeMilliMeter, [FixPoint1616\\_t](#) SignalRateRtnMegaCps)  
*Set Dmax Calibration Parameters for a given device When one of the parameter is zero, this function will get parameter from NVM.*
- [VL53L0X\\_API\\_VL53L0X\\_Error\\_VL53L0X\\_GetDmaxCalParameters](#) ([VL53L0X\\_DEV](#) Dev, [uint16\\_t](#) \*pRangeMilliMeter, [FixPoint1616\\_t](#) \*pSignalRateRtnMegaCps)  
*Get Dmax Calibration Parameters for a given device.*

## Detailed Description

Functions used to prepare and setup the device.

## Function Documentation

[VL53L0X\\_API\\_VL53L0X\\_Error\\_VL53L0X\\_SetDeviceParameters](#) ([VL53L0X\\_DEV](#) Dev, const [VL53L0X\\_DeviceParameters\\_t](#) \* pDeviceParameters)

Prepare device for operation.

### Function Description

Update device with provided parameters

- Then start ranging operation.

### Note:

This function Access to the device

### Parameters:

<i>Dev</i>	Device Handle
<i>pDeviceParameters</i>	Pointer to store current device parameters.

### Returns:

VL53L0X\_ERROR\_NONE Success  
"Other error code" See [VL53L0X\\_Error](#)

[VL53L0X\\_API\\_VL53L0X\\_Error\\_VL53L0X\\_GetDeviceParameters](#) ([VL53L0X\\_DEV](#) Dev, [VL53L0X\\_DeviceParameters\\_t](#) \* pDeviceParameters)

Retrieve current device parameters.

### Function Description

Get actual parameters of the device

- Then start ranging operation.

### Note:

This function Access to the device

### Parameters:

<i>Dev</i>	Device Handle
------------	---------------

<i>pDeviceParameters</i>	Pointer to store current device parameters.
--------------------------	---

#### Returns:

VL53L0X\_ERROR\_NONE Success  
 "Other error code" See [VL53L0X\\_Error](#)

[VL53L0X\\_API](#) [VL53L0X\\_Error](#) [VL53L0X\\_SetDeviceMode](#) ([VL53L0X\\_DEV](#) *Dev*,  
[VL53L0X\\_DeviceModes](#) *DeviceMode*)

Set a new device mode.

#### Function Description

Set device to a new mode (ranging, histogram ...)

#### Note:

This function doesn't Access to the device

#### Parameters:

<i>Dev</i>	Device Handle
<i>DeviceMode</i>	New device mode to apply Valid values are: VL53L0X_DEVICEMODE_SINGLE_RANGING VL53L0X_DEVICEMODE_CONTINUOUS_RANGING VL53L0X_DEVICEMODE_CONTINUOUS_TIMED_RANGING VL53L0X_DEVICEMODE_SINGLE_HISTOGRAM VL53L0X_HISTOGRAMMODE_REFERENCE_ONLY VL53L0X_HISTOGRAMMODE_RETURN_ONLY VL53L0X_HISTOGRAMMODE_BOTH

#### Returns:

VL53L0X\_ERROR\_NONE Success  
 VL53L0X\_ERROR\_MODE\_NOT\_SUPPORTED This error occurs when DeviceMode is not in the supported list

[VL53L0X\\_API](#) [VL53L0X\\_Error](#) [VL53L0X\\_GetDeviceMode](#) ([VL53L0X\\_DEV](#) *Dev*,  
[VL53L0X\\_DeviceModes](#) \* *pDeviceMode*)

Get current new device mode.

#### Function Description

Get actual mode of the device(ranging, histogram ...)

#### Note:

This function doesn't Access to the device

#### Parameters:

<i>Dev</i>	Device Handle
<i>pDeviceMode</i>	Pointer to current apply mode value Valid values are: VL53L0X_DEVICEMODE_SINGLE_RANGING VL53L0X_DEVICEMODE_CONTINUOUS_RANGING VL53L0X_DEVICEMODE_CONTINUOUS_TIMED_RANGING VL53L0X_DEVICEMODE_SINGLE_HISTOGRAM VL53L0X_HISTOGRAMMODE_REFERENCE_ONLY VL53L0X_HISTOGRAMMODE_RETURN_ONLY VL53L0X_HISTOGRAMMODE_BOTH

#### Returns:

VL53L0X\_ERROR\_NONE Success

**VL53L0X\_ERROR\_MODE\_NOT\_SUPPORTED** This error occurs when DeviceMode is not in the supported list

### [VL53L0X\\_API](#) [VL53L0X\\_Error](#) **VL53L0X\_SetRangeFractionEnable** ([VL53L0X\\_DEV](#) *Dev*, [uint8\\_t](#) *Enable*)

Sets the resolution of range measurements.

#### Function Description

Set resolution of range measurements to either 0.25mm if fraction enabled or 1mm if not enabled.

#### Note:

This function Accesses the device

#### Parameters:

<i>Dev</i>	Device Handle
<i>Enable</i>	Enable high resolution

#### Returns:

VL53L0X\_ERROR\_NONE Success  
"Other error code" See [VL53L0X\\_Error](#)

### [VL53L0X\\_API](#) [VL53L0X\\_Error](#) **VL53L0X\_GetFractionEnable** ([VL53L0X\\_DEV](#) *Dev*, [uint8\\_t](#) \* *pEnable*)

Gets the fraction enable parameter indicating the resolution of range measurements.

#### Function Description

Gets the fraction enable state, which translates to the resolution of range measurements as follows  
:Enabled:=0.25mm resolution, Not Enabled:=1mm resolution.

#### Note:

This function Accesses the device

#### Parameters:

<i>Dev</i>	Device Handle
<i>pEnable</i>	Output Parameter reporting the fraction enable state.

#### Returns:

VL53L0X\_ERROR\_NONE Success  
"Other error code" See [VL53L0X\\_Error](#)

### [VL53L0X\\_API](#) [VL53L0X\\_Error](#) **VL53L0X\_SetHistogramMode** ([VL53L0X\\_DEV](#) *Dev*, [VL53L0X\\_HistogramModes](#) *HistogramMode*)

Set a new Histogram mode.

#### Function Description

Set device to a new Histogram mode

#### Note:

This function doesn't Access to the device

#### Parameters:

<i>Dev</i>	Device Handle
<i>HistogramMode</i>	New device mode to apply Valid values are: VL53L0X_HISTOGRAMMODE_DISABLED

	VL53L0X_DEVICEMODE_SINGLE_HISTOGRAM VL53L0X_HISTOGRAMMODE_REFERENCE_ONLY VL53L0X_HISTOGRAMMODE_RETURN_ONLY VL53L0X_HISTOGRAMMODE_BOTH
--	--

#### Returns:

VL53L0X\_ERROR\_NONE Success

VL53L0X\_ERROR\_MODE\_NOT\_SUPPORTED This error occurs when HistogramMode is not in the supported list

"Other error code" See [VL53L0X\\_Error](#)

[VL53L0X\\_API](#) [VL53L0X\\_Error](#) **VL53L0X\_GetHistogramMode** ([VL53L0X\\_DEV](#) *Dev*, [VL53L0X\\_HistogramModes](#) \* *pHistogramMode*)

Get current new device mode.

#### Function Description

Get current Histogram mode of a Device

#### Note:

This function doesn't Access to the device

#### Parameters:

<i>Dev</i>	Device Handle
<i>pHistogramMode</i>	Pointer to current Histogram Mode value Valid values are: VL53L0X_HISTOGRAMMODE_DISABLED VL53L0X_DEVICEMODE_SINGLE_HISTOGRAM VL53L0X_HISTOGRAMMODE_REFERENCE_ONLY VL53L0X_HISTOGRAMMODE_RETURN_ONLY VL53L0X_HISTOGRAMMODE_BOTH

#### Returns:

VL53L0X\_ERROR\_NONE Success

"Other error code" See [VL53L0X\\_Error](#)

[VL53L0X\\_API](#) [VL53L0X\\_Error](#) **VL53L0X\_SetMeasurementTimingBudgetMicroSeconds** ([VL53L0X\\_DEV](#) *Dev*, [uint32\\_t](#) *MeasurementTimingBudgetMicroSeconds*)

Set Ranging Timing Budget in microseconds.

#### Function Description

Defines the maximum time allowed by the user to the device to run a full ranging sequence for the current mode (ranging, histogram, ASL ...)

#### Note:

This function Access to the device

#### Parameters:

<i>Dev</i>	Device Handle
<i>MeasurementTimingBudgetMicroSeconds</i>	Max measurement time in microseconds. Valid values are: >= 17000 microsecs when wraparound enabled >= 12000 microsecs when wraparound disabled

#### Returns:

VL53L0X\_ERROR\_NONE Success

VL53L0X\_ERROR\_INVALID\_PARAMS This error is returned if MeasurementTimingBudgetMicroSeconds out of range

"Other error code" See [VL53L0X\\_Error](#)



[VL53L0X\\_API](#) [VL53L0X\\_Error](#) [VL53L0X\\_GetMeasurementTimingBudgetMicroSeconds](#)  
([VL53L0X\\_DEV](#) *Dev*, [uint32\\_t](#) \* *pMeasurementTimingBudgetMicroSeconds*)

Get Ranging Timing Budget in microseconds.

### Function Description

Returns the programmed the maximum time allowed by the user to the device to run a full ranging sequence for the current mode (ranging, histogram, ASL ...)

### Note:

This function Access to the device

### Parameters:

<i>Dev</i>	Device Handle
<i>pMeasurementTimingBudgetMicroSeconds</i>	Max measurement time in microseconds. Valid values are: >= 17000 microseconds when wraparound enabled >= 12000 microseconds when wraparound disabled

### Returns:

VL53L0X\_ERROR\_NONE Success  
"Other error code" See [VL53L0X\\_Error](#)

[VL53L0X\\_API](#) [VL53L0X\\_Error](#) [VL53L0X\\_GetVcselPulsePeriod](#) ([VL53L0X\\_DEV](#) *Dev*,  
[VL53L0X\\_VcselPeriod](#) *VcselPeriodType*, [uint8\\_t](#) \* *pVCSELPulsePeriod*)

Gets the VCSEL pulse period.

### Function Description

This function retrieves the VCSEL pulse period for the given period type.

### Note:

This function Accesses the device

### Parameters:

<i>Dev</i>	Device Handle
<i>VcselPeriodType</i>	VCSEL period identifier (pre-range final).
<i>pVCSELPulsePeriod</i>	Pointer to VCSEL period value.

### Returns:

VL53L0X\_ERROR\_NONE Success  
VL53L0X\_ERROR\_INVALID\_PARAMS Error VcselPeriodType parameter not supported.  
"Other error code" See [VL53L0X\\_Error](#)

[VL53L0X\\_API](#) [VL53L0X\\_Error](#) [VL53L0X\\_SetVcselPulsePeriod](#) ([VL53L0X\\_DEV](#) *Dev*,  
[VL53L0X\\_VcselPeriod](#) *VcselPeriodType*, [uint8\\_t](#) *VCSELPulsePeriod*)

Sets the VCSEL pulse period.

### Function Description

This function retrieves the VCSEL pulse period for the given period type.

### Note:

This function Accesses the device

### Parameters:

<i>Dev</i>	Device Handle
------------	---------------

<i>VcsePeriodType</i>	VCSEL period identifier (pre-range final).
<i>VCSELPulsePeriod</i>	VCSEL period value

#### Returns:

VL53L0X\_ERROR\_NONE Success  
 VL53L0X\_ERROR\_INVALID\_PARAMS Error VcsePeriodType parameter not supported.  
 "Other error code" See [VL53L0X\\_Error](#)

[VL53L0X\\_API](#) [VL53L0X\\_Error](#) [VL53L0X\\_SetSequenceStepEnable](#) ([VL53L0X\\_DEV](#) *Dev*,  
[VL53L0X\\_SequenceStepId](#) *SequenceStepId*, [uint8\\_t](#) *SequenceStepEnabled*)

Sets the (on/off) state of a requested sequence step.

#### Function Description

This function enables/disables a requested sequence step.

#### Note:

This function Accesses the device

#### Parameters:

<i>Dev</i>	Device Handle
<i>SequenceStepId</i>	Sequence step identifier.
<i>SequenceStepEnabled</i>	Demanded state {0=Off,1=On} is enabled.

#### Returns:

VL53L0X\_ERROR\_NONE Success  
 VL53L0X\_ERROR\_INVALID\_PARAMS Error SequenceStepId parameter not supported.  
 "Other error code" See [VL53L0X\\_Error](#)

[VL53L0X\\_API](#) [VL53L0X\\_Error](#) [VL53L0X\\_GetSequenceStepEnable](#) ([VL53L0X\\_DEV](#) *Dev*,  
[VL53L0X\\_SequenceStepId](#) *SequenceStepId*, [uint8\\_t](#) \* *pSequenceStepEnabled*)

Gets the (on/off) state of a requested sequence step.

#### Function Description

This function retrieves the state of a requested sequence step, i.e. on/off.

#### Note:

This function Accesses the device

#### Parameters:

<i>Dev</i>	Device Handle
<i>SequenceStepId</i>	Sequence step identifier.
<i>pSequenceStepEnabled</i>	Out parameter reporting if the sequence step is enabled {0=Off,1=On}.

#### Returns:

VL53L0X\_ERROR\_NONE Success  
 VL53L0X\_ERROR\_INVALID\_PARAMS Error SequenceStepId parameter not supported.  
 "Other error code" See [VL53L0X\\_Error](#)

[VL53L0X\\_API](#) [VL53L0X\\_Error](#) [VL53L0X\\_GetSequenceStepEnables](#) ([VL53L0X\\_DEV](#) *Dev*,  
[VL53L0X\\_SchedulerSequenceSteps\\_t](#) \* *pSchedulerSequenceSteps*)

Gets the (on/off) state of all sequence steps.

## Function Description

This function retrieves the state of all sequence step in the scheduler.

### Note:

This function Accesses the device

### Parameters:

<i>Dev</i>	Device Handle
<i>pSchedulerSequenceSteps</i>	Pointer to struct containing result.

### Returns:

VL53L0X\_ERROR\_NONE Success  
 "Other error code" See [VL53L0X\\_Error](#)

[VL53L0X\\_API](#) [VL53L0X\\_Error](#) [VL53L0X\\_SetSequenceStepTimeout](#) ([VL53L0X\\_DEV](#) *Dev*,  
[VL53L0X\\_SequenceStepId](#) *SequenceStepId*, [FixPoint1616\\_t](#) *TimeOutMilliSecs*)

Sets the timeout of a requested sequence step.

## Function Description

This function sets the timeout of a requested sequence step.

### Note:

This function Accesses the device

### Parameters:

<i>Dev</i>	Device Handle
<i>SequenceStepId</i>	Sequence step identifier.
<i>TimeOutMilliSecs</i>	Demanded timeout

### Returns:

VL53L0X\_ERROR\_NONE Success  
 VL53L0X\_ERROR\_INVALID\_PARAMS Error SequenceStepId parameter not supported.  
 "Other error code" See [VL53L0X\\_Error](#)

[VL53L0X\\_API](#) [VL53L0X\\_Error](#) [VL53L0X\\_GetSequenceStepTimeout](#) ([VL53L0X\\_DEV](#) *Dev*,  
[VL53L0X\\_SequenceStepId](#) *SequenceStepId*, [FixPoint1616\\_t](#) \* *pTimeOutMilliSecs*)

Gets the timeout of a requested sequence step.

## Function Description

This function retrieves the timeout of a requested sequence step.

### Note:

This function Accesses the device

### Parameters:

<i>Dev</i>	Device Handle
<i>SequenceStepId</i>	Sequence step identifier.
<i>pTimeOutMilliSecs</i>	Timeout value.

### Returns:

VL53L0X\_ERROR\_NONE Success  
 VL53L0X\_ERROR\_INVALID\_PARAMS Error SequenceStepId parameter not supported.  
 "Other error code" See [VL53L0X\\_Error](#)

[VL53L0X API](#) [VL53L0X Error](#) [VL53L0X\\_GetNumberOfSequenceSteps](#) ([VL53L0X\\_DEV](#) *Dev*, [uint8\\_t](#) \* *pNumberOfSequenceSteps*)

Gets number of sequence steps managed by the API.

### Function Description

This function retrieves the number of sequence steps currently managed by the API

#### Note:

This function Accesses the device

#### Parameters:

<i>Dev</i>	Device Handle
<i>pNumberOfSequenceSteps</i>	Out parameter reporting the number of sequence steps.

#### Returns:

VL53L0X\_ERROR\_NONE Success  
"Other error code" See [VL53L0X\\_Error](#)

[VL53L0X API](#) [VL53L0X Error](#) [VL53L0X\\_GetSequenceStepsInfo](#) ([VL53L0X\\_SequenceStepId](#) *SequenceStepId*, [char](#) \* *pSequenceStepsString*)

Gets the name of a given sequence step.

### Function Description

This function retrieves the name of sequence steps corresponding to SequenceStepId.

#### Note:

This function doesn't Accesses the device

#### Parameters:

<i>SequenceStepId</i>	Sequence step identifier.
<i>pSequenceStepsString</i>	Pointer to Info string

#### Returns:

VL53L0X\_ERROR\_NONE Success  
"Other error code" See [VL53L0X\\_Error](#)

[VL53L0X API](#) [VL53L0X Error](#) [VL53L0X\\_SetInterMeasurementPeriodMilliseconds](#) ([VL53L0X\\_DEV](#) *Dev*, [uint32\\_t](#) *InterMeasurementPeriodMilliseconds*)

Program continuous mode Inter-Measurement period in milliseconds.

### Function Description

When trying to set too short time return INVALID\_PARAMS minimal value

#### Note:

This function Access to the device

#### Parameters:

<i>Dev</i>	Device Handle
<i>InterMeasurementPeriodMilliseconds</i>	Inter-Measurement Period in ms.

#### Returns:

VL53L0X\_ERROR\_NONE Success  
 "Other error code" See [VL53L0X\\_Error](#)

[VL53L0X\\_API](#) [VL53L0X\\_Error](#) VL53L0X\_GetInterMeasurementPeriodMilliseconds  
 (VL53L0X\_DEV Dev, uint32\_t \* pInterMeasurementPeriodMilliseconds)

Get continuous mode Inter-Measurement period in milliseconds.

#### Function Description

When trying to set too short time return INVALID\_PARAMS minimal value

#### Note:

This function Access to the device

#### Parameters:

Dev	Device Handle
pInterMeasurementPeriodMilliseconds	Pointer to programmed Inter-Measurement Period in milliseconds.

#### Returns:

VL53L0X\_ERROR\_NONE Success  
 "Other error code" See [VL53L0X\\_Error](#)

[VL53L0X\\_API](#) [VL53L0X\\_Error](#) VL53L0X\_SetXTalkCompensationEnable (VL53L0X\_DEV Dev, uint8\_t XTalkCompensationEnable)

Enable/Disable Cross talk compensation feature.

#### Note:

This function is not Implemented. Enable/Disable Cross Talk by set to zero the Cross Talk value by using [VL53L0X\\_SetXTalkCompensationRateMegaCps\(\)](#).

#### Parameters:

Dev	Device Handle
XTalkCompensationEnable	Cross talk compensation to be set 0=disabled else = enabled

#### Returns:

VL53L0X\_ERROR\_NOT\_IMPLEMENTED Not implemented

[VL53L0X\\_API](#) [VL53L0X\\_Error](#) VL53L0X\_GetXTalkCompensationEnable (VL53L0X\_DEV Dev, uint8\_t \* pXTalkCompensationEnable)

Get Cross talk compensation rate.

#### Note:

This function is not Implemented. Enable/Disable Cross Talk by set to zero the Cross Talk value by using [VL53L0X\\_SetXTalkCompensationRateMegaCps\(\)](#).

#### Parameters:

Dev	Device Handle
pXTalkCompensationEnable	Pointer to the Cross talk compensation state 0=disabled or 1 = enabled

#### Returns:

VL53L0X\_ERROR\_NOT\_IMPLEMENTED Not implemented

[VL53L0X\\_API](#) [VL53L0X\\_Error](#) VL53L0X\_SetXTalkCompensationRateMegaCps ([VL53L0X\\_DEV](#) Dev, [FixPoint1616\\_t](#) XTalkCompensationRateMegaCps)

Set Cross talk compensation rate.

#### Function Description

Set Cross talk compensation rate.

#### Note:

This function Access to the device

#### Parameters:

<i>Dev</i>	Device Handle
<i>XTalkCompensationRateMegaCps</i>	Compensation rate in Mega counts per second (16.16 fix point) see datasheet for details

#### Returns:

VL53L0X\_ERROR\_NONE Success  
"Other error code" See [VL53L0X\\_Error](#)

[VL53L0X\\_API](#) [VL53L0X\\_Error](#) VL53L0X\_GetXTalkCompensationRateMegaCps ([VL53L0X\\_DEV](#) Dev, [FixPoint1616\\_t](#) \* pXTalkCompensationRateMegaCps)

Get Cross talk compensation rate.

#### Function Description

Get Cross talk compensation rate.

#### Note:

This function Access to the device

#### Parameters:

<i>Dev</i>	Device Handle
<i>pXTalkCompensationRateMegaCps</i>	Pointer to Compensation rate in Mega counts per second (16.16 fix point) see datasheet for details

#### Returns:

VL53L0X\_ERROR\_NONE Success  
"Other error code" See [VL53L0X\\_Error](#)

[VL53L0X\\_API](#) [VL53L0X\\_Error](#) VL53L0X\_SetRefCalibration ([VL53L0X\\_DEV](#) Dev, [uint8\\_t](#) VhvSettings, [uint8\\_t](#) PhaseCal)

Set Reference Calibration Parameters.

#### Function Description

Set Reference Calibration Parameters.

#### Note:

This function Access to the device

#### Parameters:

<i>Dev</i>	Device Handle
<i>VhvSettings</i>	Parameter for VHV

<i>PhaseCal</i>	Parameter for PhaseCal
-----------------	------------------------

#### Returns:

VL53L0X\_ERROR\_NONE Success  
 "Other error code" See [VL53L0X\\_Error](#)

[VL53L0X\\_API](#) [VL53L0X\\_Error](#) VL53L0X\_GetRefCalibration ([VL53L0X\\_DEV](#) *Dev*, [uint8\\_t](#) \* *pVhvSettings*, [uint8\\_t](#) \* *pPhaseCal*)

Get Reference Calibration Parameters.

#### Function Description

Get Reference Calibration Parameters.

#### Note:

This function Access to the device

#### Parameters:

<i>Dev</i>	Device Handle
<i>pVhvSettings</i>	Pointer to VHV parameter
<i>pPhaseCal</i>	Pointer to PhaseCal Parameter

#### Returns:

VL53L0X\_ERROR\_NONE Success  
 "Other error code" See [VL53L0X\\_Error](#)

[VL53L0X\\_API](#) [VL53L0X\\_Error](#) VL53L0X\_GetNumberOfLimitCheck ([uint16\\_t](#) \* *pNumberOfLimitCheck*)

Get the number of the check limit managed by a given Device.

#### Function Description

This function give the number of the check limit managed by the Device

#### Note:

This function doesn't Access to the device

#### Parameters:

<i>pNumberOfLimitCheck</i>	Pointer to the number of check limit.
----------------------------	---------------------------------------

#### Returns:

VL53L0X\_ERROR\_NONE Success  
 "Other error code" See [VL53L0X\\_Error](#)

[VL53L0X\\_API](#) [VL53L0X\\_Error](#) VL53L0X\_GetLimitCheckInfo ([VL53L0X\\_DEV](#) *Dev*, [uint16\\_t](#) *LimitCheckId*, [char](#) \* *pLimitCheckString*)

Return a description string for a given limit check number.

#### Function Description

This function returns a description string for a given limit check number. The limit check is identified with the LimitCheckId.

#### Note:

This function doesn't Access to the device

#### Parameters:

<i>Dev</i>	Device Handle
<i>LimitCheckId</i>	Limit Check ID (0<= LimitCheckId < <a href="#">VL53L0X_GetNumberOfLimitCheck()</a> ).
<i>pLimitCheckString</i>	Pointer to the description string of the given check limit.

#### Returns:

VL53L0X\_ERROR\_NONE Success  
 VL53L0X\_ERROR\_INVALID\_PARAMS This error is returned when LimitCheckId value is out of range.  
 "Other error code" See [VL53L0X\\_Error](#)

**[VL53L0X\\_API](#) [VL53L0X\\_Error](#) VL53L0X\_GetLimitCheckStatus ([VL53L0X\\_DEV](#) *Dev*, [uint16\\_t](#) *LimitCheckId*, [uint8\\_t](#) \* *pLimitCheckStatus*)**

Return a the Status of the specified check limit.

#### Function Description

This function returns the Status of the specified check limit. The value indicate if the check is fail or not. The limit check is identified with the LimitCheckId.

#### Note:

This function doesn't Access to the device

#### Parameters:

<i>Dev</i>	Device Handle
<i>LimitCheckId</i>	Limit Check ID (0<= LimitCheckId < <a href="#">VL53L0X_GetNumberOfLimitCheck()</a> ).
<i>pLimitCheckStatus</i>	Pointer to the Limit Check Status of the given check limit. LimitCheckStatus : 0 the check is not fail 1 the check if fail or not enabled

#### Returns:

VL53L0X\_ERROR\_NONE Success  
 VL53L0X\_ERROR\_INVALID\_PARAMS This error is returned when LimitCheckId value is out of range.  
 "Other error code" See [VL53L0X\\_Error](#)

**[VL53L0X\\_API](#) [VL53L0X\\_Error](#) VL53L0X\_SetLimitCheckEnable ([VL53L0X\\_DEV](#) *Dev*, [uint16\\_t](#) *LimitCheckId*, [uint8\\_t](#) *LimitCheckEnable*)**

Enable/Disable a specific limit check.

#### Function Description

This function Enable/Disable a specific limit check. The limit check is identified with the LimitCheckId.

#### Note:

This function doesn't Access to the device

#### Parameters:

<i>Dev</i>	Device Handle
<i>LimitCheckId</i>	Limit Check ID (0<= LimitCheckId < <a href="#">VL53L0X_GetNumberOfLimitCheck()</a> ).
<i>LimitCheckEnable</i>	if 1 the check limit corresponding to LimitCheckId is Enabled if 0 the check limit corresponding to LimitCheckId is disabled

#### Returns:

VL53L0X\_ERROR\_NONE Success



VL53L0X\_ERROR\_INVALID\_PARAMS This error is returned when LimitCheckId value is out of range.  
 "Other error code" See [VL53L0X\\_Error](#)

**[VL53L0X\\_API](#) [VL53L0X\\_Error](#) VL53L0X\_GetLimitCheckEnable ([VL53L0X\\_DEV](#) Dev, [uint16\\_t](#) LimitCheckId, [uint8\\_t](#) \* pLimitCheckEnable)**

Get specific limit check enable state.

### Function Description

This function get the enable state of a specific limit check. The limit check is identified with the LimitCheckId.

#### Note:

This function Access to the device

#### Parameters:

<i>Dev</i>	Device Handle
<i>LimitCheckId</i>	Limit Check ID (0<= LimitCheckId < <a href="#">VL53L0X_GetNumberOfLimitCheck()</a> ).
<i>pLimitCheckEnable</i>	Pointer to the check limit enable value. if 1 the check limit corresponding to LimitCheckId is Enabled if 0 the check limit corresponding to LimitCheckId is disabled

#### Returns:

VL53L0X\_ERROR\_NONE Success  
 VL53L0X\_ERROR\_INVALID\_PARAMS This error is returned when LimitCheckId value is out of range.  
 "Other error code" See [VL53L0X\\_Error](#)

**[VL53L0X\\_API](#) [VL53L0X\\_Error](#) VL53L0X\_SetLimitCheckValue ([VL53L0X\\_DEV](#) Dev, [uint16\\_t](#) LimitCheckId, [FixPoint1616\\_t](#) LimitCheckValue)**

Set a specific limit check value.

### Function Description

This function set a specific limit check value. The limit check is identified with the LimitCheckId.

#### Note:

This function Access to the device

#### Parameters:

<i>Dev</i>	Device Handle
<i>LimitCheckId</i>	Limit Check ID (0<= LimitCheckId < <a href="#">VL53L0X_GetNumberOfLimitCheck()</a> ).
<i>LimitCheckValue</i>	Limit check Value for a given LimitCheckId

#### Returns:

VL53L0X\_ERROR\_NONE Success  
 VL53L0X\_ERROR\_INVALID\_PARAMS This error is returned when either LimitCheckId or LimitCheckValue value is out of range.  
 "Other error code" See [VL53L0X\\_Error](#)

**[VL53L0X\\_API](#) [VL53L0X\\_Error](#) VL53L0X\_GetLimitCheckValue ([VL53L0X\\_DEV](#) Dev, [uint16\\_t](#) LimitCheckId, [FixPoint1616\\_t](#) \* pLimitCheckValue)**

Get a specific limit check value.

## Function Description

This function get a specific limit check value from device then it updates internal values and check enables. The limit check is identified with the LimitCheckId.

### Note:

This function Access to the device

### Parameters:

<i>Dev</i>	Device Handle
<i>LimitCheckId</i>	Limit Check ID (0<= LimitCheckId < <a href="#">VL53L0X_GetNumberOfLimitCheck()</a> ).
<i>pLimitCheckValue</i>	Pointer to Limit check Value for a given LimitCheckId.

### Returns:

VL53L0X\_ERROR\_NONE Success

VL53L0X\_ERROR\_INVALID\_PARAMS This error is returned when LimitCheckId value is out of range.

"Other error code" See [VL53L0X\\_Error](#)

[VL53L0X\\_API](#) [VL53L0X\\_Error](#) VL53L0X\_GetLimitCheckCurrent ([VL53L0X\\_DEV](#) Dev, [uint16\\_t](#) LimitCheckId, [FixPoint1616\\_t](#) \* pLimitCheckCurrent)

Get the current value of the signal used for the limit check.

## Function Description

This function get a the current value of the signal used for the limit check. To obtain the latest value you should run a ranging before. The value reported is linked to the limit check identified with the LimitCheckId.

### Note:

This function Access to the device

### Parameters:

<i>Dev</i>	Device Handle
<i>LimitCheckId</i>	Limit Check ID (0<= LimitCheckId < <a href="#">VL53L0X_GetNumberOfLimitCheck()</a> ).
<i>pLimitCheckCurrent</i>	Pointer to current Value for a given LimitCheckId.

### Returns:

VL53L0X\_ERROR\_NONE Success

VL53L0X\_ERROR\_INVALID\_PARAMS This error is returned when LimitCheckId value is out of range.

"Other error code" See [VL53L0X\\_Error](#)

[VL53L0X\\_API](#) [VL53L0X\\_Error](#) VL53L0X\_SetWrapAroundCheckEnable ([VL53L0X\\_DEV](#) Dev, [uint8\\_t](#) WrapAroundCheckEnable)

Enable (or disable) Wrap around Check.

### Note:

This function Access to the device

### Parameters:

<i>Dev</i>	Device Handle
<i>WrapAroundCheckEnable</i>	Wrap around Check to be set 0=disabled, other = enabled

#### Returns:

VL53L0X\_ERROR\_NONE Success  
 "Other error code" See [VL53L0X\\_Error](#)

**[VL53L0X\\_API](#) [VL53L0X\\_Error](#) VL53L0X\_GetWrapAroundCheckEnable ([VL53L0X\\_DEV](#) Dev, [uint8\\_t](#) \* pWrapAroundCheckEnable)**

Get setup of Wrap around Check.

#### Function Description

This function get the wrapAround check enable parameters

#### Note:

This function Access to the device

#### Parameters:

<i>Dev</i>	Device Handle
<i>pWrapAroundCheckEnable</i>	Pointer to the Wrap around Check state 0=disabled or 1 = enabled

#### Returns:

VL53L0X\_ERROR\_NONE Success  
 "Other error code" See [VL53L0X\\_Error](#)

**[VL53L0X\\_API](#) [VL53L0X\\_Error](#) VL53L0X\_SetDmaxCalParameters ([VL53L0X\\_DEV](#) Dev, [uint16\\_t](#) RangeMilliMeter, [FixPoint1616\\_t](#) SignalRateRtnMegaCps)**

Set Dmax Calibration Parameters for a given device When one of the parameter is zero, this function will get parameter from NVM.

#### Note:

This function doesn't Access to the device

#### Parameters:

<i>Dev</i>	Device Handle
<i>RangeMilliMeter</i>	Calibration Distance
<i>SignalRateRtnMegaCps</i>	Signal rate return read at CalDistance

#### Returns:

VL53L0X\_ERROR\_NONE Success  
 "Other error code" See [VL53L0X\\_Error](#)

**[VL53L0X\\_API](#) [VL53L0X\\_Error](#) VL53L0X\_GetDmaxCalParameters ([VL53L0X\\_DEV](#) Dev, [uint16\\_t](#) \* pRangeMilliMeter, [FixPoint1616\\_t](#) \* pSignalRateRtnMegaCps)**

Get Dmax Calibration Parameters for a given device.

#### Note:

This function Access to the device

#### Parameters:

<i>Dev</i>	Device Handle
<i>pRangeMilliMeter</i>	Pointer to Calibration Distance
<i>pSignalRateRtnMegaCps</i>	Pointer to Signal rate return

### Returns:

VL53L0X\_ERROR\_NONE Success  
 "Other error code" See [VL53L0X\\_Error](#)

## VL53L0X Measurement Functions

Functions used for the measurements.

### Functions

- [VL53L0X\\_API\\_VL53L0X\\_Error\\_VL53L0X\\_PerformSingleMeasurement](#) ([VL53L0X\\_DEV](#) Dev)  
*Single shot measurement.*
- [VL53L0X\\_API\\_VL53L0X\\_Error\\_VL53L0X\\_PerformRefCalibration](#) ([VL53L0X\\_DEV](#) Dev, [uint8\\_t](#) \*pVhvSettings, [uint8\\_t](#) \*pPhaseCal)  
*Perform Reference Calibration.*
- [VL53L0X\\_API\\_VL53L0X\\_Error\\_VL53L0X\\_PerformXTalkMeasurement](#) ([VL53L0X\\_DEV](#) Dev, [uint32\\_t](#) TimeoutMs, [FixPoint1616\\_t](#) \*pXTalkPerSpad, [uint8\\_t](#) \*pAmbientTooHigh)  
*Perform XTalk Measurement.*
- [VL53L0X\\_API\\_VL53L0X\\_Error\\_VL53L0X\\_PerformXTalkCalibration](#) ([VL53L0X\\_DEV](#) Dev, [FixPoint1616\\_t](#) XTalkCalDistance, [FixPoint1616\\_t](#) \*pXTalkCompensationRateMegaCps)  
*Perform XTalk Calibration.*
- [VL53L0X\\_API\\_VL53L0X\\_Error\\_VL53L0X\\_PerformOffsetCalibration](#) ([VL53L0X\\_DEV](#) Dev, [FixPoint1616\\_t](#) CalDistanceMilliMeter, [int32\\_t](#) \*pOffsetMicroMeter)  
*Perform Offset Calibration.*
- [VL53L0X\\_API\\_VL53L0X\\_Error\\_VL53L0X\\_StartMeasurement](#) ([VL53L0X\\_DEV](#) Dev)  
*Start device measurement.*
- [VL53L0X\\_API\\_VL53L0X\\_Error\\_VL53L0X\\_StopMeasurement](#) ([VL53L0X\\_DEV](#) Dev)  
*Stop device measurement.*
- [VL53L0X\\_API\\_VL53L0X\\_Error\\_VL53L0X\\_GetMeasurementDataReady](#) ([VL53L0X\\_DEV](#) Dev, [uint8\\_t](#) \*pMeasurementDataReady)  
*Return Measurement Data Ready.*
- [VL53L0X\\_API\\_VL53L0X\\_Error\\_VL53L0X\\_WaitDeviceReadyForNewMeasurement](#) ([VL53L0X\\_DEV](#) Dev, [uint32\\_t](#) MaxLoop)  
*Wait for device ready for a new measurement command.*
- [VL53L0X\\_API\\_VL53L0X\\_Error\\_VL53L0X\\_GetMeasurementRefSignal](#) ([VL53L0X\\_DEV](#) Dev, [FixPoint1616\\_t](#) \*pMeasurementRefSignal)  
*Retrieve the Reference Signal after a measurements.*
- [VL53L0X\\_API\\_VL53L0X\\_Error\\_VL53L0X\\_GetRangingMeasurementData](#) ([VL53L0X\\_DEV](#) Dev, [VL53L0X\\_RangingMeasurementData\\_t](#) \*pRangingMeasurementData)  
*Retrieve the measurements from device for a given setup.*
- [VL53L0X\\_API\\_VL53L0X\\_Error\\_VL53L0X\\_GetHistogramMeasurementData](#) ([VL53L0X\\_DEV](#) Dev, [VL53L0X\\_HistogramMeasurementData\\_t](#) \*pHistogramMeasurementData)  
*Retrieve the measurements from device for a given setup.*
- [VL53L0X\\_API\\_VL53L0X\\_Error\\_VL53L0X\\_PerformSingleRangingMeasurement](#) ([VL53L0X\\_DEV](#) Dev, [VL53L0X\\_RangingMeasurementData\\_t](#) \*pRangingMeasurementData)  
*Performs a single ranging measurement and retrieve the ranging measurement data.*
- [VL53L0X\\_API\\_VL53L0X\\_Error\\_VL53L0X\\_PerformSingleHistogramMeasurement](#) ([VL53L0X\\_DEV](#) Dev, [VL53L0X\\_HistogramMeasurementData\\_t](#) \*pHistogramMeasurementData)  
*Performs a single histogram measurement and retrieve the histogram measurement data Is equivalent to VL53L0X\_PerformSingleMeasurement + VL53L0X\_GetHistogramMeasurementData.*

- [VL53L0X\\_API\\_VL53L0X\\_Error\\_VL53L0X\\_SetNumberOfROIzones](#) ([VL53L0X\\_DEV](#) Dev, [uint8\\_t](#) NumberOfROIzones)  
*Set the number of ROI Zones to be used for a specific Device.*
- [VL53L0X\\_API\\_VL53L0X\\_Error\\_VL53L0X\\_GetNumberOfROIzones](#) ([VL53L0X\\_DEV](#) Dev, [uint8\\_t](#) \*pNumberOfROIzones)  
*Get the number of ROI Zones managed by the Device.*
- [VL53L0X\\_API\\_VL53L0X\\_Error\\_VL53L0X\\_GetMaxNumberOfROIzones](#) ([VL53L0X\\_DEV](#) Dev, [uint8\\_t](#) \*pMaxNumberOfROIzones)  
*Get the Maximum number of ROI Zones managed by the Device.*

## Detailed Description

Functions used for the measurements.

## Function Documentation

### [VL53L0X\\_API\\_VL53L0X\\_Error\\_VL53L0X\\_PerformSingleMeasurement](#) ([VL53L0X\\_DEV](#) Dev)

Single shot measurement.

#### Function Description

Perform simple measurement sequence (Start measure, Wait measure to end, and returns when measurement is done). Once function returns, user can get valid data by calling [VL53L0X\\_GetRangingMeasurement](#) or [VL53L0X\\_GetHistogramMeasurement](#) depending on defined measurement mode User should Clear the interrupt in case this are enabled by using the function [VL53L0X\\_ClearInterruptMask\(\)](#).

#### Warning:

This function is a blocking function

#### Note:

This function Access to the device

#### Parameters:

<i>Dev</i>	Device Handle
------------	---------------

#### Returns:

VL53L0X\_ERROR\_NONE Success  
"Other error code" See [VL53L0X\\_Error](#)

### [VL53L0X\\_API\\_VL53L0X\\_Error\\_VL53L0X\\_PerformRefCalibration](#) ([VL53L0X\\_DEV](#) Dev, [uint8\\_t](#) \*pVhvSettings, [uint8\\_t](#) \*pPhaseCal)

Perform Reference Calibration.

Perform a reference calibration of the Device. This function should be run from time to time before doing a ranging measurement. This function will launch a special ranging measurement, so if interrupt are enable an interrupt will be done. This function will clear the interrupt generated automatically.

#### Warning:

This function is a blocking function

#### Note:

This function Access to the device

#### Parameters:

<i>Dev</i>	Device Handle
<i>pVhvSettings</i>	Pointer to vhv settings parameter.
<i>pPhaseCal</i>	Pointer to PhaseCal parameter.

#### Returns:

VL53L0X\_ERROR\_NONE Success  
 "Other error code" See [VL53L0X\\_Error](#)

[VL53L0X\\_API](#) [VL53L0X\\_Error](#) **VL53L0X\_PerformXTalkMeasurement** ([VL53L0X\\_DEV](#) *Dev*,  
[uint32\\_t](#) *TimeoutMs*, [FixPoint1616\\_t](#) \* *pXtalkPerSpad*, [uint8\\_t](#) \* *pAmbientTooHigh*)

Perform XTalk Measurement.

Measures the current cross talk from glass in front of the sensor. This functions performs a histogram measurement and uses the results to measure the crosstalk. For the function to be successful, there must be no target in front of the sensor.

#### Warning:

This function is a blocking function  
 This function is not supported when the final range vcsel clock period is set below 10 PCLKS.

#### Note:

This function Access to the device

#### Parameters:

<i>Dev</i>	Device Handle
<i>TimeoutMs</i>	Histogram measurement duration.
<i>pXtalkPerSpad</i>	Output parameter containing the crosstalk measurement result, in MCPS/Spad. Format fixpoint 16:16.
<i>pAmbientTooHigh</i>	Output parameter which indicate that pXtalkPerSpad is not good if the Ambient is too high.

#### Returns:

VL53L0X\_ERROR\_NONE Success  
 VL53L0X\_ERROR\_INVALID\_PARAMS vcsel clock period not supported for this operation. Must not be less than 10PCLKS.  
 "Other error code" See [VL53L0X\\_Error](#)

[VL53L0X\\_API](#) [VL53L0X\\_Error](#) **VL53L0X\_PerformXTalkCalibration** ([VL53L0X\\_DEV](#) *Dev*,  
[FixPoint1616\\_t](#) *XTalkCalDistance*, [FixPoint1616\\_t](#) \* *pXTalkCompensationRateMegaCps*)

Perform XTalk Calibration.

Perform a XTalk calibration of the Device. This function will launch a ranging measurement, if interrupts are enabled an interrupt will be done. This function will clear the interrupt generated automatically. This function will program a new value for the XTalk compensation and it will enable the cross talk before exit. This function will disable the VL53L0X\_CHECKENABLE\_RANGE\_IGNORE\_THRESHOLD.

#### Warning:

This function is a blocking function

#### Note:

This function Access to the device  
 This function change the device mode to VL53L0X\_DEVICEMODE\_SINGLE\_RANGING

#### Parameters:

<i>Dev</i>	Device Handle
------------	---------------

<i>XTalkCalDistance</i>	XTalkCalDistance value used for the XTalk computation.
<i>pXTalkCompensationRateMegaCps</i>	Pointer to new XTalkCompensation value.

**Returns:**

VL53L0X\_ERROR\_NONE Success  
 "Other error code" See [VL53L0X\\_Error](#)

**[VL53L0X\\_API](#) [VL53L0X\\_Error](#) VL53L0X\_PerformOffsetCalibration ([VL53L0X\\_DEV](#) Dev, [FixPoint1616\\_t](#) CalDistanceMilliMeter, [int32\\_t](#) \* pOffsetMicroMeter)**

Perform Offset Calibration.

Perform a Offset calibration of the Device. This function will launch a ranging measurement, if interrupts are enabled an interrupt will be done. This function will clear the interrupt generated automatically. This function will program a new value for the Offset calibration value This function will disable the VL53L0X\_CHECKENABLE\_RANGE\_IGNORE\_THRESHOLD.

**Warning:**

This function is a blocking function

**Note:**

This function Access to the device  
 This function does not change the device mode.

**Parameters:**

<i>Dev</i>	Device Handle
<i>CalDistanceMilliMeter</i>	Calibration distance value used for the offset compensation.
<i>pOffsetMicroMeter</i>	Pointer to new Offset value computed by the function.

**Returns:**

VL53L0X\_ERROR\_NONE Success  
 "Other error code" See [VL53L0X\\_Error](#)

**[VL53L0X\\_API](#) [VL53L0X\\_Error](#) VL53L0X\_StartMeasurement ([VL53L0X\\_DEV](#) Dev)**

Start device measurement.

Started measurement will depend on device parameters set through *VL53L0X\_SetParameters()* This is a non-blocking function. This function will change the VL53L0X\_State from VL53L0X\_STATE\_IDLE to VL53L0X\_STATE\_RUNNING.

**Note:**

This function Access to the device

**Parameters:**

<i>Dev</i>	Device Handle
------------	---------------

**Returns:**

VL53L0X\_ERROR\_NONE Success  
 VL53L0X\_ERROR\_MODE\_NOT\_SUPPORTED This error occurs when DeviceMode programmed with *VL53L0X\_SetDeviceMode* is not in the supported list: Supported mode are:  
 VL53L0X\_DEVICEMODE\_SINGLE\_RANGING,  
 VL53L0X\_DEVICEMODE\_CONTINUOUS\_RANGING,  
 VL53L0X\_DEVICEMODE\_CONTINUOUS\_TIMED\_RANGING  
 VL53L0X\_ERROR\_TIME\_OUT Time out on start measurement  
 "Other error code" See [VL53L0X\\_Error](#)

**[VL53L0X\\_API](#) [VL53L0X\\_Error](#) VL53L0X\_StopMeasurement ([VL53L0X\\_DEV](#) Dev)**

Stop device measurement.

Will set the device in standby mode at end of current measurement

Not necessary in single mode as device shall return automatically in standby mode at end of measurement. This function will change the VL53L0X\_State from VL53L0X\_STATE\_RUNNING to VL53L0X\_STATE\_IDLE.

**Note:**

This function Access to the device

**Parameters:**

<i>Dev</i>	Device Handle
------------	---------------

**Returns:**

VL53L0X\_ERROR\_NONE Success  
 "Other error code" See [VL53L0X\\_Error](#)

[VL53L0X\\_API](#) [VL53L0X\\_Error](#) VL53L0X\_GetMeasurementDataReady ([VL53L0X\\_DEV](#) *Dev*, [uint8\\_t](#) \* *pMeasurementDataReady*)

Return Measurement Data Ready.

**Function Description**

This function indicate that a measurement data is ready. This function check if interrupt mode is used then check is done accordingly. If perform function clear the interrupt, this function will not work, like in case of [VL53L0X\\_PerformSingleRangingMeasurement\(\)](#). The previous function is blocking function, VL53L0X\_GetMeasurementDataReady is used for non-blocking capture.

**Note:**

This function Access to the device

**Parameters:**

<i>Dev</i>	Device Handle
<i>pMeasurementDataReady</i>	Pointer to Measurement Data Ready. 0=data not ready, 1 = data ready

**Returns:**

VL53L0X\_ERROR\_NONE Success  
 "Other error code" See [VL53L0X\\_Error](#)

[VL53L0X\\_API](#) [VL53L0X\\_Error](#) VL53L0X\_WaitDeviceReadyForNewMeasurement ([VL53L0X\\_DEV](#) *Dev*, [uint32\\_t](#) *MaxLoop*)

Wait for device ready for a new measurement command.

Blocking function.

**Note:**

This function is not Implemented

**Parameters:**

<i>Dev</i>	Device Handle
<i>MaxLoop</i>	Max Number of polling loop (timeout).

**Returns:**

VL53L0X\_ERROR\_NOT\_IMPLEMENTED Not implemented

[VL53L0X\\_API](#) [VL53L0X\\_Error](#) VL53L0X\_GetMeasurementRefSignal ([VL53L0X\\_DEV](#) *Dev*, [FixPoint1616\\_t](#) \* *pMeasurementRefSignal*)



Retrieve the Reference Signal after a measurements.

### Function Description

Get Reference Signal from last successful Ranging measurement This function return a valid value after that you call the [VL53L0X\\_GetRangingMeasurementData\(\)](#).

#### Note:

This function Access to the device

#### Parameters:

<i>Dev</i>	Device Handle
<i>pMeasurementRef Signal</i>	Pointer to the Ref Signal to fill up.

#### Returns:

VL53L0X\_ERROR\_NONE Success  
"Other error code" See [VL53L0X\\_Error](#)

[VL53L0X\\_API](#) [VL53L0X\\_Error](#) [VL53L0X\\_GetRangingMeasurementData](#) ([VL53L0X\\_DEV](#) *Dev*,  
[VL53L0X\\_RangingMeasurementData\\_t](#) \* *pRangingMeasurementData*)

Retrieve the measurements from device for a given setup.

### Function Description

Get data from last successful Ranging measurement

#### Warning:

USER should take care about [VL53L0X\\_GetNumberOfROIzones\(\)](#) before get data. PAL will fill a NumberOfROIzones times the corresponding data structure used in the measurement function.

#### Note:

This function Access to the device

#### Parameters:

<i>Dev</i>	Device Handle
<i>pRangingMeasure mentData</i>	Pointer to the data structure to fill up.

#### Returns:

VL53L0X\_ERROR\_NONE Success  
"Other error code" See [VL53L0X\\_Error](#)

[VL53L0X\\_API](#) [VL53L0X\\_Error](#) [VL53L0X\\_GetHistogramMeasurementData](#) ([VL53L0X\\_DEV](#) *Dev*,  
[VL53L0X\\_HistogramMeasurementData\\_t](#) \* *pHistogramMeasurementData*)

Retrieve the measurements from device for a given setup.

### Function Description

Get data from last successful Histogram measurement

#### Warning:

USER should take care about [VL53L0X\\_GetNumberOfROIzones\(\)](#) before get data. PAL will fill a NumberOfROIzones times the corresponding data structure used in the measurement function.

#### Note:

This function is not Implemented

#### Parameters:

<i>Dev</i>	Device Handle
------------	---------------

<i>pHistogramMeasurementData</i>	Pointer to the histogram data structure.
----------------------------------	--

#### Returns:

VL53L0X\_ERROR\_NOT\_IMPLEMENTED Not implemented

**[VL53L0X\\_API](#) [VL53L0X\\_Error](#) VL53L0X\_PerformSingleRangingMeasurement ([VL53L0X\\_DEV](#) Dev, [VL53L0X\\_RangingMeasurementData\\_t](#) \* *pRangingMeasurementData*)**

Performs a single ranging measurement and retrieve the ranging measurement data.

#### Function Description

This function will change the device mode to VL53L0X\_DEVICEMODE\_SINGLE\_RANGING with [VL53L0X\\_SetDeviceMode\(\)](#) , It performs measurement with [VL53L0X\\_PerformSingleMeasurement\(\)](#) . It get data from last successful Ranging measurement with [VL53L0X\\_GetRangingMeasurementData](#) . Finally it clear the interrupt with [VL53L0X\\_ClearInterruptMask\(\)](#) .

#### Note:

This function Access to the device

This function change the device mode to VL53L0X\_DEVICEMODE\_SINGLE\_RANGING

#### Parameters:

<i>Dev</i>	Device Handle
<i>pRangingMeasurementData</i>	Pointer to the data structure to fill up.

#### Returns:

VL53L0X\_ERROR\_NONE Success

"Other error code" See [VL53L0X\\_Error](#)

**[VL53L0X\\_API](#) [VL53L0X\\_Error](#) VL53L0X\_PerformSingleHistogramMeasurement ([VL53L0X\\_DEV](#) Dev, [VL53L0X\\_HistogramMeasurementData\\_t](#) \* *pHistogramMeasurementData*)**

Performs a single histogram measurement and retrieve the histogram measurement data Is equivalent to VL53L0X\_PerformSingleMeasurement + VL53L0X\_GetHistogramMeasurementData.

#### Function Description

Get data from last successful Ranging measurement. This function will clear the interrupt in case of these are enabled.

#### Note:

This function is not Implemented

#### Parameters:

<i>Dev</i>	Device Handle
<i>pHistogramMeasurementData</i>	Pointer to the data structure to fill up.

#### Returns:

VL53L0X\_ERROR\_NOT\_IMPLEMENTED Not implemented

**[VL53L0X\\_API](#) [VL53L0X\\_Error](#) VL53L0X\_SetNumberOfROI Zones ([VL53L0X\\_DEV](#) Dev, [uint8\\_t](#) NumberOfROI Zones)**

Set the number of ROI Zones to be used for a specific Device.

## Function Description

Set the number of ROI Zones to be used for a specific Device. The programmed value should be less than the max number of ROI Zones given with [VL53L0X\\_GetMaxNumberOfROI Zones\(\)](#) . This version of API manage only one zone.

### Parameters:

<i>Dev</i>	Device Handle
<i>NumberOfROI Zones</i>	Number of ROI Zones to be used for a specific Device.

### Returns:

VL53L0X\_ERROR\_NONE Success  
VL53L0X\_ERROR\_INVALID\_PARAMS This error is returned if NumberOfROI Zones != 1

[VL53L0X\\_API](#) [VL53L0X\\_Error](#) VL53L0X\_GetNumberOfROI Zones ([VL53L0X\\_DEV](#) *Dev*, [uint8\\_t](#) \* *pNumberOfROI Zones*)

Get the number of ROI Zones managed by the Device.

## Function Description

Get number of ROI Zones managed by the Device USER should take care about [VL53L0X\\_GetNumberOfROI Zones\(\)](#) before get data after a perform measurement. PAL will fill a NumberOfROI Zones times the corresponding data structure used in the measurement function.

### Note:

This function doesn't Access to the device

### Parameters:

<i>Dev</i>	Device Handle
<i>pNumberOfROI Zones</i>	Pointer to the Number of ROI Zones value.

### Returns:

VL53L0X\_ERROR\_NONE Success

[VL53L0X\\_API](#) [VL53L0X\\_Error](#) VL53L0X\_GetMaxNumberOfROI Zones ([VL53L0X\\_DEV](#) *Dev*, [uint8\\_t](#) \* *pMaxNumberOfROI Zones*)

Get the Maximum number of ROI Zones managed by the Device.

## Function Description

Get Maximum number of ROI Zones managed by the Device.

### Note:

This function doesn't Access to the device

### Parameters:

<i>Dev</i>	Device Handle
<i>pMaxNumberOfROI Zones</i>	Pointer to the Maximum Number of ROI Zones value.

### Returns:

VL53L0X\_ERROR\_NONE Success

## VL53L0X Interrupt Functions

Functions used for interrupt managements.

## Functions

- [VL53L0X\\_API\\_VL53L0X\\_Error\\_VL53L0X\\_SetGpioConfig](#) ([VL53L0X\\_DEV](#) Dev, [uint8\\_t](#) Pin, [VL53L0X\\_DeviceModes](#) DeviceMode, [VL53L0X\\_GpioFunctionality](#) Functionality, [VL53L0X\\_InterruptPolarity](#) Polarity)  
*Set the configuration of GPIO pin for a given device.*
- [VL53L0X\\_API\\_VL53L0X\\_Error\\_VL53L0X\\_GetGpioConfig](#) ([VL53L0X\\_DEV](#) Dev, [uint8\\_t](#) Pin, [VL53L0X\\_DeviceModes](#) \*pDeviceMode, [VL53L0X\\_GpioFunctionality](#) \*pFunctionality, [VL53L0X\\_InterruptPolarity](#) \*pPolarity)  
*Get current configuration for GPIO pin for a given device.*
- [VL53L0X\\_API\\_VL53L0X\\_Error\\_VL53L0X\\_SetInterruptThresholds](#) ([VL53L0X\\_DEV](#) Dev, [VL53L0X\\_DeviceModes](#) DeviceMode, [FixPoint1616\\_t](#) ThresholdLow, [FixPoint1616\\_t](#) ThresholdHigh)  
*Set low and high Interrupt thresholds for a given mode (ranging, ALS, ...) for a given device.*
- [VL53L0X\\_API\\_VL53L0X\\_Error\\_VL53L0X\\_GetInterruptThresholds](#) ([VL53L0X\\_DEV](#) Dev, [VL53L0X\\_DeviceModes](#) DeviceMode, [FixPoint1616\\_t](#) \*pThresholdLow, [FixPoint1616\\_t](#) \*pThresholdHigh)  
*Get high and low Interrupt thresholds for a given mode (ranging, ALS, ...) for a given device.*
- [VL53L0X\\_API\\_VL53L0X\\_Error\\_VL53L0X\\_GetStopCompletedStatus](#) ([VL53L0X\\_DEV](#) Dev, [uint32\\_t](#) \*pStopStatus)  
*Return device stop completion status.*
- [VL53L0X\\_API\\_VL53L0X\\_Error\\_VL53L0X\\_ClearInterruptMask](#) ([VL53L0X\\_DEV](#) Dev, [uint32\\_t](#) InterruptMask)  
*Clear given system interrupt condition.*
- [VL53L0X\\_API\\_VL53L0X\\_Error\\_VL53L0X\\_GetInterruptMaskStatus](#) ([VL53L0X\\_DEV](#) Dev, [uint32\\_t](#) \*pInterruptMaskStatus)  
*Return device interrupt status.*
- [VL53L0X\\_API\\_VL53L0X\\_Error\\_VL53L0X\\_EnableInterruptMask](#) ([VL53L0X\\_DEV](#) Dev, [uint32\\_t](#) InterruptMask)  
*Configure ranging interrupt reported to system.*

## Detailed Description

Functions used for interrupt managements.

## Function Documentation

[VL53L0X\\_API\\_VL53L0X\\_Error\\_VL53L0X\\_SetGpioConfig](#) ([VL53L0X\\_DEV](#) Dev, [uint8\\_t](#) Pin, [VL53L0X\\_DeviceModes](#) DeviceMode, [VL53L0X\\_GpioFunctionality](#) Functionality, [VL53L0X\\_InterruptPolarity](#) Polarity)

Set the configuration of GPIO pin for a given device.

### Note:

This function Access to the device

### Parameters:

<i>Dev</i>	Device Handle
<i>Pin</i>	ID of the GPIO Pin
<i>Functionality</i>	Select Pin functionality. Refer to <a href="#">VL53L0X_GpioFunctionality</a>

<i>DeviceMode</i>	Device Mode associated to the Gpio.
<i>Polarity</i>	Set interrupt polarity. Active high or active low see <a href="#">VL53L0X_InterruptPolarity</a>

#### Returns:

VL53L0X\_ERROR\_NONE Success  
 VL53L0X\_ERROR\_GPIO\_NOT\_EXISTING Only Pin=0 is accepted.  
 VL53L0X\_ERROR\_GPIO\_FUNCTIONALITY\_NOT\_SUPPORTED This error occurs when Functionality programmed is not in the supported list: Supported value are:  
 VL53L0X\_GPIOFUNCTIONALITY\_OFF,  
 VL53L0X\_GPIOFUNCTIONALITY\_THRESHOLD\_CROSSED\_LOW,  
 VL53L0X\_GPIOFUNCTIONALITY\_THRESHOLD\_CROSSED\_HIGH,  
 VL53L0X\_GPIOFUNCTIONALITY\_THRESHOLD\_CROSSED\_OUT,  
 VL53L0X\_GPIOFUNCTIONALITY\_NEW\_MEASURE\_READY  
 "Other error code" See [VL53L0X\\_Error](#)

**[VL53L0X\\_API](#) [VL53L0X\\_Error](#) [VL53L0X\\_GetGpioConfig](#) ([VL53L0X\\_DEV](#) *Dev*, [uint8\\_t](#) *Pin*, [VL53L0X\\_DeviceModes](#) \* *pDeviceMode*, [VL53L0X\\_GpioFunctionality](#) \* *pFunctionality*, [VL53L0X\\_InterruptPolarity](#) \* *pPolarity*)**

Get current configuration for GPIO pin for a given device.

#### Note:

This function Access to the device

#### Parameters:

<i>Dev</i>	Device Handle
<i>Pin</i>	ID of the GPIO Pin
<i>pDeviceMode</i>	Pointer to Device Mode associated to the Gpio.
<i>pFunctionality</i>	Pointer to Pin functionality. Refer to <a href="#">VL53L0X_GpioFunctionality</a>
<i>pPolarity</i>	Pointer to interrupt polarity. Active high or active low see <a href="#">VL53L0X_InterruptPolarity</a>

#### Returns:

VL53L0X\_ERROR\_NONE Success  
 VL53L0X\_ERROR\_GPIO\_NOT\_EXISTING Only Pin=0 is accepted.  
 VL53L0X\_ERROR\_GPIO\_FUNCTIONALITY\_NOT\_SUPPORTED This error occurs when Functionality programmed is not in the supported list: Supported value are:  
 VL53L0X\_GPIOFUNCTIONALITY\_OFF,  
 VL53L0X\_GPIOFUNCTIONALITY\_THRESHOLD\_CROSSED\_LOW,  
 VL53L0X\_GPIOFUNCTIONALITY\_THRESHOLD\_CROSSED\_HIGH,  
 VL53L0X\_GPIOFUNCTIONALITY\_THRESHOLD\_CROSSED\_OUT,  
 VL53L0X\_GPIOFUNCTIONALITY\_NEW\_MEASURE\_READY  
 "Other error code" See [VL53L0X\\_Error](#)

**[VL53L0X\\_API](#) [VL53L0X\\_Error](#) [VL53L0X\\_SetInterruptThresholds](#) ([VL53L0X\\_DEV](#) *Dev*, [VL53L0X\\_DeviceModes](#) *DeviceMode*, [FixPoint1616\\_t](#) *ThresholdLow*, [FixPoint1616\\_t](#) *ThresholdHigh*)**

Set low and high Interrupt thresholds for a given mode (ranging, ALS, ...) for a given device.

#### Function Description

Set low and high Interrupt thresholds for a given mode (ranging, ALS, ...) for a given device

#### Note:

This function Access to the device  
 DeviceMode is ignored for the current device

#### Parameters:

<i>Dev</i>	Device Handle
<i>DeviceMode</i>	Device Mode for which change thresholds
<i>ThresholdLow</i>	Low threshold (mm, lux ..., depending on the mode)
<i>ThresholdHigh</i>	High threshold (mm, lux ..., depending on the mode)

#### Returns:

VL53L0X\_ERROR\_NONE Success  
 "Other error code" See [VL53L0X\\_Error](#)

[VL53L0X\\_API](#) [VL53L0X\\_Error](#) VL53L0X\_GetInterruptThresholds ([VL53L0X\\_DEV](#) *Dev*,  
[VL53L0X\\_DeviceModes](#) *DeviceMode*, [FixPoint1616\\_t](#) \* *pThresholdLow*, [FixPoint1616\\_t](#) \*  
*pThresholdHigh*)

Get high and low Interrupt thresholds for a given mode (ranging, ALS, ...) for a given device.

#### Function Description

Get high and low Interrupt thresholds for a given mode (ranging, ALS, ...) for a given device

#### Note:

This function Access to the device  
 DeviceMode is ignored for the current device

#### Parameters:

<i>Dev</i>	Device Handle
<i>DeviceMode</i>	Device Mode from which read thresholds
<i>pThresholdLow</i>	Low threshold (mm, lux ..., depending on the mode)
<i>pThresholdHigh</i>	High threshold (mm, lux ..., depending on the mode)

#### Returns:

VL53L0X\_ERROR\_NONE Success  
 "Other error code" See [VL53L0X\\_Error](#)

[VL53L0X\\_API](#) [VL53L0X\\_Error](#) VL53L0X\_GetStopCompletedStatus ([VL53L0X\\_DEV](#) *Dev*,  
[uint32\\_t](#) \* *pStopStatus*)

Return device stop completion status.

#### Function Description

Returns stop completion status. User shall call this function after a stop command

#### Note:

This function Access to the device

#### Parameters:

<i>Dev</i>	Device Handle
<i>pStopStatus</i>	Pointer to status variable to update

#### Returns:

VL53L0X\_ERROR\_NONE Success  
 "Other error code" See [VL53L0X\\_Error](#)

[VL53L0X\\_API](#) [VL53L0X\\_Error](#) VL53L0X\_ClearInterruptMask ([VL53L0X\\_DEV](#) *Dev*, [uint32\\_t](#)  
*InterruptMask*)

Clear given system interrupt condition.

## Function Description

Clear given interrupt(s).

### Note:

This function Access to the device

### Parameters:

<i>Dev</i>	Device Handle
<i>InterruptMask</i>	Mask of interrupts to clear

### Returns:

VL53L0X\_ERROR\_NONE Success

VL53L0X\_ERROR\_INTERRUPT\_NOT\_CLEARED Cannot clear interrupts

"Other error code" See [VL53L0X\\_Error](#)

[VL53L0X\\_API](#) [VL53L0X\\_Error](#) VL53L0X\_GetInterruptMaskStatus ([VL53L0X\\_DEV](#) *Dev*, [uint32\\_t](#) *pInterruptMaskStatus*)

Return device interrupt status.

## Function Description

Returns currently raised interrupts by the device. User shall be able to activate/deactivate interrupts through [VL53L0X\\_SetGpioConfig\(\)](#)

### Note:

This function Access to the device

### Parameters:

<i>Dev</i>	Device Handle
<i>pInterruptMaskStatus</i>	Pointer to status variable to update

### Returns:

VL53L0X\_ERROR\_NONE Success

"Other error code" See [VL53L0X\\_Error](#)

[VL53L0X\\_API](#) [VL53L0X\\_Error](#) VL53L0X\_EnableInterruptMask ([VL53L0X\\_DEV](#) *Dev*, [uint32\\_t](#) *InterruptMask*)

Configure ranging interrupt reported to system.

### Note:

This function is not Implemented

### Parameters:

<i>Dev</i>	Device Handle
<i>InterruptMask</i>	Mask of interrupt to Enable/disable (0:interrupt disabled or 1: interrupt enabled)

### Returns:

VL53L0X\_ERROR\_NOT\_IMPLEMENTED Not implemented

## VL53L0X SPAD Functions

Functions used for SPAD managements.

## Functions

- [VL53L0X\\_API\\_VL53L0X\\_Error\\_VL53L0X\\_SetSpadAmbientDamperThreshold](#) ([VL53L0X\\_DEV](#) Dev, [uint16\\_t](#) SpadAmbientDamperThreshold)  
*Set the SPAD Ambient Damper Threshold value.*
- [VL53L0X\\_API\\_VL53L0X\\_Error\\_VL53L0X\\_GetSpadAmbientDamperThreshold](#) ([VL53L0X\\_DEV](#) Dev, [uint16\\_t](#) \*pSpadAmbientDamperThreshold)  
*Get the current SPAD Ambient Damper Threshold value.*
- [VL53L0X\\_API\\_VL53L0X\\_Error\\_VL53L0X\\_SetSpadAmbientDamperFactor](#) ([VL53L0X\\_DEV](#) Dev, [uint16\\_t](#) SpadAmbientDamperFactor)  
*Set the SPAD Ambient Damper Factor value.*
- [VL53L0X\\_API\\_VL53L0X\\_Error\\_VL53L0X\\_GetSpadAmbientDamperFactor](#) ([VL53L0X\\_DEV](#) Dev, [uint16\\_t](#) \*pSpadAmbientDamperFactor)  
*Get the current SPAD Ambient Damper Factor value.*
- [VL53L0X\\_API\\_VL53L0X\\_Error\\_VL53L0X\\_PerformRefSpadManagement](#) ([VL53L0X\\_DEV](#) Dev, [uint32\\_t](#) \*refSpadCount, [uint8\\_t](#) \*isApertureSpads)  
*Performs Reference Spad Management.*
- [VL53L0X\\_API\\_VL53L0X\\_Error\\_VL53L0X\\_SetReferenceSpads](#) ([VL53L0X\\_DEV](#) Dev, [uint32\\_t](#) refSpadCount, [uint8\\_t](#) isApertureSpads)  
*Applies Reference SPAD configuration.*
- [VL53L0X\\_API\\_VL53L0X\\_Error\\_VL53L0X\\_GetReferenceSpads](#) ([VL53L0X\\_DEV](#) Dev, [uint32\\_t](#) \*refSpadCount, [uint8\\_t](#) \*isApertureSpads)  
*Retrieves SPAD configuration.*

## Detailed Description

Functions used for SPAD managements.

## Function Documentation

[VL53L0X\\_API\\_VL53L0X\\_Error\\_VL53L0X\\_SetSpadAmbientDamperThreshold](#) ([VL53L0X\\_DEV](#) Dev, [uint16\\_t](#) *SpadAmbientDamperThreshold*)

Set the SPAD Ambient Damper Threshold value.

### Function Description

This function set the SPAD Ambient Damper Threshold value

#### Note:

This function Access to the device

#### Parameters:

<i>Dev</i>	Device Handle
<i>SpadAmbientDamperThreshold</i>	SPAD Ambient Damper Threshold value

#### Returns:

VL53L0X\_ERROR\_NONE Success  
"Other error code" See [VL53L0X\\_Error](#)



[VL53L0X API](#) [VL53L0X Error](#) [VL53L0X\\_GetSpadAmbientDamperThreshold](#) ([VL53L0X\\_DEV](#) *Dev*, [uint16\\_t](#) \* *pSpadAmbientDamperThreshold*)

Get the current SPAD Ambient Damper Threshold value.

### Function Description

This function get the SPAD Ambient Damper Threshold value

#### Note:

This function Access to the device

#### Parameters:

<i>Dev</i>	Device Handle
<i>pSpadAmbientDamperThreshold</i>	Pointer to programmed SPAD Ambient Damper Threshold value

#### Returns:

VL53L0X\_ERROR\_NONE Success  
"Other error code" See [VL53L0X\\_Error](#)

[VL53L0X API](#) [VL53L0X Error](#) [VL53L0X\\_SetSpadAmbientDamperFactor](#) ([VL53L0X\\_DEV](#) *Dev*, [uint16\\_t](#) \* *SpadAmbientDamperFactor*)

Set the SPAD Ambient Damper Factor value.

### Function Description

This function set the SPAD Ambient Damper Factor value

#### Note:

This function Access to the device

#### Parameters:

<i>Dev</i>	Device Handle
<i>SpadAmbientDamperFactor</i>	SPAD Ambient Damper Factor value

#### Returns:

VL53L0X\_ERROR\_NONE Success  
"Other error code" See [VL53L0X\\_Error](#)

[VL53L0X API](#) [VL53L0X Error](#) [VL53L0X\\_GetSpadAmbientDamperFactor](#) ([VL53L0X\\_DEV](#) *Dev*, [uint16\\_t](#) \* *pSpadAmbientDamperFactor*)

Get the current SPAD Ambient Damper Factor value.

### Function Description

This function get the SPAD Ambient Damper Factor value

#### Note:

This function Access to the device

#### Parameters:

<i>Dev</i>	Device Handle
<i>pSpadAmbientDamperFactor</i>	Pointer to programmed SPAD Ambient Damper Factor value

#### Returns:

VL53L0X\_ERROR\_NONE Success

"Other error code" See [VL53L0X\\_Error](#)

[VL53L0X\\_API](#) [VL53L0X\\_Error](#) [VL53L0X\\_PerformRefSpadManagement](#) ([VL53L0X\\_DEV](#) *Dev*, [uint32\\_t](#) \* *refSpadCount*, [uint8\\_t](#) \* *isApertureSpads*)

Performs Reference Spad Management.

### Function Description

The reference SPAD initialization procedure determines the minimum amount of reference spads to be enables to achieve a target reference signal rate and should be performed once during initialization.

### Note:

This function Access to the device

This function change the device mode to VL53L0X\_DEVICEMODE\_SINGLE\_RANGING

### Parameters:

<i>Dev</i>	Device Handle
<i>refSpadCount</i>	Reports ref Spad Count
<i>isApertureSpads</i>	Reports if spads are of type aperture or non-aperture. 1:=aperture, 0:=Non-Aperture

### Returns:

VL53L0X\_ERROR\_NONE Success

VL53L0X\_ERROR\_REF\_SPAD\_INIT Error in the Ref Spad procedure.

"Other error code" See [VL53L0X\\_Error](#)

[VL53L0X\\_API](#) [VL53L0X\\_Error](#) [VL53L0X\\_SetReferenceSpads](#) ([VL53L0X\\_DEV](#) *Dev*, [uint32\\_t](#) *refSpadCount*, [uint8\\_t](#) *isApertureSpads*)

Applies Reference SPAD configuration.

### Function Description

This function applies a given number of reference spads, identified as either Aperture or Non-Aperture. The requested spad count and type are stored within the device specific parameters data for access by the host.

### Note:

This function Access to the device

### Parameters:

<i>Dev</i>	Device Handle
<i>refSpadCount</i>	Number of ref spads.
<i>isApertureSpads</i>	Defines if spads are of type aperture or non-aperture. 1:=aperture, 0:=Non-Aperture

### Returns:

VL53L0X\_ERROR\_NONE Success

VL53L0X\_ERROR\_REF\_SPAD\_INIT Error in the in the reference spad configuration.

"Other error code" See [VL53L0X\\_Error](#)

[VL53L0X\\_API](#) [VL53L0X\\_Error](#) [VL53L0X\\_GetReferenceSpads](#) ([VL53L0X\\_DEV](#) *Dev*, [uint32\\_t](#) \* *refSpadCount*, [uint8\\_t](#) \* *isApertureSpads*)

Retrieves SPAD configuration.

## Function Description

This function retrieves the current number of applied reference spads and also their type : Aperture or Non-Aperture.

### Note:

This function Access to the device

### Parameters:

<i>Dev</i>	Device Handle
<i>refSpadCount</i>	Number ref Spad Count
<i>isApertureSpads</i>	Reports if spads are of type aperture or non-aperture. 1:=aperture, 0:=Non-Aperture

### Returns:

VL53L0X\_ERROR\_NONE Success

VL53L0X\_ERROR\_REF\_SPAD\_INIT Error in the in the reference spad configuration.

"Other error code" See [VL53L0X\\_Error](#)

## VL53L0X Defines

VL53L0X Defines.

### Modules

- [Error and Warning code returned by API](#)
- The following DEFINE are used to identify the PAL ERROR. [Defines Device modes](#)
- Defines all possible modes for the device. [Defines Histogram modes](#)
- Defines all possible Histogram modes for the device. [List of available Power Modes](#)
- List of available Power Modes. [Defines the current status of the device](#)
- Defines the current status of the device. [Defines the Polarity](#)
- of the Interrupt Defines the Polarity of the Interrupt [Vcsel Period Defines](#)
- Defines the range measurement for which to access the vcsel period. [Defines the steps](#)
- carried out by the scheduler during a range measurement. [Defines the Polarity](#)
- of the Interrupt Defines the the sequence steps performed during ranging. [General Macro Defines](#)

### General Macro Defines. Data Structures

- struct [VL53L0X\\_Version\\_t](#)
- Defines the parameters of the Get Version Functions. struct [VL53L0X\\_DeviceInfo\\_t](#)
- Defines the parameters of the Get Device Info Functions. struct [VL53L0X\\_DeviceParameters\\_t](#)
- Defines all parameters for the device. struct [VL53L0X\\_DMaxData\\_t](#)
- Structure containing the Dmax computation parameters and data. struct [VL53L0X\\_RangingMeasurementData\\_t](#)
- struct [VL53L0X\\_HistogramMeasurementData\\_t](#)
- struct [VL53L0X\\_SpadData\\_t](#)
- Spad Configuration Data. struct [VL53L0X\\_DeviceSpecificParameters\\_t](#)
- struct [VL53L0X\\_DevData\\_t](#)  
VL53L0X PAL device ST private data structure
- End user should never access any of these field directly. struct [VL53L0X\\_RangeData\\_t](#)
- Range measurement data. struct [VL53L0X\\_HistogramData\\_t](#)

### Histogram measurement data. Macros

- #define [VL53L0X10\\_SPECIFICATION\\_VER\\_MAJOR](#) 1  
PAL SPECIFICATION major version.
- #define [VL53L0X10\\_SPECIFICATION\\_VER\\_MINOR](#) 2  
PAL SPECIFICATION minor version.

- #define [VL53L0X10\\_SPECIFICATION\\_VER\\_SUB](#) 7  
PAL SPECIFICATION sub version.
- #define [VL53L0X10\\_SPECIFICATION\\_VER\\_REVISION](#) 1440  
PAL SPECIFICATION sub version.
- #define [VL53L0X10\\_IMPLEMENTATION\\_VER\\_MAJOR](#) 1  
VL53L0X PAL IMPLEMENTATION major version.
- #define [VL53L0X10\\_IMPLEMENTATION\\_VER\\_MINOR](#) 0  
VL53L0X PAL IMPLEMENTATION minor version.
- #define [VL53L0X10\\_IMPLEMENTATION\\_VER\\_SUB](#) 9  
VL53L0X PAL IMPLEMENTATION sub version.
- #define [VL53L0X10\\_IMPLEMENTATION\\_VER\\_REVISION](#) 3673  
VL53L0X PAL IMPLEMENTATION sub version.
- #define [VL53L0X\\_SPECIFICATION\\_VER\\_MAJOR](#) 1  
PAL SPECIFICATION major version.
- #define [VL53L0X\\_SPECIFICATION\\_VER\\_MINOR](#) 2  
PAL SPECIFICATION minor version.
- #define [VL53L0X\\_SPECIFICATION\\_VER\\_SUB](#) 7  
PAL SPECIFICATION sub version.
- #define [VL53L0X\\_SPECIFICATION\\_VER\\_REVISION](#) 1440  
PAL SPECIFICATION sub version.
- #define [VL53L0X\\_IMPLEMENTATION\\_VER\\_MAJOR](#) 1  
VL53L0X PAL IMPLEMENTATION major version.
- #define [VL53L0X\\_IMPLEMENTATION\\_VER\\_MINOR](#) 0  
VL53L0X PAL IMPLEMENTATION minor version.
- #define [VL53L0X\\_IMPLEMENTATION\\_VER\\_SUB](#) 1  
VL53L0X PAL IMPLEMENTATION sub version.
- #define [VL53L0X\\_IMPLEMENTATION\\_VER\\_REVISION](#) 4606  
VL53L0X PAL IMPLEMENTATION sub version.
- #define [VL53L0X\\_DEFAULT\\_MAX\\_LOOP](#) 200
- #define [VL53L0X\\_MAX\\_STRING\\_LENGTH](#) 32
- #define [VL53L0X\\_HISTOGRAM\\_BUFFER\\_SIZE](#) 24
- #define [VL53L0X\\_REF\\_SPAD\\_BUFFER\\_SIZE](#) 6

---

## Detailed Description

VL53L0X Defines.

---

## Macro Definition Documentation

**#define VL53L0X10\_SPECIFICATION\_VER\_MAJOR 1**

PAL SPECIFICATION major version.

Definition at line 52 of file vl53l0x\_def.h.

**#define VL53L0X10\_SPECIFICATION\_VER\_MINOR 2**

PAL SPECIFICATION minor version.

Definition at line 54 of file vl53l0x\_def.h.

**#define VL53L0X10\_SPECIFICATION\_VER\_SUB 7**

PAL SPECIFICATION sub version.

Definition at line 56 of file vl53l0x\_def.h.

**#define VL53L0X10\_SPECIFICATION\_VER\_REVISION 1440**

PAL SPECIFICATION sub version.

Definition at line 58 of file vl53l0x\_def.h.

**#define VL53L0X10\_IMPLEMENTATION\_VER\_MAJOR 1**

VL53L0X PAL IMPLEMENTATION major version.

Definition at line 61 of file vl53l0x\_def.h.

**#define VL53L0X10\_IMPLEMENTATION\_VER\_MINOR 0**

VL53L0X PAL IMPLEMENTATION minor version.

Definition at line 63 of file vl53l0x\_def.h.

**#define VL53L0X10\_IMPLEMENTATION\_VER\_SUB 9**

VL53L0X PAL IMPLEMENTATION sub version.

Definition at line 65 of file vl53l0x\_def.h.

**#define VL53L0X10\_IMPLEMENTATION\_VER\_REVISION 3673**

VL53L0X PAL IMPLEMENTATION sub version.

Definition at line 67 of file vl53l0x\_def.h.

**#define VL53L0X\_SPECIFICATION\_VER\_MAJOR 1**

PAL SPECIFICATION major version.

Definition at line 70 of file vl53l0x\_def.h.

**#define VL53L0X\_SPECIFICATION\_VER\_MINOR 2**

PAL SPECIFICATION minor version.

Definition at line 72 of file vl53l0x\_def.h.

**#define VL53L0X\_SPECIFICATION\_VER\_SUB 7**

PAL SPECIFICATION sub version.

Definition at line 74 of file vl53l0x\_def.h.

**#define VL53L0X\_SPECIFICATION\_VER\_REVISION 1440**

PAL SPECIFICATION sub version.

Definition at line 76 of file vl53l0x\_def.h.

**#define VL53L0X\_IMPLEMENTATION\_VER\_MAJOR 1**

VL53L0X PAL IMPLEMENTATION major version.

Definition at line 79 of file vl53l0x\_def.h.

**#define VL53L0X\_IMPLEMENTATION\_VER\_MINOR 0**

VL53L0X PAL IMPLEMENTATION minor version.

Definition at line 81 of file vl53l0x\_def.h.

**#define VL53L0X\_IMPLEMENTATION\_VER\_SUB 1**

VL53L0X PAL IMPLEMENTATION sub version.

Definition at line 83 of file vl53l0x\_def.h.

**#define VL53L0X\_IMPLEMENTATION\_VER\_REVISION 4606**

VL53L0X PAL IMPLEMENTATION sub version.

Definition at line 85 of file vl53l0x\_def.h.

**#define VL53L0X\_DEFAULT\_MAX\_LOOP 200**

Definition at line 86 of file vl53l0x\_def.h.

**#define VL53L0X\_MAX\_STRING\_LENGTH 32**

Definition at line 87 of file vl53l0x\_def.h.

**#define VL53L0X\_HISTOGRAM\_BUFFER\_SIZE 24**

Definition at line 346 of file vl53l0x\_def.h.

**#define VL53L0X\_REF\_SPAD\_BUFFER\_SIZE 6**

Definition at line 368 of file vl53l0x\_def.h.

## Error and Warning code returned by API

The following DEFINE are used to identify the PAL ERROR.

## Macros

- #define [VL53L0X\\_ERROR\\_NONE](#) (([VL53L0X\\_Error](#)) 0)
- #define [VL53L0X\\_ERROR\\_CALIBRATION\\_WARNING](#) (([VL53L0X\\_Error](#)) -1)
- #define [VL53L0X\\_ERROR\\_MIN\\_CLIPPED](#) (([VL53L0X\\_Error](#)) -2)
- #define [VL53L0X\\_ERROR\\_UNDEFINED](#) (([VL53L0X\\_Error](#)) -3)
- #define [VL53L0X\\_ERROR\\_INVALID\\_PARAMS](#) (([VL53L0X\\_Error](#)) -4)
- #define [VL53L0X\\_ERROR\\_NOT\\_SUPPORTED](#) (([VL53L0X\\_Error](#)) -5)
- #define [VL53L0X\\_ERROR\\_RANGE\\_ERROR](#) (([VL53L0X\\_Error](#)) -6)
- #define [VL53L0X\\_ERROR\\_TIME\\_OUT](#) (([VL53L0X\\_Error](#)) -7)
- #define [VL53L0X\\_ERROR\\_MODE\\_NOT\\_SUPPORTED](#) (([VL53L0X\\_Error](#)) -8)
- #define [VL53L0X\\_ERROR\\_BUFFER\\_TOO\\_SMALL](#) (([VL53L0X\\_Error](#)) -9)
- #define [VL53L0X\\_ERROR\\_GPIO\\_NOT\\_EXISTING](#) (([VL53L0X\\_Error](#)) -10)
- #define [VL53L0X\\_ERROR\\_GPIO\\_FUNCTIONALITY\\_NOT\\_SUPPORTED](#) (([VL53L0X\\_Error](#)) -11)
- #define [VL53L0X\\_ERROR\\_INTERRUPT\\_NOT\\_CLEARED](#) (([VL53L0X\\_Error](#)) -12)
- #define [VL53L0X\\_ERROR\\_CONTROL\\_INTERFACE](#) (([VL53L0X\\_Error](#)) -20)
- #define [VL53L0X\\_ERROR\\_INVALID\\_COMMAND](#) (([VL53L0X\\_Error](#)) -30)
- #define [VL53L0X\\_ERROR\\_DIVISION\\_BY\\_ZERO](#) (([VL53L0X\\_Error](#)) -40)
- #define [VL53L0X\\_ERROR\\_REF\\_SPAD\\_INIT](#) (([VL53L0X\\_Error](#)) -50)
- #define [VL53L0X\\_ERROR\\_NOT\\_IMPLEMENTED](#) (([VL53L0X\\_Error](#)) -99)

## Typedefs

- typedef [int8\\_t](#) [VL53L0X\\_Error](#)

---

## Detailed Description

The following DEFINE are used to identify the PAL ERROR.

---

## Macro Definition Documentation

**#define VL53L0X\_ERROR\_NONE (([VL53L0X\\_Error](#)) 0)**

Definition at line 133 of file vl53l0x\_def.h.

**#define VL53L0X\_ERROR\_CALIBRATION\_WARNING (([VL53L0X\\_Error](#)) -1)**

Warning invalid calibration data may be in used *VL53L0X\_InitData()*  
*VL53L0X\_GetOffsetCalibrationData* *VL53L0X\_SetOffsetCalibrationData*

Definition at line 134 of file vl53l0x\_def.h.

**#define VL53L0X\_ERROR\_MIN\_CLIPPED (([VL53L0X\\_Error](#)) -2)**

Warning parameter passed was clipped to min before to be applied

Definition at line 139 of file vl53l0x\_def.h.

**#define VL53L0X\_ERROR\_UNDEFINED (([VL53L0X\\_Error](#)) -3)**

Unqualified error

Definition at line 142 of file vl53l0x\_def.h.

**#define VL53L0X\_ERROR\_INVALID\_PARAMS (([VL53L0X\\_Error](#)) -4)**

Parameter passed is invalid or out of range

Definition at line 144 of file vl53l0x\_def.h.

**#define VL53L0X\_ERROR\_NOT\_SUPPORTED ((VL53L0X\_Error) -5)**

Function is not supported in current mode or configuration

Definition at line 146 of file vl53l0x\_def.h.

**#define VL53L0X\_ERROR\_RANGE\_ERROR ((VL53L0X\_Error) -6)**

Device report a ranging error interrupt status

Definition at line 148 of file vl53l0x\_def.h.

**#define VL53L0X\_ERROR\_TIME\_OUT ((VL53L0X\_Error) -7)**

Aborted due to time out

Definition at line 150 of file vl53l0x\_def.h.

**#define VL53L0X\_ERROR\_MODE\_NOT\_SUPPORTED ((VL53L0X\_Error) -8)**

Asked mode is not supported by the device

Definition at line 152 of file vl53l0x\_def.h.

**#define VL53L0X\_ERROR\_BUFFER\_TOO\_SMALL ((VL53L0X\_Error) -9)**

...

Definition at line 154 of file vl53l0x\_def.h.

**#define VL53L0X\_ERROR\_GPIO\_NOT\_EXISTING ((VL53L0X\_Error) -10)**

User tried to setup a non-existing GPIO pin

Definition at line 156 of file vl53l0x\_def.h.

**#define VL53L0X\_ERROR\_GPIO\_FUNCTIONALITY\_NOT\_SUPPORTED ((VL53L0X\_Error) -11)**

unsupported GPIO functionality

Definition at line 158 of file vl53l0x\_def.h.

**#define VL53L0X\_ERROR\_INTERRUPT\_NOT\_CLEARED ((VL53L0X\_Error) -12)**

Error during interrupt clear

Definition at line 160 of file vl53l0x\_def.h.

**#define VL53L0X\_ERROR\_CONTROL\_INTERFACE ((VL53L0X\_Error) -20)**

error reported from IO functions

Definition at line 162 of file vl53l0x\_def.h.

**#define VL53L0X\_ERROR\_INVALID\_COMMAND ((VL53L0X\_Error) -30)**

The command is not allowed in the current device state (power down)

Definition at line 164 of file vl53l0x\_def.h.

**#define VL53L0X\_ERROR\_DIVISION\_BY\_ZERO ((VL53L0X\_Error) -40)**

In the function a division by zero occurs

Definition at line 167 of file vl53l0x\_def.h.



**#define VL53L0X\_ERROR\_REF\_SPAD\_INIT (([VL53L0X\\_Error](#)) -50)**

Error during reference SPAD initialization

Definition at line 169 of file vl53l0x\_def.h.

**#define VL53L0X\_ERROR\_NOT\_IMPLEMENTED (([VL53L0X\\_Error](#)) -99)**

Tells requested functionality has not been implemented yet or not compatible with the device

Definition at line 171 of file vl53l0x\_def.h.

---

## Typedef Documentation

typedef [int8\\_t](#) [VL53L0X\\_Error](#)

Definition at line 131 of file vl53l0x\_def.h.

## Defines Device modes

Defines all possible modes for the device.

### Macros

- #define [VL53L0X\\_DEVICEMODE\\_SINGLE\\_RANGING](#) (([VL53L0X\\_DeviceModes](#)) 0)
- #define [VL53L0X\\_DEVICEMODE\\_CONTINUOUS\\_RANGING](#) (([VL53L0X\\_DeviceModes](#)) 1)
- #define [VL53L0X\\_DEVICEMODE\\_SINGLE\\_HISTOGRAM](#) (([VL53L0X\\_DeviceModes](#)) 2)
- #define [VL53L0X\\_DEVICEMODE\\_CONTINUOUS\\_TIMED\\_RANGING](#) (([VL53L0X\\_DeviceModes](#)) 3)
- #define [VL53L0X\\_DEVICEMODE\\_SINGLE\\_ALS](#) (([VL53L0X\\_DeviceModes](#)) 10)
- #define [VL53L0X\\_DEVICEMODE\\_GPIO\\_DRIVE](#) (([VL53L0X\\_DeviceModes](#)) 20)
- #define [VL53L0X\\_DEVICEMODE\\_GPIO\\_OSC](#) (([VL53L0X\\_DeviceModes](#)) 21)

### Typedefs

- typedef [uint8\\_t](#) [VL53L0X\\_DeviceModes](#)

---

## Detailed Description

Defines all possible modes for the device.

---

## Macro Definition Documentation

**#define VL53L0X\_DEVICEMODE\_SINGLE\_RANGING (([VL53L0X\\_DeviceModes](#)) 0)**

Definition at line 183 of file vl53l0x\_def.h.

**#define VL53L0X\_DEVICEMODE\_CONTINUOUS\_RANGING (([VL53L0X\\_DeviceModes](#)) 1)**

Definition at line 184 of file vl53l0x\_def.h.

**#define VL53L0X\_DEVICEMODE\_SINGLE\_HISTOGRAM (([VL53L0X\\_DeviceModes](#)) 2)**

Definition at line 185 of file vl53l0x\_def.h.

**#define VL53L0X\_DEVICEMODE\_CONTINUOUS\_TIMED\_RANGING (([VL53L0X\\_DeviceModes](#)) 3)**

Definition at line 186 of file vl53l0x\_def.h.

**#define VL53L0X\_DEVICEMODE\_SINGLE\_ALS (([VL53L0X\\_DeviceModes](#)) 10)**

Definition at line 187 of file vl53l0x\_def.h.

**#define VL53L0X\_DEVICEMODE\_GPIO\_DRIVE (([VL53L0X\\_DeviceModes](#)) 20)**

Definition at line 188 of file vl53l0x\_def.h.

**#define VL53L0X\_DEVICEMODE\_GPIO\_OSC (([VL53L0X\\_DeviceModes](#)) 21)**

Definition at line 189 of file vl53l0x\_def.h.

## Typedef Documentation

**typedef [uint8\\_t](#) [VL53L0X\\_DeviceModes](#)**

Definition at line 181 of file vl53l0x\_def.h.

## Defines Histogram modes

Defines all possible Histogram modes for the device.

### Macros

- **#define [VL53L0X\\_HISTOGRAMMODE\\_DISABLED](#) (([VL53L0X\\_HistogramModes](#)) 0)**
- **#define [VL53L0X\\_HISTOGRAMMODE\\_REFERENCE\\_ONLY](#) (([VL53L0X\\_HistogramModes](#)) 1)**
- **#define [VL53L0X\\_HISTOGRAMMODE\\_RETURN\\_ONLY](#) (([VL53L0X\\_HistogramModes](#)) 2)**
- **#define [VL53L0X\\_HISTOGRAMMODE\\_BOTH](#) (([VL53L0X\\_HistogramModes](#)) 3)**

### Typedefs

- **typedef [uint8\\_t](#) [VL53L0X\\_HistogramModes](#)**

## Detailed Description

Defines all possible Histogram modes for the device.

## Macro Definition Documentation

**#define VL53L0X\_HISTOGRAMMODE\_DISABLED** (([VL53L0X\\_HistogramModes](#)) 0)

Histogram Disabled

Definition at line 201 of file vl53l0x\_def.h.

**#define VL53L0X\_HISTOGRAMMODE\_REFERENCE\_ONLY** (([VL53L0X\\_HistogramModes](#)) 1)

Histogram Reference array only

Definition at line 203 of file vl53l0x\_def.h.

**#define VL53L0X\_HISTOGRAMMODE\_RETURN\_ONLY** (([VL53L0X\\_HistogramModes](#)) 2)

Histogram Return array only

Definition at line 205 of file vl53l0x\_def.h.

**#define VL53L0X\_HISTOGRAMMODE\_BOTH** (([VL53L0X\\_HistogramModes](#)) 3)

Histogram both Reference and Return Arrays

Definition at line 207 of file vl53l0x\_def.h.

---

## Typedef Documentation

**typedef** [uint8\\_t](#) [VL53L0X\\_HistogramModes](#)

Definition at line 199 of file vl53l0x\_def.h.

## List of available Power Modes

List of available Power Modes.

### Macros

- **#define** [VL53L0X\\_POWERMODE\\_STANDBY\\_LEVEL1](#) (([VL53L0X\\_PowerModes](#)) 0)
- **#define** [VL53L0X\\_POWERMODE\\_STANDBY\\_LEVEL2](#) (([VL53L0X\\_PowerModes](#)) 1)
- **#define** [VL53L0X\\_POWERMODE\\_IDLE\\_LEVEL1](#) (([VL53L0X\\_PowerModes](#)) 2)
- **#define** [VL53L0X\\_POWERMODE\\_IDLE\\_LEVEL2](#) (([VL53L0X\\_PowerModes](#)) 3)

### Typedefs

- **typedef** [uint8\\_t](#) [VL53L0X\\_PowerModes](#)

---

## Detailed Description

List of available Power Modes.

---

## Macro Definition Documentation

**#define VL53L0X\_POWERMODE\_STANDBY\_LEVEL1** (([VL53L0X\\_PowerModes](#)) 0)

Standby level 1

Definition at line 220 of file vl53l0x\_def.h.

**#define VL53L0X\_POWERMODE\_STANDBY\_LEVEL2** (([VL53L0X\\_PowerModes](#)) 1)

Standby level 2

Definition at line 222 of file vl53l0x\_def.h.

**#define VL53L0X\_POWERMODE\_IDLE\_LEVEL1** (([VL53L0X\\_PowerModes](#)) 2)

Idle level 1

Definition at line 224 of file vl53l0x\_def.h.

**#define VL53L0X\_POWERMODE\_IDLE\_LEVEL2** (([VL53L0X\\_PowerModes](#)) 3)

Idle level 2

Definition at line 226 of file vl53l0x\_def.h.

---

## Typedef Documentation

**typedef** [uint8\\_t](#) [VL53L0X\\_PowerModes](#)

Definition at line 218 of file vl53l0x\_def.h.

## Defines the current status of the device

Defines the current status of the device.

### Macros

- **#define** [VL53L0X\\_STATE\\_POWERDOWN](#) (([VL53L0X\\_State](#)) 0)
- **#define** [VL53L0X\\_STATE\\_WAIT\\_STATICINIT](#) (([VL53L0X\\_State](#)) 1)
- **#define** [VL53L0X\\_STATE\\_STANDBY](#) (([VL53L0X\\_State](#)) 2)
- **#define** [VL53L0X\\_STATE\\_IDLE](#) (([VL53L0X\\_State](#)) 3)
- **#define** [VL53L0X\\_STATE\\_RUNNING](#) (([VL53L0X\\_State](#)) 4)
- **#define** [VL53L0X\\_STATE\\_UNKNOWN](#) (([VL53L0X\\_State](#)) 98)
- **#define** [VL53L0X\\_STATE\\_ERROR](#) (([VL53L0X\\_State](#)) 99)

### Typedefs

- **typedef** [uint8\\_t](#) [VL53L0X\\_State](#)

---

## Detailed Description

Defines the current status of the device.

---

## Macro Definition Documentation

**#define VL53L0X\_STATE\_POWERDOWN ((VL53L0X\_State) 0)**

Device is in HW reset

Definition at line 275 of file vl53l0x\_def.h.

**#define VL53L0X\_STATE\_WAIT\_STATICINIT ((VL53L0X\_State) 1)**

Device is initialized and wait for static initialization

Definition at line 277 of file vl53l0x\_def.h.

**#define VL53L0X\_STATE\_STANDBY ((VL53L0X\_State) 2)**

Device is in Low power Standby mode

Definition at line 279 of file vl53l0x\_def.h.

**#define VL53L0X\_STATE\_IDLE ((VL53L0X\_State) 3)**

Device has been initialized and ready to do measurements

Definition at line 281 of file vl53l0x\_def.h.

**#define VL53L0X\_STATE\_RUNNING ((VL53L0X\_State) 4)**

Device is performing measurement

Definition at line 283 of file vl53l0x\_def.h.

**#define VL53L0X\_STATE\_UNKNOWN ((VL53L0X\_State) 98)**

Device is in unknown state and need to be rebooted

Definition at line 285 of file vl53l0x\_def.h.

**#define VL53L0X\_STATE\_ERROR ((VL53L0X\_State) 99)**

Device is in error state and need to be rebooted

Definition at line 287 of file vl53l0x\_def.h.

## Typedef Documentation

**typedef [uint8\\_t VL53L0X\\_State](#)**

Definition at line 273 of file vl53l0x\_def.h.

## Defines the Polarity

of the Interrupt Defines the Polarity of the Interrupt

### Macros

- **#define [VL53L0X\\_INTERRUPTPOLARITY\\_LOW](#) ((VL53L0X\_InterruptPolarity)0)**
- **#define [VL53L0X\\_INTERRUPTPOLARITY\\_HIGH](#) ((VL53L0X\_InterruptPolarity) 1)**

## Typedefs

- typedef [uint8\\_t VL53L0X\\_InterruptPolarity](#)
- 

## Detailed Description

of the Interrupt Defines the Polarity of the Interrupt

---

## Macro Definition Documentation

**#define VL53L0X\_INTERRUPTPOLARITY\_LOW (([VL53L0X\\_InterruptPolarity](#)) 0)**

Set active low polarity best setup for falling edge.

Definition at line 498 of file vl53l0x\_def.h.

**#define VL53L0X\_INTERRUPTPOLARITY\_HIGH (([VL53L0X\\_InterruptPolarity](#)) 1)**

Set active high polarity best setup for rising edge.

Definition at line 500 of file vl53l0x\_def.h.

---

## Typedef Documentation

typedef [uint8\\_t VL53L0X\\_InterruptPolarity](#)

Definition at line 496 of file vl53l0x\_def.h.

---

## Vcsel Period Defines

Defines the range measurement for which to access the vcsel period.

### Macros

- #define [VL53L0X\\_VCSEL\\_PERIOD\\_PRE\\_RANGE](#) (([VL53L0X\\_VcselPeriod](#)) 0)
- #define [VL53L0X\\_VCSEL\\_PERIOD\\_FINAL\\_RANGE](#) (([VL53L0X\\_VcselPeriod](#)) 1)

### Typedefs

- typedef [uint8\\_t VL53L0X\\_VcselPeriod](#)
- 

## Detailed Description

Defines the range measurement for which to access the vcsel period.

---

## Macro Definition Documentation

**#define VL53L0X\_VCSEL\_PERIOD\_PRE\_RANGE** (([VL53L0X\\_VcseIPeriod](#)) 0)

Identifies the pre-range vcsel period.

Definition at line 512 of file vl53l0x\_def.h.

**#define VL53L0X\_VCSEL\_PERIOD\_FINAL\_RANGE** (([VL53L0X\\_VcseIPeriod](#)) 1)

Identifies the final range vcsel period.

Definition at line 514 of file vl53l0x\_def.h.

---

## Typedef Documentation

**typedef** [uint8\\_t VL53L0X\\_VcseIPeriod](#)

Definition at line 510 of file vl53l0x\_def.h.

## Defines the steps

carried out by the scheduler during a range measurement.

### Data Structures

- struct [VL53L0X\\_SchedulerSequenceSteps\\_t](#)
- 

## Detailed Description

carried out by the scheduler during a range measurement.

Defines the states of all the steps in the scheduler i.e. enabled/disabled.

## Defines the Polarity

of the Interrupt Defines the the sequence steps performed during ranging.

### Macros

- #define [VL53L0X\\_SEQUENCESTEP\\_TCC](#) (([VL53L0X\\_VcseIPeriod](#)) 0)
- #define [VL53L0X\\_SEQUENCESTEP\\_DSS](#) (([VL53L0X\\_VcseIPeriod](#)) 1)
- #define [VL53L0X\\_SEQUENCESTEP\\_MSRC](#) (([VL53L0X\\_VcseIPeriod](#)) 2)
- #define [VL53L0X\\_SEQUENCESTEP\\_PRE\\_RANGE](#) (([VL53L0X\\_VcseIPeriod](#)) 3)
- #define [VL53L0X\\_SEQUENCESTEP\\_FINAL\\_RANGE](#) (([VL53L0X\\_VcseIPeriod](#)) 4)
- #define [VL53L0X\\_SEQUENCESTEP\\_NUMBER\\_OF\\_CHECKS](#) 5

### Typedefs

- typedef [uint8\\_t VL53L0X\\_SequenceStepId](#)
-

## Detailed Description

of the Interrupt Defines the the sequence steps performed during ranging.

---

## Macro Definition Documentation

**#define VL53L0X\_SEQUENCESTEP\_TCC** (([VL53L0X\\_VcselPeriod](#)) 0)

Target CentreCheck identifier.

Definition at line 542 of file vl53l0x\_def.h.

**#define VL53L0X\_SEQUENCESTEP\_DSS** (([VL53L0X\\_VcselPeriod](#)) 1)

Dynamic Spad Selection function Identifier.

Definition at line 544 of file vl53l0x\_def.h.

**#define VL53L0X\_SEQUENCESTEP\_MSRC** (([VL53L0X\\_VcselPeriod](#)) 2)

Minimum Signal Rate Check function Identifier.

Definition at line 546 of file vl53l0x\_def.h.

**#define VL53L0X\_SEQUENCESTEP\_PRE\_RANGE** (([VL53L0X\\_VcselPeriod](#)) 3)

Pre-Range check Identifier.

Definition at line 548 of file vl53l0x\_def.h.

**#define VL53L0X\_SEQUENCESTEP\_FINAL\_RANGE** (([VL53L0X\\_VcselPeriod](#)) 4)

Final Range Check Identifier.

Definition at line 550 of file vl53l0x\_def.h.

**#define VL53L0X\_SEQUENCESTEP\_NUMBER\_OF\_CHECKS** 5

Number of Sequence Step Managed by the API.

Definition at line 553 of file vl53l0x\_def.h.

---

## Typedef Documentation

**typedef** [uint8\\_t](#) [VL53L0X\\_SequenceStepId](#)

Definition at line 540 of file vl53l0x\_def.h.

## General Macro Defines

General Macro Defines.

### Macros

- #define [VL53L0X\\_SETPARAMETERFIELD](#)(Dev, field, value) [PALDevDataSet](#)(Dev, CurrentParameters.field, value)



- #define [VL53L0X\\_GETPARAMETERFIELD](#)(Dev, field, variable) variable = [PALDevDataGet](#)(Dev, CurrentParameters).field
- #define [VL53L0X\\_SETARRAYPARAMETERFIELD](#)(Dev, field, index, value) [PALDevDataSet](#)(Dev, CurrentParameters.field[index], value)
- #define [VL53L0X\\_GETARRAYPARAMETERFIELD](#)(Dev, field, index, variable) variable = [PALDevDataGet](#)(Dev, CurrentParameters).field[index]
- #define [VL53L0X\\_SETDEVICESPECIFICPARAMETER](#)(Dev, field, value) [PALDevDataSet](#)(Dev, DeviceSpecificParameters.field, value)
- #define [VL53L0X\\_GETDEVICESPECIFICPARAMETER](#)(Dev, field) [PALDevDataGet](#)(Dev, DeviceSpecificParameters).field
- #define [VL53L0X\\_FIXPOINT1616TOFIXPOINT97](#)(Value) ([uint16\\_t](#))((Value>>9)&0xFFFF)
- #define [VL53L0X\\_FIXPOINT97TOFIXPOINT1616](#)(Value) ([FixPoint1616\\_t](#))(Value<<9)
- #define [VL53L0X\\_FIXPOINT1616TOFIXPOINT88](#)(Value) ([uint16\\_t](#))((Value>>8)&0xFFFF)
- #define [VL53L0X\\_FIXPOINT88TOFIXPOINT1616](#)(Value) ([FixPoint1616\\_t](#))(Value<<8)
- #define [VL53L0X\\_FIXPOINT1616TOFIXPOINT412](#)(Value) ([uint16\\_t](#))((Value>>4)&0xFFFF)
- #define [VL53L0X\\_FIXPOINT412TOFIXPOINT1616](#)(Value) ([FixPoint1616\\_t](#))(Value<<4)
- #define [VL53L0X\\_FIXPOINT1616TOFIXPOINT313](#)(Value) ([uint16\\_t](#))((Value>>3)&0xFFFF)
- #define [VL53L0X\\_FIXPOINT313TOFIXPOINT1616](#)(Value) ([FixPoint1616\\_t](#))(Value<<3)
- #define [VL53L0X\\_FIXPOINT1616TOFIXPOINT08](#)(Value) ([uint8\\_t](#))((Value>>8)&0x00FF)
- #define [VL53L0X\\_FIXPOINT08TOFIXPOINT1616](#)(Value) ([FixPoint1616\\_t](#))(Value<<8)
- #define [VL53L0X\\_FIXPOINT1616TOFIXPOINT53](#)(Value) ([uint8\\_t](#))((Value>>13)&0x00FF)
- #define [VL53L0X\\_FIXPOINT53TOFIXPOINT1616](#)(Value) ([FixPoint1616\\_t](#))(Value<<13)
- #define [VL53L0X\\_FIXPOINT1616TOFIXPOINT102](#)(Value) ([uint16\\_t](#))((Value>>14)&0x0FFF)
- #define [VL53L0X\\_FIXPOINT102TOFIXPOINT1616](#)(Value) ([FixPoint1616\\_t](#))(Value<<12)
- #define [VL53L0X\\_MAKEUINT16](#)(lsb, msb)

## Detailed Description

General Macro Defines.

## Macro Definition Documentation

**#define VL53L0X\_SETPARAMETERFIELD( Dev, field, value) [PALDevDataSet](#)(Dev, CurrentParameters.field, value)**

Definition at line 566 of file vl53l0x\_def.h.

**#define VL53L0X\_GETPARAMETERFIELD( Dev, field, variable) variable = [PALDevDataGet](#)(Dev, CurrentParameters).field**

Definition at line 569 of file vl53l0x\_def.h.

**#define VL53L0X\_SETARRAYPARAMETERFIELD( Dev, field, index, value) [PALDevDataSet](#)(Dev, CurrentParameters.field[index], value)**

Definition at line 573 of file vl53l0x\_def.h.

**#define VL53L0X\_GETARRAYPARAMETERFIELD( Dev, field, index, variable) variable = [PALDevDataGet](#)(Dev, CurrentParameters).field[index]**

Definition at line 576 of file vl53l0x\_def.h.

```
#define VL53L0X_SETDEVICESTRUCTUREPARAMETER( Dev, field,  
value) PALDevDataSet(Dev, DeviceSpecificParameters.field, value)
```

Definition at line 580 of file vl53l0x\_def.h.

```
#define VL53L0X_GETDEVICESTRUCTUREPARAMETER( Dev, field) PALDevDataGet(Dev,  
DeviceSpecificParameters).field
```

Definition at line 583 of file vl53l0x\_def.h.

```
#define VL53L0X_FIXPOINT1616TOFIXPOINT97( Value) (uint16\_t)((Value>>9)&0xFFFF)
```

Definition at line 587 of file vl53l0x\_def.h.

```
#define VL53L0X_FIXPOINT97TOFIXPOINT1616( Value) (FixPoint1616\_t)(Value<<9)
```

Definition at line 589 of file vl53l0x\_def.h.

```
#define VL53L0X_FIXPOINT1616TOFIXPOINT88( Value) (uint16\_t)((Value>>8)&0xFFFF)
```

Definition at line 592 of file vl53l0x\_def.h.

```
#define VL53L0X_FIXPOINT88TOFIXPOINT1616( Value) (FixPoint1616\_t)(Value<<8)
```

Definition at line 594 of file vl53l0x\_def.h.

```
#define VL53L0X_FIXPOINT1616TOFIXPOINT412( Value) (uint16\_t)((Value>>4)&0xFFFF)
```

Definition at line 597 of file vl53l0x\_def.h.

```
#define VL53L0X_FIXPOINT412TOFIXPOINT1616( Value) (FixPoint1616\_t)(Value<<4)
```

Definition at line 599 of file vl53l0x\_def.h.

```
#define VL53L0X_FIXPOINT1616TOFIXPOINT313( Value) (uint16\_t)((Value>>3)&0xFFFF)
```

Definition at line 602 of file vl53l0x\_def.h.

```
#define VL53L0X_FIXPOINT313TOFIXPOINT1616( Value) (FixPoint1616\_t)(Value<<3)
```

Definition at line 604 of file vl53l0x\_def.h.

```
#define VL53L0X_FIXPOINT1616TOFIXPOINT08( Value) (uint8\_t)((Value>>8)&0x00FF)
```

Definition at line 607 of file vl53l0x\_def.h.

```
#define VL53L0X_FIXPOINT08TOFIXPOINT1616( Value) (FixPoint1616\_t)(Value<<8)
```

Definition at line 609 of file vl53l0x\_def.h.

```
#define VL53L0X_FIXPOINT1616TOFIXPOINT53( Value) (uint8\_t)((Value>>13)&0x00FF)
```

Definition at line 612 of file vl53l0x\_def.h.

```
#define VL53L0X_FIXPOINT53TOFIXPOINT1616( Value) (FixPoint1616\_t)(Value<<13)
```

Definition at line 614 of file vl53l0x\_def.h.

```
#define VL53L0X_FIXPOINT1616TOFIXPOINT102( Value) (uint16\_t)((Value>>14)&0x0FFF)
```

Definition at line 617 of file vl53l0x\_def.h.

```
#define VL53L0X_FIXPOINT102TOFIXPOINT1616( Value) (FixPoint1616\_t)(Value<<12)
```

Definition at line 619 of file vl53l0x\_def.h.

```
#define VL53L0X_MAKEUINT16( lsb, msb)
```

```
Value: (uint16\_t) (((uint16\_t)msb)<<8) + \
(uint16\_t)lsb)
```

Definition at line 622 of file vl53l0x\_def.h.

## VL53L0X cut1.1 Device Specific Defines

Device specific defines.

### Modules

- [Device Error](#)
  - Device Error code. [Check Enable list](#)
  - Check Enable code. [Gpio Functionality](#)
  - Defines the different functionalities for the device GPIO(s) [Define Registers](#)
- List of all the defined registers.

### Detailed Description

Device specific defines.

To be adapted by implementer for the targeted device. VL53L0X cut1.1 Device Specific Defines

## Device Error

Device Error code.

### Macros

- #define [VL53L0X\\_DEVICEERROR\\_NONE](#) (([VL53L0X\\_DeviceError](#)) 0)

- #define [VL53L0X\\_DEVICEERROR\\_VCSELCONTINUITYTESTFAILURE](#) (([VL53L0X\\_DeviceError](#)) 1)
- #define [VL53L0X\\_DEVICEERROR\\_VCSELWATCHDOGTESTFAILURE](#) (([VL53L0X\\_DeviceError](#)) 2)
- #define [VL53L0X\\_DEVICEERROR\\_NOVHVVALUEFOUND](#) (([VL53L0X\\_DeviceError](#)) 3)
- #define [VL53L0X\\_DEVICEERROR\\_MSRCNOTARGET](#) (([VL53L0X\\_DeviceError](#)) 4)
- #define [VL53L0X\\_DEVICEERROR\\_SNRCHECK](#) (([VL53L0X\\_DeviceError](#)) 5)
- #define [VL53L0X\\_DEVICEERROR\\_RANGEPHASECHECK](#) (([VL53L0X\\_DeviceError](#)) 6)
- #define [VL53L0X\\_DEVICEERROR\\_SIGMATHRESHOLDCHECK](#) (([VL53L0X\\_DeviceError](#)) 7)
- #define [VL53L0X\\_DEVICEERROR\\_TCC](#) (([VL53L0X\\_DeviceError](#)) 8)
- #define [VL53L0X\\_DEVICEERROR\\_PHASECONSISTENCY](#) (([VL53L0X\\_DeviceError](#)) 9)
- #define [VL53L0X\\_DEVICEERROR\\_MINCLIP](#) (([VL53L0X\\_DeviceError](#)) 10)
- #define [VL53L0X\\_DEVICEERROR\\_RANGECOMPLETE](#) (([VL53L0X\\_DeviceError](#)) 11)
- #define [VL53L0X\\_DEVICEERROR\\_ALGOUNDERFLOW](#) (([VL53L0X\\_DeviceError](#)) 12)
- #define [VL53L0X\\_DEVICEERROR\\_ALGOOVERFLOW](#) (([VL53L0X\\_DeviceError](#)) 13)
- #define [VL53L0X\\_DEVICEERROR\\_RANGEIGNORETHRESHOLD](#) (([VL53L0X\\_DeviceError](#)) 14)

## Typedefs

- typedef [uint8\\_t VL53L0X\\_DeviceError](#)

---

## Detailed Description

Device Error code.

This enum is Device specific it should be updated in the implementation Use `VL53L0X_GetStatusErrorString()` to get the string. It is related to Status Register of the Device.

---

## Macro Definition Documentation

**#define VL53L0X\_DEVICEERROR\_NONE** (([VL53L0X\\_DeviceError](#)) 0)

0 NoError

Definition at line 56 of file vl53l0x\_device.h.

**#define VL53L0X\_DEVICEERROR\_VCSELCONTINUITYTESTFAILURE** (([VL53L0X\\_DeviceError](#)) 1)

Definition at line 58 of file vl53l0x\_device.h.

**#define VL53L0X\_DEVICEERROR\_VCSELWATCHDOGTESTFAILURE** (([VL53L0X\\_DeviceError](#)) 2)

Definition at line 59 of file vl53l0x\_device.h.

**#define VL53L0X\_DEVICEERROR\_NOVHVVALUEFOUND** (([VL53L0X\\_DeviceError](#)) 3)

Definition at line 60 of file vl53l0x\_device.h.

**#define VL53L0X\_DEVICEERROR\_MSRCNOTARGET** (([VL53L0X\\_DeviceError](#)) 4)

Definition at line 61 of file vl53l0x\_device.h.

**#define VL53L0X\_DEVICEERROR\_SNRCHECK (([VL53L0X\\_DeviceError](#)) 5)**

Definition at line 62 of file vl53l0x\_device.h.

**#define VL53L0X\_DEVICEERROR\_RANGEPHASECHECK (([VL53L0X\\_DeviceError](#)) 6)**

Definition at line 63 of file vl53l0x\_device.h.

**#define VL53L0X\_DEVICEERROR\_SIGMATHRESHOLDCHECK (([VL53L0X\\_DeviceError](#)) 7)**

Definition at line 64 of file vl53l0x\_device.h.

**#define VL53L0X\_DEVICEERROR\_TCC (([VL53L0X\\_DeviceError](#)) 8)**

Definition at line 65 of file vl53l0x\_device.h.

**#define VL53L0X\_DEVICEERROR\_PHASECONSISTENCY (([VL53L0X\\_DeviceError](#)) 9)**

Definition at line 66 of file vl53l0x\_device.h.

**#define VL53L0X\_DEVICEERROR\_MINCLIP (([VL53L0X\\_DeviceError](#)) 10)**

Definition at line 67 of file vl53l0x\_device.h.

**#define VL53L0X\_DEVICEERROR\_RANGECOMPLETE (([VL53L0X\\_DeviceError](#)) 11)**

Definition at line 68 of file vl53l0x\_device.h.

**#define VL53L0X\_DEVICEERROR\_ALGOUNDERFLOW (([VL53L0X\\_DeviceError](#)) 12)**

Definition at line 69 of file vl53l0x\_device.h.

**#define VL53L0X\_DEVICEERROR\_ALGOOVERFLOW (([VL53L0X\\_DeviceError](#)) 13)**

Definition at line 70 of file vl53l0x\_device.h.

**#define VL53L0X\_DEVICEERROR\_RANGEIGNORETHRESHOLD (([VL53L0X\\_DeviceError](#)) 14)**

Definition at line 71 of file vl53l0x\_device.h.

---

## Typedef Documentation

**typedef [uint8\\_t](#) [VL53L0X\\_DeviceError](#)**

Definition at line 54 of file vl53l0x\_device.h.

## Check Enable list

Check Enable code.

### Macros

- #define [VL53L0X\\_CHECKENABLE\\_SIGMA\\_FINAL\\_RANGE](#) 0
- #define [VL53L0X\\_CHECKENABLE\\_SIGNAL\\_RATE\\_FINAL\\_RANGE](#) 1
- #define [VL53L0X\\_CHECKENABLE\\_SIGNAL\\_REF\\_CLIP](#) 2
- #define [VL53L0X\\_CHECKENABLE\\_RANGE\\_IGNORE\\_THRESHOLD](#) 3
- #define [VL53L0X\\_CHECKENABLE\\_SIGNAL\\_RATE\\_MSRC](#) 4
- #define [VL53L0X\\_CHECKENABLE\\_SIGNAL\\_RATE\\_PRE\\_RANGE](#) 5
- #define [VL53L0X\\_CHECKENABLE\\_NUMBER\\_OF\\_CHECKS](#) 6

---

### Detailed Description

Check Enable code.

Define used to specify the LimitCheckId. Use [VL53L0X\\_GetLimitCheckInfo\(\)](#) to get the string.

---

### Macro Definition Documentation

**#define VL53L0X\_CHECKENABLE\_SIGMA\_FINAL\_RANGE 0**

Definition at line 84 of file vl53l0x\_device.h.

**#define VL53L0X\_CHECKENABLE\_SIGNAL\_RATE\_FINAL\_RANGE 1**

Definition at line 85 of file vl53l0x\_device.h.

**#define VL53L0X\_CHECKENABLE\_SIGNAL\_REF\_CLIP 2**

Definition at line 86 of file vl53l0x\_device.h.

**#define VL53L0X\_CHECKENABLE\_RANGE\_IGNORE\_THRESHOLD 3**

Definition at line 87 of file vl53l0x\_device.h.

**#define VL53L0X\_CHECKENABLE\_SIGNAL\_RATE\_MSRC 4**

Definition at line 88 of file vl53l0x\_device.h.

**#define VL53L0X\_CHECKENABLE\_SIGNAL\_RATE\_PRE\_RANGE 5**

Definition at line 89 of file vl53l0x\_device.h.

**#define VL53L0X\_CHECKENABLE\_NUMBER\_OF\_CHECKS 6**

## Gpio Functionality

Defines the different functionalities for the device GPIO(s)

### Macros

- `#define VL53L0X_GPIOFUNCTIONALITY_OFF ((VL53L0X_GpioFunctionality) 0)`
- `#define VL53L0X_GPIOFUNCTIONALITY_THRESHOLD_CROSSED_LOW ((VL53L0X_GpioFunctionality) 1)`
- `#define VL53L0X_GPIOFUNCTIONALITY_THRESHOLD_CROSSED_HIGH ((VL53L0X_GpioFunctionality) 2)`
- `#define VL53L0X_GPIOFUNCTIONALITY_THRESHOLD_CROSSED_OUT ((VL53L0X_GpioFunctionality) 3)`
- `#define VL53L0X_GPIOFUNCTIONALITY_NEW_MEASURE_READY ((VL53L0X_GpioFunctionality) 4)`

### Typedefs

- `typedef uint8_t VL53L0X_GpioFunctionality`

## Detailed Description

Defines the different functionalities for the device GPIO(s)

### Macro Definition Documentation

**`#define VL53L0X_GPIOFUNCTIONALITY_OFF ((VL53L0X_GpioFunctionality) 0)`**

NO Interrupt

Definition at line 102 of file vl53l0x\_device.h.

**`#define VL53L0X_GPIOFUNCTIONALITY_THRESHOLD_CROSSED_LOW ((VL53L0X_GpioFunctionality) 1)`**

Level Low (value < thresh\_low)

Definition at line 104 of file vl53l0x\_device.h.

**`#define VL53L0X_GPIOFUNCTIONALITY_THRESHOLD_CROSSED_HIGH ((VL53L0X_GpioFunctionality) 2)`**

Level High (value > thresh\_high)

Definition at line 106 of file vl53l0x\_device.h.

## #define

**VL53L0X\_GPIOFUNCTIONALITY\_THRESHOLD\_CROSSED\_OUT** (([VL53L0X\\_GpioFunctionality](#)) 3)

Out Of Window (value < thresh\_low OR value > thresh\_high)

Definition at line 108 of file vl53l0x\_device.h.

## #define

**VL53L0X\_GPIOFUNCTIONALITY\_NEW\_MEASURE\_READY** (([VL53L0X\\_GpioFunctionality](#)) 4)

New Sample Ready

Definition at line 111 of file vl53l0x\_device.h.

## Typedef Documentation

**typedef** [uint8\\_t VL53L0X\\_GpioFunctionality](#)

Definition at line 100 of file vl53l0x\_device.h.

## Define Registers

List of all the defined registers.

### Macros

- #define [VL53L0X\\_REG\\_SYSRANGE\\_START](#) 0x000
- #define [VL53L0X\\_REG\\_SYSRANGE\\_MODE\\_MASK](#) 0x0F  
*mask existing bit in [VL53L0X\\_REG\\_SYSRANGE\\_START](#)*
- #define [VL53L0X\\_REG\\_SYSRANGE\\_MODE\\_START\\_STOP](#) 0x01  
*bit 0 in [VL53L0X\\_REG\\_SYSRANGE\\_START](#) write 1 toggle state in continuous mode and arm next shot in single shot mode*
- #define [VL53L0X\\_REG\\_SYSRANGE\\_MODE\\_SINGLESOT](#) 0x00  
*bit 1 write 0 in [VL53L0X\\_REG\\_SYSRANGE\\_START](#) set single shot mode*
- #define [VL53L0X\\_REG\\_SYSRANGE\\_MODE\\_BACKTOBACK](#) 0x02  
*bit 1 write 1 in [VL53L0X\\_REG\\_SYSRANGE\\_START](#) set back-to-back operation mode*
- #define [VL53L0X\\_REG\\_SYSRANGE\\_MODE\\_TIMED](#) 0x04  
*bit 2 write 1 in [VL53L0X\\_REG\\_SYSRANGE\\_START](#) set timed operation mode*
- #define [VL53L0X\\_REG\\_SYSRANGE\\_MODE\\_HISTOGRAM](#) 0x08  
*bit 3 write 1 in [VL53L0X\\_REG\\_SYSRANGE\\_START](#) set histogram operation mode*
- #define [VL53L0X\\_REG\\_SYSTEM\\_THRESH\\_HIGH](#) 0x000C
- #define [VL53L0X\\_REG\\_SYSTEM\\_THRESH\\_LOW](#) 0x000E
- #define [VL53L0X\\_REG\\_SYSTEM\\_SEQUENCE\\_CONFIG](#) 0x0001
- #define [VL53L0X\\_REG\\_SYSTEM\\_RANGE\\_CONFIG](#) 0x0009
- #define [VL53L0X\\_REG\\_SYSTEM\\_INTERMEASUREMENT\\_PERIOD](#) 0x0004
- #define [VL53L0X\\_REG\\_SYSTEM\\_INTERRUPT\\_CONFIG\\_GPIO](#) 0x000A
- #define [VL53L0X\\_REG\\_SYSTEM\\_INTERRUPT\\_GPIO\\_DISABLED](#) 0x00
- #define [VL53L0X\\_REG\\_SYSTEM\\_INTERRUPT\\_GPIO\\_LEVEL\\_LOW](#) 0x01
- #define [VL53L0X\\_REG\\_SYSTEM\\_INTERRUPT\\_GPIO\\_LEVEL\\_HIGH](#) 0x02
- #define [VL53L0X\\_REG\\_SYSTEM\\_INTERRUPT\\_GPIO\\_OUT\\_OF\\_WINDOW](#) 0x03
- #define [VL53L0X\\_REG\\_SYSTEM\\_INTERRUPT\\_GPIO\\_NEW\\_SAMPLE\\_READY](#) 0x04
- #define [VL53L0X\\_REG\\_GPIO\\_HV\\_MUX\\_ACTIVE\\_HIGH](#) 0x0084



- #define [VL53L0X\\_REG\\_SYSTEM\\_INTERRUPT\\_CLEAR](#) 0x000B
- #define [VL53L0X\\_REG\\_RESULT\\_INTERRUPT\\_STATUS](#) 0x0013
- #define [VL53L0X\\_REG\\_RESULT\\_RANGE\\_STATUS](#) 0x0014
- #define [VL53L0X\\_REG\\_RESULT\\_CORE\\_PAGE](#) 1
- #define [VL53L0X\\_REG\\_RESULT\\_CORE\\_AMBIENT\\_WINDOW\\_EVENTS RTN](#) 0x00BC
- #define [VL53L0X\\_REG\\_RESULT\\_CORE\\_RANGING\\_TOTAL\\_EVENTS RTN](#) 0x00C0
- #define [VL53L0X\\_REG\\_RESULT\\_CORE\\_AMBIENT\\_WINDOW\\_EVENTS REF](#) 0x00D0
- #define [VL53L0X\\_REG\\_RESULT\\_CORE\\_RANGING\\_TOTAL\\_EVENTS REF](#) 0x00D4
- #define [VL53L0X\\_REG\\_RESULT\\_PEAK\\_SIGNAL\\_RATE REF](#) 0x00B6
- #define [VL53L0X\\_REG\\_ALGO\\_PART\\_TO\\_PART\\_RANGE\\_OFFSET MM](#) 0x0028
- #define [VL53L0X\\_REG\\_I2C\\_SLAVE\\_DEVICE\\_ADDRESS](#) 0x008a
- #define [VL53L0X\\_REG\\_MSRC\\_CONFIG\\_CONTROL](#) 0x0060
- #define [VL53L0X\\_REG\\_PRE\\_RANGE\\_CONFIG\\_MIN\\_SNR](#) 0x0027
- #define [VL53L0X\\_REG\\_PRE\\_RANGE\\_CONFIG\\_VALID\\_PHASE\\_LOW](#) 0x0056
- #define [VL53L0X\\_REG\\_PRE\\_RANGE\\_CONFIG\\_VALID\\_PHASE\\_HIGH](#) 0x0057
- #define [VL53L0X\\_REG\\_PRE\\_RANGE\\_MIN\\_COUNT\\_RATE RTN LIMIT](#) 0x0064
- #define [VL53L0X\\_REG\\_FINAL\\_RANGE\\_CONFIG\\_MIN\\_SNR](#) 0x0067
- #define [VL53L0X\\_REG\\_FINAL\\_RANGE\\_CONFIG\\_VALID\\_PHASE\\_LOW](#) 0x0047
- #define [VL53L0X\\_REG\\_FINAL\\_RANGE\\_CONFIG\\_VALID\\_PHASE\\_HIGH](#) 0x0048
- #define [VL53L0X\\_REG\\_FINAL\\_RANGE\\_CONFIG\\_MIN\\_COUNT\\_RATE RTN LIMIT](#) 0x0044
- #define [VL53L0X\\_REG\\_PRE\\_RANGE\\_CONFIG\\_SIGMA\\_THRESH HI](#) 0x0061
- #define [VL53L0X\\_REG\\_PRE\\_RANGE\\_CONFIG\\_SIGMA\\_THRESH LO](#) 0x0062
- #define [VL53L0X\\_REG\\_PRE\\_RANGE\\_CONFIG\\_VCSEL\\_PERIOD](#) 0x0050
- #define [VL53L0X\\_REG\\_PRE\\_RANGE\\_CONFIG\\_TIMEOUT\\_MACROP HI](#) 0x0051
- #define [VL53L0X\\_REG\\_PRE\\_RANGE\\_CONFIG\\_TIMEOUT\\_MACROP LO](#) 0x0052
- #define [VL53L0X\\_REG\\_SYSTEM\\_HISTOGRAM\\_BIN](#) 0x0081
- #define [VL53L0X\\_REG\\_HISTOGRAM\\_CONFIG\\_INITIAL\\_PHASE SELECT](#) 0x0033
- #define [VL53L0X\\_REG\\_HISTOGRAM\\_CONFIG\\_READOUT\\_CTRL](#) 0x0055
- #define [VL53L0X\\_REG\\_FINAL\\_RANGE\\_CONFIG\\_VCSEL\\_PERIOD](#) 0x0070
- #define [VL53L0X\\_REG\\_FINAL\\_RANGE\\_CONFIG\\_TIMEOUT\\_MACROP HI](#) 0x0071
- #define [VL53L0X\\_REG\\_FINAL\\_RANGE\\_CONFIG\\_TIMEOUT\\_MACROP LO](#) 0x0072
- #define [VL53L0X\\_REG\\_CROSSTALK\\_COMPENSATION\\_PEAK\\_RATE MCPS](#) 0x0020
- #define [VL53L0X\\_REG\\_MSRC\\_CONFIG\\_TIMEOUT\\_MACROP](#) 0x0046
- #define [VL53L0X\\_REG\\_SOFT\\_RESET\\_GO2\\_SOFT\\_RESET N](#) 0x00bf
- #define [VL53L0X\\_REG\\_IDENTIFICATION\\_MODEL\\_ID](#) 0x00c0
- #define [VL53L0X\\_REG\\_IDENTIFICATION\\_REVISION\\_ID](#) 0x00c2
- #define [VL53L0X\\_REG\\_OSC\\_CALIBRATE\\_VAL](#) 0x00f8
- #define [VL53L0X\\_SIGMA\\_ESTIMATE\\_MAX\\_VALUE](#) 65535
- #define [VL53L0X\\_REG\\_GLOBAL\\_CONFIG\\_VCSEL\\_WIDTH](#) 0x032
- #define [VL53L0X\\_REG\\_GLOBAL\\_CONFIG\\_SPAD\\_ENABLES REF 0](#) 0x0B0
- #define [VL53L0X\\_REG\\_GLOBAL\\_CONFIG\\_SPAD\\_ENABLES REF 1](#) 0x0B1
- #define [VL53L0X\\_REG\\_GLOBAL\\_CONFIG\\_SPAD\\_ENABLES REF 2](#) 0x0B2
- #define [VL53L0X\\_REG\\_GLOBAL\\_CONFIG\\_SPAD\\_ENABLES REF 3](#) 0x0B3
- #define [VL53L0X\\_REG\\_GLOBAL\\_CONFIG\\_SPAD\\_ENABLES REF 4](#) 0x0B4
- #define [VL53L0X\\_REG\\_GLOBAL\\_CONFIG\\_SPAD\\_ENABLES REF 5](#) 0x0B5
- #define [VL53L0X\\_REG\\_GLOBAL\\_CONFIG\\_REF\\_EN\\_START\\_SELECT](#) 0xB6
- #define [VL53L0X\\_REG\\_DYNAMIC\\_SPAD\\_NUM\\_REQUESTED REF SPAD](#) 0x4E /\* 0x14E \*/
- #define [VL53L0X\\_REG\\_DYNAMIC\\_SPAD\\_REF\\_EN\\_START\\_OFFSET](#) 0x4F /\* 0x14F \*/
- #define [VL53L0X\\_REG\\_POWER\\_MANAGEMENT\\_GO1\\_POWER\\_FORCE](#) 0x80
- #define [VL53L0X\\_SPEED\\_OF\\_LIGHT\\_IN\\_AIR](#) 2997
- #define [VL53L0X\\_REG\\_VHV\\_CONFIG\\_PAD\\_SCL\\_SDA\\_EXTSUP HV](#) 0x0089
- #define [VL53L0X\\_REG\\_ALGO\\_PHASECAL\\_LIM](#) 0x0030 /\* 0x130 \*/
- #define [VL53L0X\\_REG\\_ALGO\\_PHASECAL\\_CONFIG\\_TIMEOUT](#) 0x0030

---

## Detailed Description

List of all the defined registers.

---

### Macro Definition Documentation

**#define VL53L0X\_REG\_SYSRANGE\_START 0x000**

Definition at line 123 of file vl53l0x\_device.h.

**#define VL53L0X\_REG\_SYSRANGE\_MODE\_MASK 0x0F**

mask existing bit in [VL53L0X\\_REG\\_SYSRANGE\\_START](#)

Definition at line 125 of file vl53l0x\_device.h.

**#define VL53L0X\_REG\_SYSRANGE\_MODE\_START\_STOP 0x01**

bit 0 in [VL53L0X\\_REG\\_SYSRANGE\\_START](#) write 1 toggle state in continuous mode and arm next shot in single shot mode

Definition at line 128 of file vl53l0x\_device.h.

**#define VL53L0X\_REG\_SYSRANGE\_MODE\_SINGLESOT 0x00**

bit 1 write 0 in [VL53L0X\\_REG\\_SYSRANGE\\_START](#) set single shot mode

Definition at line 130 of file vl53l0x\_device.h.

**#define VL53L0X\_REG\_SYSRANGE\_MODE\_BACKTOBACK 0x02**

bit 1 write 1 in [VL53L0X\\_REG\\_SYSRANGE\\_START](#) set back-to-back operation mode

Definition at line 133 of file vl53l0x\_device.h.

**#define VL53L0X\_REG\_SYSRANGE\_MODE\_TIMED 0x04**

bit 2 write 1 in [VL53L0X\\_REG\\_SYSRANGE\\_START](#) set timed operation mode

Definition at line 136 of file vl53l0x\_device.h.

**#define VL53L0X\_REG\_SYSRANGE\_MODE\_HISTOGRAM 0x08**

bit 3 write 1 in [VL53L0X\\_REG\\_SYSRANGE\\_START](#) set histogram operation mode

Definition at line 139 of file vl53l0x\_device.h.

**#define VL53L0X\_REG\_SYSTEM\_THRESH\_HIGH 0x000C**

Definition at line 142 of file vl53l0x\_device.h.

**#define VL53L0X\_REG\_SYSTEM\_THRESH\_LOW 0x000E**

Definition at line 143 of file vl53l0x\_device.h.

**#define VL53L0X\_REG\_SYSTEM\_SEQUENCE\_CONFIG 0x0001**

Definition at line 146 of file vl53l0x\_device.h.

**#define VL53L0X\_REG\_SYSTEM\_RANGE\_CONFIG 0x0009**

Definition at line 147 of file vl53l0x\_device.h.

**#define VL53L0X\_REG\_SYSTEM\_INTERMEASUREMENT\_PERIOD 0x0004**

Definition at line 148 of file vl53l0x\_device.h.

**#define VL53L0X\_REG\_SYSTEM\_INTERRUPT\_CONFIG\_GPIO 0x000A**

Definition at line 151 of file vl53l0x\_device.h.

**#define VL53L0X\_REG\_SYSTEM\_INTERRUPT\_GPIO\_DISABLED 0x00**

Definition at line 152 of file vl53l0x\_device.h.

**#define VL53L0X\_REG\_SYSTEM\_INTERRUPT\_GPIO\_LEVEL\_LOW 0x01**

Definition at line 153 of file vl53l0x\_device.h.

**#define VL53L0X\_REG\_SYSTEM\_INTERRUPT\_GPIO\_LEVEL\_HIGH 0x02**

Definition at line 154 of file vl53l0x\_device.h.

**#define VL53L0X\_REG\_SYSTEM\_INTERRUPT\_GPIO\_OUT\_OF\_WINDOW 0x03**

Definition at line 155 of file vl53l0x\_device.h.

**#define VL53L0X\_REG\_SYSTEM\_INTERRUPT\_GPIO\_NEW\_SAMPLE\_READY 0x04**

Definition at line 156 of file vl53l0x\_device.h.

**#define VL53L0X\_REG\_GPIO\_HV\_MUX\_ACTIVE\_HIGH 0x0084**

Definition at line 158 of file vl53l0x\_device.h.

**#define VL53L0X\_REG\_SYSTEM\_INTERRUPT\_CLEAR 0x000B**

Definition at line 161 of file vl53l0x\_device.h.

**#define VL53L0X\_REG\_RESULT\_INTERRUPT\_STATUS 0x0013**

Definition at line 164 of file vl53l0x\_device.h.

**#define VL53L0X\_REG\_RESULT\_RANGE\_STATUS 0x0014**

Definition at line 165 of file vl53l0x\_device.h.

**#define VL53L0X\_REG\_RESULT\_CORE\_PAGE 1**

Definition at line 167 of file vl53l0x\_device.h.

**#define VL53L0X\_REG\_RESULT\_CORE\_AMBIENT\_WINDOW\_EVENTS\_RTN 0x00BC**

Definition at line 168 of file vl53l0x\_device.h.

**#define VL53L0X\_REG\_RESULT\_CORE\_RANGING\_TOTAL\_EVENTS\_RTN 0x00C0**

Definition at line 169 of file vl53l0x\_device.h.

**#define VL53L0X\_REG\_RESULT\_CORE\_AMBIENT\_WINDOW\_EVENTS\_REF 0x00D0**

Definition at line 170 of file vl53l0x\_device.h.

**#define VL53L0X\_REG\_RESULT\_CORE\_RANGING\_TOTAL\_EVENTS\_REF 0x00D4**

Definition at line 171 of file vl53l0x\_device.h.

**#define VL53L0X\_REG\_RESULT\_PEAK\_SIGNAL\_RATE\_REF 0x00B6**

Definition at line 172 of file vl53l0x\_device.h.

**#define VL53L0X\_REG\_ALGO\_PART\_TO\_PART\_RANGE\_OFFSET\_MM 0x0028**

Definition at line 176 of file vl53l0x\_device.h.

**#define VL53L0X\_REG\_I2C\_SLAVE\_DEVICE\_ADDRESS 0x008a**

Definition at line 178 of file vl53l0x\_device.h.

**#define VL53L0X\_REG\_MSRC\_CONFIG\_CONTROL 0x0060**

Definition at line 181 of file vl53l0x\_device.h.

**#define VL53L0X\_REG\_PRE\_RANGE\_CONFIG\_MIN\_SNR 0x0027**

Definition at line 183 of file vl53l0x\_device.h.

**#define VL53L0X\_REG\_PRE\_RANGE\_CONFIG\_VALID\_PHASE\_LOW 0x0056**

Definition at line 184 of file vl53l0x\_device.h.

**#define VL53L0X\_REG\_PRE\_RANGE\_CONFIG\_VALID\_PHASE\_HIGH 0x0057**

Definition at line 185 of file vl53l0x\_device.h.

**#define VL53L0X\_REG\_PRE\_RANGE\_MIN\_COUNT\_RATE\_RTN\_LIMIT 0x0064**

Definition at line 186 of file vl53l0x\_device.h.

**#define VL53L0X\_REG\_FINAL\_RANGE\_CONFIG\_MIN\_SNR 0x0067**

Definition at line 188 of file vl53l0x\_device.h.

**#define VL53L0X\_REG\_FINAL\_RANGE\_CONFIG\_VALID\_PHASE\_LOW 0x0047**

Definition at line 189 of file vl53l0x\_device.h.

**#define VL53L0X\_REG\_FINAL\_RANGE\_CONFIG\_VALID\_PHASE\_HIGH 0x0048**

Definition at line 190 of file vl53l0x\_device.h.

**#define VL53L0X\_REG\_FINAL\_RANGE\_CONFIG\_MIN\_COUNT\_RATE\_RTN\_LIMIT 0x0044**

Definition at line 191 of file vl53l0x\_device.h.

**#define VL53L0X\_REG\_PRE\_RANGE\_CONFIG\_SIGMA\_THRESH\_HI 0x0061**

Definition at line 194 of file vl53l0x\_device.h.

**#define VL53L0X\_REG\_PRE\_RANGE\_CONFIG\_SIGMA\_THRESH\_LO 0x0062**

Definition at line 195 of file vl53l0x\_device.h.

**#define VL53L0X\_REG\_PRE\_RANGE\_CONFIG\_VCSEL\_PERIOD 0x0050**

Definition at line 198 of file vl53l0x\_device.h.

**#define VL53L0X\_REG\_PRE\_RANGE\_CONFIG\_TIMEOUT\_MACROP\_HI 0x0051**

Definition at line 199 of file vl53l0x\_device.h.

**#define VL53L0X\_REG\_PRE\_RANGE\_CONFIG\_TIMEOUT\_MACROP\_LO 0x0052**

Definition at line 200 of file vl53l0x\_device.h.

**#define VL53L0X\_REG\_SYSTEM\_HISTOGRAM\_BIN 0x0081**

Definition at line 202 of file vl53l0x\_device.h.

**#define VL53L0X\_REG\_HISTOGRAM\_CONFIG\_INITIAL\_PHASE\_SELECT 0x0033**

Definition at line 203 of file vl53l0x\_device.h.

**#define VL53L0X\_REG\_HISTOGRAM\_CONFIG\_READOUT\_CTRL 0x0055**

Definition at line 204 of file vl53l0x\_device.h.

**#define VL53L0X\_REG\_FINAL\_RANGE\_CONFIG\_VCSEL\_PERIOD 0x0070**

Definition at line 206 of file vl53l0x\_device.h.

**#define VL53L0X\_REG\_FINAL\_RANGE\_CONFIG\_TIMEOUT\_MACROP\_HI 0x0071**

Definition at line 207 of file vl53l0x\_device.h.

**#define VL53L0X\_REG\_FINAL\_RANGE\_CONFIG\_TIMEOUT\_MACROP\_LO 0x0072**

Definition at line 208 of file vl53l0x\_device.h.

**#define VL53L0X\_REG\_CROSSTALK\_COMPENSATION\_PEAK\_RATE\_MCPS 0x0020**

Definition at line 209 of file vl53l0x\_device.h.

**#define VL53L0X\_REG\_MSRC\_CONFIG\_TIMEOUT\_MACROP 0x0046**

Definition at line 211 of file vl53l0x\_device.h.

**#define VL53L0X\_REG\_SOFT\_RESET\_GO2\_SOFT\_RESET\_N 0x00bf**

Definition at line 214 of file vl53l0x\_device.h.

**#define VL53L0X\_REG\_IDENTIFICATION\_MODEL\_ID 0x00c0**

Definition at line 215 of file vl53l0x\_device.h.

**#define VL53L0X\_REG\_IDENTIFICATION\_REVISION\_ID 0x00c2**

Definition at line 216 of file vl53l0x\_device.h.

**#define VL53L0X\_REG\_OSC\_CALIBRATE\_VAL 0x00f8**

Definition at line 218 of file vl53l0x\_device.h.

**#define VL53L0X\_SIGMA\_ESTIMATE\_MAX\_VALUE 65535**

Definition at line 221 of file vl53l0x\_device.h.

**#define VL53L0X\_REG\_GLOBAL\_CONFIG\_VCSEL\_WIDTH 0x032**

Definition at line 224 of file vl53l0x\_device.h.

**#define VL53L0X\_REG\_GLOBAL\_CONFIG\_SPAD\_ENABLES\_REF\_0 0x0B0**

Definition at line 225 of file vl53l0x\_device.h.

**#define VL53L0X\_REG\_GLOBAL\_CONFIG\_SPAD\_ENABLES\_REF\_1 0x0B1**

Definition at line 226 of file vl53l0x\_device.h.

**#define VL53L0X\_REG\_GLOBAL\_CONFIG\_SPAD\_ENABLES\_REF\_2 0x0B2**

Definition at line 227 of file vl53l0x\_device.h.

**#define VL53L0X\_REG\_GLOBAL\_CONFIG\_SPAD\_ENABLES\_REF\_3 0x0B3**

Definition at line 228 of file vl53l0x\_device.h.

**#define VL53L0X\_REG\_GLOBAL\_CONFIG\_SPAD\_ENABLES\_REF\_4 0x0B4**

Definition at line 229 of file vl53l0x\_device.h.

**#define VL53L0X\_REG\_GLOBAL\_CONFIG\_SPAD\_ENABLES\_REF\_5 0x0B5**

Definition at line 230 of file vl53l0x\_device.h.

**#define VL53L0X\_REG\_GLOBAL\_CONFIG\_REF\_EN\_START\_SELECT 0xB6**

Definition at line 232 of file vl53l0x\_device.h.

**#define VL53L0X\_REG\_DYNAMIC\_SPAD\_NUM\_REQUESTED\_REF\_SPAD 0x4E /\* 0x14E \*/**

Definition at line 233 of file vl53l0x\_device.h.

**#define VL53L0X\_REG\_DYNAMIC\_SPAD\_REF\_EN\_START\_OFFSET 0x4F /\* 0x14F \*/**

Definition at line 234 of file vl53l0x\_device.h.

**#define VL53L0X\_REG\_POWER\_MANAGEMENT\_GO1\_POWER\_FORCE 0x80**

Definition at line 235 of file vl53l0x\_device.h.

**#define VL53L0X\_SPEED\_OF\_LIGHT\_IN\_AIR 2997**

Definition at line 241 of file vl53l0x\_device.h.

**#define VL53L0X\_REG\_VHV\_CONFIG\_PAD\_SCL\_SDA\_\_EXTSUP\_HV 0x0089**

Definition at line 243 of file vl53l0x\_device.h.

**#define VL53L0X\_REG\_ALGO\_PHASECAL\_LIM 0x0030 /\* 0x130 \*/**

Definition at line 245 of file vl53l0x\_device.h.

**#define VL53L0X\_REG\_ALGO\_PHASECAL\_CONFIG\_TIMEOUT 0x0030**

Definition at line 246 of file vl53l0x\_device.h.

---

## Data Structure Documentation

### VL53L0X\_Dev\_t Struct Reference

Generic PAL device type that does link between API and platform abstraction layer.

#include <vl53l0x\_platform.h>

#### Data Fields

- [VL53L0X\\_DevData\\_t Data](#)
- [uint8\\_t I2cDevAddr](#)
- [uint8\\_t comms\\_type](#)
- [uint16\\_t comms\\_speed\\_khz](#)

---

#### Detailed Description

Generic PAL device type that does link between API and platform abstraction layer.

Definition at line 58 of file vl53l0x\_platform.h.

---

#### Field Documentation

##### [VL53L0X\\_DevData\\_t](#) VL53L0X\_Dev\_t::Data

embed ST Ewok Dev data as "Data" user specific field

Definition at line 59 of file vl53l0x\_platform.h.

##### [uint8\\_t](#) VL53L0X\_Dev\_t::I2cDevAddr

i2c device address user specific field

Definition at line 62 of file vl53l0x\_platform.h.



#### **uint8\_t VL53L0X\_Dev\_t::comms\_type**

Type of comms : VL53L0X\_COMMS\_I2C or VL53L0X\_COMMS\_SPI

Definition at line 63 of file vl53l0x\_platform.h.

#### **uint16\_t VL53L0X\_Dev\_t::comms\_speed\_khz**

Comms speed [kHz] : typically 400kHz for I2C

Definition at line 64 of file vl53l0x\_platform.h.

---

The documentation for this struct was generated from the following file:

- [vl53l0x\\_platform.h](#)
- 

## **VL53L0X\_DevData\_t Struct Reference**

VL53L0X PAL device ST private data structure

End user should never access any of these field directly.

```
#include <vl53l0x_def.h>
```

### **Data Fields**

- [VL53L0X\\_DMaxData\\_t DMaxData](#)
  - [int32\\_t Part2PartOffsetNVMMicroMeter](#)
  - [int32\\_t Part2PartOffsetAdjustmentNVMMicroMeter](#)
  - [VL53L0X\\_DeviceParameters\\_t CurrentParameters](#)
  - [VL53L0X\\_RangingMeasurementData\\_t LastRangeMeasure](#)
  - [VL53L0X\\_HistogramMeasurementData\\_t LastHistogramMeasure](#)
  - [VL53L0X\\_DeviceSpecificParameters\\_t DeviceSpecificParameters](#)
  - [VL53L0X\\_SpadData\\_t SpadData](#)
  - [uint8\\_t SequenceConfig](#)
  - [uint8\\_t RangeFractionalEnable](#)
  - [VL53L0X\\_State PalState](#)
  - [VL53L0X\\_PowerModes PowerMode](#)
  - [uint16\\_t SigmaEstRefArray](#)
  - [uint16\\_t SigmaEstEffPulseWidth](#)
  - [uint16\\_t SigmaEstEffAmbWidth](#)
  - [uint8\\_t StopVariable](#)
  - [uint16\\_t targetRefRate](#)
  - [FixPoint1616\\_t SigmaEstimate](#)
  - [FixPoint1616\\_t SignalEstimate](#)
  - [FixPoint1616\\_t LastSignalRefMcps](#)
  - [uint8\\_t \\* pTuningSettingsPointer](#)
  - [uint8\\_t UseInternalTuningSettings](#)
  - [uint16\\_t LinearityCorrectiveGain](#)
  - [uint16\\_t DmaxCalRangeMilliMeter](#)
  - [FixPoint1616\\_t DmaxCalSignalRateRtnMegaCps](#)
- 

### **Detailed Description**

VL53L0X PAL device ST private data structure

End user should never access any of these field directly.

These must never access directly but only via macro

Definition at line 433 of file vl53l0x\_def.h.

## Field Documentation

### [VL53L0X\\_DMaxData\\_t VL53L0X\\_DevData\\_t::DMaxData](#)

Dmax Data

Definition at line 434 of file vl53l0x\_def.h.

### [int32\\_t VL53L0X\\_DevData\\_t::Part2PartOffsetNVMMicroMeter](#)

backed up NVM value

Definition at line 436 of file vl53l0x\_def.h.

### [int32\\_t VL53L0X\\_DevData\\_t::Part2PartOffsetAdjustmentNVMMicroMeter](#)

backed up NVM value representing additional offset adjustment

Definition at line 438 of file vl53l0x\_def.h.

### [VL53L0X\\_DeviceParameters\\_t VL53L0X\\_DevData\\_t::CurrentParameters](#)

Current Device Parameter

Definition at line 440 of file vl53l0x\_def.h.

### [VL53L0X\\_RangingMeasurementData\\_t VL53L0X\\_DevData\\_t::LastRangeMeasure](#)

Ranging Data

Definition at line 442 of file vl53l0x\_def.h.

### [VL53L0X\\_HistogramMeasurementData\\_t VL53L0X\\_DevData\\_t::LastHistogramMeasure](#)

Histogram Data

Definition at line 444 of file vl53l0x\_def.h.

### [VL53L0X\\_DeviceSpecificParameters\\_t VL53L0X\\_DevData\\_t::DeviceSpecificParameters](#)

Parameters specific to the device

Definition at line 446 of file vl53l0x\_def.h.

### [VL53L0X\\_SpadData\\_t VL53L0X\\_DevData\\_t::SpadData](#)

Spad Data

Definition at line 448 of file vl53l0x\_def.h.

### [uint8\\_t VL53L0X\\_DevData\\_t::SequenceConfig](#)

Internal value for the sequence config

Definition at line 450 of file vl53l0x\_def.h.

### [uint8\\_t VL53L0X\\_DevData\\_t::RangeFractionalEnable](#)

Enable/Disable fractional part of ranging data

Definition at line 452 of file vl53l0x\_def.h.

#### **VL53L0X\_State VL53L0X\_DevData\_t::PalState**

Current state of the PAL for this device

Definition at line 454 of file vl53l0x\_def.h.

#### **VL53L0X\_PowerModes VL53L0X\_DevData\_t::PowerMode**

Current Power Mode

Definition at line 456 of file vl53l0x\_def.h.

#### **uint16\_t VL53L0X\_DevData\_t::SigmaEstRefArray**

Reference array sigma value in 1/100th of [mm] e.g. 100 = 1mm

Definition at line 458 of file vl53l0x\_def.h.

#### **uint16\_t VL53L0X\_DevData\_t::SigmaEstEffPulseWidth**

Effective Pulse width for sigma estimate in 1/100th of ns e.g. 900 = 9.0ns

Definition at line 460 of file vl53l0x\_def.h.

#### **uint16\_t VL53L0X\_DevData\_t::SigmaEstEffAmbWidth**

Effective Ambient width for sigma estimate in 1/100th of ns e.g. 500 = 5.0ns

Definition at line 463 of file vl53l0x\_def.h.

#### **uint8\_t VL53L0X\_DevData\_t::StopVariable**

StopVariable used during the stop sequence

Definition at line 466 of file vl53l0x\_def.h.

#### **uint16\_t VL53L0X\_DevData\_t::targetRefRate**

Target Ambient Rate for Ref spad management

Definition at line 468 of file vl53l0x\_def.h.

#### **FixPoint1616\_t VL53L0X\_DevData\_t::SigmaEstimate**

Sigma Estimate - based on ambient & VCSEL rates and signal\_total\_events

Definition at line 470 of file vl53l0x\_def.h.

#### **FixPoint1616\_t VL53L0X\_DevData\_t::SignalEstimate**

Signal Estimate - based on ambient & VCSEL rates and cross talk

Definition at line 473 of file vl53l0x\_def.h.

#### **FixPoint1616\_t VL53L0X\_DevData\_t::LastSignalRefMcps**

Latest Signal ref in Mcps

Definition at line 475 of file vl53l0x\_def.h.

#### **uint8\_t\* VL53L0X\_DevData\_t::pTuningSettingsPointer**

Pointer for Tuning Settings table

Definition at line 477 of file vl53l0x\_def.h.

#### **uint8\_t VL53L0X\_DevData\_t::UseInternalTuningSettings**

Indicate if we use Tuning Settings table

Definition at line 479 of file vl53l0x\_def.h.

#### **uint16\_t VL53L0X\_DevData\_t::LinearityCorrectiveGain**

Linearity Corrective Gain value in x1000

Definition at line 481 of file vl53l0x\_def.h.

#### **uint16\_t VL53L0X\_DevData\_t::DmaxCalRangeMilliMeter**

Dmax Calibration Range millimeter

Definition at line 483 of file vl53l0x\_def.h.

#### **FixPoint1616\_t VL53L0X\_DevData\_t::DmaxCalSignalRateRtnMegaCps**

Dmax Calibration Signal Rate Return MegaCps

Definition at line 485 of file vl53l0x\_def.h.

---

The documentation for this struct was generated from the following file:

- [vl53l0x\\_def.h](#)
- 

## **VL53L0X\_DeviceInfo\_t Struct Reference**

Defines the parameters of the Get Device Info Functions.

```
#include <vl53l0x_def.h>
```

### **Data Fields**

- char [Name](#) [[VL53L0X\\_MAX\\_STRING\\_LENGTH](#)]
  - char [Type](#) [[VL53L0X\\_MAX\\_STRING\\_LENGTH](#)]
  - char [ProductId](#) [[VL53L0X\\_MAX\\_STRING\\_LENGTH](#)]
  - [uint8\\_t](#) [ProductType](#)
  - [uint8\\_t](#) [ProductRevisionMajor](#)
  - [uint8\\_t](#) [ProductRevisionMinor](#)
- 

### **Detailed Description**

Defines the parameters of the Get Device Info Functions.

Definition at line 110 of file vl53l0x\_def.h.

---

### **Field Documentation**

**char VL53L0X\_DeviceInfo\_t::Name**[\[VL53L0X\\_MAX\\_STRING\\_LENGTH\]](#)

Name of the Device e.g. Left\_Distance

Definition at line 111 of file vl53l0x\_def.h.

**char VL53L0X\_DeviceInfo\_t::Type**[\[VL53L0X\\_MAX\\_STRING\\_LENGTH\]](#)

Type of the Device e.g VL53L0X

Definition at line 113 of file vl53l0x\_def.h.

**char VL53L0X\_DeviceInfo\_t::ProductId**[\[VL53L0X\\_MAX\\_STRING\\_LENGTH\]](#)

Product Identifier String

Definition at line 115 of file vl53l0x\_def.h.

**uint8\_t VL53L0X\_DeviceInfo\_t::ProductType**

Product Type, VL53L0X = 1, VL53L1 = 2

Definition at line 117 of file vl53l0x\_def.h.

**uint8\_t VL53L0X\_DeviceInfo\_t::ProductRevisionMajor**

Product revision major

Definition at line 119 of file vl53l0x\_def.h.

**uint8\_t VL53L0X\_DeviceInfo\_t::ProductRevisionMinor**

Product revision minor

Definition at line 121 of file vl53l0x\_def.h.

---

The documentation for this struct was generated from the following file:

- [vl53l0x\\_def.h](#)
- 

## VL53L0X\_DeviceParameters\_t Struct Reference

Defines all parameters for the device.

```
#include <vl53l0x_def.h>
```

### Data Fields

- [VL53L0X\\_DeviceModes DeviceMode](#)
  - [VL53L0X\\_HistogramModes HistogramMode](#)
  - [uint32\\_t MeasurementTimingBudgetMicroSeconds](#)
  - [uint32\\_t InterMeasurementPeriodMilliSeconds](#)
  - [uint8\\_t XTalkCompensationEnable](#)
  - [uint16\\_t XTalkCompensationRangeMilliMeter](#)
  - [FixPoint1616\\_t XTalkCompensationRateMegaCps](#)
  - [int32\\_t RangeOffsetMicroMeters](#)
  - [uint8\\_t LimitChecksEnable](#) [\[VL53L0X\\_CHECKENABLE\\_NUMBER\\_OF\\_CHECKS\]](#)
  - [uint8\\_t LimitChecksStatus](#) [\[VL53L0X\\_CHECKENABLE\\_NUMBER\\_OF\\_CHECKS\]](#)
  - [FixPoint1616\\_t LimitChecksValue](#) [\[VL53L0X\\_CHECKENABLE\\_NUMBER\\_OF\\_CHECKS\]](#)
  - [uint8\\_t WrapAroundCheckEnable](#)
- 

### Detailed Description

Defines all parameters for the device.

Definition at line 234 of file vl53l0x\_def.h.

---

## Field Documentation

### [VL53L0X\\_DeviceModes](#) **VL53L0X\_DeviceParameters\_t::DeviceMode**

Defines type of measurement to be done for the next measure

Definition at line 235 of file vl53l0x\_def.h.

### [VL53L0X\\_HistogramModes](#) **VL53L0X\_DeviceParameters\_t::HistogramMode**

Defines type of histogram measurement to be done for the next measure

Definition at line 237 of file vl53l0x\_def.h.

### [uint32\\_t](#) **VL53L0X\_DeviceParameters\_t::MeasurementTimingBudgetMicroSeconds**

Defines the allowed total time for a single measurement

Definition at line 240 of file vl53l0x\_def.h.

### [uint32\\_t](#) **VL53L0X\_DeviceParameters\_t::InterMeasurementPeriodMilliSeconds**

Defines time between two consecutive measurements (between two measurement starts). If set to 0 means back-to-back mode

Definition at line 242 of file vl53l0x\_def.h.

### [uint8\\_t](#) **VL53L0X\_DeviceParameters\_t::XTalkCompensationEnable**

Tells if Crosstalk compensation shall be enable or not

Definition at line 245 of file vl53l0x\_def.h.

### [uint16\\_t](#) **VL53L0X\_DeviceParameters\_t::XTalkCompensationRangeMilliMeter**

CrossTalk compensation range in millimeter

Definition at line 247 of file vl53l0x\_def.h.

### [FixPoint1616\\_t](#) **VL53L0X\_DeviceParameters\_t::XTalkCompensationRateMegaCps**

CrossTalk compensation rate in Mega counts per seconds. Expressed in 16.16 fixed point format.

Definition at line 249 of file vl53l0x\_def.h.

### [int32\\_t](#) **VL53L0X\_DeviceParameters\_t::RangeOffsetMicroMeters**

Range offset adjustment (mm).

Definition at line 252 of file vl53l0x\_def.h.

### [uint8\\_t](#)

### **VL53L0X\_DeviceParameters\_t::LimitChecksEnable[[VL53L0X\\_CHECKENABLE\\_NUMBER\\_OF\\_CHECKS](#)]**

This Array store all the Limit Check enable for this device.

Definition at line 255 of file vl53l0x\_def.h.

### [uint8\\_t](#)

### **VL53L0X\_DeviceParameters\_t::LimitChecksStatus[[VL53L0X\\_CHECKENABLE\\_NUMBER\\_OF\\_CHECKS](#)]**

This Array store all the Status of the check linked to last measurement.

Definition at line 257 of file vl53l0x\_def.h.

## [FixPoint1616\\_t](#)

**VL53L0X\_DeviceParameters\_t::LimitChecksValue**[\[VL53L0X\\_CHECKENABLE\\_NUMBER\\_OF\\_CHECKS\]](#)

This Array store all the Limit Check value for this device

Definition at line 260 of file vl53l0x\_def.h.

## [uint8\\_t VL53L0X\\_DeviceParameters\\_t::WrapAroundCheckEnable](#)

Tells if Wrap Around Check shall be enable or not

Definition at line 263 of file vl53l0x\_def.h.

---

The documentation for this struct was generated from the following file:

- [vl53l0x\\_def.h](#)

---

## VL53L0X\_DeviceSpecificParameters\_t Struct Reference

```
#include <vl53l0x_def.h>
```

### Data Fields

- [FixPoint1616\\_t OscFrequencyMHz](#)
- [uint16\\_t LastEncodedTimeout](#)
- [VL53L0X\\_GpioFunctionality Pin0GpioFunctionality](#)
- [uint32\\_t FinalRangeTimeoutMicroSecs](#)
- [uint8\\_t FinalRangeVcselPulsePeriod](#)
- [uint32\\_t PreRangeTimeoutMicroSecs](#)
- [uint8\\_t PreRangeVcselPulsePeriod](#)
- [uint16\\_t SigmaEstRefArray](#)
- [uint16\\_t SigmaEstEffPulseWidth](#)
- [uint16\\_t SigmaEstEffAmbWidth](#)
- [uint8\\_t ReadDataFromDeviceDone](#)
- [uint8\\_t ModuleId](#)
- [uint8\\_t Revision](#)
- char [ProductId](#) [\[VL53L0X\\_MAX\\_STRING\\_LENGTH\]](#)
- [uint8\\_t ReferenceSpadCount](#)
- [uint8\\_t ReferenceSpadType](#)
- [uint8\\_t RefSpadsInitialised](#)
- [uint32\\_t PartUIDUpper](#)
- [uint32\\_t PartUIDLower](#)
- [FixPoint1616\\_t SignalRateMeasFixed400mm](#)

---

### Detailed Description

Definition at line 381 of file vl53l0x\_def.h.

---

## Field Documentation

### [FixPoint1616 t VL53L0X\\_DeviceSpecificParameters\\_t::OscFrequencyMHz](#)

Definition at line 382 of file vl53l0x\_def.h.

### [uint16 t VL53L0X\\_DeviceSpecificParameters\\_t::LastEncodedTimeout](#)

Definition at line 384 of file vl53l0x\_def.h.

### [VL53L0X\\_GpioFunctionality VL53L0X\\_DeviceSpecificParameters\\_t::Pin0GpioFunctionality](#)

Definition at line 387 of file vl53l0x\_def.h.

### [uint32 t VL53L0X\\_DeviceSpecificParameters\\_t::FinalRangeTimeoutMicroSecs](#)

Execution time of the final range

Definition at line 390 of file vl53l0x\_def.h.

### [uint8 t VL53L0X\\_DeviceSpecificParameters\\_t::FinalRangeVcselPulsePeriod](#)

Vcsel pulse period (pll clocks) for the final range measurement

Definition at line 392 of file vl53l0x\_def.h.

### [uint32 t VL53L0X\\_DeviceSpecificParameters\\_t::PreRangeTimeoutMicroSecs](#)

Execution time of the final range

Definition at line 394 of file vl53l0x\_def.h.

### [uint8 t VL53L0X\\_DeviceSpecificParameters\\_t::PreRangeVcselPulsePeriod](#)

Vcsel pulse period (pll clocks) for the pre-range measurement

Definition at line 396 of file vl53l0x\_def.h.

### [uint16 t VL53L0X\\_DeviceSpecificParameters\\_t::SigmaEstRefArray](#)

Reference array sigma value in 1/100th of [mm] e.g. 100 = 1mm

Definition at line 399 of file vl53l0x\_def.h.

### [uint16 t VL53L0X\\_DeviceSpecificParameters\\_t::SigmaEstEffPulseWidth](#)

Effective Pulse width for sigma estimate in 1/100th of ns e.g. 900 = 9.0ns

Definition at line 401 of file vl53l0x\_def.h.

### [uint16 t VL53L0X\\_DeviceSpecificParameters\\_t::SigmaEstEffAmbWidth](#)

Effective Ambient width for sigma estimate in 1/100th of ns e.g. 500 = 5.0ns

Definition at line 404 of file vl53l0x\_def.h.

### [uint8 t VL53L0X\\_DeviceSpecificParameters\\_t::ReadDataFromDeviceDone](#)

Definition at line 409 of file vl53l0x\_def.h.



## **[uint8\\_t VL53L0X\\_DeviceSpecificParameters\\_t::ModuleId](#)**

Definition at line 411 of file vl53l0x\_def.h.

## **[uint8\\_t VL53L0X\\_DeviceSpecificParameters\\_t::Revision](#)**

Definition at line 412 of file vl53l0x\_def.h.

## **char VL53L0X\_DeviceSpecificParameters\_t::ProductId[[VL53L0X\\_MAX\\_STRING\\_LENGTH](#)]**

Definition at line 413 of file vl53l0x\_def.h.

## **[uint8\\_t VL53L0X\\_DeviceSpecificParameters\\_t::ReferenceSpadCount](#)**

Definition at line 415 of file vl53l0x\_def.h.

## **[uint8\\_t VL53L0X\\_DeviceSpecificParameters\\_t::ReferenceSpadType](#)**

Definition at line 416 of file vl53l0x\_def.h.

## **[uint8\\_t VL53L0X\\_DeviceSpecificParameters\\_t::RefSpadsInitialised](#)**

Definition at line 417 of file vl53l0x\_def.h.

## **[uint32\\_t VL53L0X\\_DeviceSpecificParameters\\_t::PartUIDUpper](#)**

Unique Part ID Upper

Definition at line 418 of file vl53l0x\_def.h.

## **[uint32\\_t VL53L0X\\_DeviceSpecificParameters\\_t::PartUIDLower](#)**

Unique Part ID Lower

Definition at line 419 of file vl53l0x\_def.h.

## **[FixPoint1616\\_t VL53L0X\\_DeviceSpecificParameters\\_t::SignalRateMeasFixed400mm](#)**

Peek Signal rate at 400 mm

Definition at line 420 of file vl53l0x\_def.h.

---

The documentation for this struct was generated from the following file:

- [vl53l0x\\_def.h](#)
- 

## **VL53L0X\_DMaxData\_t Struct Reference**

Structure containing the Dmax computation parameters and data.

```
#include <vl53l0x_def.h>
```

## Data Fields

- [int32\\_t AmbTuningWindowFactor\\_K](#)
  - [int32\\_t RetSignalAt0mm](#)
- 

## Detailed Description

Structure containing the Dmax computation parameters and data.

Definition at line 295 of file vl53l0x\_def.h.

---

## Field Documentation

### [int32\\_t VL53L0X\\_DMaxData\\_t::AmbTuningWindowFactor\\_K](#)

internal algo tuning (\*1000)

Definition at line 296 of file vl53l0x\_def.h.

### [int32\\_t VL53L0X\\_DMaxData\\_t::RetSignalAt0mm](#)

intermediate dmax computation value caching

Definition at line 298 of file vl53l0x\_def.h.

---

The documentation for this struct was generated from the following file:

- [vl53l0x\\_def.h](#)
- 

## VL53L0X\_HistogramData\_t Struct Reference

Histogram measurement data.

```
#include <vl53l0x_def.h>
```

---

## Detailed Description

Histogram measurement data.

---

The documentation for this struct was generated from the following file:

- [vl53l0x\\_def.h](#)
- 

## VL53L0X\_HistogramMeasurementData\_t Struct Reference

```
#include <vl53l0x_def.h>
```

## Data Fields

- [uint32\\_t HistogramData \[VL53L0X\\_HISTOGRAM\\_BUFFER\\_SIZE\]](#)
- [uint8\\_t HistogramType](#)

- [uint8\\_t FirstBin](#)
  - [uint8\\_t BufferSize](#)
  - [uint8\\_t NumberOfBins](#)
  - [VL53L0X\\_DeviceError\\_ErrorStatus](#)
- 

## Detailed Description

Definition at line 352 of file vl53l0x\_def.h.

---

## Field Documentation

### [uint32\\_t](#)

**VL53L0X\_HistogramMeasurementData\_t::HistogramData[[VL53L0X\\_HISTOGRAM\\_BUFFER\\_SIZE](#)]**

Histogram data

Definition at line 354 of file vl53l0x\_def.h.

### [uint8\\_t](#) **VL53L0X\_HistogramMeasurementData\_t::HistogramType**

Indicate the types of histogram data : Return only, Reference only, both Return and Reference

Definition at line 356 of file vl53l0x\_def.h.

### [uint8\\_t](#) **VL53L0X\_HistogramMeasurementData\_t::FirstBin**

First Bin value

Definition at line 358 of file vl53l0x\_def.h.

### [uint8\\_t](#) **VL53L0X\_HistogramMeasurementData\_t::BufferSize**

Buffer Size - Set by the user.

Definition at line 359 of file vl53l0x\_def.h.

### [uint8\\_t](#) **VL53L0X\_HistogramMeasurementData\_t::NumberOfBins**

Number of bins filled by the histogram measurement

Definition at line 360 of file vl53l0x\_def.h.

### [VL53L0X\\_DeviceError](#) **VL53L0X\_HistogramMeasurementData\_t::ErrorStatus**

Error status of the current measurement.

see [VL53L0X\\_DeviceError](#) *VL53L0X\_GetStatusErrorString()*

Definition at line 363 of file vl53l0x\_def.h.

---

The documentation for this struct was generated from the following file:

- [vl53l0x\\_def.h](#)
- 

## VL53L0X\_RangeData\_t Struct Reference

Range measurement data.

```
#include <vl53l0x_def.h>
```

---

## Detailed Description

Range measurement data.

---

The documentation for this struct was generated from the following file:

- [vl53l0x\\_def.h](#)
- 

## VL53L0X\_RangingMeasurementData\_t Struct Reference

```
#include <vl53l0x_def.h>
```

### Data Fields

- [uint32\\_t](#) [TimeStamp](#)
  - [uint32\\_t](#) [MeasurementTimeUsec](#)
  - [uint16\\_t](#) [RangeMilliMeter](#)
  - [uint16\\_t](#) [RangeDMaxMilliMeter](#)
  - [FixPoint1616\\_t](#) [SignalRateRtnMegaCps](#)
  - [FixPoint1616\\_t](#) [AmbientRateRtnMegaCps](#)
  - [uint16\\_t](#) [EffectiveSpadRtnCount](#)
  - [uint8\\_t](#) [ZoneId](#)
  - [uint8\\_t](#) [RangeFractionalPart](#)
  - [uint8\\_t](#) [RangeStatus](#)
- 

## Detailed Description

Definition at line 306 of file vl53l0x\_def.h.

---

## Field Documentation

### [uint32\\_t](#) VL53L0X\_RangingMeasurementData\_t::TimeStamp

32-bit time stamp.

Definition at line 307 of file vl53l0x\_def.h.

### [uint32\\_t](#) VL53L0X\_RangingMeasurementData\_t::MeasurementTimeUsec

Give the Measurement time needed by the device to do the measurement.

Definition at line 308 of file vl53l0x\_def.h.

### [uint16\\_t](#) VL53L0X\_RangingMeasurementData\_t::RangeMilliMeter

range distance in millimeter.

Definition at line 313 of file vl53l0x\_def.h.

#### **uint16\_t VL53L0X\_RangingMeasurementData\_t::RangeDMaxMilliMeter**

Tells what is the maximum detection distance of the device in current setup and environment conditions (Filled when applicable)

Definition at line 315 of file vl53l0x\_def.h.

#### **FixPoint1616\_t VL53L0X\_RangingMeasurementData\_t::SignalRateRtnMegaCps**

Return signal rate (MCPS)

these is a 16.16 fix point value, which is effectively a measure of target reflectance.

Definition at line 320 of file vl53l0x\_def.h.

#### **FixPoint1616\_t VL53L0X\_RangingMeasurementData\_t::AmbientRateRtnMegaCps**

Return ambient rate (MCPS)

these is a 16.16 fix point value, which is effectively a measure of the ambient light.

Definition at line 324 of file vl53l0x\_def.h.

#### **uint16\_t VL53L0X\_RangingMeasurementData\_t::EffectiveSpadRtnCount**

Return the effective SPAD count for the return signal. To obtain Real value it should be divided by 256

Definition at line 329 of file vl53l0x\_def.h.

#### **uint8\_t VL53L0X\_RangingMeasurementData\_t::Zoneld**

Denotes which zone and range scheduler stage the range data relates to.

Definition at line 333 of file vl53l0x\_def.h.

#### **uint8\_t VL53L0X\_RangingMeasurementData\_t::RangeFractionalPart**

Fractional part of range distance. Final value is a FixPoint168 value.

Definition at line 336 of file vl53l0x\_def.h.

#### **uint8\_t VL53L0X\_RangingMeasurementData\_t::RangeStatus**

Range Status for the current measurement. This is device dependent. Value = 0 means value is valid. See [RangeStatus](#)

Definition at line 339 of file vl53l0x\_def.h.

---

The documentation for this struct was generated from the following file:

- [vl53l0x\\_def.h](#)
- 

## **VL53L0X\_SchedulerSequenceSteps\_t Struct Reference**

```
#include <vl53l0x_def.h>
```

### **Data Fields**

- [uint8\\_t TccOn](#)
- [uint8\\_t MsrcOn](#)
- [uint8\\_t DssOn](#)

- [uint8\\_t PreRangeOn](#)
  - [uint8\\_t FinalRangeOn](#)
- 

## Detailed Description

Definition at line 525 of file vl53l0x\_def.h.

---

## Field Documentation

### [uint8\\_t VL53L0X\\_SchedulerSequenceSteps\\_t::TccOn](#)

Reports if Target Centre Check On

Definition at line 526 of file vl53l0x\_def.h.

### [uint8\\_t VL53L0X\\_SchedulerSequenceSteps\\_t::MsrcOn](#)

Reports if MSRC On

Definition at line 527 of file vl53l0x\_def.h.

### [uint8\\_t VL53L0X\\_SchedulerSequenceSteps\\_t::DssOn](#)

Reports if DSS On

Definition at line 528 of file vl53l0x\_def.h.

### [uint8\\_t VL53L0X\\_SchedulerSequenceSteps\\_t::PreRangeOn](#)

Reports if Pre-Range On

Definition at line 529 of file vl53l0x\_def.h.

### [uint8\\_t VL53L0X\\_SchedulerSequenceSteps\\_t::FinalRangeOn](#)

Reports if Final-Range On

Definition at line 530 of file vl53l0x\_def.h.

---

The documentation for this struct was generated from the following file:

- [vl53l0x\\_def.h](#)
- 

## VL53L0X\_SpadData\_t Struct Reference

Spad Configuration Data.

```
#include <vl53l0x_def.h>
```

### Data Fields

- [uint8\\_t RefSpadEnables](#) [[VL53L0X\\_REF\\_SPAD\\_BUFFER\\_SIZE](#)]
  - [uint8\\_t RefGoodSpadMap](#) [[VL53L0X\\_REF\\_SPAD\\_BUFFER\\_SIZE](#)]
-

## Detailed Description

Spad Configuration Data.

Definition at line 374 of file vl53l0x\_def.h.

---

## Field Documentation

[\*\*uint8\\_t VL53L0X\\_SpadData\\_t::RefSpadEnables\[VL53L0X\\_REF\\_SPAD\\_BUFFER\\_SIZE\]\*\*](#)

Reference Spad Enables

Definition at line 375 of file vl53l0x\_def.h.

[\*\*uint8\\_t VL53L0X\\_SpadData\\_t::RefGoodSpadMap\[VL53L0X\\_REF\\_SPAD\\_BUFFER\\_SIZE\]\*\*](#)

Reference Spad Good Spad Map

Definition at line 377 of file vl53l0x\_def.h.

---

The documentation for this struct was generated from the following file:

- [vl53l0x\\_def.h](#)
- 

## VL53L0X\_Version\_t Struct Reference

Defines the parameters of the Get Version Functions.

```
#include <vl53l0x_def.h>
```

### Data Fields

- [uint32\\_t revision](#)
  - [uint8\\_t major](#)
  - [uint8\\_t minor](#)
  - [uint8\\_t build](#)
- 

## Detailed Description

Defines the parameters of the Get Version Functions.

Definition at line 100 of file vl53l0x\_def.h.

---

## Field Documentation

[\*\*uint32\\_t VL53L0X\\_Version\\_t::revision\*\*](#)

revision number

Definition at line 101 of file vl53l0x\_def.h.

[\*\*uint8\\_t VL53L0X\\_Version\\_t::major\*\*](#)

major number

Definition at line 102 of file vl53l0x\_def.h.

#### [uint8\\_t VL53L0X\\_Version\\_t::minor](#)

minor number

Definition at line 103 of file vl53l0x\_def.h.

#### [uint8\\_t VL53L0X\\_Version\\_t::build](#)

build number

Definition at line 104 of file vl53l0x\_def.h.

---

The documentation for this struct was generated from the following file:

- [vl53l0x\\_def.h](#)

---

## File Documentation

### PAL\_disclaimer.c File Reference

no code doxygen doc only

---

### Detailed Description

no code doxygen doc only

---

### vl53l0x\_api.h File Reference

```
#include "vl53l0x_api_strings.h"
#include "vl53l0x_def.h"
#include "vl53l0x_platform.h"
```

### Macros

- #define [VL53L0X\\_API](#)

### Functions

- [VL53L0X\\_API VL53L0X\\_Error VL53L0X\\_GetVersion](#) ([VL53L0X\\_Version\\_t](#) \*pVersion)  
*Return the VL53L0X PAL Implementation Version.*
- [VL53L0X\\_API VL53L0X\\_Error VL53L0X\\_GetPalSpecVersion](#) ([VL53L0X\\_Version\\_t](#) \*pPalSpecVersion)  
*Return the PAL Specification Version used for the current implementation.*
- [VL53L0X\\_API VL53L0X\\_Error VL53L0X\\_GetProductRevision](#) ([VL53L0X\\_DEV](#) Dev, [uint8\\_t](#) \*pProductRevisionMajor, [uint8\\_t](#) \*pProductRevisionMinor)  
*Reads the Product Revision for a for given Device This function can be used to distinguish cut1.0 from cut1.1.*
- [VL53L0X\\_API VL53L0X\\_Error VL53L0X\\_GetDeviceInfo](#) ([VL53L0X\\_DEV](#) Dev, [VL53L0X\\_DeviceInfo\\_t](#) \*pVL53L0X\_DeviceInfo)  
*Reads the Device information for given Device.*
- [VL53L0X\\_API VL53L0X\\_Error VL53L0X\\_GetDeviceErrorStatus](#) ([VL53L0X\\_DEV](#) Dev, [VL53L0X\\_DeviceError](#) \*pDeviceErrorStatus)



*Read current status of the error register for the selected device.*

- [VL53L0X\\_API\\_VL53L0X\\_Error\\_VL53L0X\\_GetRangeStatusString](#) ([uint8\\_t](#) RangeStatus, char \*pRangeStatusString)  
*Human readable Range Status string for a given RangeStatus.*
- [VL53L0X\\_API\\_VL53L0X\\_Error\\_VL53L0X\\_GetDeviceErrorString](#) ([VL53L0X\\_DeviceError](#) ErrorCode, char \*pDeviceErrorString)  
*Human readable error string for a given Error Code.*
- [VL53L0X\\_API\\_VL53L0X\\_Error\\_VL53L0X\\_GetPalErrorString](#) ([VL53L0X\\_Error](#) PalErrorCode, char \*pPalErrorString)  
*Human readable error string for current PAL error status.*
- [VL53L0X\\_API\\_VL53L0X\\_Error\\_VL53L0X\\_GetPalStateString](#) ([VL53L0X\\_State](#) PalStateCode, char \*pPalStateString)  
*Human readable PAL State string.*
- [VL53L0X\\_API\\_VL53L0X\\_Error\\_VL53L0X\\_GetPalState](#) ([VL53L0X\\_DEV](#) Dev, [VL53L0X\\_State](#) \*pPalState)  
*Reads the internal state of the PAL for a given Device.*
- [VL53L0X\\_API\\_VL53L0X\\_Error\\_VL53L0X\\_SetPowerMode](#) ([VL53L0X\\_DEV](#) Dev, [VL53L0X\\_PowerModes](#) PowerMode)  
*Set the power mode for a given Device The power mode can be Standby or Idle.*
- [VL53L0X\\_API\\_VL53L0X\\_Error\\_VL53L0X\\_GetPowerMode](#) ([VL53L0X\\_DEV](#) Dev, [VL53L0X\\_PowerModes](#) \*pPowerMode)  
*Get the power mode for a given Device.*
- [VL53L0X\\_API\\_VL53L0X\\_Error\\_VL53L0X\\_SetOffsetCalibrationDataMicroMeter](#) ([VL53L0X\\_DEV](#) Dev, [int32\\_t](#) OffsetCalibrationDataMicroMeter)  
*Set or over-hide part to part calibration offset.*
- [VL53L0X\\_API\\_VL53L0X\\_Error\\_VL53L0X\\_GetOffsetCalibrationDataMicroMeter](#) ([VL53L0X\\_DEV](#) Dev, [int32\\_t](#) \*pOffsetCalibrationDataMicroMeter)  
*Get part to part calibration offset.*
- [VL53L0X\\_API\\_VL53L0X\\_Error\\_VL53L0X\\_SetLinearityCorrectiveGain](#) ([VL53L0X\\_DEV](#) Dev, [int16\\_t](#) LinearityCorrectiveGain)  
*Set the linearity corrective gain.*
- [VL53L0X\\_API\\_VL53L0X\\_Error\\_VL53L0X\\_GetLinearityCorrectiveGain](#) ([VL53L0X\\_DEV](#) Dev, [uint16\\_t](#) \*pLinearityCorrectiveGain)  
*Get the linearity corrective gain.*
- [VL53L0X\\_API\\_VL53L0X\\_Error\\_VL53L0X\\_SetGroupParamHold](#) ([VL53L0X\\_DEV](#) Dev, [uint8\\_t](#) GroupParamHold)  
*Set Group parameter Hold state.*
- [VL53L0X\\_API\\_VL53L0X\\_Error\\_VL53L0X\\_GetUpperLimitMilliMeter](#) ([VL53L0X\\_DEV](#) Dev, [uint16\\_t](#) \*pUpperLimitMilliMeter)  
*Get the maximal distance for actual setup.*
- [VL53L0X\\_Error\\_VL53L0X\\_GetTotalSignalRate](#) ([VL53L0X\\_DEV](#) Dev, [FixPoint1616\\_t](#) \*pTotalSignalRate)  
*Get the Total Signal Rate.*
- [VL53L0X\\_API\\_VL53L0X\\_Error\\_VL53L0X\\_SetDeviceAddress](#) ([VL53L0X\\_DEV](#) Dev, [uint8\\_t](#) DeviceAddress)  
*Set new device address.*
- [VL53L0X\\_API\\_VL53L0X\\_Error\\_VL53L0X\\_DataInit](#) ([VL53L0X\\_DEV](#) Dev)  
*One time device initialization.*
- [VL53L0X\\_API\\_VL53L0X\\_Error\\_VL53L0X\\_SetTuningSettingBuffer](#) ([VL53L0X\\_DEV](#) Dev, [uint8\\_t](#) \*pTuningSettingBuffer, [uint8\\_t](#) UseInternalTuningSettings)  
*Set the tuning settings pointer.*

- [VL53L0X\\_API\\_VL53L0X\\_Error\\_VL53L0X\\_GetTuningSettingBuffer](#) ([VL53L0X\\_DEV](#) Dev, [uint8\\_t](#) \*ppTuningSettingBuffer, [uint8\\_t](#) \*pUseInternalTuningSettings)  
*Get the tuning settings pointer and the internal external switch value.*
- [VL53L0X\\_API\\_VL53L0X\\_Error\\_VL53L0X\\_StaticInit](#) ([VL53L0X\\_DEV](#) Dev)  
*Do basic device init (and eventually patch loading) This function will change the VL53L0X\_State from VL53L0X\_STATE\_WAIT\_STATICINIT to VL53L0X\_STATE\_IDLE.*
- [VL53L0X\\_API\\_VL53L0X\\_Error\\_VL53L0X\\_WaitDeviceBooted](#) ([VL53L0X\\_DEV](#) Dev)  
*Wait for device booted after chip enable (hardware standby) This function can be run only when VL53L0X\_State is VL53L0X\_STATE\_POWERDOWN.*
- [VL53L0X\\_API\\_VL53L0X\\_Error\\_VL53L0X\\_ResetDevice](#) ([VL53L0X\\_DEV](#) Dev)  
*Do an hard reset or soft reset (depending on implementation) of the device call of this function, device must be in same state as right after a power-up sequence. This function will change the VL53L0X\_State to VL53L0X\_STATE\_POWERDOWN.*
- [VL53L0X\\_API\\_VL53L0X\\_Error\\_VL53L0X\\_SetDeviceParameters](#) ([VL53L0X\\_DEV](#) Dev, const [VL53L0X\\_DeviceParameters\\_t](#) \*pDeviceParameters)  
*Prepare device for operation.*
- [VL53L0X\\_API\\_VL53L0X\\_Error\\_VL53L0X\\_GetDeviceParameters](#) ([VL53L0X\\_DEV](#) Dev, [VL53L0X\\_DeviceParameters\\_t](#) \*pDeviceParameters)  
*Retrieve current device parameters.*
- [VL53L0X\\_API\\_VL53L0X\\_Error\\_VL53L0X\\_SetDeviceMode](#) ([VL53L0X\\_DEV](#) Dev, [VL53L0X\\_DeviceModes](#) DeviceMode)  
*Set a new device mode.*
- [VL53L0X\\_API\\_VL53L0X\\_Error\\_VL53L0X\\_GetDeviceMode](#) ([VL53L0X\\_DEV](#) Dev, [VL53L0X\\_DeviceModes](#) \*pDeviceMode)  
*Get current new device mode.*
- [VL53L0X\\_API\\_VL53L0X\\_Error\\_VL53L0X\\_SetRangeFractionEnable](#) ([VL53L0X\\_DEV](#) Dev, [uint8\\_t](#) Enable)  
*Sets the resolution of range measurements.*
- [VL53L0X\\_API\\_VL53L0X\\_Error\\_VL53L0X\\_GetFractionEnable](#) ([VL53L0X\\_DEV](#) Dev, [uint8\\_t](#) \*pEnable)  
*Gets the fraction enable parameter indicating the resolution of range measurements.*
- [VL53L0X\\_API\\_VL53L0X\\_Error\\_VL53L0X\\_SetHistogramMode](#) ([VL53L0X\\_DEV](#) Dev, [VL53L0X\\_HistogramModes](#) HistogramMode)  
*Set a new Histogram mode.*
- [VL53L0X\\_API\\_VL53L0X\\_Error\\_VL53L0X\\_GetHistogramMode](#) ([VL53L0X\\_DEV](#) Dev, [VL53L0X\\_HistogramModes](#) \*pHistogramMode)  
*Get current new device mode.*
- [VL53L0X\\_API\\_VL53L0X\\_Error\\_VL53L0X\\_SetMeasurementTimingBudgetMicroSeconds](#) ([VL53L0X\\_DEV](#) Dev, [uint32\\_t](#) MeasurementTimingBudgetMicroSeconds)  
*Set Ranging Timing Budget in microseconds.*
- [VL53L0X\\_API\\_VL53L0X\\_Error\\_VL53L0X\\_GetMeasurementTimingBudgetMicroSeconds](#) ([VL53L0X\\_DEV](#) Dev, [uint32\\_t](#) \*pMeasurementTimingBudgetMicroSeconds)  
*Get Ranging Timing Budget in microseconds.*
- [VL53L0X\\_API\\_VL53L0X\\_Error\\_VL53L0X\\_GetVcselPulsePeriod](#) ([VL53L0X\\_DEV](#) Dev, [VL53L0X\\_VcselPeriod](#) VcselPeriodType, [uint8\\_t](#) \*pVCSELPulsePeriod)  
*Gets the VCSEL pulse period.*
- [VL53L0X\\_API\\_VL53L0X\\_Error\\_VL53L0X\\_SetVcselPulsePeriod](#) ([VL53L0X\\_DEV](#) Dev, [VL53L0X\\_VcselPeriod](#) VcselPeriodType, [uint8\\_t](#) VCSELPulsePeriod)  
*Sets the VCSEL pulse period.*
- [VL53L0X\\_API\\_VL53L0X\\_Error\\_VL53L0X\\_SetSequenceStepEnable](#) ([VL53L0X\\_DEV](#) Dev, [VL53L0X\\_SequenceStepId](#) SequenceStepId, [uint8\\_t](#) SequenceStepEnabled)  
*Sets the (on/off) state of a requested sequence step.*

- [VL53L0X\\_API\\_VL53L0X\\_Error\\_VL53L0X\\_GetSequenceStepEnable](#) ([VL53L0X\\_DEV](#) Dev, [VL53L0X\\_SequenceStepId](#) SequenceStepId, [uint8\\_t](#) \*pSequenceStepEnabled)  
*Gets the (on/off) state of a requested sequence step.*
- [VL53L0X\\_API\\_VL53L0X\\_Error\\_VL53L0X\\_GetSequenceStepEnables](#) ([VL53L0X\\_DEV](#) Dev, [VL53L0X\\_SchedulerSequenceSteps\\_t](#) \*pSchedulerSequenceSteps)  
*Gets the (on/off) state of all sequence steps.*
- [VL53L0X\\_API\\_VL53L0X\\_Error\\_VL53L0X\\_SetSequenceStepTimeout](#) ([VL53L0X\\_DEV](#) Dev, [VL53L0X\\_SequenceStepId](#) SequenceStepId, [FixPoint1616\\_t](#) TimeOutMilliSecs)  
*Sets the timeout of a requested sequence step.*
- [VL53L0X\\_API\\_VL53L0X\\_Error\\_VL53L0X\\_GetSequenceStepTimeout](#) ([VL53L0X\\_DEV](#) Dev, [VL53L0X\\_SequenceStepId](#) SequenceStepId, [FixPoint1616\\_t](#) \*pTimeOutMilliSecs)  
*Gets the timeout of a requested sequence step.*
- [VL53L0X\\_API\\_VL53L0X\\_Error\\_VL53L0X\\_GetNumberOfSequenceSteps](#) ([VL53L0X\\_DEV](#) Dev, [uint8\\_t](#) \*pNumberOfSequenceSteps)  
*Gets number of sequence steps managed by the API.*
- [VL53L0X\\_API\\_VL53L0X\\_Error\\_VL53L0X\\_GetSequenceStepsInfo](#) ([VL53L0X\\_SequenceStepId](#) SequenceStepId, char \*pSequenceStepsString)  
*Gets the name of a given sequence step.*
- [VL53L0X\\_API\\_VL53L0X\\_Error\\_VL53L0X\\_SetInterMeasurementPeriodMilliSeconds](#) ([VL53L0X\\_DEV](#) Dev, [uint32\\_t](#) InterMeasurementPeriodMilliSeconds)  
*Program continuous mode Inter-Measurement period in milliseconds.*
- [VL53L0X\\_API\\_VL53L0X\\_Error\\_VL53L0X\\_GetInterMeasurementPeriodMilliSeconds](#) ([VL53L0X\\_DEV](#) Dev, [uint32\\_t](#) \*pInterMeasurementPeriodMilliSeconds)  
*Get continuous mode Inter-Measurement period in milliseconds.*
- [VL53L0X\\_API\\_VL53L0X\\_Error\\_VL53L0X\\_SetXTalkCompensationEnable](#) ([VL53L0X\\_DEV](#) Dev, [uint8\\_t](#) XTalkCompensationEnable)  
*Enable/Disable Cross talk compensation feature.*
- [VL53L0X\\_API\\_VL53L0X\\_Error\\_VL53L0X\\_GetXTalkCompensationEnable](#) ([VL53L0X\\_DEV](#) Dev, [uint8\\_t](#) \*pXTalkCompensationEnable)  
*Get Cross talk compensation rate.*
- [VL53L0X\\_API\\_VL53L0X\\_Error\\_VL53L0X\\_SetXTalkCompensationRateMegaCps](#) ([VL53L0X\\_DEV](#) Dev, [FixPoint1616\\_t](#) XTalkCompensationRateMegaCps)  
*Set Cross talk compensation rate.*
- [VL53L0X\\_API\\_VL53L0X\\_Error\\_VL53L0X\\_GetXTalkCompensationRateMegaCps](#) ([VL53L0X\\_DEV](#) Dev, [FixPoint1616\\_t](#) \*pXTalkCompensationRateMegaCps)  
*Get Cross talk compensation rate.*
- [VL53L0X\\_API\\_VL53L0X\\_Error\\_VL53L0X\\_SetRefCalibration](#) ([VL53L0X\\_DEV](#) Dev, [uint8\\_t](#) VhvSettings, [uint8\\_t](#) PhaseCal)  
*Set Reference Calibration Parameters.*
- [VL53L0X\\_API\\_VL53L0X\\_Error\\_VL53L0X\\_GetRefCalibration](#) ([VL53L0X\\_DEV](#) Dev, [uint8\\_t](#) \*pVhvSettings, [uint8\\_t](#) \*pPhaseCal)  
*Get Reference Calibration Parameters.*
- [VL53L0X\\_API\\_VL53L0X\\_Error\\_VL53L0X\\_GetNumberOfLimitCheck](#) ([uint16\\_t](#) \*pNumberOfLimitCheck)  
*Get the number of the check limit managed by a given Device.*
- [VL53L0X\\_API\\_VL53L0X\\_Error\\_VL53L0X\\_GetLimitCheckInfo](#) ([VL53L0X\\_DEV](#) Dev, [uint16\\_t](#) LimitCheckId, char \*pLimitCheckString)  
*Return a description string for a given limit check number.*
- [VL53L0X\\_API\\_VL53L0X\\_Error\\_VL53L0X\\_GetLimitCheckStatus](#) ([VL53L0X\\_DEV](#) Dev, [uint16\\_t](#) LimitCheckId, [uint8\\_t](#) \*pLimitCheckStatus)  
*Return a the Status of the specified check limit.*

- [VL53L0X\\_API\\_VL53L0X\\_Error\\_VL53L0X\\_SetLimitCheckEnable](#) (VL53L0X\_DEV Dev, [uint16\\_t](#) LimitCheckId, [uint8\\_t](#) LimitCheckEnable)  
*Enable/Disable a specific limit check.*
- [VL53L0X\\_API\\_VL53L0X\\_Error\\_VL53L0X\\_GetLimitCheckEnable](#) (VL53L0X\_DEV Dev, [uint16\\_t](#) LimitCheckId, [uint8\\_t](#) \*pLimitCheckEnable)  
*Get specific limit check enable state.*
- [VL53L0X\\_API\\_VL53L0X\\_Error\\_VL53L0X\\_SetLimitCheckValue](#) (VL53L0X\_DEV Dev, [uint16\\_t](#) LimitCheckId, [FixPoint1616\\_t](#) LimitCheckValue)  
*Set a specific limit check value.*
- [VL53L0X\\_API\\_VL53L0X\\_Error\\_VL53L0X\\_GetLimitCheckValue](#) (VL53L0X\_DEV Dev, [uint16\\_t](#) LimitCheckId, [FixPoint1616\\_t](#) \*pLimitCheckValue)  
*Get a specific limit check value.*
- [VL53L0X\\_API\\_VL53L0X\\_Error\\_VL53L0X\\_GetLimitCheckCurrent](#) (VL53L0X\_DEV Dev, [uint16\\_t](#) LimitCheckId, [FixPoint1616\\_t](#) \*pLimitCheckCurrent)  
*Get the current value of the signal used for the limit check.*
- [VL53L0X\\_API\\_VL53L0X\\_Error\\_VL53L0X\\_SetWrapAroundCheckEnable](#) (VL53L0X\_DEV Dev, [uint8\\_t](#) WrapAroundCheckEnable)  
*Enable (or disable) Wrap around Check.*
- [VL53L0X\\_API\\_VL53L0X\\_Error\\_VL53L0X\\_GetWrapAroundCheckEnable](#) (VL53L0X\_DEV Dev, [uint8\\_t](#) \*pWrapAroundCheckEnable)  
*Get setup of Wrap around Check.*
- [VL53L0X\\_API\\_VL53L0X\\_Error\\_VL53L0X\\_SetDmaxCalParameters](#) (VL53L0X\_DEV Dev, [uint16\\_t](#) RangeMilliMeter, [FixPoint1616\\_t](#) SignalRateRtnMegaCps)  
*Set Dmax Calibration Parameters for a given device When one of the parameter is zero, this function will get parameter from NVM.*
- [VL53L0X\\_API\\_VL53L0X\\_Error\\_VL53L0X\\_GetDmaxCalParameters](#) (VL53L0X\_DEV Dev, [uint16\\_t](#) \*pRangeMilliMeter, [FixPoint1616\\_t](#) \*pSignalRateRtnMegaCps)  
*Get Dmax Calibration Parameters for a given device.*
- [VL53L0X\\_API\\_VL53L0X\\_Error\\_VL53L0X\\_PerformSingleMeasurement](#) (VL53L0X\_DEV Dev)  
*Single shot measurement.*
- [VL53L0X\\_API\\_VL53L0X\\_Error\\_VL53L0X\\_PerformRefCalibration](#) (VL53L0X\_DEV Dev, [uint8\\_t](#) \*pVhvSettings, [uint8\\_t](#) \*pPhaseCal)  
*Perform Reference Calibration.*
- [VL53L0X\\_API\\_VL53L0X\\_Error\\_VL53L0X\\_PerformXTalkMeasurement](#) (VL53L0X\_DEV Dev, [uint32\\_t](#) TimeoutMs, [FixPoint1616\\_t](#) \*pXTalkPerSpad, [uint8\\_t](#) \*pAmbientTooHigh)  
*Perform XTalk Measurement.*
- [VL53L0X\\_API\\_VL53L0X\\_Error\\_VL53L0X\\_PerformXTalkCalibration](#) (VL53L0X\_DEV Dev, [FixPoint1616\\_t](#) XTalkCalDistance, [FixPoint1616\\_t](#) \*pXTalkCompensationRateMegaCps)  
*Perform XTalk Calibration.*
- [VL53L0X\\_API\\_VL53L0X\\_Error\\_VL53L0X\\_PerformOffsetCalibration](#) (VL53L0X\_DEV Dev, [FixPoint1616\\_t](#) CalDistanceMilliMeter, [int32\\_t](#) \*pOffsetMicroMeter)  
*Perform Offset Calibration.*
- [VL53L0X\\_API\\_VL53L0X\\_Error\\_VL53L0X\\_StartMeasurement](#) (VL53L0X\_DEV Dev)  
*Start device measurement.*
- [VL53L0X\\_API\\_VL53L0X\\_Error\\_VL53L0X\\_StopMeasurement](#) (VL53L0X\_DEV Dev)  
*Stop device measurement.*
- [VL53L0X\\_API\\_VL53L0X\\_Error\\_VL53L0X\\_GetMeasurementDataReady](#) (VL53L0X\_DEV Dev, [uint8\\_t](#) \*pMeasurementDataReady)  
*Return Measurement Data Ready.*
- [VL53L0X\\_API\\_VL53L0X\\_Error\\_VL53L0X\\_WaitDeviceReadyForNewMeasurement](#) (VL53L0X\_DEV Dev, [uint32\\_t](#) MaxLoop)  
*Wait for device ready for a new measurement command.*

- [VL53L0X\\_API\\_VL53L0X\\_Error\\_VL53L0X\\_GetMeasurementRefSignal](#) ([VL53L0X\\_DEV](#) Dev, [FixPoint1616\\_t](#) \*pMeasurementRefSignal)  
*Retrieve the Reference Signal after a measurements.*
- [VL53L0X\\_API\\_VL53L0X\\_Error\\_VL53L0X\\_GetRangingMeasurementData](#) ([VL53L0X\\_DEV](#) Dev, [VL53L0X\\_RangingMeasurementData\\_t](#) \*pRangingMeasurementData)  
*Retrieve the measurements from device for a given setup.*
- [VL53L0X\\_API\\_VL53L0X\\_Error\\_VL53L0X\\_GetHistogramMeasurementData](#) ([VL53L0X\\_DEV](#) Dev, [VL53L0X\\_HistogramMeasurementData\\_t](#) \*pHistogramMeasurementData)  
*Retrieve the measurements from device for a given setup.*
- [VL53L0X\\_API\\_VL53L0X\\_Error\\_VL53L0X\\_PerformSingleRangingMeasurement](#) ([VL53L0X\\_DEV](#) Dev, [VL53L0X\\_RangingMeasurementData\\_t](#) \*pRangingMeasurementData)  
*Performs a single ranging measurement and retrieve the ranging measurement data.*
- [VL53L0X\\_API\\_VL53L0X\\_Error\\_VL53L0X\\_PerformSingleHistogramMeasurement](#) ([VL53L0X\\_DEV](#) Dev, [VL53L0X\\_HistogramMeasurementData\\_t](#) \*pHistogramMeasurementData)  
*Performs a single histogram measurement and retrieve the histogram measurement data Is equivalent to VL53L0X\_PerformSingleMeasurement + VL53L0X\_GetHistogramMeasurementData.*
- [VL53L0X\\_API\\_VL53L0X\\_Error\\_VL53L0X\\_SetNumberOfROIzones](#) ([VL53L0X\\_DEV](#) Dev, [uint8\\_t](#) NumberOfROIzones)  
*Set the number of ROI Zones to be used for a specific Device.*
- [VL53L0X\\_API\\_VL53L0X\\_Error\\_VL53L0X\\_GetNumberOfROIzones](#) ([VL53L0X\\_DEV](#) Dev, [uint8\\_t](#) \*pNumberOfROIzones)  
*Get the number of ROI Zones managed by the Device.*
- [VL53L0X\\_API\\_VL53L0X\\_Error\\_VL53L0X\\_GetMaxNumberOfROIzones](#) ([VL53L0X\\_DEV](#) Dev, [uint8\\_t](#) \*pMaxNumberOfROIzones)  
*Get the Maximum number of ROI Zones managed by the Device.*
- [VL53L0X\\_API\\_VL53L0X\\_Error\\_VL53L0X\\_SetGpioConfig](#) ([VL53L0X\\_DEV](#) Dev, [uint8\\_t](#) Pin, [VL53L0X\\_DeviceModes](#) DeviceMode, [VL53L0X\\_GpioFunctionality](#) Functionality, [VL53L0X\\_InterruptPolarity](#) Polarity)  
*Set the configuration of GPIO pin for a given device.*
- [VL53L0X\\_API\\_VL53L0X\\_Error\\_VL53L0X\\_GetGpioConfig](#) ([VL53L0X\\_DEV](#) Dev, [uint8\\_t](#) Pin, [VL53L0X\\_DeviceModes](#) \*pDeviceMode, [VL53L0X\\_GpioFunctionality](#) \*pFunctionality, [VL53L0X\\_InterruptPolarity](#) \*pPolarity)  
*Get current configuration for GPIO pin for a given device.*
- [VL53L0X\\_API\\_VL53L0X\\_Error\\_VL53L0X\\_SetInterruptThresholds](#) ([VL53L0X\\_DEV](#) Dev, [VL53L0X\\_DeviceModes](#) DeviceMode, [FixPoint1616\\_t](#) ThresholdLow, [FixPoint1616\\_t](#) ThresholdHigh)  
*Set low and high Interrupt thresholds for a given mode (ranging, ALS, ...) for a given device.*
- [VL53L0X\\_API\\_VL53L0X\\_Error\\_VL53L0X\\_GetInterruptThresholds](#) ([VL53L0X\\_DEV](#) Dev, [VL53L0X\\_DeviceModes](#) DeviceMode, [FixPoint1616\\_t](#) \*pThresholdLow, [FixPoint1616\\_t](#) \*pThresholdHigh)  
*Get high and low Interrupt thresholds for a given mode (ranging, ALS, ...) for a given device.*
- [VL53L0X\\_API\\_VL53L0X\\_Error\\_VL53L0X\\_GetStopCompletedStatus](#) ([VL53L0X\\_DEV](#) Dev, [uint32\\_t](#) \*pStopStatus)  
*Return device stop completion status.*
- [VL53L0X\\_API\\_VL53L0X\\_Error\\_VL53L0X\\_ClearInterruptMask](#) ([VL53L0X\\_DEV](#) Dev, [uint32\\_t](#) InterruptMask)  
*Clear given system interrupt condition.*
- [VL53L0X\\_API\\_VL53L0X\\_Error\\_VL53L0X\\_GetInterruptMaskStatus](#) ([VL53L0X\\_DEV](#) Dev, [uint32\\_t](#) \*pInterruptMaskStatus)  
*Return device interrupt status.*
- [VL53L0X\\_API\\_VL53L0X\\_Error\\_VL53L0X\\_EnableInterruptMask](#) ([VL53L0X\\_DEV](#) Dev, [uint32\\_t](#) InterruptMask)  
*Configure ranging interrupt reported to system.*

- [VL53L0X\\_API\\_VL53L0X\\_Error\\_VL53L0X\\_SetSpadAmbientDamperThreshold](#) (VL53L0X\_DEV Dev, [uint16\\_t](#) SpadAmbientDamperThreshold)  
*Set the SPAD Ambient Damper Threshold value.*
- [VL53L0X\\_API\\_VL53L0X\\_Error\\_VL53L0X\\_GetSpadAmbientDamperThreshold](#) (VL53L0X\_DEV Dev, [uint16\\_t](#) \*pSpadAmbientDamperThreshold)  
*Get the current SPAD Ambient Damper Threshold value.*
- [VL53L0X\\_API\\_VL53L0X\\_Error\\_VL53L0X\\_SetSpadAmbientDamperFactor](#) (VL53L0X\_DEV Dev, [uint16\\_t](#) SpadAmbientDamperFactor)  
*Set the SPAD Ambient Damper Factor value.*
- [VL53L0X\\_API\\_VL53L0X\\_Error\\_VL53L0X\\_GetSpadAmbientDamperFactor](#) (VL53L0X\_DEV Dev, [uint16\\_t](#) \*pSpadAmbientDamperFactor)  
*Get the current SPAD Ambient Damper Factor value.*
- [VL53L0X\\_API\\_VL53L0X\\_Error\\_VL53L0X\\_PerformRefSpadManagement](#) (VL53L0X\_DEV Dev, [uint32\\_t](#) \*refSpadCount, [uint8\\_t](#) \*isApertureSpads)  
*Performs Reference Spad Management.*
- [VL53L0X\\_API\\_VL53L0X\\_Error\\_VL53L0X\\_SetReferenceSpads](#) (VL53L0X\_DEV Dev, [uint32\\_t](#) refSpadCount, [uint8\\_t](#) isApertureSpads)  
*Applies Reference SPAD configuration.*
- [VL53L0X\\_API\\_VL53L0X\\_Error\\_VL53L0X\\_GetReferenceSpads](#) (VL53L0X\_DEV Dev, [uint32\\_t](#) \*refSpadCount, [uint8\\_t](#) \*isApertureSpads)  
*Retrieves SPAD configuration.*

## Macro Definition Documentation

### #define VL53L0X\_API

Definition at line 48 of file vl53l0x\_api.h.

## vl53l0x\_api\_calibration.h File Reference

```
#include "vl53l0x_def.h"
#include "vl53l0x_platform.h"
```

### Functions

- [VL53L0X\\_Error\\_VL53L0X\\_perform\\_xtalk\\_calibration](#) (VL53L0X\_DEV Dev, [FixPoint1616\\_t](#) XTalkCalDistance, [FixPoint1616\\_t](#) \*pXTalkCompensationRateMegaCps)
- [VL53L0X\\_Error\\_VL53L0X\\_perform\\_offset\\_calibration](#) (VL53L0X\_DEV Dev, [FixPoint1616\\_t](#) CalDistanceMilliMeter, [int32\\_t](#) \*pOffsetMicroMeter)
- [VL53L0X\\_Error\\_VL53L0X\\_set\\_offset\\_calibration\\_data\\_micro\\_meter](#) (VL53L0X\_DEV Dev, [int32\\_t](#) OffsetCalibrationDataMicroMeter)
- [VL53L0X\\_Error\\_VL53L0X\\_get\\_offset\\_calibration\\_data\\_micro\\_meter](#) (VL53L0X\_DEV Dev, [int32\\_t](#) \*pOffsetCalibrationDataMicroMeter)
- [VL53L0X\\_Error\\_VL53L0X\\_apply\\_offset\\_adjustment](#) (VL53L0X\_DEV Dev)
- [VL53L0X\\_Error\\_VL53L0X\\_perform\\_ref\\_spad\\_management](#) (VL53L0X\_DEV Dev, [uint32\\_t](#) \*refSpadCount, [uint8\\_t](#) \*isApertureSpads)
- [VL53L0X\\_Error\\_VL53L0X\\_set\\_reference\\_spads](#) (VL53L0X\_DEV Dev, [uint32\\_t](#) count, [uint8\\_t](#) isApertureSpads)
- [VL53L0X\\_Error\\_VL53L0X\\_get\\_reference\\_spads](#) (VL53L0X\_DEV Dev, [uint32\\_t](#) \*pSpadCount, [uint8\\_t](#) \*pIsApertureSpads)



- [VL53L0X\\_Error VL53L0X\\_perform\\_phase\\_calibration](#) ([VL53L0X\\_DEV](#) Dev, [uint8\\_t](#) \*pPhaseCal, const [uint8\\_t](#) get\_data\_enable, const [uint8\\_t](#) restore\_config)
- [VL53L0X\\_Error VL53L0X\\_perform\\_ref\\_calibration](#) ([VL53L0X\\_DEV](#) Dev, [uint8\\_t](#) \*pVhvSettings, [uint8\\_t](#) \*pPhaseCal, [uint8\\_t](#) get\_data\_enable)
- [VL53L0X\\_Error VL53L0X\\_set\\_ref\\_calibration](#) ([VL53L0X\\_DEV](#) Dev, [uint8\\_t](#) VhvSettings, [uint8\\_t](#) PhaseCal)
- [VL53L0X\\_Error VL53L0X\\_get\\_ref\\_calibration](#) ([VL53L0X\\_DEV](#) Dev, [uint8\\_t](#) \*pVhvSettings, [uint8\\_t](#) \*pPhaseCal)

---

## Function Documentation

[VL53L0X\\_Error VL53L0X\\_perform\\_xtalk\\_calibration](#) ([VL53L0X\\_DEV](#) Dev, [FixPoint1616\\_t](#) XTalkCalDistance, [FixPoint1616\\_t](#) \* pXTalkCompensationRateMegaCps)

[VL53L0X\\_Error VL53L0X\\_perform\\_offset\\_calibration](#) ([VL53L0X\\_DEV](#) Dev, [FixPoint1616\\_t](#) CalDistanceMilliMeter, [int32\\_t](#) \* pOffsetMicroMeter)

[VL53L0X\\_Error VL53L0X\\_set\\_offset\\_calibration\\_data\\_micro\\_meter](#) ([VL53L0X\\_DEV](#) Dev, [int32\\_t](#) OffsetCalibrationDataMicroMeter)

[VL53L0X\\_Error VL53L0X\\_get\\_offset\\_calibration\\_data\\_micro\\_meter](#) ([VL53L0X\\_DEV](#) Dev, [int32\\_t](#) \* pOffsetCalibrationDataMicroMeter)

[VL53L0X\\_Error VL53L0X\\_apply\\_offset\\_adjustment](#) ([VL53L0X\\_DEV](#) Dev)

[VL53L0X\\_Error VL53L0X\\_perform\\_ref\\_spad\\_management](#) ([VL53L0X\\_DEV](#) Dev, [uint32\\_t](#) \* refSpadCount, [uint8\\_t](#) \* isApertureSpads)

[VL53L0X\\_Error VL53L0X\\_set\\_reference\\_spads](#) ([VL53L0X\\_DEV](#) Dev, [uint32\\_t](#) count, [uint8\\_t](#) isApertureSpads)

[VL53L0X\\_Error VL53L0X\\_get\\_reference\\_spads](#) ([VL53L0X\\_DEV](#) Dev, [uint32\\_t](#) \* pSpadCount, [uint8\\_t](#) \* pIsApertureSpads)

[VL53L0X\\_Error VL53L0X\\_perform\\_phase\\_calibration](#) ([VL53L0X\\_DEV](#) Dev, [uint8\\_t](#) \* pPhaseCal, const [uint8\\_t](#) get\_data\_enable, const [uint8\\_t](#) restore\_config)

[VL53L0X\\_Error VL53L0X\\_perform\\_ref\\_calibration](#) ([VL53L0X\\_DEV](#) Dev, [uint8\\_t](#) \* pVhvSettings, [uint8\\_t](#) \* pPhaseCal, [uint8\\_t](#) get\_data\_enable)

[VL53L0X\\_Error VL53L0X\\_set\\_ref\\_calibration](#) ([VL53L0X\\_DEV](#) Dev, [uint8\\_t](#) VhvSettings, [uint8\\_t](#) PhaseCal)

[VL53L0X\\_Error VL53L0X\\_get\\_ref\\_calibration](#) ([VL53L0X\\_DEV](#) Dev, [uint8\\_t](#) \* pVhvSettings, [uint8\\_t](#) \* pPhaseCal)

---

## vl53l0x\_api\_core.h File Reference

```
#include "vl53l0x_def.h"
#include "vl53l0x_platform.h"
```

## Functions

- [VL53L0X\\_Error VL53L0X\\_reverse\\_bytes](#) (uint8\_t \*data, uint32\_t size)
- [VL53L0X\\_Error VL53L0X\\_measurement\\_poll\\_for\\_completion](#) (VL53L0X\_DEV Dev)
- [uint8\\_t VL53L0X\\_encode\\_vcsel\\_period](#) (uint8\_t vcsel\_period\_pclks)
- [uint8\\_t VL53L0X\\_decode\\_vcsel\\_period](#) (uint8\_t vcsel\_period\_reg)
- [uint32\\_t VL53L0X\\_isqrt](#) (uint32\_t num)
- [uint32\\_t VL53L0X\\_quadrature\\_sum](#) (uint32\_t a, uint32\_t b)
- [VL53L0X\\_Error VL53L0X\\_get\\_info\\_from\\_device](#) (VL53L0X\_DEV Dev, uint8\_t option)
- [VL53L0X\\_Error VL53L0X\\_set\\_vcsel\\_pulse\\_period](#) (VL53L0X\_DEV Dev, VL53L0X\_VcselPeriod VcselPeriodType, uint8\_t VCSELPulsePeriodPCLK)
- [VL53L0X\\_Error VL53L0X\\_get\\_vcsel\\_pulse\\_period](#) (VL53L0X\_DEV Dev, VL53L0X\_VcselPeriod VcselPeriodType, uint8\_t \*pVCSELPulsePeriodPCLK)
- [uint32\\_t VL53L0X\\_decode\\_timeout](#) (uint16\_t encoded\_timeout)
- [VL53L0X\\_Error VL53L0X\\_get\\_sequence\\_step\\_timeout](#) (VL53L0X\_DEV Dev, VL53L0X\_SequenceStepId SequenceStepId, uint32\_t \*pTimeOutMicroSecs)
- [VL53L0X\\_Error VL53L0X\\_set\\_sequence\\_step\\_timeout](#) (VL53L0X\_DEV Dev, VL53L0X\_SequenceStepId SequenceStepId, uint32\_t TimeOutMicroSecs)
- [VL53L0X\\_Error VL53L0X\\_set\\_measurement\\_timing\\_budget\\_micro\\_seconds](#) (VL53L0X\_DEV Dev, uint32\_t MeasurementTimingBudgetMicroSeconds)
- [VL53L0X\\_Error VL53L0X\\_get\\_measurement\\_timing\\_budget\\_micro\\_seconds](#) (VL53L0X\_DEV Dev, uint32\_t \*pMeasurementTimingBudgetMicroSeconds)
- [VL53L0X\\_Error VL53L0X\\_load\\_tuning\\_settings](#) (VL53L0X\_DEV Dev, uint8\_t \*pTuningSettingBuffer)
- [VL53L0X\\_Error VL53L0X\\_calc\\_sigma\\_estimate](#) (VL53L0X\_DEV Dev, VL53L0X\_RangingMeasurementData\_t \*pRangingMeasurementData, FixPoint1616\_t \*pSigmaEstimate, uint32\_t \*pDmax\_mm)
- [VL53L0X\\_Error VL53L0X\\_get\\_total\\_xtalk\\_rate](#) (VL53L0X\_DEV Dev, VL53L0X\_RangingMeasurementData\_t \*pRangingMeasurementData, FixPoint1616\_t \*ptotal\_xtalk\_rate\_mcps)
- [VL53L0X\\_Error VL53L0X\\_get\\_total\\_signal\\_rate](#) (VL53L0X\_DEV Dev, VL53L0X\_RangingMeasurementData\_t \*pRangingMeasurementData, FixPoint1616\_t \*ptotal\_signal\_rate\_mcps)
- [VL53L0X\\_Error VL53L0X\\_get\\_pal\\_range\\_status](#) (VL53L0X\_DEV Dev, uint8\_t DeviceRangeStatus, FixPoint1616\_t SignalRate, uint16\_t EffectiveSpadRtnCount, VL53L0X\_RangingMeasurementData\_t \*pRangingMeasurementData, uint8\_t \*pPalRangeStatus)
- [uint32\\_t VL53L0X\\_calc\\_timeout\\_mclks](#) (VL53L0X\_DEV Dev, uint32\_t timeout\_period\_us, uint8\_t vcsel\_period\_pclks)
- [uint16\\_t VL53L0X\\_encode\\_timeout](#) (uint32\_t timeout\_macro\_clks)



## Function Documentation

[VL53L0X\\_Error](#) VL53L0X\_reverse\_bytes ([uint8\\_t](#) \* data, [uint32\\_t](#) size)

[VL53L0X\\_Error](#) VL53L0X\_measurement\_poll\_for\_completion ([VL53L0X\\_DEV](#) Dev)

[uint8\\_t](#) VL53L0X\_encode\_vcsel\_period ([uint8\\_t](#) vcsel\_period\_pclks)

[uint8\\_t](#) VL53L0X\_decode\_vcsel\_period ([uint8\\_t](#) vcsel\_period\_reg)

[uint32\\_t](#) VL53L0X\_isqrt ([uint32\\_t](#) num)

[uint32\\_t](#) VL53L0X\_quadrature\_sum ([uint32\\_t](#) a, [uint32\\_t](#) b)

[VL53L0X\\_Error](#) VL53L0X\_get\_info\_from\_device ([VL53L0X\\_DEV](#) Dev, [uint8\\_t](#) option)

[VL53L0X\\_Error](#) VL53L0X\_set\_vcsel\_pulse\_period ([VL53L0X\\_DEV](#) Dev, [VL53L0X\\_VcselPeriod](#) VcselPeriodType, [uint8\\_t](#) VCSELPulsePeriodPCLK)

[VL53L0X\\_Error](#) VL53L0X\_get\_vcsel\_pulse\_period ([VL53L0X\\_DEV](#) Dev, [VL53L0X\\_VcselPeriod](#) VcselPeriodType, [uint8\\_t](#) \* pVCSELPulsePeriodPCLK)

[uint32\\_t](#) VL53L0X\_decode\_timeout ([uint16\\_t](#) encoded\_timeout)

[VL53L0X\\_Error](#) get\_sequence\_step\_timeout ([VL53L0X\\_DEV](#) Dev, [VL53L0X\\_SequenceStepId](#) SequenceStepId, [uint32\\_t](#) \* pTimeOutMicroSecs)

[VL53L0X\\_Error](#) set\_sequence\_step\_timeout ([VL53L0X\\_DEV](#) Dev, [VL53L0X\\_SequenceStepId](#) SequenceStepId, [uint32\\_t](#) TimeOutMicroSecs)

[VL53L0X\\_Error](#) VL53L0X\_set\_measurement\_timing\_budget\_micro\_seconds ([VL53L0X\\_DEV](#) Dev, [uint32\\_t](#) MeasurementTimingBudgetMicroSeconds)

[VL53L0X\\_Error](#) VL53L0X\_get\_measurement\_timing\_budget\_micro\_seconds ([VL53L0X\\_DEV](#) Dev, [uint32\\_t](#) \* pMeasurementTimingBudgetMicroSeconds)

[VL53L0X\\_Error](#) VL53L0X\_load\_tuning\_settings ([VL53L0X\\_DEV](#) Dev, [uint8\\_t](#) \* pTuningSettingBuffer)

[VL53L0X\\_Error](#) VL53L0X\_calc\_sigma\_estimate ([VL53L0X\\_DEV](#) Dev, [VL53L0X\\_RangingMeasurementData\\_t](#) \* pRangingMeasurementData, [FixPoint1616\\_t](#) \* pSigmaEstimate, [uint32\\_t](#) \* pDmax\_mm)

[VL53L0X\\_Error](#) VL53L0X\_get\_total\_xtalk\_rate ([VL53L0X\\_DEV](#) Dev, [VL53L0X\\_RangingMeasurementData\\_t](#) \* pRangingMeasurementData, [FixPoint1616\\_t](#) \* ptotal\_xtalk\_rate\_mcps)

[VL53L0X\\_Error](#) VL53L0X\_get\_total\_signal\_rate ([VL53L0X\\_DEV](#) Dev, [VL53L0X\\_RangingMeasurementData\\_t](#) \* pRangingMeasurementData, [FixPoint1616\\_t](#) \* ptotal\_signal\_rate\_mcps)

[VL53L0X\\_Error](#) VL53L0X\_get\_pal\_range\_status ([VL53L0X\\_DEV](#) Dev, [uint8\\_t](#) DeviceRangeStatus, [FixPoint1616\\_t](#) SignalRate, [uint16\\_t](#) EffectiveSpadRtnCount,

[VL53L0X\\_RangingMeasurementData\\_t](#) \* *pRangingMeasurementData*, [uint8\\_t](#) \* *pPalRangeStatus*)

[uint32\\_t](#) VL53L0X\_calc\_timeout\_mclks ([VL53L0X\\_DEV](#) *Dev*, [uint32\\_t](#) *timeout\_period\_us*, [uint8\\_t](#) *vcSEL\_period\_pclks*)

[uint16\\_t](#) VL53L0X\_encode\_timeout ([uint32\\_t](#) *timeout\_macro\_clks*)

## vl53l0x\_api\_ranging.h File Reference

```
#include "vl53l0x_def.h"
#include "vl53l0x_platform.h"
```

## vl53l0x\_api\_strings.h File Reference

```
#include "vl53l0x_def.h"
#include "vl53l0x_platform.h"
```

### Macros

- #define [VL53L0X\\_STRING\\_DEVICE\\_INFO\\_NAME](#) "VL53L0X cut1.0"
- #define [VL53L0X\\_STRING\\_DEVICE\\_INFO\\_NAME\\_TS0](#) "VL53L0X TS0"
- #define [VL53L0X\\_STRING\\_DEVICE\\_INFO\\_NAME\\_TS1](#) "VL53L0X TS1"
- #define [VL53L0X\\_STRING\\_DEVICE\\_INFO\\_NAME\\_TS2](#) "VL53L0X TS2"
- #define [VL53L0X\\_STRING\\_DEVICE\\_INFO\\_NAME\\_ES1](#) "VL53L0X ES1 or later"
- #define [VL53L0X\\_STRING\\_DEVICE\\_INFO\\_TYPE](#) "VL53L0X"
- #define [VL53L0X\\_STRING\\_ERROR\\_NONE](#) "No Error"
- #define [VL53L0X\\_STRING\\_ERROR\\_CALIBRATION\\_WARNING](#) "Calibration Warning Error"
- #define [VL53L0X\\_STRING\\_ERROR\\_MIN\\_CLIPPED](#) "Min clipped error"
- #define [VL53L0X\\_STRING\\_ERROR\\_UNDEFINED](#) "Undefined error"
- #define [VL53L0X\\_STRING\\_ERROR\\_INVALID\\_PARAMS](#) "Invalid parameters error"
- #define [VL53L0X\\_STRING\\_ERROR\\_NOT\\_SUPPORTED](#) "Not supported error"
- #define [VL53L0X\\_STRING\\_ERROR\\_RANGE\\_ERROR](#) "Range error"
- #define [VL53L0X\\_STRING\\_ERROR\\_TIME\\_OUT](#) "Time out error"
- #define [VL53L0X\\_STRING\\_ERROR\\_MODE\\_NOT\\_SUPPORTED](#) "Mode not supported error"
- #define [VL53L0X\\_STRING\\_ERROR\\_BUFFER\\_TOO\\_SMALL](#) "Buffer too small"
- #define [VL53L0X\\_STRING\\_ERROR\\_GPIO\\_NOT\\_EXISTING](#) "GPIO not existing"
- #define [VL53L0X\\_STRING\\_ERROR\\_GPIO\\_FUNCTIONALITY\\_NOT\\_SUPPORTED](#) "GPIO funct not supported"
- #define [VL53L0X\\_STRING\\_ERROR\\_INTERRUPT\\_NOT\\_CLEARED](#) "Interrupt not Cleared"
- #define [VL53L0X\\_STRING\\_ERROR\\_CONTROL\\_INTERFACE](#) "Control Interface Error"
- #define [VL53L0X\\_STRING\\_ERROR\\_INVALID\\_COMMAND](#) "Invalid Command Error"
- #define [VL53L0X\\_STRING\\_ERROR\\_DIVISION\\_BY\\_ZERO](#) "Division by zero Error"
- #define [VL53L0X\\_STRING\\_ERROR\\_REF\\_SPAD\\_INIT](#) "Reference Spad Init Error"
- #define [VL53L0X\\_STRING\\_ERROR\\_NOT\\_IMPLEMENTED](#) "Not implemented error"
- #define [VL53L0X\\_STRING\\_UNKNOWN\\_ERROR\\_CODE](#) "Unknown Error Code"
- #define [VL53L0X\\_STRING\\_RANGESTATUS\\_NONE](#) "No Update"
- #define [VL53L0X\\_STRING\\_RANGESTATUS\\_RANGEVALID](#) "Range Valid"
- #define [VL53L0X\\_STRING\\_RANGESTATUS\\_SIGMA](#) "Sigma Fail"
- #define [VL53L0X\\_STRING\\_RANGESTATUS\\_SIGNAL](#) "Signal Fail"
- #define [VL53L0X\\_STRING\\_RANGESTATUS\\_MINRANGE](#) "Min Range Fail"
- #define [VL53L0X\\_STRING\\_RANGESTATUS\\_PHASE](#) "Phase Fail"

- `#define VL53L0X_STRING_RANGESTATUS_HW "Hardware Fail"`
- `#define VL53L0X_STRING_STATE_POWERDOWN "POWERDOWN State"`
- `#define VL53L0X_STRING_STATE_WAIT_STATICINIT "Wait for staticinit State"`
- `#define VL53L0X_STRING_STATE_STANDBY "STANDBY State"`
- `#define VL53L0X_STRING_STATE_IDLE "IDLE State"`
- `#define VL53L0X_STRING_STATE_RUNNING "RUNNING State"`
- `#define VL53L0X_STRING_STATE_UNKNOWN "UNKNOWN State"`
- `#define VL53L0X_STRING_STATE_ERROR "ERROR State"`
- `#define VL53L0X_STRING_DEVICEERROR_NONE "No Update"`
- `#define VL53L0X_STRING_DEVICEERROR_VCSELCONTINUITYTESTFAILURE "VCSEL Continuity Test Failure"`
- `#define VL53L0X_STRING_DEVICEERROR_VCSELWATCHDOGTESTFAILURE "VCSEL Watchdog Test Failure"`
- `#define VL53L0X_STRING_DEVICEERROR_NOVHVVALUEFOUND "No VHV Value found"`
- `#define VL53L0X_STRING_DEVICEERROR_MSRCNOTARGET "MSRC No Target Error"`
- `#define VL53L0X_STRING_DEVICEERROR_SNRCHECK "SNR Check Exit"`
- `#define VL53L0X_STRING_DEVICEERROR_RANGEPHASECHECK "Range Phase Check Error"`
- `#define VL53L0X_STRING_DEVICEERROR_SIGMATHRESHOLDCHECK "Sigma Threshold Check Error"`
- `#define VL53L0X_STRING_DEVICEERROR_TCC "TCC Error"`
- `#define VL53L0X_STRING_DEVICEERROR_PHASECONSISTENCY "Phase Consistency Error"`
- `#define VL53L0X_STRING_DEVICEERROR_MINCLIP "Min Clip Error"`
- `#define VL53L0X_STRING_DEVICEERROR_RANGECOMPLETE "Range Complete"`
- `#define VL53L0X_STRING_DEVICEERROR_ALGOUNDERFLOW "Range Algo Underflow Error"`
- `#define VL53L0X_STRING_DEVICEERROR_ALGOOVERFLOW "Range Algo Overflow Error"`
- `#define VL53L0X_STRING_DEVICEERROR_RANGEIGNORETHRESHOLD "Range Ignore Threshold Error"`
- `#define VL53L0X_STRING_DEVICEERROR_UNKNOWN "Unknown error code"`
- `#define VL53L0X_STRING_CHECKENABLE_SIGMA_FINAL_RANGE "SIGMA FINAL RANGE"`
- `#define VL53L0X_STRING_CHECKENABLE_SIGNAL_RATE_FINAL_RANGE "SIGNAL RATE FINAL RANGE"`
- `#define VL53L0X_STRING_CHECKENABLE_SIGNAL_REF_CLIP "SIGNAL REF CLIP"`
- `#define VL53L0X_STRING_CHECKENABLE_RANGE_IGNORE_THRESHOLD "RANGE IGNORE THRESHOLD"`
- `#define VL53L0X_STRING_CHECKENABLE_SIGNAL_RATE_MSRC "SIGNAL RATE MSRC"`
- `#define VL53L0X_STRING_CHECKENABLE_SIGNAL_RATE_PRE_RANGE "SIGNAL RATE PRE RANGE"`
- `#define VL53L0X_STRING_SEQUENCESTEP_TCC "TCC"`
- `#define VL53L0X_STRING_SEQUENCESTEP_DSS "DSS"`
- `#define VL53L0X_STRING_SEQUENCESTEP_MSRC "MSRC"`
- `#define VL53L0X_STRING_SEQUENCESTEP_PRE_RANGE "PRE RANGE"`
- `#define VL53L0X_STRING_SEQUENCESTEP_FINAL_RANGE "FINAL RANGE"`

## Functions

- `VL53L0X_Error VL53L0X_get_device_info (VL53L0X_DEV Dev, VL53L0X_DeviceInfo_t *pVL53L0X_DeviceInfo)`
- `VL53L0X_Error VL53L0X_get_device_error_string (VL53L0X_DeviceError ErrorCode, char *pDeviceErrorString)`
- `VL53L0X_Error VL53L0X_get_range_status_string (uint8_t RangeStatus, char *pRangeStatusString)`
- `VL53L0X_Error VL53L0X_get_pal_error_string (VL53L0X_Error PalErrorCode, char *pPalErrorString)`
- `VL53L0X_Error VL53L0X_get_pal_state_string (VL53L0X_State PalStateCode, char *pPalStateString)`
- `VL53L0X_Error VL53L0X_get_sequence_steps_info (VL53L0X_SequenceStepId SequenceStepId, char *pSequenceStepsString)`
- `VL53L0X_Error VL53L0X_get_limit_check_info (VL53L0X_DEV Dev, uint16_t LimitCheckId, char *pLimitCheckString)`

## Macro Definition Documentation

**#define VL53L0X\_STRING\_DEVICE\_INFO\_NAME "VL53L0X cut1.0"**

Definition at line 145 of file vl53l0x\_api\_strings.h.

**#define VL53L0X\_STRING\_DEVICE\_INFO\_NAME\_TS0 "VL53L0X TS0"**

Definition at line 146 of file vl53l0x\_api\_strings.h.

**#define VL53L0X\_STRING\_DEVICE\_INFO\_NAME\_TS1 "VL53L0X TS1"**

Definition at line 147 of file vl53l0x\_api\_strings.h.

**#define VL53L0X\_STRING\_DEVICE\_INFO\_NAME\_TS2 "VL53L0X TS2"**

Definition at line 148 of file vl53l0x\_api\_strings.h.

**#define VL53L0X\_STRING\_DEVICE\_INFO\_NAME\_ES1 "VL53L0X ES1 or later"**

Definition at line 149 of file vl53l0x\_api\_strings.h.

**#define VL53L0X\_STRING\_DEVICE\_INFO\_TYPE "VL53L0X"**

Definition at line 150 of file vl53l0x\_api\_strings.h.

**#define VL53L0X\_STRING\_ERROR\_NONE "No Error"**

Definition at line 153 of file vl53l0x\_api\_strings.h.

**#define VL53L0X\_STRING\_ERROR\_CALIBRATION\_WARNING "Calibration Warning Error"**

Definition at line 155 of file vl53l0x\_api\_strings.h.

**#define VL53L0X\_STRING\_ERROR\_MIN\_CLIPPED "Min clipped error"**

Definition at line 157 of file vl53l0x\_api\_strings.h.

**#define VL53L0X\_STRING\_ERROR\_UNDEFINED "Undefined error"**

Definition at line 159 of file vl53l0x\_api\_strings.h.

**#define VL53L0X\_STRING\_ERROR\_INVALID\_PARAMS "Invalid parameters error"**

Definition at line 161 of file vl53l0x\_api\_strings.h.

**#define VL53L0X\_STRING\_ERROR\_NOT\_SUPPORTED "Not supported error"**

Definition at line 163 of file vl53l0x\_api\_strings.h.

**#define VL53L0X\_STRING\_ERROR\_RANGE\_ERROR "Range error"**

Definition at line 165 of file vl53l0x\_api\_strings.h.

**#define VL53L0X\_STRING\_ERROR\_TIME\_OUT "Time out error"**

Definition at line 167 of file vl53l0x\_api\_strings.h.

**#define VL53L0X\_STRING\_ERROR\_MODE\_NOT\_SUPPORTED "Mode not supported error"**

Definition at line 169 of file vl53l0x\_api\_strings.h.

**#define VL53L0X\_STRING\_ERROR\_BUFFER\_TOO\_SMALL "Buffer too small"**

Definition at line 171 of file vl53l0x\_api\_strings.h.

**#define VL53L0X\_STRING\_ERROR\_GPIO\_NOT\_EXISTING "GPIO not existing"**

Definition at line 173 of file vl53l0x\_api\_strings.h.

**#define VL53L0X\_STRING\_ERROR\_GPIO\_FUNCTIONALITY\_NOT\_SUPPORTED "GPIO funct not supported"**

Definition at line 175 of file vl53l0x\_api\_strings.h.

**#define VL53L0X\_STRING\_ERROR\_INTERRUPT\_NOT\_CLEARED "Interrupt not Cleared"**

Definition at line 177 of file vl53l0x\_api\_strings.h.

**#define VL53L0X\_STRING\_ERROR\_CONTROL\_INTERFACE "Control Interface Error"**

Definition at line 179 of file vl53l0x\_api\_strings.h.

**#define VL53L0X\_STRING\_ERROR\_INVALID\_COMMAND "Invalid Command Error"**

Definition at line 181 of file vl53l0x\_api\_strings.h.

**#define VL53L0X\_STRING\_ERROR\_DIVISION\_BY\_ZERO "Division by zero Error"**

Definition at line 183 of file vl53l0x\_api\_strings.h.

**#define VL53L0X\_STRING\_ERROR\_REF\_SPAD\_INIT "Reference Spad Init Error"**

Definition at line 185 of file vl53l0x\_api\_strings.h.

**#define VL53L0X\_STRING\_ERROR\_NOT\_IMPLEMENTED "Not implemented error"**

Definition at line 187 of file vl53l0x\_api\_strings.h.

**#define VL53L0X\_STRING\_UNKNOW\_ERROR\_CODE "Unknown Error Code"**

Definition at line 190 of file vl53l0x\_api\_strings.h.

**#define VL53L0X\_STRING\_RANGESTATUS\_NONE "No Update"**

Definition at line 196 of file vl53l0x\_api\_strings.h.

**#define VL53L0X\_STRING\_RANGESTATUS\_RANGEVALID "Range Valid"**

Definition at line 197 of file vl53l0x\_api\_strings.h.

**#define VL53L0X\_STRING\_RANGESTATUS\_SIGMA "Sigma Fail"**

Definition at line 198 of file vl53l0x\_api\_strings.h.

**#define VL53L0X\_STRING\_RANGESTATUS\_SIGNAL "Signal Fail"**

Definition at line 199 of file vl53l0x\_api\_strings.h.

**#define VL53L0X\_STRING\_RANGESTATUS\_MINRANGE "Min Range Fail"**

Definition at line 200 of file vl53l0x\_api\_strings.h.

**#define VL53L0X\_STRING\_RANGESTATUS\_PHASE "Phase Fail"**

Definition at line 201 of file vl53l0x\_api\_strings.h.

**#define VL53L0X\_STRING\_RANGESTATUS\_HW "Hardware Fail"**

Definition at line 202 of file vl53l0x\_api\_strings.h.

**#define VL53L0X\_STRING\_STATE\_POWERDOWN "POWERDOWN State"**

Definition at line 206 of file vl53l0x\_api\_strings.h.

**#define VL53L0X\_STRING\_STATE\_WAIT\_STATICINIT "Wait for staticinit State"**

Definition at line 207 of file vl53l0x\_api\_strings.h.

**#define VL53L0X\_STRING\_STATE\_STANDBY "STANDBY State"**

Definition at line 209 of file vl53l0x\_api\_strings.h.

**#define VL53L0X\_STRING\_STATE\_IDLE "IDLE State"**

Definition at line 210 of file vl53l0x\_api\_strings.h.

**#define VL53L0X\_STRING\_STATE\_RUNNING "RUNNING State"**

Definition at line 211 of file vl53l0x\_api\_strings.h.

**#define VL53L0X\_STRING\_STATE\_UNKNOWN "UNKNOWN State"**

Definition at line 212 of file vl53l0x\_api\_strings.h.

**#define VL53L0X\_STRING\_STATE\_ERROR "ERROR State"**

Definition at line 213 of file vl53l0x\_api\_strings.h.

**#define VL53L0X\_STRING\_DEVICEERROR\_NONE "No Update"**

Definition at line 217 of file vl53l0x\_api\_strings.h.

**#define VL53L0X\_STRING\_DEVICEERROR\_VCSELCONTINUITYTESTFAILURE "VCSEL Continuity Test Failure"**

Definition at line 218 of file vl53l0x\_api\_strings.h.

**#define VL53L0X\_STRING\_DEVICEERROR\_VCSELWATCHDOGTESTFAILURE "VCSEL Watchdog Test Failure"**

Definition at line 220 of file vl53l0x\_api\_strings.h.

**#define VL53L0X\_STRING\_DEVICEERROR\_NOVHVVALUEFOUND "No VHV Value found"**

Definition at line 222 of file vl53l0x\_api\_strings.h.

**#define VL53L0X\_STRING\_DEVICEERROR\_MSRCNOTARGET "MSRC No Target Error"**

Definition at line 224 of file vl53l0x\_api\_strings.h.

**#define VL53L0X\_STRING\_DEVICEERROR\_SNRCHECK "SNR Check Exit"**

Definition at line 226 of file vl53l0x\_api\_strings.h.

**#define VL53L0X\_STRING\_DEVICEERROR\_RANGEPHASECHECK "Range Phase Check Error"**

Definition at line 228 of file vl53l0x\_api\_strings.h.

**#define VL53L0X\_STRING\_DEVICEERROR\_SIGMATHRESHOLDCHECK "Sigma Threshold Check Error"**

Definition at line 230 of file vl53l0x\_api\_strings.h.

**#define VL53L0X\_STRING\_DEVICEERROR\_TCC "TCC Error"**

Definition at line 232 of file vl53l0x\_api\_strings.h.

**#define VL53L0X\_STRING\_DEVICEERROR\_PHASECONSISTENCY "Phase Consistency Error"**

Definition at line 234 of file vl53l0x\_api\_strings.h.

**#define VL53L0X\_STRING\_DEVICEERROR\_MINCLIP "Min Clip Error"**

Definition at line 236 of file vl53l0x\_api\_strings.h.

**#define VL53L0X\_STRING\_DEVICEERROR\_RANGECOMPLETE "Range Complete"**

Definition at line 238 of file vl53l0x\_api\_strings.h.

**#define VL53L0X\_STRING\_DEVICEERROR\_ALGOUNDERFLOW "Range Algo Underflow Error"**

Definition at line 240 of file vl53l0x\_api\_strings.h.

**#define VL53L0X\_STRING\_DEVICEERROR\_ALGOOVERFLOW "Range Algo Overflow Error"**

Definition at line 242 of file vl53l0x\_api\_strings.h.

**#define VL53L0X\_STRING\_DEVICEERROR\_RANGEIGNORETHRESHOLD "Range Ignore Threshold Error"**

Definition at line 244 of file vl53l0x\_api\_strings.h.

**#define VL53L0X\_STRING\_DEVICEERROR\_UNKNOWN "Unknown error code"**

Definition at line 246 of file vl53l0x\_api\_strings.h.

**#define VL53L0X\_STRING\_CHECKENABLE\_SIGMA\_FINAL\_RANGE "SIGMA FINAL RANGE"**

Definition at line 250 of file vl53l0x\_api\_strings.h.

**#define VL53L0X\_STRING\_CHECKENABLE\_SIGNAL\_RATE\_FINAL\_RANGE "SIGNAL RATE FINAL RANGE"**

Definition at line 252 of file vl53l0x\_api\_strings.h.



**#define VL53L0X\_STRING\_CHECKENABLE\_SIGNAL\_REF\_CLIP "SIGNAL REF CLIP"**

Definition at line 254 of file vl53l0x\_api\_strings.h.

**#define VL53L0X\_STRING\_CHECKENABLE\_RANGE\_IGNORE\_THRESHOLD "RANGE IGNORE THRESHOLD"**

Definition at line 256 of file vl53l0x\_api\_strings.h.

**#define VL53L0X\_STRING\_CHECKENABLE\_SIGNAL\_RATE\_MSRC "SIGNAL RATE MSRC"**

Definition at line 258 of file vl53l0x\_api\_strings.h.

**#define VL53L0X\_STRING\_CHECKENABLE\_SIGNAL\_RATE\_PRE\_RANGE "SIGNAL RATE PRE RANGE"**

Definition at line 260 of file vl53l0x\_api\_strings.h.

**#define VL53L0X\_STRING\_SEQUENCESTEP\_TCC "TCC"**

Definition at line 264 of file vl53l0x\_api\_strings.h.

**#define VL53L0X\_STRING\_SEQUENCESTEP\_DSS "DSS"**

Definition at line 265 of file vl53l0x\_api\_strings.h.

**#define VL53L0X\_STRING\_SEQUENCESTEP\_MSRC "MSRC"**

Definition at line 266 of file vl53l0x\_api\_strings.h.

**#define VL53L0X\_STRING\_SEQUENCESTEP\_PRE\_RANGE "PRE RANGE"**

Definition at line 267 of file vl53l0x\_api\_strings.h.

**#define VL53L0X\_STRING\_SEQUENCESTEP\_FINAL\_RANGE "FINAL RANGE"**

Definition at line 268 of file vl53l0x\_api\_strings.h.

---

## Function Documentation

[VL53L0X\\_Error](#) VL53L0X\_get\_device\_info ([VL53L0X\\_DEV](#) Dev, [VL53L0X\\_DeviceInfo\\_t](#) \* pVL53L0X\_DeviceInfo)

[VL53L0X\\_Error](#) VL53L0X\_get\_device\_error\_string ([VL53L0X\\_DeviceError](#) ErrorCode, char \* pDeviceErrorString)

[VL53L0X\\_Error](#) VL53L0X\_get\_range\_status\_string ([uint8\\_t](#) RangeStatus, char \* pRangeStatusString)

[VL53L0X\\_Error](#) VL53L0X\_get\_pal\_error\_string ([VL53L0X\\_Error](#) PalErrorCode, char \* pPalErrorString)

[VL53L0X\\_Error](#) VL53L0X\_get\_pal\_state\_string ([VL53L0X\\_State](#) PalStateCode, char \* pPalStateString)

[VL53L0X\\_Error](#) VL53L0X\_get\_sequence\_steps\_info ([VL53L0X\\_SequenceStepId](#) SequenceStepId, char \* pSequenceStepsString)

[VL53L0X\\_Error](#) VL53L0X\_get\_limit\_check\_info ([VL53L0X\\_DEV](#) Dev, [uint16\\_t](#) LimitCheckId, char \* pLimitCheckString)

## vl53l0x\_def.h File Reference

Type definitions for VL53L0X API.

```
#include "vl53l0x_device.h"
```

```
#include "vl53l0x_types.h"
```

## Data Structures

- struct [VL53L0X\\_Version\\_t](#)
- Defines the parameters of the Get Version Functions. struct [VL53L0X\\_DeviceInfo\\_t](#)
- Defines the parameters of the Get Device Info Functions. struct [VL53L0X\\_DeviceParameters\\_t](#)
- Defines all parameters for the device. struct [VL53L0X\\_DMaxData\\_t](#)
- Structure containing the Dmax computation parameters and data. struct [VL53L0X\\_RangingMeasurementData\\_t](#)
- struct [VL53L0X\\_HistogramMeasurementData\\_t](#)
- struct [VL53L0X\\_SpadData\\_t](#)
- Spad Configuration Data. struct [VL53L0X\\_DeviceSpecificParameters\\_t](#)
- struct [VL53L0X\\_DevData\\_t](#)  
VL53L0X PAL device ST private data structure
- End user should never access any of these field directly. struct [VL53L0X\\_SchedulerSequenceSteps\\_t](#)

## Macros

- #define [VL53L0X10\\_SPECIFICATION\\_VER\\_MAJOR](#) 1  
PAL SPECIFICATION major version.
- #define [VL53L0X10\\_SPECIFICATION\\_VER\\_MINOR](#) 2  
PAL SPECIFICATION minor version.
- #define [VL53L0X10\\_SPECIFICATION\\_VER\\_SUB](#) 7  
PAL SPECIFICATION sub version.

- #define [VL53L0X10\\_SPECIFICATION\\_VER\\_REVISION](#) 1440  
*PAL SPECIFICATION sub version.*
- #define [VL53L0X10\\_IMPLEMENTATION\\_VER\\_MAJOR](#) 1  
*VL53L0X PAL IMPLEMENTATION major version.*
- #define [VL53L0X10\\_IMPLEMENTATION\\_VER\\_MINOR](#) 0  
*VL53L0X PAL IMPLEMENTATION minor version.*
- #define [VL53L0X10\\_IMPLEMENTATION\\_VER\\_SUB](#) 9  
*VL53L0X PAL IMPLEMENTATION sub version.*
- #define [VL53L0X10\\_IMPLEMENTATION\\_VER\\_REVISION](#) 3673  
*VL53L0X PAL IMPLEMENTATION sub version.*
- #define [VL53L0X\\_SPECIFICATION\\_VER\\_MAJOR](#) 1  
*PAL SPECIFICATION major version.*
- #define [VL53L0X\\_SPECIFICATION\\_VER\\_MINOR](#) 2  
*PAL SPECIFICATION minor version.*
- #define [VL53L0X\\_SPECIFICATION\\_VER\\_SUB](#) 7  
*PAL SPECIFICATION sub version.*
- #define [VL53L0X\\_SPECIFICATION\\_VER\\_REVISION](#) 1440  
*PAL SPECIFICATION sub version.*
- #define [VL53L0X\\_IMPLEMENTATION\\_VER\\_MAJOR](#) 1  
*VL53L0X PAL IMPLEMENTATION major version.*
- #define [VL53L0X\\_IMPLEMENTATION\\_VER\\_MINOR](#) 0  
*VL53L0X PAL IMPLEMENTATION minor version.*
- #define [VL53L0X\\_IMPLEMENTATION\\_VER\\_SUB](#) 1  
*VL53L0X PAL IMPLEMENTATION sub version.*
- #define [VL53L0X\\_IMPLEMENTATION\\_VER\\_REVISION](#) 4606  
*VL53L0X PAL IMPLEMENTATION sub version.*
- #define [VL53L0X\\_DEFAULT\\_MAX\\_LOOP](#) 200
- #define [VL53L0X\\_MAX\\_STRING\\_LENGTH](#) 32
- #define [VL53L0X\\_ERROR\\_NONE](#) (([VL53L0X\\_Error](#)) 0)
- #define [VL53L0X\\_ERROR\\_CALIBRATION\\_WARNING](#) (([VL53L0X\\_Error](#)) -1)
- #define [VL53L0X\\_ERROR\\_MIN\\_CLIPPED](#) (([VL53L0X\\_Error](#)) -2)
- #define [VL53L0X\\_ERROR\\_UNDEFINED](#) (([VL53L0X\\_Error](#)) -3)
- #define [VL53L0X\\_ERROR\\_INVALID\\_PARAMS](#) (([VL53L0X\\_Error](#)) -4)
- #define [VL53L0X\\_ERROR\\_NOT\\_SUPPORTED](#) (([VL53L0X\\_Error](#)) -5)
- #define [VL53L0X\\_ERROR\\_RANGE\\_ERROR](#) (([VL53L0X\\_Error](#)) -6)
- #define [VL53L0X\\_ERROR\\_TIME\\_OUT](#) (([VL53L0X\\_Error](#)) -7)
- #define [VL53L0X\\_ERROR\\_MODE\\_NOT\\_SUPPORTED](#) (([VL53L0X\\_Error](#)) -8)
- #define [VL53L0X\\_ERROR\\_BUFFER\\_TOO\\_SMALL](#) (([VL53L0X\\_Error](#)) -9)
- #define [VL53L0X\\_ERROR\\_GPIO\\_NOT\\_EXISTING](#) (([VL53L0X\\_Error](#)) -10)
- #define [VL53L0X\\_ERROR\\_GPIO\\_FUNCTIONALITY\\_NOT\\_SUPPORTED](#) (([VL53L0X\\_Error](#)) -11)
- #define [VL53L0X\\_ERROR\\_INTERRUPT\\_NOT\\_CLEARED](#) (([VL53L0X\\_Error](#)) -12)
- #define [VL53L0X\\_ERROR\\_CONTROL\\_INTERFACE](#) (([VL53L0X\\_Error](#)) -20)
- #define [VL53L0X\\_ERROR\\_INVALID\\_COMMAND](#) (([VL53L0X\\_Error](#)) -30)
- #define [VL53L0X\\_ERROR\\_DIVISION\\_BY\\_ZERO](#) (([VL53L0X\\_Error](#)) -40)
- #define [VL53L0X\\_ERROR\\_REF\\_SPAD\\_INIT](#) (([VL53L0X\\_Error](#)) -50)
- #define [VL53L0X\\_ERROR\\_NOT\\_IMPLEMENTED](#) (([VL53L0X\\_Error](#)) -99)
- #define [VL53L0X\\_DEVICEMODE\\_SINGLE\\_RANGING](#) (([VL53L0X\\_DeviceModes](#)) 0)
- #define [VL53L0X\\_DEVICEMODE\\_CONTINUOUS\\_RANGING](#) (([VL53L0X\\_DeviceModes](#)) 1)
- #define [VL53L0X\\_DEVICEMODE\\_SINGLE\\_HISTOGRAM](#) (([VL53L0X\\_DeviceModes](#)) 2)
- #define [VL53L0X\\_DEVICEMODE\\_CONTINUOUS\\_TIMED\\_RANGING](#) (([VL53L0X\\_DeviceModes](#)) 3)
- #define [VL53L0X\\_DEVICEMODE\\_SINGLE\\_ALS](#) (([VL53L0X\\_DeviceModes](#)) 10)
- #define [VL53L0X\\_DEVICEMODE\\_GPIO\\_DRIVE](#) (([VL53L0X\\_DeviceModes](#)) 20)
- #define [VL53L0X\\_DEVICEMODE\\_GPIO\\_OSC](#) (([VL53L0X\\_DeviceModes](#)) 21)

- #define [VL53L0X\\_HISTOGRAMMODE\\_DISABLED](#) (([VL53L0X\\_HistogramModes](#)) 0)
- #define [VL53L0X\\_HISTOGRAMMODE\\_REFERENCE\\_ONLY](#) (([VL53L0X\\_HistogramModes](#)) 1)
- #define [VL53L0X\\_HISTOGRAMMODE\\_RETURN\\_ONLY](#) (([VL53L0X\\_HistogramModes](#)) 2)
- #define [VL53L0X\\_HISTOGRAMMODE\\_BOTH](#) (([VL53L0X\\_HistogramModes](#)) 3)
- #define [VL53L0X\\_POWERMODE\\_STANDBY\\_LEVEL1](#) (([VL53L0X\\_PowerModes](#)) 0)
- #define [VL53L0X\\_POWERMODE\\_STANDBY\\_LEVEL2](#) (([VL53L0X\\_PowerModes](#)) 1)
- #define [VL53L0X\\_POWERMODE\\_IDLE\\_LEVEL1](#) (([VL53L0X\\_PowerModes](#)) 2)
- #define [VL53L0X\\_POWERMODE\\_IDLE\\_LEVEL2](#) (([VL53L0X\\_PowerModes](#)) 3)
- #define [VL53L0X\\_STATE\\_POWERDOWN](#) (([VL53L0X\\_State](#)) 0)
- #define [VL53L0X\\_STATE\\_WAIT\\_STATICINIT](#) (([VL53L0X\\_State](#)) 1)
- #define [VL53L0X\\_STATE\\_STANDBY](#) (([VL53L0X\\_State](#)) 2)
- #define [VL53L0X\\_STATE\\_IDLE](#) (([VL53L0X\\_State](#)) 3)
- #define [VL53L0X\\_STATE\\_RUNNING](#) (([VL53L0X\\_State](#)) 4)
- #define [VL53L0X\\_STATE\\_UNKNOWN](#) (([VL53L0X\\_State](#)) 98)
- #define [VL53L0X\\_STATE\\_ERROR](#) (([VL53L0X\\_State](#)) 99)
- #define [VL53L0X\\_HISTOGRAM\\_BUFFER\\_SIZE](#) 24
- #define [VL53L0X\\_REF\\_SPAD\\_BUFFER\\_SIZE](#) 6
- #define [VL53L0X\\_INTERRUPTPOLARITY\\_LOW](#) (([VL53L0X\\_InterruptPolarity](#)) 0)
- #define [VL53L0X\\_INTERRUPTPOLARITY\\_HIGH](#) (([VL53L0X\\_InterruptPolarity](#)) 1)
- #define [VL53L0X\\_VCSEL\\_PERIOD\\_PRE\\_RANGE](#) (([VL53L0X\\_VcSELPeriod](#)) 0)
- #define [VL53L0X\\_VCSEL\\_PERIOD\\_FINAL\\_RANGE](#) (([VL53L0X\\_VcSELPeriod](#)) 1)
- #define [VL53L0X\\_SEQUENCESTEP\\_TCC](#) (([VL53L0X\\_VcSELPeriod](#)) 0)
- #define [VL53L0X\\_SEQUENCESTEP\\_DSS](#) (([VL53L0X\\_VcSELPeriod](#)) 1)
- #define [VL53L0X\\_SEQUENCESTEP\\_MSRC](#) (([VL53L0X\\_VcSELPeriod](#)) 2)
- #define [VL53L0X\\_SEQUENCESTEP\\_PRE\\_RANGE](#) (([VL53L0X\\_VcSELPeriod](#)) 3)
- #define [VL53L0X\\_SEQUENCESTEP\\_FINAL\\_RANGE](#) (([VL53L0X\\_VcSELPeriod](#)) 4)
- #define [VL53L0X\\_SEQUENCESTEP\\_NUMBER\\_OF\\_CHECKS](#) 5
- #define [VL53L0X\\_SETPARAMETERFIELD](#)(Dev, field, value) [PALDevDataSet](#)(Dev, CurrentParameters.field, value)
- #define [VL53L0X\\_GETPARAMETERFIELD](#)(Dev, field, variable) variable = [PALDevDataGet](#)(Dev, CurrentParameters).field
- #define [VL53L0X\\_SETARRAYPARAMETERFIELD](#)(Dev, field, index, value) [PALDevDataSet](#)(Dev, CurrentParameters.field[index], value)
- #define [VL53L0X\\_GETARRAYPARAMETERFIELD](#)(Dev, field, index, variable) variable = [PALDevDataGet](#)(Dev, CurrentParameters).field[index]
- #define [VL53L0X\\_SETDEVICESTRUCTUREPARAMETER](#)(Dev, field, value) [PALDevDataSet](#)(Dev, DeviceSpecificParameters.field, value)
- #define [VL53L0X\\_GETDEVICESTRUCTUREPARAMETER](#)(Dev, field) [PALDevDataGet](#)(Dev, DeviceSpecificParameters).field
- #define [VL53L0X\\_FIXPOINT1616TOFIXPOINT97](#)(Value) ([uint16\\_t](#))((Value>>9)&0xFFFF)
- #define [VL53L0X\\_FIXPOINT97TOFIXPOINT1616](#)(Value) ([FixPoint1616\\_t](#))(Value<<9)
- #define [VL53L0X\\_FIXPOINT1616TOFIXPOINT88](#)(Value) ([uint16\\_t](#))((Value>>8)&0xFFFF)
- #define [VL53L0X\\_FIXPOINT88TOFIXPOINT1616](#)(Value) ([FixPoint1616\\_t](#))(Value<<8)
- #define [VL53L0X\\_FIXPOINT1616TOFIXPOINT412](#)(Value) ([uint16\\_t](#))((Value>>4)&0xFFFF)
- #define [VL53L0X\\_FIXPOINT412TOFIXPOINT1616](#)(Value) ([FixPoint1616\\_t](#))(Value<<4)
- #define [VL53L0X\\_FIXPOINT1616TOFIXPOINT313](#)(Value) ([uint16\\_t](#))((Value>>3)&0xFFFF)
- #define [VL53L0X\\_FIXPOINT313TOFIXPOINT1616](#)(Value) ([FixPoint1616\\_t](#))(Value<<3)
- #define [VL53L0X\\_FIXPOINT1616TOFIXPOINT08](#)(Value) ([uint8\\_t](#))((Value>>8)&0x00FF)
- #define [VL53L0X\\_FIXPOINT08TOFIXPOINT1616](#)(Value) ([FixPoint1616\\_t](#))(Value<<8)
- #define [VL53L0X\\_FIXPOINT1616TOFIXPOINT53](#)(Value) ([uint8\\_t](#))((Value>>13)&0x00FF)
- #define [VL53L0X\\_FIXPOINT53TOFIXPOINT1616](#)(Value) ([FixPoint1616\\_t](#))(Value<<13)
- #define [VL53L0X\\_FIXPOINT1616TOFIXPOINT102](#)(Value) ([uint16\\_t](#))((Value>>14)&0x0FFF)
- #define [VL53L0X\\_FIXPOINT102TOFIXPOINT1616](#)(Value) ([FixPoint1616\\_t](#))(Value<<12)
- #define [VL53L0X\\_MAKEUINT16](#)(lsb, msb)

## Typedefs

- typedef [int8\\_t](#) [VL53L0X\\_Error](#)

- typedef [uint8\\_t VL53L0X\\_DeviceModes](#)
- typedef [uint8\\_t VL53L0X\\_HistogramModes](#)
- typedef [uint8\\_t VL53L0X\\_PowerModes](#)
- typedef [uint8\\_t VL53L0X\\_State](#)
- typedef [uint8\\_t VL53L0X\\_InterruptPolarity](#)
- typedef [uint8\\_t VL53L0X\\_VcselPeriod](#)
- typedef [uint8\\_t VL53L0X\\_SequenceStepId](#)

## Detailed Description

Type definitions for VL53L0X API.

## vl53l0x\_device.h File Reference

```
#include "vl53l0x_types.h"
```

### Macros

- #define [VL53L0X\\_DEVICEERROR\\_NONE](#) (([VL53L0X\\_DeviceError](#)) 0)
- #define [VL53L0X\\_DEVICEERROR\\_VCSELCONTINUITYTESTFAILURE](#) (([VL53L0X\\_DeviceError](#)) 1)
- #define [VL53L0X\\_DEVICEERROR\\_VCSELWATCHDOGTESTFAILURE](#) (([VL53L0X\\_DeviceError](#)) 2)
- #define [VL53L0X\\_DEVICEERROR\\_NOVHVVALUEFOUND](#) (([VL53L0X\\_DeviceError](#)) 3)
- #define [VL53L0X\\_DEVICEERROR\\_MSRCNOTARGET](#) (([VL53L0X\\_DeviceError](#)) 4)
- #define [VL53L0X\\_DEVICEERROR\\_SNRCHECK](#) (([VL53L0X\\_DeviceError](#)) 5)
- #define [VL53L0X\\_DEVICEERROR\\_RANGEPHASECHECK](#) (([VL53L0X\\_DeviceError](#)) 6)
- #define [VL53L0X\\_DEVICEERROR\\_SIGMATHRESHOLDCHECK](#) (([VL53L0X\\_DeviceError](#)) 7)
- #define [VL53L0X\\_DEVICEERROR\\_TCC](#) (([VL53L0X\\_DeviceError](#)) 8)
- #define [VL53L0X\\_DEVICEERROR\\_PHASECONSISTENCY](#) (([VL53L0X\\_DeviceError](#)) 9)
- #define [VL53L0X\\_DEVICEERROR\\_MINCLIP](#) (([VL53L0X\\_DeviceError](#)) 10)
- #define [VL53L0X\\_DEVICEERROR\\_RANGECOMPLETE](#) (([VL53L0X\\_DeviceError](#)) 11)
- #define [VL53L0X\\_DEVICEERROR\\_ALGOUNDERFLOW](#) (([VL53L0X\\_DeviceError](#)) 12)
- #define [VL53L0X\\_DEVICEERROR\\_ALGOOVERFLOW](#) (([VL53L0X\\_DeviceError](#)) 13)
- #define [VL53L0X\\_DEVICEERROR\\_RANGEIGNORETHRESHOLD](#) (([VL53L0X\\_DeviceError](#)) 14)
- #define [VL53L0X\\_CHECKENABLE\\_SIGMA\\_FINAL\\_RANGE](#) 0
- #define [VL53L0X\\_CHECKENABLE\\_SIGNAL\\_RATE\\_FINAL\\_RANGE](#) 1
- #define [VL53L0X\\_CHECKENABLE\\_SIGNAL\\_REF\\_CLIP](#) 2
- #define [VL53L0X\\_CHECKENABLE\\_RANGE\\_IGNORE\\_THRESHOLD](#) 3
- #define [VL53L0X\\_CHECKENABLE\\_SIGNAL\\_RATE\\_MSRC](#) 4
- #define [VL53L0X\\_CHECKENABLE\\_SIGNAL\\_RATE\\_PRE\\_RANGE](#) 5
- #define [VL53L0X\\_CHECKENABLE\\_NUMBER\\_OF\\_CHECKS](#) 6
- #define [VL53L0X\\_GPIOFUNCTIONALITY\\_OFF](#) (([VL53L0X\\_GpioFunctionality](#)) 0)
- #define [VL53L0X\\_GPIOFUNCTIONALITY\\_THRESHOLD\\_CROSSED\\_LOW](#) (([VL53L0X\\_GpioFunctionality](#)) 1)
- #define [VL53L0X\\_GPIOFUNCTIONALITY\\_THRESHOLD\\_CROSSED\\_HIGH](#) (([VL53L0X\\_GpioFunctionality](#)) 2)
- #define [VL53L0X\\_GPIOFUNCTIONALITY\\_THRESHOLD\\_CROSSED\\_OUT](#) (([VL53L0X\\_GpioFunctionality](#)) 3)

- #define [VL53L0X\\_GPIOFUNCTIONALITY\\_NEW\\_MEASURE\\_READY](#) (([VL53L0X\\_GpioFunctionality](#)) 4)
- #define [VL53L0X\\_REG\\_SYSRANGE\\_START](#) 0x000
- #define [VL53L0X\\_REG\\_SYSRANGE\\_MODE\\_MASK](#) 0x0F  
*mask existing bit in [VL53L0X\\_REG\\_SYSRANGE\\_START](#)*
- #define [VL53L0X\\_REG\\_SYSRANGE\\_MODE\\_START\\_STOP](#) 0x01  
*bit 0 in [VL53L0X\\_REG\\_SYSRANGE\\_START](#) write 1 toggle state in continuous mode and arm next shot in single shot mode*
- #define [VL53L0X\\_REG\\_SYSRANGE\\_MODE\\_SINGLESOT](#) 0x00  
*bit 1 write 0 in [VL53L0X\\_REG\\_SYSRANGE\\_START](#) set single shot mode*
- #define [VL53L0X\\_REG\\_SYSRANGE\\_MODE\\_BACKTOBACK](#) 0x02  
*bit 1 write 1 in [VL53L0X\\_REG\\_SYSRANGE\\_START](#) set back-to-back operation mode*
- #define [VL53L0X\\_REG\\_SYSRANGE\\_MODE\\_TIMED](#) 0x04  
*bit 2 write 1 in [VL53L0X\\_REG\\_SYSRANGE\\_START](#) set timed operation mode*
- #define [VL53L0X\\_REG\\_SYSRANGE\\_MODE\\_HISTOGRAM](#) 0x08  
*bit 3 write 1 in [VL53L0X\\_REG\\_SYSRANGE\\_START](#) set histogram operation mode*
- #define [VL53L0X\\_REG\\_SYSTEM\\_THRESH\\_HIGH](#) 0x000C
- #define [VL53L0X\\_REG\\_SYSTEM\\_THRESH\\_LOW](#) 0x000E
- #define [VL53L0X\\_REG\\_SYSTEM\\_SEQUENCE\\_CONFIG](#) 0x0001
- #define [VL53L0X\\_REG\\_SYSTEM\\_RANGE\\_CONFIG](#) 0x0009
- #define [VL53L0X\\_REG\\_SYSTEM\\_INTERMEASUREMENT\\_PERIOD](#) 0x0004
- #define [VL53L0X\\_REG\\_SYSTEM\\_INTERRUPT\\_CONFIG\\_GPIO](#) 0x000A
- #define [VL53L0X\\_REG\\_SYSTEM\\_INTERRUPT\\_GPIO\\_DISABLED](#) 0x00
- #define [VL53L0X\\_REG\\_SYSTEM\\_INTERRUPT\\_GPIO\\_LEVEL\\_LOW](#) 0x01
- #define [VL53L0X\\_REG\\_SYSTEM\\_INTERRUPT\\_GPIO\\_LEVEL\\_HIGH](#) 0x02
- #define [VL53L0X\\_REG\\_SYSTEM\\_INTERRUPT\\_GPIO\\_OUT\\_OF\\_WINDOW](#) 0x03
- #define [VL53L0X\\_REG\\_SYSTEM\\_INTERRUPT\\_GPIO\\_NEW\\_SAMPLE\\_READY](#) 0x04
- #define [VL53L0X\\_REG\\_GPIO\\_HV\\_MUX\\_ACTIVE\\_HIGH](#) 0x0084
- #define [VL53L0X\\_REG\\_SYSTEM\\_INTERRUPT\\_CLEAR](#) 0x000B
- #define [VL53L0X\\_REG\\_RESULT\\_INTERRUPT\\_STATUS](#) 0x0013
- #define [VL53L0X\\_REG\\_RESULT\\_RANGE\\_STATUS](#) 0x0014
- #define [VL53L0X\\_REG\\_RESULT\\_CORE\\_PAGE](#) 1
- #define [VL53L0X\\_REG\\_RESULT\\_CORE\\_AMBIENT\\_WINDOW\\_EVENTS RTN](#) 0x00BC
- #define [VL53L0X\\_REG\\_RESULT\\_CORE\\_RANGING\\_TOTAL\\_EVENTS RTN](#) 0x00C0
- #define [VL53L0X\\_REG\\_RESULT\\_CORE\\_AMBIENT\\_WINDOW\\_EVENTS REF](#) 0x00D0
- #define [VL53L0X\\_REG\\_RESULT\\_CORE\\_RANGING\\_TOTAL\\_EVENTS REF](#) 0x00D4
- #define [VL53L0X\\_REG\\_RESULT\\_PEAK\\_SIGNAL\\_RATE\\_REF](#) 0x00B6
- #define [VL53L0X\\_REG\\_ALGO\\_PART\\_TO\\_PART\\_RANGE\\_OFFSET\\_MM](#) 0x0028
- #define [VL53L0X\\_REG\\_I2C\\_SLAVE\\_DEVICE\\_ADDRESS](#) 0x008a
- #define [VL53L0X\\_REG\\_MSRC\\_CONFIG\\_CONTROL](#) 0x0060
- #define [VL53L0X\\_REG\\_PRE\\_RANGE\\_CONFIG\\_MIN\\_SNR](#) 0x0027
- #define [VL53L0X\\_REG\\_PRE\\_RANGE\\_CONFIG\\_VALID\\_PHASE\\_LOW](#) 0x0056
- #define [VL53L0X\\_REG\\_PRE\\_RANGE\\_CONFIG\\_VALID\\_PHASE\\_HIGH](#) 0x0057
- #define [VL53L0X\\_REG\\_PRE\\_RANGE\\_MIN\\_COUNT\\_RATE RTN LIMIT](#) 0x0064
- #define [VL53L0X\\_REG\\_FINAL\\_RANGE\\_CONFIG\\_MIN\\_SNR](#) 0x0067
- #define [VL53L0X\\_REG\\_FINAL\\_RANGE\\_CONFIG\\_VALID\\_PHASE\\_LOW](#) 0x0047
- #define [VL53L0X\\_REG\\_FINAL\\_RANGE\\_CONFIG\\_VALID\\_PHASE\\_HIGH](#) 0x0048
- #define [VL53L0X\\_REG\\_FINAL\\_RANGE\\_CONFIG\\_MIN\\_COUNT\\_RATE RTN LIMIT](#) 0x0044
- #define [VL53L0X\\_REG\\_PRE\\_RANGE\\_CONFIG\\_SIGMA\\_THRESH\\_HI](#) 0x0061
- #define [VL53L0X\\_REG\\_PRE\\_RANGE\\_CONFIG\\_SIGMA\\_THRESH\\_LO](#) 0x0062
- #define [VL53L0X\\_REG\\_PRE\\_RANGE\\_CONFIG\\_VCSEL\\_PERIOD](#) 0x0050
- #define [VL53L0X\\_REG\\_PRE\\_RANGE\\_CONFIG\\_TIMEOUT\\_MACROP\\_HI](#) 0x0051
- #define [VL53L0X\\_REG\\_PRE\\_RANGE\\_CONFIG\\_TIMEOUT\\_MACROP\\_LO](#) 0x0052
- #define [VL53L0X\\_REG\\_SYSTEM\\_HISTOGRAM\\_BIN](#) 0x0081
- #define [VL53L0X\\_REG\\_HISTOGRAM\\_CONFIG\\_INITIAL\\_PHASE\\_SELECT](#) 0x0033



- #define [VL53L0X\\_REG\\_HISTOGRAM\\_CONFIG\\_READOUT\\_CTRL](#) 0x0055
- #define [VL53L0X\\_REG\\_FINAL\\_RANGE\\_CONFIG\\_VCSEL\\_PERIOD](#) 0x0070
- #define [VL53L0X\\_REG\\_FINAL\\_RANGE\\_CONFIG\\_TIMEOUT\\_MACROP\\_HI](#) 0x0071
- #define [VL53L0X\\_REG\\_FINAL\\_RANGE\\_CONFIG\\_TIMEOUT\\_MACROP\\_LO](#) 0x0072
- #define [VL53L0X\\_REG\\_CROSSTALK\\_COMPENSATION\\_PEAK\\_RATE\\_MCPS](#) 0x0020
- #define [VL53L0X\\_REG\\_MSRC\\_CONFIG\\_TIMEOUT\\_MACROP](#) 0x0046
- #define [VL53L0X\\_REG\\_SOFT\\_RESET\\_GO2\\_SOFT\\_RESET\\_N](#) 0x00bf
- #define [VL53L0X\\_REG\\_IDENTIFICATION\\_MODEL\\_ID](#) 0x00c0
- #define [VL53L0X\\_REG\\_IDENTIFICATION\\_REVISION\\_ID](#) 0x00c2
- #define [VL53L0X\\_REG\\_OSC\\_CALIBRATE\\_VAL](#) 0x00f8
- #define [VL53L0X\\_SIGMA\\_ESTIMATE\\_MAX\\_VALUE](#) 65535
- #define [VL53L0X\\_REG\\_GLOBAL\\_CONFIG\\_VCSEL\\_WIDTH](#) 0x032
- #define [VL53L0X\\_REG\\_GLOBAL\\_CONFIG\\_SPAD\\_ENABLES\\_REF\\_0](#) 0x0B0
- #define [VL53L0X\\_REG\\_GLOBAL\\_CONFIG\\_SPAD\\_ENABLES\\_REF\\_1](#) 0x0B1
- #define [VL53L0X\\_REG\\_GLOBAL\\_CONFIG\\_SPAD\\_ENABLES\\_REF\\_2](#) 0x0B2
- #define [VL53L0X\\_REG\\_GLOBAL\\_CONFIG\\_SPAD\\_ENABLES\\_REF\\_3](#) 0x0B3
- #define [VL53L0X\\_REG\\_GLOBAL\\_CONFIG\\_SPAD\\_ENABLES\\_REF\\_4](#) 0x0B4
- #define [VL53L0X\\_REG\\_GLOBAL\\_CONFIG\\_SPAD\\_ENABLES\\_REF\\_5](#) 0x0B5
- #define [VL53L0X\\_REG\\_GLOBAL\\_CONFIG\\_REF\\_EN\\_START\\_SELECT](#) 0xB6
- #define [VL53L0X\\_REG\\_DYNAMIC\\_SPAD\\_NUM\\_REQUESTED\\_REF\\_SPAD](#) 0x4E /\* 0x14E \*/
- #define [VL53L0X\\_REG\\_DYNAMIC\\_SPAD\\_REF\\_EN\\_START\\_OFFSET](#) 0x4F /\* 0x14F \*/
- #define [VL53L0X\\_REG\\_POWER\\_MANAGEMENT\\_GO1\\_POWER\\_FORCE](#) 0x80
- #define [VL53L0X\\_SPEED\\_OF\\_LIGHT\\_IN\\_AIR](#) 2997
- #define [VL53L0X\\_REG\\_VHV\\_CONFIG\\_PAD\\_SCL\\_SDA\\_EXTSUP\\_HV](#) 0x0089
- #define [VL53L0X\\_REG\\_ALGO\\_PHASECAL\\_LIM](#) 0x0030 /\* 0x130 \*/
- #define [VL53L0X\\_REG\\_ALGO\\_PHASECAL\\_CONFIG\\_TIMEOUT](#) 0x0030

## Typedefs

- typedef [uint8\\_t](#) [VL53L0X\\_DeviceError](#)
- typedef [uint8\\_t](#) [VL53L0X\\_GpioFunctionality](#)

---

## vl53l0x\_doxydoc.c File Reference

---

## vl53l0x\_i2c\_platform.h File Reference

```
#include "vl53l0x_def.h"
#include <stdint.h>
#include <stdarg.h>
```

## Macros

- #define [I2C](#) 0x01
- #define [SPI](#) 0x00
- #define [COMMS\\_BUFFER\\_SIZE](#) 64
- #define [BYTES\\_PER\\_WORD](#) 2
- #define [BYTES\\_PER\\_DWORD](#) 4
- #define [VL53L0X\\_MAX\\_STRING\\_LENGTH\\_PLT](#) 256

## Typedefs

- typedef unsigned char [bool\\_t](#)  
*Typedef defining .*

## Functions

- [`int32\_t VL53L0X\_comms\_initialise`](#) ([`uint8\_t`](#) comms\_type, [`uint16\_t`](#) comms\_speed\_khz)  
*Initialise platform comms.*
- [`int VL53L0\_i2c\_init`](#) (char \*comPortStr, unsigned int baudRate)  
*Initialise platform serial comms.*
- [`int32\_t VL53L0X\_comms\_close`](#) (void)  
*Close platform comms.*
- [`int32\_t VL53L0X\_cycle\_power`](#) (void)  
*Cycle Power to Device.*
- [`int32\_t VL53L0X\_write\_multi`](#) ([`uint8\_t`](#) address, [`uint8\_t`](#) index, [`uint8\_t`](#) \*pdata, [`int32\_t`](#) count)  
*Writes the supplied byte buffer to the device.*
- [`int32\_t VL53L0X\_read\_multi`](#) ([`uint8\_t`](#) address, [`uint8\_t`](#) index, [`uint8\_t`](#) \*pdata, [`int32\_t`](#) count)  
*Reads the requested number of bytes from the device.*
- [`int32\_t VL53L0X\_write\_byte`](#) ([`uint8\_t`](#) address, [`uint8\_t`](#) index, [`uint8\_t`](#) data)  
*Writes a single byte to the device.*
- [`int32\_t VL53L0X\_write\_word`](#) ([`uint8\_t`](#) address, [`uint8\_t`](#) index, [`uint16\_t`](#) data)  
*Writes a single word (16-bit unsigned) to the device.*
- [`int32\_t VL53L0X\_write\_dword`](#) ([`uint8\_t`](#) address, [`uint8\_t`](#) index, [`uint32\_t`](#) data)  
*Writes a single dword (32-bit unsigned) to the device.*
- [`int32\_t VL53L0X\_read\_byte`](#) ([`uint8\_t`](#) address, [`uint8\_t`](#) index, [`uint8\_t`](#) \*pdata)  
*Reads a single byte from the device.*
- [`int32\_t VL53L0X\_read\_word`](#) ([`uint8\_t`](#) address, [`uint8\_t`](#) index, [`uint16\_t`](#) \*pdata)  
*Reads a single word (16-bit unsigned) from the device.*
- [`int32\_t VL53L0X\_read\_dword`](#) ([`uint8\_t`](#) address, [`uint8\_t`](#) index, [`uint32\_t`](#) \*pdata)  
*Reads a single dword (32-bit unsigned) from the device.*
- [`int32\_t VL53L0X\_platform\_wait\_us`](#) ([`int32\_t`](#) wait\_us)  
*Implements a programmable wait in us.*
- [`int32\_t VL53L0X\_wait\_ms`](#) ([`int32\_t`](#) wait\_ms)  
*Implements a programmable wait in ms.*
- [`int32\_t VL53L0X\_set\_gpio`](#) ([`uint8\_t`](#) level)  
*Set GPIO value.*
- [`int32\_t VL53L0X\_get\_gpio`](#) ([`uint8\_t`](#) \*plevel)  
*Get GPIO value.*
- [`int32\_t VL53L0X\_release\_gpio`](#) (void)  
*Release force on GPIO.*
- [`int32\_t VL53L0X\_get\_timer\_frequency`](#) ([`int32\_t`](#) \*ptimer\_freq\_hz)  
*Get the frequency of the timer used for ranging results time stamps.*
- [`int32\_t VL53L0X\_get\_timer\_value`](#) ([`int32\_t`](#) \*ptimer\_count)  
*Get the timer value in units of timer\_freq\_hz (see VL53L0X\_get\_timestamp\_frequency())*

---

## Macro Definition Documentation

### #define I2C 0x01

Definition at line 55 of file vl53l0x\_i2c\_platform.h.

### #define SPI 0x00



Definition at line 56 of file vl53l0x\_i2c\_platform.h.

**#define COMMS\_BUFFER\_SIZE 64**

Definition at line 58 of file vl53l0x\_i2c\_platform.h.

**#define BYTES\_PER\_WORD 2**

Definition at line 60 of file vl53l0x\_i2c\_platform.h.

**#define BYTES\_PER\_DWORD 4**

Definition at line 61 of file vl53l0x\_i2c\_platform.h.

**#define VL53L0X\_MAX\_STRING\_LENGTH\_PLT 256**

Definition at line 63 of file vl53l0x\_i2c\_platform.h.

## Typedef Documentation

**typedef unsigned char [bool\\_t](#)**

Typedef defining .

The developer should modify this to suit the platform being deployed. Typedef defining 8 bit unsigned char type.

The developer should modify this to suit the platform being deployed.

Definition at line 51 of file vl53l0x\_i2c\_platform.h.

## Function Documentation

**[int32\\_t](#) VL53L0X\_comms\_initialise ([uint8\\_t](#) *comms\_type*, [uint16\\_t](#) *comms\_speed\_khz*)**

Initialise platform comms.

### Parameters:

<i>comms_type</i>	- selects between I2C and SPI
<i>comms_speed_khz</i>	- unsigned short containing the I2C speed in kHz

### Returns:

status - status 0 = ok, 1 = error

**int VL53L0\_i2c\_init (char \* *comPortStr*, unsigned int *baudRate*)**

Initialise platform serial comms.

#### Parameters:

<i>comPortStr</i>	- String to indicate the comm port
<i>baudRate</i>	- Bau rate

#### Returns:

status - status 0 = ok, 1 = error

### [int32\\_t VL53L0X\\_comms\\_close \(void \)](#)

Close platform comms.

#### Returns:

status - status 0 = ok, 1 = error

### [int32\\_t VL53L0X\\_cycle\\_power \(void \)](#)

Cycle Power to Device.

#### Returns:

status - status 0 = ok, 1 = error

### [int32\\_t VL53L0X\\_write\\_multi \(uint8\\_t address, uint8\\_t index, uint8\\_t \\* pdata, int32\\_t count\)](#)

Writes the supplied byte buffer to the device.

Wrapper for SystemVerilog Write Multi task

```
1 Example:
2
3 uint8_t *spad_enables;
4
5 int status = VL53L0X_write_multi(RET_SPAD_EN_0, spad_enables, 36);
```

#### Parameters:

<i>address</i>	- uint8_t device address value
<i>index</i>	- uint8_t register index value
<i>pdata</i>	- pointer to uint8_t buffer containing the data to be written
<i>count</i>	- number of bytes in the supplied byte buffer

#### Returns:

status - SystemVerilog status 0 = ok, 1 = error

### [int32\\_t VL53L0X\\_read\\_multi \(uint8\\_t address, uint8\\_t index, uint8\\_t \\* pdata, int32\\_t count\)](#)

Reads the requested number of bytes from the device.

Wrapper for SystemVerilog Read Multi task

```
1 Example:
2
3 uint8_t buffer[COMMS_BUFFER_SIZE];
4
5 int status = status = VL53L0X_read_multi(DEVICE_ID, buffer, 2)
```

#### Parameters:

<i>address</i>	- uint8_t device address value
<i>index</i>	- uint8_t register index value
<i>pdata</i>	- pointer to the uint8_t buffer to store read data
<i>count</i>	- number of uint8_t's to read

#### Returns:

status - SystemVerilog status 0 = ok, 1 = error

**[int32\\_t](#) VL53L0X\_write\_byte ([uint8\\_t](#) address, [uint8\\_t](#) index, [uint8\\_t](#) data)**

Writes a single byte to the device.

Wrapper for SystemVerilog Write Byte task

```
1 Example:
2
3 uint8_t page_number = MAIN_SELECT_PAGE;
4
5 int status = VL53L0X_write_byte(PAGE_SELECT, page_number);
```

#### Parameters:

<i>address</i>	- uint8_t device address value
<i>index</i>	- uint8_t register index value
<i>data</i>	- uint8_t data value to write

#### Returns:

status - SystemVerilog status 0 = ok, 1 = error

**[int32\\_t](#) VL53L0X\_write\_word ([uint8\\_t](#) address, [uint8\\_t](#) index, [uint16\\_t](#) data)**

Writes a single word (16-bit unsigned) to the device.

Manages the big-endian nature of the device (first byte written is the MS byte). Uses SystemVerilog Write Multi task.

```
1 Example:
2
3 uint16_t nvm_ctrl_pulse_width = 0x0004;
4
5 int status = VL53L0X_write_word(NVM_CTRL_PULSE_WIDTH_MSB, nvm_ctrl_pulse_width);
```

#### Parameters:

<i>address</i>	- uint8_t device address value
<i>index</i>	- uint8_t register index value
<i>data</i>	- uint16_t data value write

#### Returns:

status - SystemVerilog status 0 = ok, 1 = error

**[int32\\_t](#) VL53L0X\_write\_dword ([uint8\\_t](#) address, [uint8\\_t](#) index, [uint32\\_t](#) data)**

Writes a single dword (32-bit unsigned) to the device.

Manages the big-endian nature of the device (first byte written is the MS byte). Uses SystemVerilog Write Multi task.

```

1 Example:
2
3 uint32_t nvm_data = 0x0004;
4
5 int status = VL53L0X_write_dword(NVM_CTRL__DATAIN_MMM, nvm_data);

```

#### Parameters:

<i>address</i>	- uint8_t device address value
<i>index</i>	- uint8_t register index value
<i>data</i>	- uint32_t data value to write

#### Returns:

status - SystemVerilog status 0 = ok, 1 = error

**[int32\\_t](#) VL53L0X\_read\_byte ([uint8\\_t](#) address, [uint8\\_t](#) index, [uint8\\_t](#)\* pdata)**

Reads a single byte from the device.

Uses SystemVerilog Read Byte task.

```

1 Example:
2
3 uint8_t device_status = 0;
4
5 int status = VL53L0X_read_byte(STATUS, &device_status);

```

#### Parameters:

<i>address</i>	- uint8_t device address value
<i>index</i>	- uint8_t register index value
<i>pdata</i>	- pointer to uint8_t data value

#### Returns:

status - SystemVerilog status 0 = ok, 1 = error

**[int32\\_t](#) VL53L0X\_read\_word ([uint8\\_t](#) address, [uint8\\_t](#) index, [uint16\\_t](#)\* pdata)**

Reads a single word (16-bit unsigned) from the device.

Manages the big-endian nature of the device (first byte read is the MS byte). Uses SystemVerilog Read Multi task.

```

1 Example:
2
3 uint16_t timeout = 0;
4
5 int status = VL53L0X_read_word(TIMEOUT_OVERALL_PERIODS_MSB, &timeout);

```

#### Parameters:

<i>address</i>	- uint8_t device address value
<i>index</i>	- uint8_t register index value
<i>pdata</i>	- pointer to uint16_t data value

#### Returns:

status - SystemVerilog status 0 = ok, 1 = error

**[int32\\_t](#) VL53L0X\_read\_dword ([uint8\\_t](#) address, [uint8\\_t](#) index, [uint32\\_t](#)\* pdata)**

Reads a single dword (32-bit unsigned) from the device.

Manages the big-endian nature of the device (first byte read is the MS byte). Uses SystemVerilog Read Multi task.

```
1 Example:
2
3 uint32 t range 1 = 0;
4
5 int status = VL53L0X_read_dword(RANGE_1_MMM, &range_1);
```

#### Parameters:

<i>address</i>	- uint8_t device address value
<i>index</i>	- uint8_t register index value
<i>pdata</i>	- pointer to uint32_t data value

#### Returns:

status - SystemVerilog status 0 = ok, 1 = error

### [int32\\_t VL53L0X\\_platform\\_wait\\_us \(int32\\_t wait\\_us\)](#)

Implements a programmable wait in us.

Wrapper for SystemVerilog Wait in micro seconds task

#### Parameters:

<i>wait_us</i>	- integer wait in micro seconds
----------------	---------------------------------

#### Returns:

status - SystemVerilog status 0 = ok, 1 = error

### [int32\\_t VL53L0X\\_wait\\_ms \(int32\\_t wait\\_ms\)](#)

Implements a programmable wait in ms.

Wrapper for SystemVerilog Wait in milli seconds task

#### Parameters:

<i>wait_ms</i>	- integer wait in milli seconds
----------------	---------------------------------

#### Returns:

status - SystemVerilog status 0 = ok, 1 = error

### [int32\\_t VL53L0X\\_set\\_gpio \(uint8\\_t level\)](#)

Set GPIO value.

#### Parameters:

<i>level</i>	- input level - either 0 or 1
--------------	-------------------------------

#### Returns:

status - SystemVerilog status 0 = ok, 1 = error

### [int32\\_t VL53L0X\\_get\\_gpio \(uint8\\_t \\* plevel\)](#)

Get GPIO value.

#### Parameters:

<i>plevel</i>	- uint8_t pointer to store GPIO level (0 or 1)
---------------	--

#### Returns:

status - SystemVerilog status 0 = ok, 1 = error

### [int32\\_t](#) VL53L0X\_release\_gpio (void )

Release force on GPIO.

#### Returns:

status - SystemVerilog status 0 = ok, 1 = error

### [int32\\_t](#) VL53L0X\_get\_timer\_frequency ([int32\\_t](#) \* *ptimer\_freq\_hz*)

Get the frequency of the timer used for ranging results time stamps.

#### Parameters:

out	<i>ptimer_freq_hz</i>	: pointer for timer frequency
-----	-----------------------	-------------------------------

#### Returns:

status : 0 = ok, 1 = error

### [int32\\_t](#) VL53L0X\_get\_timer\_value ([int32\\_t](#) \* *ptimer\_count*)

Get the timer value in units of timer\_freq\_hz (see VL53L0X\_get\_timestamp\_frequency())

#### Parameters:

out	<i>ptimer_count</i>	: pointer for timer count value
-----	---------------------	---------------------------------

#### Returns:

status : 0 = ok, 1 = error

## vl53l0x\_interrupt\_threshold\_settings.h File Reference

### Variables

- [uint8\\_t](#) [InterruptThresholdSettings](#) []

### Variable Documentation

#### [uint8\\_t](#) [InterruptThresholdSettings](#) []

Definition at line 39 of file vl53l0x\_interrupt\_threshold\_settings.h.

## vl53l0x\_platform.h File Reference

Function prototype definitions for Ewok Platform layer.

```
#include "vl53l0x_def.h"
#include "vl53l0x_platform_log.h"
#include "vl53l0x_i2c_platform.h"
```

### Data Structures

- struct [VL53L0X\\_Dev\\_t](#)

**Generic PAL device type that does link between API and platform abstraction layer. Macros**

- #define [PALDevDataGet](#)(Dev, field) (Dev->Data.field)  
*Get ST private structure [VL53L0X\\_DevData\\_t](#) data access.*
- #define [PALDevDataSet](#)(Dev, field, data) (Dev->Data.field)=(data)  
*Set ST private structure [VL53L0X\\_DevData\\_t](#) data field.*

### Typedefs

- typedef [VL53L0X\\_Dev\\_t](#) \* [VL53L0X\\_DEV](#)  
*Declare the device Handle as a pointer of the structure [VL53L0X\\_Dev\\_t](#).*

### Functions

- [VL53L0X\\_Error VL53L0X\\_LockSequenceAccess](#) ([VL53L0X\\_DEV](#) Dev)  
*Lock comms interface to serialize all commands to a shared I2C interface for a specific device.*
- [VL53L0X\\_Error VL53L0X\\_UnlockSequenceAccess](#) ([VL53L0X\\_DEV](#) Dev)  
*Unlock comms interface to serialize all commands to a shared I2C interface for a specific device.*
- [VL53L0X\\_Error VL53L0X\\_WriteMulti](#) ([VL53L0X\\_DEV](#) Dev, [uint8\\_t](#) index, [uint8\\_t](#) \*pdata, [uint32\\_t](#) count)  
*Writes the supplied byte buffer to the device.*
- [VL53L0X\\_Error VL53L0X\\_ReadMulti](#) ([VL53L0X\\_DEV](#) Dev, [uint8\\_t](#) index, [uint8\\_t](#) \*pdata, [uint32\\_t](#) count)  
*Reads the requested number of bytes from the device.*
- [VL53L0X\\_Error VL53L0X\\_WrByte](#) ([VL53L0X\\_DEV](#) Dev, [uint8\\_t](#) index, [uint8\\_t](#) data)  
*Write single byte register.*
- [VL53L0X\\_Error VL53L0X\\_WrWord](#) ([VL53L0X\\_DEV](#) Dev, [uint8\\_t](#) index, [uint16\\_t](#) data)  
*Write word register.*
- [VL53L0X\\_Error VL53L0X\\_WrDWord](#) ([VL53L0X\\_DEV](#) Dev, [uint8\\_t](#) index, [uint32\\_t](#) data)  
*Write double word (4 byte) register.*
- [VL53L0X\\_Error VL53L0X\\_RdByte](#) ([VL53L0X\\_DEV](#) Dev, [uint8\\_t](#) index, [uint8\\_t](#) \*data)  
*Read single byte register.*
- [VL53L0X\\_Error VL53L0X\\_RdWord](#) ([VL53L0X\\_DEV](#) Dev, [uint8\\_t](#) index, [uint16\\_t](#) \*data)  
*Read word (2byte) register.*
- [VL53L0X\\_Error VL53L0X\\_RdDWord](#) ([VL53L0X\\_DEV](#) Dev, [uint8\\_t](#) index, [uint32\\_t](#) \*data)  
*Read dword (4byte) register.*
- [VL53L0X\\_Error VL53L0X\\_UpdateByte](#) ([VL53L0X\\_DEV](#) Dev, [uint8\\_t](#) index, [uint8\\_t](#) AndData, [uint8\\_t](#) OrData)  
*Threat safe Update (read/modify/write) single byte register.*
- [VL53L0X\\_Error VL53L0X\\_PollingDelay](#) ([VL53L0X\\_DEV](#) Dev)  
*execute delay in all polling API call*

## Detailed Description

Function prototype definitions for Ewok Platform layer.

---

## vl53l0x\_platform\_log.h File Reference

```
#include <stdio.h>
#include <string.h>
```

### Macros

- `#define VL53L0X\_ErrLog(...) (void)0`
- `#define LOG\_FUNCTION\_START(module, fmt, ...) (void)0`
- `#define LOG\_FUNCTION\_END(module, status, ...) (void)0`
- `#define LOG\_FUNCTION\_END\_FMT(module, status, fmt, ...) (void)0`
- `#define VL53L0X\_COPYSTRING(str, ...) strcpy(str, ##__VA_ARGS__)`

### Enumerations

- enum { [TRACE\\_LEVEL\\_NONE](#), [TRACE\\_LEVEL\\_ERRORS](#), [TRACE\\_LEVEL\\_WARNING](#), [TRACE\\_LEVEL\\_INFO](#), [TRACE\\_LEVEL\\_DEBUG](#), [TRACE\\_LEVEL\\_ALL](#), [TRACE\\_LEVEL\\_IGNORE](#) }
- enum { [TRACE\\_FUNCTION\\_NONE](#) = 0, [TRACE\\_FUNCTION\\_I2C](#) = 1, [TRACE\\_FUNCTION\\_ALL](#) = 0x7fffffff }
- enum { [TRACE\\_MODULE\\_NONE](#) = 0x0, [TRACE\\_MODULE\\_API](#) = 0x1, [TRACE\\_MODULE\\_PLATFORM](#) = 0x2, [TRACE\\_MODULE\\_ALL](#) = 0x7fffffff }

---

## Macro Definition Documentation

**`#define VL53L0X_ErrLog( ...) (void)0`**

Definition at line 103 of file vl53l0x\_platform\_log.h.

**`#define _LOG_FUNCTION_START( module, fmt, ...) (void)0`**

Definition at line 104 of file vl53l0x\_platform\_log.h.

**`#define _LOG_FUNCTION_END( module, status, ...) (void)0`**

Definition at line 105 of file vl53l0x\_platform\_log.h.

**`#define _LOG_FUNCTION_END_FMT( module, status, fmt, ...) (void)0`**

Definition at line 106 of file vl53l0x\_platform\_log.h.

**`#define VL53L0X_COPYSTRING( str, ...) strcpy(str, ##__VA_ARGS__)`**

Definition at line 109 of file vl53l0x\_platform\_log.h.

---



## Enumeration Type Documentation

### anonymous enum

#### Enumerator

*TRACE\_LEVEL\_NONE*  
*TRACE\_LEVEL\_ERRORS*  
*TRACE\_LEVEL\_WARNING*  
*TRACE\_LEVEL\_INFO*  
*TRACE\_LEVEL\_DEBUG*  
*TRACE\_LEVEL\_ALL*  
*TRACE\_LEVEL\_IGNORE*

Definition at line 49 of file vl53l0x\_platform\_log.h.

### anonymous enum

#### Enumerator

*TRACE\_FUNCTION\_NONE*  
*TRACE\_FUNCTION\_I2C*  
*TRACE\_FUNCTION\_ALL*

Definition at line 59 of file vl53l0x\_platform\_log.h.

### anonymous enum

#### Enumerator

*TRACE\_MODULE\_NONE*  
*TRACE\_MODULE\_API*  
*TRACE\_MODULE\_PLATFORM*  
*TRACE\_MODULE\_ALL*

Definition at line 65 of file vl53l0x\_platform\_log.h.

---

## vl53l0x\_tuning.h File Reference

#include "vl53l0x\_def.h"

### Variables

- [uint8\\_t DefaultTuningSettings](#) []

---

## Variable Documentation

### [uint8\\_t](#) DefaultTuningSettings[]

Definition at line 41 of file vl53l0x\_tuning.h.

---

## vl53l0x\_types.h File Reference

```
#include <stdint.h>
#include <stddef.h>
```

### Typedefs

- typedef [uint32\\_t](#) [FixPoint1616\\_t](#)  
*use where fractional values are expected*
  - typedef unsigned long long [uint64\\_t](#)
  - typedef unsigned int [uint32\\_t](#)  
*Typedef defining 32 bit unsigned int type.*
  - typedef int [int32\\_t](#)  
*Typedef defining 32 bit int type.*
  - typedef unsigned short [uint16\\_t](#)  
*Typedef defining 16 bit unsigned short type.*
  - typedef short [int16\\_t](#)  
*Typedef defining 16 bit short type.*
  - typedef unsigned char [uint8\\_t](#)  
*Typedef defining 8 bit unsigned char type.*
  - typedef signed char [int8\\_t](#)  
*Typedef defining 8 bit char type.*
- 

### Typedef Documentation

#### typedef unsigned long long [uint64\\_t](#)

Definition at line 69 of file vl53l0x\_types.h.

#### typedef unsigned int [uint32\\_t](#)

Typedef defining 32 bit unsigned int type.

The developer should modify this to suit the platform being deployed.

Definition at line 75 of file vl53l0x\_types.h.

#### typedef int [int32\\_t](#)

Typedef defining 32 bit int type.

The developer should modify this to suit the platform being deployed.

Definition at line 80 of file vl53l0x\_types.h.

#### typedef unsigned short [uint16\\_t](#)

Typedef defining 16 bit unsigned short type.

The developer should modify this to suit the platform being deployed.

Definition at line 85 of file vl53l0x\_types.h.

### **typedef short [int16\\_t](#)**

Typedef defining 16 bit short type.

The developer should modify this to suit the platform being deployed.

Definition at line 90 of file vl53l0x\_types.h.

### **typedef unsigned char [uint8\\_t](#)**

Typedef defining 8 bit unsigned char type.

The developer should modify this to suit the platform being deployed.

Definition at line 95 of file vl53l0x\_types.h.

### **typedef signed char [int8\\_t](#)**

Typedef defining 8 bit char type.

The developer should modify this to suit the platform being deployed.

Definition at line 100 of file vl53l0x\_types.h.

### **typedef [uint32\\_t](#) [FixPoint1616\\_t](#)**

use where fractional values are expected

Given a floating point value f it's .16 bit point is (int)(f\*(1<<16))

Definition at line 109 of file vl53l0x\_types.h.

---

## Index

<a href="#">_LOG_FUNCTION_END</a>	<a href="#">build</a>
vl53l0x_platform_log.h, 132	VL53L0X_Version_t, 100
<a href="#">_LOG_FUNCTION_END_FMT</a>	<a href="#">BYTES_PER_DWORD</a>
vl53l0x_platform_log.h, 132	vl53l0x_i2c_platform.h, 125
<a href="#">_LOG_FUNCTION_START</a>	<a href="#">BYTES_PER_WORD</a>
vl53l0x_platform_log.h, 132	vl53l0x_i2c_platform.h, 125
<a href="#">AmbientRateRtnMegaCps</a>	<a href="#">Check Enable list</a> , 74
VL53L0X_RangingMeasurementData_t, 97	<a href="#">VL53L0X_CHECKENABLE_NUMBER_OF_C</a>
<a href="#">AmbTuningWindowFactor_K</a>	<a href="#">HECKS</a> , 74
VL53L0X_DMaxData_t, 94	<a href="#">VL53L0X_CHECKENABLE_RANGE_IGNORE_THRESHOLD</a> , 74
<a href="#">Basic type definition</a> , 11	<a href="#">VL53L0X_CHECKENABLE_SIGMA_FINAL_RANGE</a> , 74
<a href="#">bool_t</a>	<a href="#">VL53L0X_CHECKENABLE_SIGNAL_RATE_FINAL_RANGE</a> , 74
vl53l0x_i2c_platform.h, 125	
<a href="#">BufferSize</a>	
VL53L0X_HistogramMeasurementData_t, 95	

VL53L0X\_CHECKENABLE\_SIGNAL\_RATE\_MSRC, 74  
 VL53L0X\_CHECKENABLE\_SIGNAL\_RATE\_PRE\_RANGE, 74  
 VL53L0X\_CHECKENABLE\_SIGNAL\_REF\_CLIP, 74  
 COMMS\_BUFFER\_SIZE  
 vl53l0x\_i2c\_platform.h, 125  
 comms\_speed\_khz  
 VL53L0X\_Dev\_t, 85  
 comms\_type  
 VL53L0X\_Dev\_t, 85  
 CurrentParameters  
 VL53L0X\_DevData\_t, 86  
 Data  
 VL53L0X\_Dev\_t, 84  
 DefaultTuningSettings  
 vl53l0x\_tuning.h, 133  
 Define Registers, 76  
 VL53L0X\_REG\_ALGO\_PART\_TO\_PART\_RANGE\_OFFSET\_MM, 80  
 VL53L0X\_REG\_ALGO\_PHASECAL\_CONFIG\_TIMEOUT, 84  
 VL53L0X\_REG\_ALGO\_PHASECAL\_LIM, 84  
 VL53L0X\_REG\_CROSSTALK\_COMPENSATION\_PEAK\_RATE\_MCPS, 82  
 VL53L0X\_REG\_DYNAMIC\_SPAD\_NUM\_REQUESTED\_REF\_SPAD, 83  
 VL53L0X\_REG\_DYNAMIC\_SPAD\_REF\_EN\_START\_OFFSET, 83  
 VL53L0X\_REG\_FINAL\_RANGE\_CONFIG\_MIN\_COUNT\_RATE\_RTN\_LIMIT, 81  
 VL53L0X\_REG\_FINAL\_RANGE\_CONFIG\_MIN\_SNR, 81  
 VL53L0X\_REG\_FINAL\_RANGE\_CONFIG\_TIMEOUT\_MACROP\_HI, 82  
 VL53L0X\_REG\_FINAL\_RANGE\_CONFIG\_TIMEOUT\_MACROP\_LO, 82  
 VL53L0X\_REG\_FINAL\_RANGE\_CONFIG\_VALID\_PHASE\_HIGH, 81  
 VL53L0X\_REG\_FINAL\_RANGE\_CONFIG\_VALID\_PHASE\_LOW, 81  
 VL53L0X\_REG\_FINAL\_RANGE\_CONFIG\_VCSEL\_PERIOD, 82  
 VL53L0X\_REG\_GLOBAL\_CONFIG\_REF\_EN\_START\_SELECT, 83  
 VL53L0X\_REG\_GLOBAL\_CONFIG\_SPAD\_ENABLED\_REF\_0, 83  
 VL53L0X\_REG\_GLOBAL\_CONFIG\_SPAD\_ENABLED\_REF\_1, 83  
 VL53L0X\_REG\_GLOBAL\_CONFIG\_SPAD\_ENABLED\_REF\_2, 83  
 VL53L0X\_REG\_GLOBAL\_CONFIG\_SPAD\_ENABLED\_REF\_3, 83  
 VL53L0X\_REG\_GLOBAL\_CONFIG\_SPAD\_ENABLED\_REF\_4, 83  
 VL53L0X\_REG\_GLOBAL\_CONFIG\_SPAD\_ENABLED\_REF\_5, 83  
 VL53L0X\_REG\_GLOBAL\_CONFIG\_VCSEL\_WIDTH, 83  
 VL53L0X\_REG\_GPIO\_HV\_MUX\_ACTIVE\_HIGH, 79  
 VL53L0X\_REG\_HISTOGRAM\_CONFIG\_INITIAL\_PHASE\_SELECT, 82  
 VL53L0X\_REG\_HISTOGRAM\_CONFIG\_READOUT\_CTRL, 82  
 VL53L0X\_REG\_I2C\_SLAVE\_DEVICE\_ADDRESS, 80  
 VL53L0X\_REG\_IDENTIFICATION\_MODEL\_ID, 82  
 VL53L0X\_REG\_IDENTIFICATION\_REVISION\_ID, 82  
 VL53L0X\_REG\_MSRC\_CONFIG\_CONTROL, 80  
 VL53L0X\_REG\_MSRC\_CONFIG\_TIMEOUT\_MACROP, 82  
 VL53L0X\_REG\_OSC\_CALIBRATE\_VAL, 82  
 VL53L0X\_REG\_POWER\_MANAGEMENT\_GO1\_POWER\_FORCE, 83  
 VL53L0X\_REG\_PRE\_RANGE\_CONFIG\_MIN\_SNR, 80  
 VL53L0X\_REG\_PRE\_RANGE\_CONFIG\_SIGMA\_THRESH\_HI, 81  
 VL53L0X\_REG\_PRE\_RANGE\_CONFIG\_SIGMA\_THRESH\_LO, 81  
 VL53L0X\_REG\_PRE\_RANGE\_CONFIG\_TIMEOUT\_MACROP\_HI, 81  
 VL53L0X\_REG\_PRE\_RANGE\_CONFIG\_TIMEOUT\_MACROP\_LO, 81  
 VL53L0X\_REG\_PRE\_RANGE\_CONFIG\_VALID\_PHASE\_HIGH, 81  
 VL53L0X\_REG\_PRE\_RANGE\_CONFIG\_VALID\_PHASE\_LOW, 81  
 VL53L0X\_REG\_PRE\_RANGE\_CONFIG\_VCSEL\_PERIOD, 81  
 VL53L0X\_REG\_PRE\_RANGE\_MIN\_COUNT\_RATE\_RTN\_LIMIT, 81  
 VL53L0X\_REG\_RESULT\_CORE\_AMBIENT\_WINDOW\_EVENTS\_REF, 80  
 VL53L0X\_REG\_RESULT\_CORE\_AMBIENT\_WINDOW\_EVENTS\_RTN, 80  
 VL53L0X\_REG\_RESULT\_CORE\_PAGE, 80  
 VL53L0X\_REG\_RESULT\_CORE\_RANGING\_TOTAL\_EVENTS\_REF, 80  
 VL53L0X\_REG\_RESULT\_CORE\_RANGING\_TOTAL\_EVENTS\_RTN, 80  
 VL53L0X\_REG\_RESULT\_INTERRUPT\_STATUS, 80  
 VL53L0X\_REG\_RESULT\_PEAK\_SIGNAL\_RATE\_REF, 80  
 VL53L0X\_REG\_RESULT\_RANGE\_STATUS, 80  
 VL53L0X\_REG\_SOFT\_RESET\_GO2\_SOFT\_RESET\_N, 82  
 VL53L0X\_REG\_SYSRANGE\_MODE\_BACKTOBACK, 78  
 VL53L0X\_REG\_SYSRANGE\_MODE\_HISTOGRAM, 78  
 VL53L0X\_REG\_SYSRANGE\_MODE\_MASK, 78

VL53L0X\_REG\_SYSRANGE\_MODE\_SINGLES  
ESHOT, 78

VL53L0X\_REG\_SYSRANGE\_MODE\_START  
\_STOP, 78

VL53L0X\_REG\_SYSRANGE\_MODE\_TIMED  
, 78

VL53L0X\_REG\_SYSRANGE\_START, 78

VL53L0X\_REG\_SYSTEM\_HISTOGRAM\_BI  
N, 82

VL53L0X\_REG\_SYSTEM\_INTERMEASURE  
MENT\_PERIOD, 79

VL53L0X\_REG\_SYSTEM\_INTERRUPT\_CLE  
AR, 79

VL53L0X\_REG\_SYSTEM\_INTERRUPT\_CON  
FIG\_GPIO, 79

VL53L0X\_REG\_SYSTEM\_INTERRUPT\_GPI  
O\_DISABLED, 79

VL53L0X\_REG\_SYSTEM\_INTERRUPT\_GPI  
O\_LEVEL\_HIGH, 79

VL53L0X\_REG\_SYSTEM\_INTERRUPT\_GPI  
O\_LEVEL\_LOW, 79

VL53L0X\_REG\_SYSTEM\_INTERRUPT\_GPI  
O\_NEW\_SAMPLE\_READY, 79

VL53L0X\_REG\_SYSTEM\_INTERRUPT\_GPI  
O\_OUT\_OF\_WINDOW, 79

VL53L0X\_REG\_SYSTEM\_RANGE\_CONFIG,  
79

VL53L0X\_REG\_SYSTEM\_SEQUENCE\_CON  
FIG, 79

VL53L0X\_REG\_SYSTEM\_THRESH\_HIGH,  
78

VL53L0X\_REG\_SYSTEM\_THRESH\_LOW,  
79

VL53L0X\_REG\_VHV\_CONFIG\_PAD\_SCL\_S  
DA\_EXTSUP\_HV, 84

VL53L0X\_SIGMA\_ESTIMATE\_MAX\_VALU  
E, 83

VL53L0X\_SPEED\_OF\_LIGHT\_IN\_AIR, 84

Defines Device modes, 61

VL53L0X\_DEVICEMODE\_CONTINUOUS\_R  
ANGING, 61

VL53L0X\_DEVICEMODE\_CONTINUOUS\_TI  
MED\_RANGING, 62

VL53L0X\_DEVICEMODE\_GPIO\_DRIVE, 62

VL53L0X\_DEVICEMODE\_GPIO\_OSC, 62

VL53L0X\_DEVICEMODE\_SINGLE\_ALS, 62

VL53L0X\_DEVICEMODE\_SINGLE\_HISTOG  
RAM, 62

VL53L0X\_DEVICEMODE\_SINGLE\_RANGIN  
G, 61

VL53L0X\_DeviceModes, 62

Defines Histogram modes, 62

VL53L0X\_HISTOGRAMMODE\_BOTH, 63

VL53L0X\_HISTOGRAMMODE\_DISABLED,  
63

VL53L0X\_HISTOGRAMMODE\_REFERENC  
E\_ONLY, 63

VL53L0X\_HISTOGRAMMODE\_RETURN\_O  
NLY, 63

VL53L0X\_HistogramModes, 63

Defines the current status of the device, 64

VL53L0X\_State, 65

VL53L0X\_STATE\_ERROR, 65

VL53L0X\_STATE\_IDLE, 65

VL53L0X\_STATE\_POWERDOWN, 65

VL53L0X\_STATE\_RUNNING, 65

VL53L0X\_STATE\_STANDBY, 65

VL53L0X\_STATE\_UNKNOWN, 65

VL53L0X\_STATE\_WAIT\_STATICINIT, 65

Defines the Polarity, 65, 67

VL53L0X\_InterruptPolarity, 66

VL53L0X\_INTERRUPTPOLARITY\_HIGH, 66

VL53L0X\_INTERRUPTPOLARITY\_LOW, 66

VL53L0X\_SEQUENCESTEP\_DSS, 68

VL53L0X\_SEQUENCESTEP\_FINAL\_RANGE  
, 68

VL53L0X\_SEQUENCESTEP\_MSRC, 68

VL53L0X\_SEQUENCESTEP\_NUMBER\_OF\_  
CHECKS, 68

VL53L0X\_SEQUENCESTEP\_PRE\_RANGE,  
68

VL53L0X\_SEQUENCESTEP\_TCC, 68

VL53L0X\_SequenceStepId, 68

Defines the steps, 67

Device Error, 71

VL53L0X\_DeviceError, 73

VL53L0X\_DEVICEERROR\_ALGOOVERFLO  
W, 73

VL53L0X\_DEVICEERROR\_ALGOUNDERFL  
OW, 73

VL53L0X\_DEVICEERROR\_MINCLIP, 73

VL53L0X\_DEVICEERROR\_MSRCNOTARGE  
T, 72

VL53L0X\_DEVICEERROR\_NONE, 72

VL53L0X\_DEVICEERROR\_NOHVVALUEF  
OUND, 72

VL53L0X\_DEVICEERROR\_PHASECONSIST  
ENCY, 73

VL53L0X\_DEVICEERROR\_RANGECOMPLE  
TE, 73

VL53L0X\_DEVICEERROR\_RANGEIGNORE  
THRESHOLD, 73

VL53L0X\_DEVICEERROR\_RANGEPHASEC  
HECK, 73

VL53L0X\_DEVICEERROR\_SIGMATHRESH  
OLDCHECK, 73

VL53L0X\_DEVICEERROR\_SNRCHECK, 73

VL53L0X\_DEVICEERROR\_TCC, 73

VL53L0X\_DEVICEERROR\_VCSELCONTIN  
UITYTESTFAILURE, 72

VL53L0X\_DEVICEERROR\_VCSELWATCHD  
OGTESTFAILURE, 72

DeviceMode

VL53L0X\_DeviceParameters\_t, 90

DeviceSpecificParameters

VL53L0X\_DevData\_t, 86

DmaxCalRangeMilliMeter

VL53L0X\_DevData\_t, 88

DmaxCalSignalRateRtnMegaCps

VL53L0X\_DevData\_t, 88

DMaxData  
     VL53L0X\_DevData\_t, 86  
 DssOn  
     VL53L0X\_SchedulerSequenceSteps\_t, 98  
 EffectiveSpadRtnCount  
     VL53L0X\_RangingMeasurementData\_t, 97  
 Error and Warning code returned by API, 58  
     VL53L0X\_Error, 61  
     VL53L0X\_ERROR\_BUFFER\_TOO\_SMALL, 60  
     VL53L0X\_ERROR\_CALIBRATION\_WARNING, 59  
     VL53L0X\_ERROR\_CONTROL\_INTERFACE, 60  
     VL53L0X\_ERROR\_DIVISION\_BY\_ZERO, 60  
     VL53L0X\_ERROR\_GPIO\_FUNCTIONALITY\_NOT\_SUPPORTED, 60  
     VL53L0X\_ERROR\_GPIO\_NOT\_EXISTING, 60  
     VL53L0X\_ERROR\_INTERRUPT\_NOT\_CLEARED, 60  
     VL53L0X\_ERROR\_INVALID\_COMMAND, 60  
     VL53L0X\_ERROR\_INVALID\_PARAMS, 59  
     VL53L0X\_ERROR\_MIN\_CLIPPED, 59  
     VL53L0X\_ERROR\_MODE\_NOT\_SUPPORTED, 60  
     VL53L0X\_ERROR\_NONE, 59  
     VL53L0X\_ERROR\_NOT\_IMPLEMENTED, 61  
     VL53L0X\_ERROR\_NOT\_SUPPORTED, 60  
     VL53L0X\_ERROR\_RANGE\_ERROR, 60  
     VL53L0X\_ERROR\_REF\_SPAD\_INIT, 61  
     VL53L0X\_ERROR\_TIME\_OUT, 60  
     VL53L0X\_ERROR\_UNDEFINED, 59  
 ErrorStatus  
     VL53L0X\_HistogramMeasurementData\_t, 95  
 FinalRangeOn  
     VL53L0X\_SchedulerSequenceSteps\_t, 98  
 FinalRangeTimeoutMicroSecs  
     VL53L0X\_DeviceSpecificParameters\_t, 92  
 FinalRangeVcselPulsePeriod  
     VL53L0X\_DeviceSpecificParameters\_t, 92  
 FirstBin  
     VL53L0X\_HistogramMeasurementData\_t, 95  
 FixPoint1616\_t  
     vl53l0x\_types.h, 135  
 General Macro Defines, 68  
     VL53L0X\_FIXPOINT08TOFIXPOINT1616, 71  
     VL53L0X\_FIXPOINT102TOFIXPOINT1616, 71  
     VL53L0X\_FIXPOINT1616TOFIXPOINT08, 70  
     VL53L0X\_FIXPOINT1616TOFIXPOINT102, 71  
     VL53L0X\_FIXPOINT1616TOFIXPOINT313, 70  
     VL53L0X\_FIXPOINT1616TOFIXPOINT412, 70  
     VL53L0X\_FIXPOINT1616TOFIXPOINT53, 71  
     VL53L0X\_FIXPOINT1616TOFIXPOINT88, 70  
     VL53L0X\_FIXPOINT1616TOFIXPOINT97, 70  
     VL53L0X\_FIXPOINT313TOFIXPOINT1616, 70  
     VL53L0X\_FIXPOINT412TOFIXPOINT1616, 70  
     VL53L0X\_FIXPOINT53TOFIXPOINT1616, 71  
     VL53L0X\_FIXPOINT88TOFIXPOINT1616, 70  
     VL53L0X\_FIXPOINT97TOFIXPOINT1616, 70  
     VL53L0X\_GETARRAYPARAMETERFIELD, 69  
     VL53L0X\_GETDEVICESTRUCTUREPARAMETER, 70  
     VL53L0X\_GETPARAMETERFIELD, 69  
     VL53L0X\_MAKEUINT16, 71  
     VL53L0X\_SETARRAYPARAMETERFIELD, 69  
     VL53L0X\_SETDEVICESTRUCTUREPARAMETER, 70  
     VL53L0X\_SETPARAMETERFIELD, 69  
 get\_sequence\_step\_timeout  
     vl53l0x\_api\_core.h, 109  
 Gpio Functionality, 75  
     VL53L0X\_GpioFunctionality, 76  
     VL53L0X\_GPIOFUNCTIONALITY\_NEW\_MEASURE\_READY, 76  
     VL53L0X\_GPIOFUNCTIONALITY\_OFF, 75  
     VL53L0X\_GPIOFUNCTIONALITY\_THRESHOLD\_CROSSED\_HIGH, 75  
     VL53L0X\_GPIOFUNCTIONALITY\_THRESHOLD\_CROSSED\_LOW, 75  
     VL53L0X\_GPIOFUNCTIONALITY\_THRESHOLD\_CROSSED\_OUT, 76  
 HistogramData  
     VL53L0X\_HistogramMeasurementData\_t, 95  
 HistogramMode  
     VL53L0X\_DeviceParameters\_t, 90  
 HistogramType  
     VL53L0X\_HistogramMeasurementData\_t, 95  
 I2C  
     vl53l0x\_i2c\_platform.h, 124  
 I2cDevAddr  
     VL53L0X\_Dev\_t, 84  
 int16\_t  
     vl53l0x\_types.h, 135  
 int32\_t  
     vl53l0x\_types.h, 134  
 int8\_t  
     vl53l0x\_types.h, 135  
 InterMeasurementPeriodMilliseconds  
     VL53L0X\_DeviceParameters\_t, 90  
 InterruptThresholdSettings  
     vl53l0x\_interrupt\_threshold\_settings.h, 130  
 LastEncodedTimeout  
     VL53L0X\_DeviceSpecificParameters\_t, 92  
 LastHistogramMeasure  
     VL53L0X\_DevData\_t, 86  
 LastRangeMeasure  
     VL53L0X\_DevData\_t, 86  
 LastSignalRefMcps  
     VL53L0X\_DevData\_t, 87  
 LimitChecksEnable

- VL53L0X\_DeviceParameters\_t, 90
- LimitChecksStatus
  - VL53L0X\_DeviceParameters\_t, 90
- LimitChecksValue
  - VL53L0X\_DeviceParameters\_t, 91
- LinearityCorrectiveGain
  - VL53L0X\_DevData\_t, 88
- List of available Power Modes, 63
  - VL53L0X\_POWERMODE\_IDLE\_LEVEL1, 64
  - VL53L0X\_POWERMODE\_IDLE\_LEVEL2, 64
  - VL53L0X\_POWERMODE\_STANDBY\_LEVE
    - L1, 64
  - VL53L0X\_POWERMODE\_STANDBY\_LEVE
    - L2, 64
- VL53L0X\_PowerModes, 64
- major
  - VL53L0X\_Version\_t, 99
- MeasurementTimeUsec
  - VL53L0X\_RangingMeasurementData\_t, 96
- MeasurementTimingBudgetMicroSeconds
  - VL53L0X\_DeviceParameters\_t, 90
- minor
  - VL53L0X\_Version\_t, 100
- ModuleId
  - VL53L0X\_DeviceSpecificParameters\_t, 93
- MsrcOn
  - VL53L0X\_SchedulerSequenceSteps\_t, 98
- Name
  - VL53L0X\_DeviceInfo\_t, 88
- NumberOfBins
  - VL53L0X\_HistogramMeasurementData\_t, 95
- OscFrequencyMHz
  - VL53L0X\_DeviceSpecificParameters\_t, 92
- PAL Register Access Functions, 8
  - VL53L0X\_LockSequenceAccess, 8
  - VL53L0X\_RdByte, 10
  - VL53L0X\_RdDWord, 11
  - VL53L0X\_RdWord, 10
  - VL53L0X\_ReadMulti, 9
  - VL53L0X\_UnlockSequenceAccess, 9
  - VL53L0X\_UpdateByte, 11
  - VL53L0X\_WrByte, 9
  - VL53L0X\_WrDWord, 10
  - VL53L0X\_WriteMulti, 9
  - VL53L0X\_WrWord, 10
- PAL\_disclaimer.c, 100
- PALDevDataGet
  - VL53L0 Platform Functions, 7
- PALDevDataSet
  - VL53L0 Platform Functions, 7
- PalState
  - VL53L0X\_DevData\_t, 87
- Part2PartOffsetAdjustmentNVMMMicroMeter
  - VL53L0X\_DevData\_t, 86
- Part2PartOffsetNVMMMicroMeter
  - VL53L0X\_DevData\_t, 86
- PartUIDLower
  - VL53L0X\_DeviceSpecificParameters\_t, 93
- PartUIDUpper
  - VL53L0X\_DeviceSpecificParameters\_t, 93
- Pin0GpioFunctionality
  - VL53L0X\_DeviceSpecificParameters\_t, 92
- PowerMode
  - VL53L0X\_DevData\_t, 87
- PreRangeOn
  - VL53L0X\_SchedulerSequenceSteps\_t, 98
- PreRangeTimeoutMicroSecs
  - VL53L0X\_DeviceSpecificParameters\_t, 92
- PreRangeVcseI PulsePeriod
  - VL53L0X\_DeviceSpecificParameters\_t, 92
- ProductId
  - VL53L0X\_DeviceInfo\_t, 89
  - VL53L0X\_DeviceSpecificParameters\_t, 93
- ProductRevisionMajor
  - VL53L0X\_DeviceInfo\_t, 89
- ProductRevisionMinor
  - VL53L0X\_DeviceInfo\_t, 89
- ProductType
  - VL53L0X\_DeviceInfo\_t, 89
- pTuningSettingsPointer
  - VL53L0X\_DevData\_t, 87
- RangeDMaxMilliMeter
  - VL53L0X\_RangingMeasurementData\_t, 97
- RangeFractionalEnable
  - VL53L0X\_DevData\_t, 86
- RangeFractionalPart
  - VL53L0X\_RangingMeasurementData\_t, 97
- RangeMilliMeter
  - VL53L0X\_RangingMeasurementData\_t, 96
- RangeOffsetMicroMeters
  - VL53L0X\_DeviceParameters\_t, 90
- RangeStatus
  - VL53L0X\_RangingMeasurementData\_t, 97
- ReadDataFromDeviceDone
  - VL53L0X\_DeviceSpecificParameters\_t, 92
- ReferenceSpadCount
  - VL53L0X\_DeviceSpecificParameters\_t, 93
- ReferenceSpadType
  - VL53L0X\_DeviceSpecificParameters\_t, 93
- RefGoodSpadMap
  - VL53L0X\_SpadData\_t, 99
- RefSpadEnables
  - VL53L0X\_SpadData\_t, 99
- RefSpadsInitialised
  - VL53L0X\_DeviceSpecificParameters\_t, 93
- RetSignalAt0mm
  - VL53L0X\_DMaxData\_t, 94
- revision
  - VL53L0X\_Version\_t, 99
- Revision
  - VL53L0X\_DeviceSpecificParameters\_t, 93
- SequenceConfig
  - VL53L0X\_DevData\_t, 86
- set\_sequence\_step\_timeout
  - vl53l0x\_api\_core.h, 109
- SigmaEstEffAmbWidth
  - VL53L0X\_DevData\_t, 87
- SigmaEstEffPulseWidth
  - VL53L0X\_DevData\_t, 87



VL53L0X\_DeviceSpecificParameters\_t, 92

SigmaEstimate

VL53L0X\_DevData\_t, 87

SigmaEstRefArray

VL53L0X\_DevData\_t, 87

VL53L0X\_DeviceSpecificParameters\_t, 92

SignalEstimate

VL53L0X\_DevData\_t, 87

SignalRateMeasFixed400mm

VL53L0X\_DeviceSpecificParameters\_t, 93

SignalRateRtnMegaCps

VL53L0X\_RangingMeasurementData\_t, 97

SpadData

VL53L0X\_DevData\_t, 86

SPI

vl53l0x\_i2c\_platform.h, 124

StopVariable

VL53L0X\_DevData\_t, 87

targetRefRate

VL53L0X\_DevData\_t, 87

TccOn

VL53L0X\_SchedulerSequenceSteps\_t, 98

TimeStamp

VL53L0X\_RangingMeasurementData\_t, 96

TRACE\_FUNCTION\_ALL

vl53l0x\_platform\_log.h, 133

TRACE\_FUNCTION\_I2C

vl53l0x\_platform\_log.h, 133

TRACE\_FUNCTION\_NONE

vl53l0x\_platform\_log.h, 133

TRACE\_LEVEL\_ALL

vl53l0x\_platform\_log.h, 133

TRACE\_LEVEL\_DEBUG

vl53l0x\_platform\_log.h, 133

TRACE\_LEVEL\_ERRORS

vl53l0x\_platform\_log.h, 133

TRACE\_LEVEL\_IGNORE

vl53l0x\_platform\_log.h, 133

TRACE\_LEVEL\_INFO

vl53l0x\_platform\_log.h, 133

TRACE\_LEVEL\_NONE

vl53l0x\_platform\_log.h, 133

TRACE\_LEVEL\_WARNING

vl53l0x\_platform\_log.h, 133

TRACE\_MODULE\_ALL

vl53l0x\_platform\_log.h, 133

TRACE\_MODULE\_API

vl53l0x\_platform\_log.h, 133

TRACE\_MODULE\_NONE

vl53l0x\_platform\_log.h, 133

TRACE\_MODULE\_PLATFORM

vl53l0x\_platform\_log.h, 133

Type

VL53L0X\_DeviceInfo\_t, 88

uint16\_t

vl53l0x\_types.h, 134

uint32\_t

vl53l0x\_types.h, 134

uint64\_t

vl53l0x\_types.h, 134

uint8\_t

vl53l0x\_types.h, 135

UseInternalTuningSettings

VL53L0X\_DevData\_t, 87

Vcsel Period Defines, 66

VL53L0X\_VCSEL\_PERIOD\_FINAL\_RANGE, 67

VL53L0X\_VCSEL\_PERIOD\_PRE\_RANGE, 67

VL53L0X\_VcselPeriod, 67

VL53L0 Platform Functions, 6

PALDevDataGet, 7

PALDevDataSet, 7

VL53L0X\_DEV, 7

VL53L0X\_PollingDelay, 7

VL53L0\_i2c\_init

vl53l0x\_i2c\_platform.h, 125

VL53L0X cut1.1 Device Specific Defines, 71

VL53L0X cut1.1 Function Definition, 11

VL53L0X Defines, 55

VL53L0X\_DEFAULT\_MAX\_LOOP, 58

VL53L0X\_HISTOGRAM\_BUFFER\_SIZE, 58

VL53L0X\_IMPLEMENTATION\_VER\_MAJOR, 58

VL53L0X\_IMPLEMENTATION\_VER\_MINOR, 58

VL53L0X\_IMPLEMENTATION\_VER\_REVISION, 58

VL53L0X\_IMPLEMENTATION\_VER\_SUB, 58

VL53L0X\_MAX\_STRING\_LENGTH, 58

VL53L0X\_REF\_SPAD\_BUFFER\_SIZE, 58

VL53L0X\_SPECIFICATION\_VER\_MAJOR, 57

VL53L0X\_SPECIFICATION\_VER\_MINOR, 57

VL53L0X\_SPECIFICATION\_VER\_REVISION, 58

VL53L0X\_SPECIFICATION\_VER\_SUB, 57

VL53L0X10\_IMPLEMENTATION\_VER\_MAJOR, 57

VL53L0X10\_IMPLEMENTATION\_VER\_MINOR, 57

VL53L0X10\_IMPLEMENTATION\_VER\_REVISION, 57

VL53L0X10\_IMPLEMENTATION\_VER\_SUB, 57

VL53L0X10\_SPECIFICATION\_VER\_MAJOR, 56

VL53L0X10\_SPECIFICATION\_VER\_MINOR, 56

VL53L0X10\_SPECIFICATION\_VER\_REVISION, 57

VL53L0X10\_SPECIFICATION\_VER\_SUB, 57

VL53L0X General Functions, 12

VL53L0X\_GetDeviceErrorStatus, 14

VL53L0X\_GetDeviceErrorString, 15

VL53L0X\_GetDeviceInfo, 14

VL53L0X\_GetLinearityCorrectiveGain, 18

VL53L0X\_GetOffsetCalibrationDataMicroMeter, 17



- VL53L0X\_GetPalErrorString, 15
- VL53L0X\_GetPalSpecVersion, 13
- VL53L0X\_GetPalState, 16
- VL53L0X\_GetPalStateString, 16
- VL53L0X\_GetPowerMode, 17
- VL53L0X\_GetProductRevision, 14
- VL53L0X\_GetRangeStatusString, 15
- VL53L0X\_GetTotalSignalRate, 19
- VL53L0X\_GetUpperLimitMilliMeter, 19
- VL53L0X\_GetVersion, 13
- VL53L0X\_SetGroupParamHold, 18
- VL53L0X\_SetLinearityCorrectiveGain, 18
- VL53L0X\_SetOffsetCalibrationDataMicroMeter, 17
- VL53L0X\_SetPowerMode, 16
- VL53L0X Init Functions, 19
  - VL53L0X\_DataInit, 20
  - VL53L0X\_GetTuningSettingBuffer, 21
  - VL53L0X\_ResetDevice, 22
  - VL53L0X\_SetDeviceAddress, 20
  - VL53L0X\_SetTuningSettingBuffer, 21
  - VL53L0X\_StaticInit, 22
  - VL53L0X\_WaitDeviceBooted, 22
- VL53L0X Interrupt Functions, 47
  - VL53L0X\_ClearInterruptMask, 50
  - VL53L0X\_EnableInterruptMask, 51
  - VL53L0X\_GetGpioConfig, 49
  - VL53L0X\_GetInterruptMaskStatus, 51
  - VL53L0X\_GetInterruptThresholds, 50
  - VL53L0X\_GetStopCompletedStatus, 50
  - VL53L0X\_SetGpioConfig, 48
  - VL53L0X\_SetInterruptThresholds, 49
- VL53L0X Measurement Functions, 40
  - VL53L0X\_GetHistogramMeasurementData, 45
  - VL53L0X\_GetMaxNumberOfROI Zones, 47
  - VL53L0X\_GetMeasurementDataReady, 44
  - VL53L0X\_GetMeasurementRefSignal, 44
  - VL53L0X\_GetNumberOfROI Zones, 47
  - VL53L0X\_GetRangingMeasurementData, 45
  - VL53L0X\_PerformOffsetCalibration, 43
  - VL53L0X\_PerformRefCalibration, 41
  - VL53L0X\_PerformSingleHistogramMeasurement, 46
  - VL53L0X\_PerformSingleMeasurement, 41
  - VL53L0X\_PerformSingleRangingMeasurement, 46
  - VL53L0X\_PerformXTalkCalibration, 42
  - VL53L0X\_PerformXTalkMeasurement, 42
  - VL53L0X\_SetNumberOfROI Zones, 46
  - VL53L0X\_StartMeasurement, 43
  - VL53L0X\_StopMeasurement, 43
  - VL53L0X\_WaitDeviceReadyForNewMeasurement, 44
- VL53L0X Parameters Functions, 22
  - VL53L0X\_GetDeviceMode, 26
  - VL53L0X\_GetDeviceParameters, 25
  - VL53L0X\_GetDmaxCalParameters, 39
  - VL53L0X\_GetFractionEnable, 27
  - VL53L0X\_GetHistogramMode, 28
  - VL53L0X\_GetInterMeasurementPeriodMilliSeconds, 33
  - VL53L0X\_GetLimitCheckCurrent, 38
  - VL53L0X\_GetLimitCheckEnable, 37
  - VL53L0X\_GetLimitCheckInfo, 35
  - VL53L0X\_GetLimitCheckStatus, 36
  - VL53L0X\_GetLimitCheckValue, 37
  - VL53L0X\_GetMeasurementTimingBudgetMicroSeconds, 29
  - VL53L0X\_GetNumberOfLimitCheck, 35
  - VL53L0X\_GetNumberOfSequenceSteps, 32
  - VL53L0X\_GetRefCalibration, 35
  - VL53L0X\_GetSequenceStepEnable, 30
  - VL53L0X\_GetSequenceStepEnables, 30
  - VL53L0X\_GetSequenceStepsInfo, 32
  - VL53L0X\_GetSequenceStepTimeout, 31
  - VL53L0X\_GetVcselPulsePeriod, 29
  - VL53L0X\_GetWrapAroundCheckEnable, 39
  - VL53L0X\_GetXTalkCompensationEnable, 33
  - VL53L0X\_GetXTalkCompensationRateMegaCps, 34
  - VL53L0X\_SetDeviceMode, 26
  - VL53L0X\_SetDeviceParameters, 25
  - VL53L0X\_SetDmaxCalParameters, 39
  - VL53L0X\_SetHistogramMode, 27
  - VL53L0X\_SetInterMeasurementPeriodMilliSeconds, 32
  - VL53L0X\_SetLimitCheckEnable, 36
  - VL53L0X\_SetLimitCheckValue, 37
  - VL53L0X\_SetMeasurementTimingBudgetMicroSeconds, 28
  - VL53L0X\_SetRangeFractionEnable, 27
  - VL53L0X\_SetRefCalibration, 34
  - VL53L0X\_SetSequenceStepEnable, 30
  - VL53L0X\_SetSequenceStepTimeout, 31
  - VL53L0X\_SetVcselPulsePeriod, 29
  - VL53L0X\_SetWrapAroundCheckEnable, 38
  - VL53L0X\_SetXTalkCompensationEnable, 33
  - VL53L0X\_SetXTalkCompensationRateMegaCps, 34
- VL53L0X SPAD Functions, 51
  - VL53L0X\_GetReferenceSpads, 54
  - VL53L0X\_GetSpadAmbientDamperFactor, 53
  - VL53L0X\_GetSpadAmbientDamperThreshold, 53
  - VL53L0X\_PerformRefSpadManagement, 54
  - VL53L0X\_SetReferenceSpads, 54
  - VL53L0X\_SetSpadAmbientDamperFactor, 53
  - VL53L0X\_SetSpadAmbientDamperThreshold, 52
- VL53L0X\_API
  - vl53l0x\_api.h, 106
  - vl53l0x\_api.h, 100
  - VL53L0X\_API, 106
  - vl53l0x\_api\_calibration.h, 106
  - VL53L0X\_apply\_offset\_adjustment, 107
  - VL53L0X\_get\_offset\_calibration\_data\_micro\_meter, 107
  - VL53L0X\_get\_ref\_calibration, 107
  - VL53L0X\_get\_reference\_spads, 107

VL53L0X\_perform\_offset\_calibration, 107  
 VL53L0X\_perform\_phase\_calibration, 107  
 VL53L0X\_perform\_ref\_calibration, 107  
 VL53L0X\_perform\_ref\_spad\_management, 107  
 VL53L0X\_perform\_xtalk\_calibration, 107  
 VL53L0X\_set\_offset\_calibration\_data\_micro\_meter, 107  
 VL53L0X\_set\_ref\_calibration, 107  
 VL53L0X\_set\_reference\_spads, 107  
 vl53l0x\_api\_core.h, 107  
 get\_sequence\_step\_timeout, 109  
 set\_sequence\_step\_timeout, 109  
 VL53L0X\_calc\_sigma\_estimate, 109  
 VL53L0X\_calc\_timeout\_mclks, 110  
 VL53L0X\_decode\_timeout, 109  
 VL53L0X\_decode\_vcsel\_period, 109  
 VL53L0X\_encode\_timeout, 110  
 VL53L0X\_encode\_vcsel\_period, 109  
 VL53L0X\_get\_info\_from\_device, 109  
 VL53L0X\_get\_measurement\_timing\_budget\_micro\_seconds, 109  
 VL53L0X\_get\_pal\_range\_status, 109  
 VL53L0X\_get\_total\_signal\_rate, 109  
 VL53L0X\_get\_total\_xtalk\_rate, 109  
 VL53L0X\_get\_vcsel\_pulse\_period, 109  
 VL53L0X\_isqrt, 109  
 VL53L0X\_load\_tuning\_settings, 109  
 VL53L0X\_measurement\_poll\_for\_completion, 109  
 VL53L0X\_quadrature\_sum, 109  
 VL53L0X\_reverse\_bytes, 109  
 VL53L0X\_set\_measurement\_timing\_budget\_micro\_seconds, 109  
 VL53L0X\_set\_vcsel\_pulse\_period, 109  
 vl53l0x\_api\_ranging.h, 110  
 vl53l0x\_api\_strings.h, 110  
 VL53L0X\_get\_device\_error\_string, 118  
 VL53L0X\_get\_device\_info, 118  
 VL53L0X\_get\_limit\_check\_info, 118  
 VL53L0X\_get\_pal\_error\_string, 118  
 VL53L0X\_get\_pal\_state\_string, 118  
 VL53L0X\_get\_range\_status\_string, 118  
 VL53L0X\_get\_sequence\_steps\_info, 118  
 VL53L0X\_STRING\_CHECKENABLE\_RANGE\_IGNORE\_THRESHOLD, 117  
 VL53L0X\_STRING\_CHECKENABLE\_SIGMA\_FINAL\_RANGE, 116  
 VL53L0X\_STRING\_CHECKENABLE\_SIGNAL\_RATE\_FINAL\_RANGE, 116  
 VL53L0X\_STRING\_CHECKENABLE\_SIGNAL\_RATE\_MSRC, 117  
 VL53L0X\_STRING\_CHECKENABLE\_SIGNAL\_RATE\_PRE\_RANGE, 117  
 VL53L0X\_STRING\_CHECKENABLE\_SIGNAL\_REF\_CLIP, 117  
 VL53L0X\_STRING\_DEVICE\_INFO\_NAME, 112  
 VL53L0X\_STRING\_DEVICE\_INFO\_NAME\_ES1, 112  
 VL53L0X\_STRING\_DEVICE\_INFO\_NAME\_TS0, 112  
 VL53L0X\_STRING\_DEVICE\_INFO\_NAME\_TS1, 112  
 VL53L0X\_STRING\_DEVICE\_INFO\_NAME\_TS2, 112  
 VL53L0X\_STRING\_DEVICE\_INFO\_TYPE, 112  
 VL53L0X\_STRING\_DEVICEERROR\_ALGO\_OVERFLOW, 116  
 VL53L0X\_STRING\_DEVICEERROR\_ALGO\_UNDERFLOW, 116  
 VL53L0X\_STRING\_DEVICEERROR\_MINCLIP, 116  
 VL53L0X\_STRING\_DEVICEERROR\_MSRC\_NOTARGET, 115  
 VL53L0X\_STRING\_DEVICEERROR\_NONE, 115  
 VL53L0X\_STRING\_DEVICEERROR\_NOHV\_VALUEFOUND, 115  
 VL53L0X\_STRING\_DEVICEERROR\_PHASECONSISTENCY, 116  
 VL53L0X\_STRING\_DEVICEERROR\_RANGECOMPLETE, 116  
 VL53L0X\_STRING\_DEVICEERROR\_RANGEIGNORETHRESHOLD, 116  
 VL53L0X\_STRING\_DEVICEERROR\_RANGEPHASECHECK, 115  
 VL53L0X\_STRING\_DEVICEERROR\_SIGMA\_THRESHOLDCHECK, 116  
 VL53L0X\_STRING\_DEVICEERROR\_SNRCHECK, 115  
 VL53L0X\_STRING\_DEVICEERROR\_TCC, 116  
 VL53L0X\_STRING\_DEVICEERROR\_UNKNOWN, 116  
 VL53L0X\_STRING\_DEVICEERROR\_VCSELCONTINUITYTESTFAILURE, 115  
 VL53L0X\_STRING\_DEVICEERROR\_VCSELWATCHDOGTESTFAILURE, 115  
 VL53L0X\_STRING\_ERROR\_BUFFER\_TOO\_SMALL, 113  
 VL53L0X\_STRING\_ERROR\_CALIBRATION\_WARNING, 112  
 VL53L0X\_STRING\_ERROR\_CONTROL\_INTERFACE, 113  
 VL53L0X\_STRING\_ERROR\_DIVISION\_BY\_ZERO, 113  
 VL53L0X\_STRING\_ERROR\_GPIO\_FUNCTIONALITY\_NOT\_SUPPORTED, 113  
 VL53L0X\_STRING\_ERROR\_GPIO\_NOT\_EXISTING, 113  
 VL53L0X\_STRING\_ERROR\_INTERRUPT\_NOT\_CLEARED, 113  
 VL53L0X\_STRING\_ERROR\_INVALID\_COMMAND, 113  
 VL53L0X\_STRING\_ERROR\_INVALID\_PARAMS, 112  
 VL53L0X\_STRING\_ERROR\_MIN\_CLIPPED, 112

VL53L0X\_STRING\_ERROR\_MODE\_NOT\_SUPPORTED, 113

VL53L0X\_STRING\_ERROR\_NONE, 112

VL53L0X\_STRING\_ERROR\_NOT\_IMPLEMENTED, 114

VL53L0X\_STRING\_ERROR\_NOT\_SUPPORTED, 113

VL53L0X\_STRING\_ERROR\_RANGE\_ERROR, 113

VL53L0X\_STRING\_ERROR\_REF\_SPAD\_INIT, 113

VL53L0X\_STRING\_ERROR\_TIME\_OUT, 113

VL53L0X\_STRING\_ERROR\_UNDEFINED, 112

VL53L0X\_STRING\_RANGESTATUS\_HW, 114

VL53L0X\_STRING\_RANGESTATUS\_MINRANGE, 114

VL53L0X\_STRING\_RANGESTATUS\_NONE, 114

VL53L0X\_STRING\_RANGESTATUS\_PHASE, 114

VL53L0X\_STRING\_RANGESTATUS\_RANGE\_EVALID, 114

VL53L0X\_STRING\_RANGESTATUS\_SIGMA, 114

VL53L0X\_STRING\_RANGESTATUS\_SIGNAL, 114

VL53L0X\_STRING\_SEQUENCESTEP\_DSS, 117

VL53L0X\_STRING\_SEQUENCESTEP\_FINAL\_RANGE, 117

VL53L0X\_STRING\_SEQUENCESTEP\_MSRC, 117

VL53L0X\_STRING\_SEQUENCESTEP\_PRRANGE, 117

VL53L0X\_STRING\_SEQUENCESTEP\_TCC, 117

VL53L0X\_STRING\_STATE\_ERROR, 115

VL53L0X\_STRING\_STATE\_IDLE, 115

VL53L0X\_STRING\_STATE\_POWERDOWN, 114

VL53L0X\_STRING\_STATE\_RUNNING, 115

VL53L0X\_STRING\_STATE\_STANDBY, 114

VL53L0X\_STRING\_STATE\_UNKNOWN, 115

VL53L0X\_STRING\_STATE\_WAIT\_STATICINIT, 114

VL53L0X\_STRING\_UNKNOW\_ERROR\_CODE, 114

VL53L0X\_apply\_offset\_adjustment  
vl53l0x\_api\_calibration.h, 107

VL53L0X\_calc\_sigma\_estimate  
vl53l0x\_api\_core.h, 109

VL53L0X\_calc\_timeout\_melks  
vl53l0x\_api\_core.h, 110

VL53L0X\_CHECKENABLE\_NUMBER\_OF\_CHECKS  
Check Enable list, 74

VL53L0X\_CHECKENABLE\_RANGE\_IGNORE\_THRESHOLD  
Check Enable list, 74

VL53L0X\_CHECKENABLE\_SIGMA\_FINAL\_RANGE  
Check Enable list, 74

VL53L0X\_CHECKENABLE\_SIGNAL\_RATE\_FINAL\_RANGE  
Check Enable list, 74

VL53L0X\_CHECKENABLE\_SIGNAL\_RATE\_MSRC  
Check Enable list, 74

VL53L0X\_CHECKENABLE\_SIGNAL\_RATE\_PRRANGE  
Check Enable list, 74

VL53L0X\_CHECKENABLE\_SIGNAL\_REF\_CLIP  
Check Enable list, 74

VL53L0X\_ClearInterruptMask  
VL53L0X Interrupt Functions, 50

VL53L0X\_comms\_close  
vl53l0x\_i2c\_platform.h, 126

VL53L0X\_comms\_initialise  
vl53l0x\_i2c\_platform.h, 125

VL53L0X\_COPYSTRING  
vl53l0x\_platform\_log.h, 132

VL53L0X\_cycle\_power  
vl53l0x\_i2c\_platform.h, 126

VL53L0X\_DataInit  
VL53L0X Init Functions, 20

VL53L0X\_decode\_timeout  
vl53l0x\_api\_core.h, 109

VL53L0X\_decode\_vcsel\_period  
vl53l0x\_api\_core.h, 109

vl53l0x\_def.h, 118

VL53L0X\_DEFAULT\_MAX\_LOOP  
VL53L0X Defines, 58

VL53L0X\_DEV  
VL53L0 Platform Functions, 7

VL53L0X\_Dev\_t, 84

comms\_speed\_khz, 85

comms\_type, 85

Data, 84

I2cDevAddr, 84

VL53L0X\_DevData\_t, 85

CurrentParameters, 86

DeviceSpecificParameters, 86

DmaxCalRangeMilliMeter, 88

DmaxCalSignalRateRtnMegaCps, 88

DMaxData, 86

LastHistogramMeasure, 86

LastRangeMeasure, 86

LastSignalRefMcps, 87

LinearityCorrectiveGain, 88

PalState, 87

Part2PartOffsetAdjustmentNVMMicroMeter, 86

Part2PartOffsetNVMMicroMeter, 86

PowerMode, 87

pTuningSettingsPointer, 87

RangeFractionalEnable, 86

SequenceConfig, 86

SigmaEstEffAmbWidth, 87

- SigmaEstEffPulseWidth, 87
- SigmaEstimate, 87
- SigmaEstRefArray, 87
- SignalEstimate, 87
- SpadData, 86
- StopVariable, 87
- targetRefRate, 87
- UseInternalTuningSettings, 87
- vl53l0x\_device.h, 121
- VL53L0X\_DeviceError
  - Device Error, 73
- VL53L0X\_DEVICEERROR\_ALGOOVERFLOW
  - Device Error, 73
- VL53L0X\_DEVICEERROR\_ALGOUNDERFLOW
  - Device Error, 73
- VL53L0X\_DEVICEERROR\_MINCLIP
  - Device Error, 73
- VL53L0X\_DEVICEERROR\_MSRCNOTARGET
  - Device Error, 72
- VL53L0X\_DEVICEERROR\_NONE
  - Device Error, 72
- VL53L0X\_DEVICEERROR\_NOVHVVALUEFO
  - UND
    - Device Error, 72
- VL53L0X\_DEVICEERROR\_PHASECONSISTE
  - NCY
    - Device Error, 73
- VL53L0X\_DEVICEERROR\_RANGECOMPLETE
  - Device Error, 73
- VL53L0X\_DEVICEERROR\_RANGEIGNORETH
  - RESHOLD
    - Device Error, 73
- VL53L0X\_DEVICEERROR\_RANGEPHASECH
  - ECK
    - Device Error, 73
- VL53L0X\_DEVICEERROR\_SIGMATHRESHOL
  - DCHECK
    - Device Error, 73
- VL53L0X\_DEVICEERROR\_SNRCHECK
  - Device Error, 73
- VL53L0X\_DEVICEERROR\_TCC
  - Device Error, 73
- VL53L0X\_DEVICEERROR\_VCSELCONTINU
  - ITYTESTFAILURE
    - Device Error, 72
- VL53L0X\_DEVICEERROR\_VCSELWATCHDO
  - GTESTFAILURE
    - Device Error, 72
- VL53L0X\_DeviceInfo\_t, 88
  - Name, 88
  - ProductId, 89
  - ProductRevisionMajor, 89
  - ProductRevisionMinor, 89
  - ProductType, 89
  - Type, 88
- VL53L0X\_DEVICEMODE\_CONTINUOUS\_RA
  - NGING
    - Defines Device modes, 61
- VL53L0X\_DEVICEMODE\_CONTINUOUS\_TIM
  - ED\_RANGING
    - Defines Device modes, 62
- VL53L0X\_DEVICEMODE\_GPIO\_DRIVE
  - Defines Device modes, 62
- VL53L0X\_DEVICEMODE\_GPIO\_OSC
  - Defines Device modes, 62
- VL53L0X\_DEVICEMODE\_SINGLE\_ALS
  - Defines Device modes, 62
- VL53L0X\_DEVICEMODE\_SINGLE\_HISTOGR
  - AM
    - Defines Device modes, 62
- VL53L0X\_DEVICEMODE\_SINGLE\_RANGING
  - Defines Device modes, 61
- VL53L0X\_DeviceModes
  - Defines Device modes, 62
- VL53L0X\_DeviceParameters\_t, 89
  - DeviceMode, 90
  - HistogramMode, 90
  - InterMeasurementPeriodMilliseconds, 90
  - LimitChecksEnable, 90
  - LimitChecksStatus, 90
  - LimitChecksValue, 91
  - MeasurementTimingBudgetMicroSeconds, 90
  - RangeOffsetMicroMeters, 90
  - WrapAroundCheckEnable, 91
  - XTalkCompensationEnable, 90
  - XTalkCompensationRangeMilliMeter, 90
  - XTalkCompensationRateMegaCps, 90
- VL53L0X\_DeviceSpecificParameters\_t, 91
  - FinalRangeTimeoutMicroSecs, 92
  - FinalRangeVcselPulsePeriod, 92
  - LastEncodedTimeout, 92
  - ModuleId, 93
  - OscFrequencyMHz, 92
  - PartUIDLower, 93
  - PartUIDUpper, 93
  - Pin0GpioFunctionality, 92
  - PreRangeTimeoutMicroSecs, 92
  - PreRangeVcselPulsePeriod, 92
  - ProductId, 93
  - ReadDataFromDeviceDone, 92
  - ReferenceSpadCount, 93
  - ReferenceSpadType, 93
  - RefSpadsInitialised, 93
  - Revision, 93
  - SigmaEstEffAmbWidth, 92
  - SigmaEstEffPulseWidth, 92
  - SigmaEstRefArray, 92
  - SignalRateMeasFixed400mm, 93
- VL53L0X\_DMaxData\_t, 93
  - AmbTuningWindowFactor\_K, 94
  - RetSignalAt0mm, 94
- vl53l0x\_doxydoc.c, 123
- VL53L0X\_EnableInterruptMask
  - VL53L0X Interrupt Functions, 51
- VL53L0X\_encode\_timeout
  - vl53l0x\_api\_core.h, 110
- VL53L0X\_encode\_vcsel\_period
  - vl53l0x\_api\_core.h, 109

VL53L0X\_ErrLog  
vl53l0x\_platform\_log.h, 132

VL53L0X\_Error  
Error and Warning code returned by API, 61

VL53L0X\_ERROR\_BUFFER\_TOO\_SMALL  
Error and Warning code returned by API, 60

VL53L0X\_ERROR\_CALIBRATION\_WARNING  
Error and Warning code returned by API, 59

VL53L0X\_ERROR\_CONTROL\_INTERFACE  
Error and Warning code returned by API, 60

VL53L0X\_ERROR\_DIVISION\_BY\_ZERO  
Error and Warning code returned by API, 60

VL53L0X\_ERROR\_GPIO\_FUNCTIONALITY\_NOT\_SUPPORTED  
Error and Warning code returned by API, 60

VL53L0X\_ERROR\_GPIO\_NOT\_EXISTING  
Error and Warning code returned by API, 60

VL53L0X\_ERROR\_INTERRUPT\_NOT\_CLEARED  
Error and Warning code returned by API, 60

VL53L0X\_ERROR\_INVALID\_COMMAND  
Error and Warning code returned by API, 60

VL53L0X\_ERROR\_INVALID\_PARAMS  
Error and Warning code returned by API, 59

VL53L0X\_ERROR\_MIN\_CLIPPED  
Error and Warning code returned by API, 59

VL53L0X\_ERROR\_MODE\_NOT\_SUPPORTED  
Error and Warning code returned by API, 60

VL53L0X\_ERROR\_NONE  
Error and Warning code returned by API, 59

VL53L0X\_ERROR\_NOT\_IMPLEMENTED  
Error and Warning code returned by API, 61

VL53L0X\_ERROR\_NOT\_SUPPORTED  
Error and Warning code returned by API, 60

VL53L0X\_ERROR\_RANGE\_ERROR  
Error and Warning code returned by API, 60

VL53L0X\_ERROR\_REF\_SPAD\_INIT  
Error and Warning code returned by API, 61

VL53L0X\_ERROR\_TIME\_OUT  
Error and Warning code returned by API, 60

VL53L0X\_ERROR\_UNDEFINED  
Error and Warning code returned by API, 59

VL53L0X\_FIXPOINT08TOFIXPOINT1616  
General Macro Defines, 71

VL53L0X\_FIXPOINT102TOFIXPOINT1616  
General Macro Defines, 71

VL53L0X\_FIXPOINT1616TOFIXPOINT08  
General Macro Defines, 70

VL53L0X\_FIXPOINT1616TOFIXPOINT102  
General Macro Defines, 71

VL53L0X\_FIXPOINT1616TOFIXPOINT313  
General Macro Defines, 70

VL53L0X\_FIXPOINT1616TOFIXPOINT412  
General Macro Defines, 70

VL53L0X\_FIXPOINT1616TOFIXPOINT53  
General Macro Defines, 71

VL53L0X\_FIXPOINT1616TOFIXPOINT88  
General Macro Defines, 70

VL53L0X\_FIXPOINT1616TOFIXPOINT97  
General Macro Defines, 70

VL53L0X\_FIXPOINT313TOFIXPOINT1616  
General Macro Defines, 70

VL53L0X\_FIXPOINT412TOFIXPOINT1616  
General Macro Defines, 70

VL53L0X\_FIXPOINT53TOFIXPOINT1616  
General Macro Defines, 71

VL53L0X\_FIXPOINT88TOFIXPOINT1616  
General Macro Defines, 70

VL53L0X\_FIXPOINT97TOFIXPOINT1616  
General Macro Defines, 70

VL53L0X\_get\_device\_error\_string  
vl53l0x\_api\_strings.h, 118

VL53L0X\_get\_device\_info  
vl53l0x\_api\_strings.h, 118

VL53L0X\_get\_gpio  
vl53l0x\_i2c\_platform.h, 129

VL53L0X\_get\_info\_from\_device  
vl53l0x\_api\_core.h, 109

VL53L0X\_get\_limit\_check\_info  
vl53l0x\_api\_strings.h, 118

VL53L0X\_get\_measurement\_timing\_budget\_micro\_seconds  
vl53l0x\_api\_core.h, 109

VL53L0X\_get\_offset\_calibration\_data\_micro\_meter  
vl53l0x\_api\_calibration.h, 107

VL53L0X\_get\_pal\_error\_string  
vl53l0x\_api\_strings.h, 118

VL53L0X\_get\_pal\_range\_status  
vl53l0x\_api\_core.h, 109

VL53L0X\_get\_pal\_state\_string  
vl53l0x\_api\_strings.h, 118

VL53L0X\_get\_range\_status\_string  
vl53l0x\_api\_strings.h, 118

VL53L0X\_get\_ref\_calibration  
vl53l0x\_api\_calibration.h, 107

VL53L0X\_get\_reference\_spads  
vl53l0x\_api\_calibration.h, 107

VL53L0X\_get\_sequence\_steps\_info  
vl53l0x\_api\_strings.h, 118

VL53L0X\_get\_timer\_frequency  
vl53l0x\_i2c\_platform.h, 130

VL53L0X\_get\_timer\_value  
vl53l0x\_i2c\_platform.h, 130

VL53L0X\_get\_total\_signal\_rate  
vl53l0x\_api\_core.h, 109

VL53L0X\_get\_total\_xtalk\_rate  
vl53l0x\_api\_core.h, 109

VL53L0X\_get\_vcsel\_pulse\_period  
vl53l0x\_api\_core.h, 109

VL53L0X\_GETARRAYPARAMETERFIELD  
General Macro Defines, 69

VL53L0X\_GetDeviceErrorStatus  
VL53L0X General Functions, 14

VL53L0X\_GetDeviceErrorString  
VL53L0X General Functions, 15

VL53L0X\_GetDeviceInfo  
VL53L0X General Functions, 14

VL53L0X\_GetDeviceMode  
VL53L0X Parameters Functions, 26



VL53L0X\_GetDeviceParameters  
 VL53L0X Parameters Functions, 25

VL53L0X\_GETDEVICESPECIFICPARAMETER  
 General Macro Defines, 70

VL53L0X\_GetDmaxCalParameters  
 VL53L0X Parameters Functions, 39

VL53L0X\_GetFractionEnable  
 VL53L0X Parameters Functions, 27

VL53L0X\_GetGpioConfig  
 VL53L0X Interrupt Functions, 49

VL53L0X\_GetHistogramMeasurementData  
 VL53L0X Measurement Functions, 45

VL53L0X\_GetHistogramMode  
 VL53L0X Parameters Functions, 28

VL53L0X\_GetInterMeasurementPeriodMilliseconds  
 VL53L0X Parameters Functions, 33

VL53L0X\_GetInterruptMaskStatus  
 VL53L0X Interrupt Functions, 51

VL53L0X\_GetInterruptThresholds  
 VL53L0X Interrupt Functions, 50

VL53L0X\_GetLimitCheckCurrent  
 VL53L0X Parameters Functions, 38

VL53L0X\_GetLimitCheckEnable  
 VL53L0X Parameters Functions, 37

VL53L0X\_GetLimitCheckInfo  
 VL53L0X Parameters Functions, 35

VL53L0X\_GetLimitCheckStatus  
 VL53L0X Parameters Functions, 36

VL53L0X\_GetLimitCheckValue  
 VL53L0X Parameters Functions, 37

VL53L0X\_GetLinearityCorrectiveGain  
 VL53L0X General Functions, 18

VL53L0X\_GetMaxNumberOfROI Zones  
 VL53L0X Measurement Functions, 47

VL53L0X\_GetMeasurementDataReady  
 VL53L0X Measurement Functions, 44

VL53L0X\_GetMeasurementRefSignal  
 VL53L0X Measurement Functions, 44

VL53L0X\_GetMeasurementTimingBudgetMicroSeconds  
 VL53L0X Parameters Functions, 29

VL53L0X\_GetNumberOfLimitCheck  
 VL53L0X Parameters Functions, 35

VL53L0X\_GetNumberOfROI Zones  
 VL53L0X Measurement Functions, 47

VL53L0X\_GetNumberOfSequenceSteps  
 VL53L0X Parameters Functions, 32

VL53L0X\_GetOffsetCalibrationDataMicroMeter  
 VL53L0X General Functions, 17

VL53L0X\_GetPalErrorString  
 VL53L0X General Functions, 15

VL53L0X\_GetPalSpecVersion  
 VL53L0X General Functions, 13

VL53L0X\_GetPalState  
 VL53L0X General Functions, 16

VL53L0X\_GetPalStateString  
 VL53L0X General Functions, 16

VL53L0X\_GETPARAMETERFIELD  
 General Macro Defines, 69

VL53L0X\_GetPowerMode  
 VL53L0X General Functions, 17

VL53L0X\_GetProductRevision  
 VL53L0X General Functions, 14

VL53L0X\_GetRangeStatusString  
 VL53L0X General Functions, 15

VL53L0X\_GetRangingMeasurementData  
 VL53L0X Measurement Functions, 45

VL53L0X\_GetRefCalibration  
 VL53L0X Parameters Functions, 35

VL53L0X\_GetReferenceSpads  
 VL53L0X SPAD Functions, 54

VL53L0X\_GetSequenceStepEnable  
 VL53L0X Parameters Functions, 30

VL53L0X\_GetSequenceStepEnables  
 VL53L0X Parameters Functions, 30

VL53L0X\_GetSequenceStepsInfo  
 VL53L0X Parameters Functions, 32

VL53L0X\_GetSequenceStepTimeout  
 VL53L0X Parameters Functions, 31

VL53L0X\_GetSpadAmbientDamperFactor  
 VL53L0X SPAD Functions, 53

VL53L0X\_GetSpadAmbientDamperThreshold  
 VL53L0X SPAD Functions, 53

VL53L0X\_GetStopCompletedStatus  
 VL53L0X Interrupt Functions, 50

VL53L0X\_GetTotalSignalRate  
 VL53L0X General Functions, 19

VL53L0X\_GetTuningSettingBuffer  
 VL53L0X Init Functions, 21

VL53L0X\_GetUpperLimitMilliMeter  
 VL53L0X General Functions, 19

VL53L0X\_GetVcSEL PulsePeriod  
 VL53L0X Parameters Functions, 29

VL53L0X\_GetVersion  
 VL53L0X General Functions, 13

VL53L0X\_GetWrapAroundCheckEnable  
 VL53L0X Parameters Functions, 39

VL53L0X\_GetXTalkCompensationEnable  
 VL53L0X Parameters Functions, 33

VL53L0X\_GetXTalkCompensationRateMegaCps  
 VL53L0X Parameters Functions, 34

VL53L0X\_GpioFunctionality  
 Gpio Functionality, 76

VL53L0X\_GPIOFUNCTIONALITY\_NEW\_MEASURE\_READY  
 Gpio Functionality, 76

VL53L0X\_GPIOFUNCTIONALITY\_OFF  
 Gpio Functionality, 75

VL53L0X\_GPIOFUNCTIONALITY\_THRESHOLD\_CROSSED\_HIGH  
 Gpio Functionality, 75

VL53L0X\_GPIOFUNCTIONALITY\_THRESHOLD\_CROSSED\_LOW  
 Gpio Functionality, 75

VL53L0X\_GPIOFUNCTIONALITY\_THRESHOLD\_CROSSED\_OUT  
 Gpio Functionality, 76

VL53L0X\_HISTOGRAM\_BUFFER\_SIZE  
 VL53L0X Defines, 58

VL53L0X\_HistogramData\_t, 94  
 VL53L0X\_HistogramMeasurementData\_t, 94  
   BufferSize, 95  
   ErrorStatus, 95  
   FirstBin, 95  
   HistogramData, 95  
   HistogramType, 95  
   NumberOfBins, 95  
 VL53L0X\_HISTOGRAMMODE\_BOTH  
   Defines Histogram modes, 63  
 VL53L0X\_HISTOGRAMMODE\_DISABLED  
   Defines Histogram modes, 63  
 VL53L0X\_HISTOGRAMMODE\_REFERENCE\_ONLY  
   Defines Histogram modes, 63  
 VL53L0X\_HISTOGRAMMODE\_RETURN\_ONLY  
   Defines Histogram modes, 63  
 VL53L0X\_HistogramModes  
   Defines Histogram modes, 63  
 vl53l0x\_i2c\_platform.h, 123  
   bool\_t, 125  
   BYTES\_PER\_DWORD, 125  
   BYTES\_PER\_WORD, 125  
   COMMS\_BUFFER\_SIZE, 125  
   I2C, 124  
   SPI, 124  
   VL53L0\_i2c\_init, 125  
   VL53L0X\_comms\_close, 126  
   VL53L0X\_comms\_initialise, 125  
   VL53L0X\_cycle\_power, 126  
   VL53L0X\_get\_gpio, 129  
   VL53L0X\_get\_timer\_frequency, 130  
   VL53L0X\_get\_timer\_value, 130  
   VL53L0X\_MAX\_STRING\_LENGTH\_PLT, 125  
   VL53L0X\_platform\_wait\_us, 129  
   VL53L0X\_read\_byte, 128  
   VL53L0X\_read\_dword, 128  
   VL53L0X\_read\_multi, 126  
   VL53L0X\_read\_word, 128  
   VL53L0X\_release\_gpio, 130  
   VL53L0X\_set\_gpio, 129  
   VL53L0X\_wait\_ms, 129  
   VL53L0X\_write\_byte, 127  
   VL53L0X\_write\_dword, 127  
   VL53L0X\_write\_multi, 126  
   VL53L0X\_write\_word, 127  
 VL53L0X\_IMPLEMENTATION\_VER\_MAJOR  
   VL53L0X Defines, 58  
 VL53L0X\_IMPLEMENTATION\_VER\_MINOR  
   VL53L0X Defines, 58  
 VL53L0X\_IMPLEMENTATION\_VER\_REVISION  
   VL53L0X Defines, 58  
 VL53L0X\_IMPLEMENTATION\_VER\_SUB  
   VL53L0X Defines, 58  
 vl53l0x\_interrupt\_threshold\_settings.h, 130  
   InterruptThresholdSettings, 130  
 VL53L0X\_InterruptPolarity  
   Defines the Polarity, 66  
 VL53L0X\_INTERRUPTPOLARITY\_HIGH  
   Defines the Polarity, 66  
 VL53L0X\_INTERRUPTPOLARITY\_LOW  
   Defines the Polarity, 66  
 VL53L0X\_isqrt  
   vl53l0x\_api\_core.h, 109  
 VL53L0X\_load\_tuning\_settings  
   vl53l0x\_api\_core.h, 109  
 VL53L0X\_LockSequenceAccess  
   PAL Register Access Functions, 8  
 VL53L0X\_MAKEUINT16  
   General Macro Defines, 71  
 VL53L0X\_MAX\_STRING\_LENGTH  
   VL53L0X Defines, 58  
 VL53L0X\_MAX\_STRING\_LENGTH\_PLT  
   vl53l0x\_i2c\_platform.h, 125  
 VL53L0X\_measurement\_poll\_for\_completion  
   vl53l0x\_api\_core.h, 109  
 VL53L0X\_perform\_offset\_calibration  
   vl53l0x\_api\_calibration.h, 107  
 VL53L0X\_perform\_phase\_calibration  
   vl53l0x\_api\_calibration.h, 107  
 VL53L0X\_perform\_ref\_calibration  
   vl53l0x\_api\_calibration.h, 107  
 VL53L0X\_perform\_ref\_spad\_management  
   vl53l0x\_api\_calibration.h, 107  
 VL53L0X\_perform\_xtalk\_calibration  
   vl53l0x\_api\_calibration.h, 107  
 VL53L0X\_PerformOffsetCalibration  
   VL53L0X Measurement Functions, 43  
 VL53L0X\_PerformRefCalibration  
   VL53L0X Measurement Functions, 41  
 VL53L0X\_PerformRefSpadManagement  
   VL53L0X SPAD Functions, 54  
 VL53L0X\_PerformSingleHistogramMeasurement  
   VL53L0X Measurement Functions, 46  
 VL53L0X\_PerformSingleMeasurement  
   VL53L0X Measurement Functions, 41  
 VL53L0X\_PerformSingleRangingMeasurement  
   VL53L0X Measurement Functions, 46  
 VL53L0X\_PerformXTalkCalibration  
   VL53L0X Measurement Functions, 42  
 VL53L0X\_PerformXTalkMeasurement  
   VL53L0X Measurement Functions, 42  
 vl53l0x\_platform.h, 131  
 vl53l0x\_platform\_log.h, 132  
   \_LOG\_FUNCTION\_END, 132  
   \_LOG\_FUNCTION\_END\_FMT, 132  
   \_LOG\_FUNCTION\_START, 132  
   TRACE\_FUNCTION\_ALL, 133  
   TRACE\_FUNCTION\_I2C, 133  
   TRACE\_FUNCTION\_NONE, 133  
   TRACE\_LEVEL\_ALL, 133  
   TRACE\_LEVEL\_DEBUG, 133  
   TRACE\_LEVEL\_ERRORS, 133  
   TRACE\_LEVEL\_IGNORE, 133  
   TRACE\_LEVEL\_INFO, 133  
   TRACE\_LEVEL\_NONE, 133  
   TRACE\_LEVEL\_WARNING, 133

TRACE\_MODULE\_ALL, 133  
TRACE\_MODULE\_API, 133  
TRACE\_MODULE\_NONE, 133  
TRACE\_MODULE\_PLATFORM, 133  
VL53L0X\_COPYSTRING, 132  
VL53L0X\_ErrLog, 132  
VL53L0X\_platform\_wait\_us  
  vl53l0x\_i2c\_platform.h, 129  
VL53L0X\_PollingDelay  
  VL53L0 Platform Functions, 7  
VL53L0X\_POWERMODE\_IDLE\_LEVEL1  
  List of available Power Modes, 64  
VL53L0X\_POWERMODE\_IDLE\_LEVEL2  
  List of available Power Modes, 64  
VL53L0X\_POWERMODE\_STANDBY\_LEVEL1  
  List of available Power Modes, 64  
VL53L0X\_POWERMODE\_STANDBY\_LEVEL2  
  List of available Power Modes, 64  
VL53L0X\_PowerModes  
  List of available Power Modes, 64  
VL53L0X\_quadrature\_sum  
  vl53l0x\_api\_core.h, 109  
VL53L0X\_RangeData\_t, 95  
VL53L0X\_RangingMeasurementData\_t, 96  
  AmbientRateRtnMegaCps, 97  
  EffectiveSpadRtnCount, 97  
  MeasurementTimeUsec, 96  
  RangeDMaxMilliMeter, 97  
  RangeFractionalPart, 97  
  RangeMilliMeter, 96  
  RangeStatus, 97  
  SignalRateRtnMegaCps, 97  
  TimeStamp, 96  
  ZoneId, 97  
VL53L0X\_RdByte  
  PAL Register Access Functions, 10  
VL53L0X\_RdDWord  
  PAL Register Access Functions, 11  
VL53L0X\_RdWord  
  PAL Register Access Functions, 10  
VL53L0X\_read\_byte  
  vl53l0x\_i2c\_platform.h, 128  
VL53L0X\_read\_dword  
  vl53l0x\_i2c\_platform.h, 128  
VL53L0X\_read\_multi  
  vl53l0x\_i2c\_platform.h, 126  
VL53L0X\_read\_word  
  vl53l0x\_i2c\_platform.h, 128  
VL53L0X\_ReadMulti  
  PAL Register Access Functions, 9  
VL53L0X\_REF\_SPAD\_BUFFER\_SIZE  
  VL53L0X Defines, 58  
VL53L0X\_REG\_ALGO\_PART\_TO\_PART\_RANGE\_OFFSET\_MM  
  Define Registers, 80  
VL53L0X\_REG\_ALGO\_PHASECAL\_CONFIG\_TIMEOUT  
  Define Registers, 84  
VL53L0X\_REG\_ALGO\_PHASECAL\_LIM  
  Define Registers, 84  
VL53L0X\_REG\_CROSSTALK\_COMPENSATION\_PEAK\_RATE\_MCPS  
  Define Registers, 82  
VL53L0X\_REG\_DYNAMIC\_SPAD\_NUM\_REQUESTED\_REF\_SPAD  
  Define Registers, 83  
VL53L0X\_REG\_DYNAMIC\_SPAD\_REF\_EN\_START\_OFFSET  
  Define Registers, 83  
VL53L0X\_REG\_FINAL\_RANGE\_CONFIG\_MIN\_COUNT\_RATE\_RTN\_LIMIT  
  Define Registers, 81  
VL53L0X\_REG\_FINAL\_RANGE\_CONFIG\_MIN\_SNR  
  Define Registers, 81  
VL53L0X\_REG\_FINAL\_RANGE\_CONFIG\_TIMEOUT\_MACROP\_HI  
  Define Registers, 82  
VL53L0X\_REG\_FINAL\_RANGE\_CONFIG\_TIMEOUT\_MACROP\_LO  
  Define Registers, 82  
VL53L0X\_REG\_FINAL\_RANGE\_CONFIG\_VALID\_PHASE\_HIGH  
  Define Registers, 81  
VL53L0X\_REG\_FINAL\_RANGE\_CONFIG\_VALID\_PHASE\_LOW  
  Define Registers, 81  
VL53L0X\_REG\_FINAL\_RANGE\_CONFIG\_VCSSEL\_PERIOD  
  Define Registers, 82  
VL53L0X\_REG\_GLOBAL\_CONFIG\_REF\_EN\_START\_SELECT  
  Define Registers, 83  
VL53L0X\_REG\_GLOBAL\_CONFIG\_SPAD\_ENABLES\_REF\_0  
  Define Registers, 83  
VL53L0X\_REG\_GLOBAL\_CONFIG\_SPAD\_ENABLES\_REF\_1  
  Define Registers, 83  
VL53L0X\_REG\_GLOBAL\_CONFIG\_SPAD\_ENABLES\_REF\_2  
  Define Registers, 83  
VL53L0X\_REG\_GLOBAL\_CONFIG\_SPAD\_ENABLES\_REF\_3  
  Define Registers, 83  
VL53L0X\_REG\_GLOBAL\_CONFIG\_SPAD\_ENABLES\_REF\_4  
  Define Registers, 83  
VL53L0X\_REG\_GLOBAL\_CONFIG\_SPAD\_ENABLES\_REF\_5  
  Define Registers, 83  
VL53L0X\_REG\_GLOBAL\_CONFIG\_VCSEL\_WIDTH  
  Define Registers, 83  
VL53L0X\_REG\_GPIO\_HV\_MUX\_ACTIVE\_HIGH  
  Define Registers, 79  
VL53L0X\_REG\_HISTOGRAM\_CONFIG\_INITIAL\_PHASE\_SELECT  
  Define Registers, 82



VL53L0X\_REG\_HISTOGRAM\_CONFIG\_READ  
OUT\_CTRL  
Define Registers, 82

VL53L0X\_REG\_I2C\_SLAVE\_DEVICE\_ADDRESS  
SS  
Define Registers, 80

VL53L0X\_REG\_IDENTIFICATION\_MODEL\_ID  
D  
Define Registers, 82

VL53L0X\_REG\_IDENTIFICATION\_REVISION\_ID  
ID  
Define Registers, 82

VL53L0X\_REG\_MSRC\_CONFIG\_CONTROL  
Define Registers, 80

VL53L0X\_REG\_MSRC\_CONFIG\_TIMEOUT\_MACROP  
ACROP  
Define Registers, 82

VL53L0X\_REG\_OSC\_CALIBRATE\_VAL  
Define Registers, 82

VL53L0X\_REG\_POWER\_MANAGEMENT\_GO1\_POWER\_FORCE  
1\_POWER\_FORCE  
Define Registers, 83

VL53L0X\_REG\_PRE\_RANGE\_CONFIG\_MIN\_SNR  
NR  
Define Registers, 80

VL53L0X\_REG\_PRE\_RANGE\_CONFIG\_SIGMA\_THRESH\_HI  
A\_THRESH\_HI  
Define Registers, 81

VL53L0X\_REG\_PRE\_RANGE\_CONFIG\_SIGMA\_THRESH\_LO  
A\_THRESH\_LO  
Define Registers, 81

VL53L0X\_REG\_PRE\_RANGE\_CONFIG\_TIMEOUT\_MACROP\_HI  
OUT\_MACROP\_HI  
Define Registers, 81

VL53L0X\_REG\_PRE\_RANGE\_CONFIG\_TIMEOUT\_MACROP\_LO  
OUT\_MACROP\_LO  
Define Registers, 81

VL53L0X\_REG\_PRE\_RANGE\_CONFIG\_VALID\_PHASE\_HIGH  
D\_PHASE\_HIGH  
Define Registers, 81

VL53L0X\_REG\_PRE\_RANGE\_CONFIG\_VALID\_PHASE\_LOW  
D\_PHASE\_LOW  
Define Registers, 81

VL53L0X\_REG\_PRE\_RANGE\_CONFIG\_VCSEL\_PERIOD  
L\_PERIOD  
Define Registers, 81

VL53L0X\_REG\_PRE\_RANGE\_MIN\_COUNT\_RATE\_RTN\_LIMIT  
ATE\_RTN\_LIMIT  
Define Registers, 81

VL53L0X\_REG\_RESULT\_CORE\_AMBIENT\_WINDOW\_EVENTS\_REF  
WINDOW\_EVENTS\_REF  
Define Registers, 80

VL53L0X\_REG\_RESULT\_CORE\_AMBIENT\_WINDOW\_EVENTS\_RTN  
WINDOW\_EVENTS\_RTN  
Define Registers, 80

VL53L0X\_REG\_RESULT\_CORE\_PAGE  
Define Registers, 80

VL53L0X\_REG\_RESULT\_CORE\_RANGING\_TOTAL\_EVENTS\_REF  
OTAL\_EVENTS\_REF  
Define Registers, 80

VL53L0X\_REG\_RESULT\_CORE\_RANGING\_TOTAL\_EVENTS\_RTN  
OTAL\_EVENTS\_RTN  
Define Registers, 80

VL53L0X\_REG\_RESULT\_INTERRUPT\_STATUS  
S  
Define Registers, 80

VL53L0X\_REG\_RESULT\_PEAK\_SIGNAL\_RATE\_REF  
E\_REF  
Define Registers, 80

VL53L0X\_REG\_RESULT\_RANGE\_STATUS  
Define Registers, 80

VL53L0X\_REG\_SOFT\_RESET\_GO2\_SOFT\_RESET\_N  
SET\_N  
Define Registers, 82

VL53L0X\_REG\_SYSRANGE\_MODE\_BACKTOBACK  
BACK  
Define Registers, 78

VL53L0X\_REG\_SYSRANGE\_MODE\_HISTOGRAM  
AM  
Define Registers, 78

VL53L0X\_REG\_SYSRANGE\_MODE\_MASK  
Define Registers, 78

VL53L0X\_REG\_SYSRANGE\_MODE\_SINGLES  
HOT  
Define Registers, 78

VL53L0X\_REG\_SYSRANGE\_MODE\_START\_STOP  
TOP  
Define Registers, 78

VL53L0X\_REG\_SYSRANGE\_MODE\_TIMED  
Define Registers, 78

VL53L0X\_REG\_SYSRANGE\_START  
Define Registers, 78

VL53L0X\_REG\_SYSTEM\_HISTOGRAM\_BIN  
Define Registers, 82

VL53L0X\_REG\_SYSTEM\_INTERMEASUREMENT\_PERIOD  
ENT\_PERIOD  
Define Registers, 79

VL53L0X\_REG\_SYSTEM\_INTERRUPT\_CLEAR  
R  
Define Registers, 79

VL53L0X\_REG\_SYSTEM\_INTERRUPT\_CONFIG\_GPIO  
G\_GPIO  
Define Registers, 79

VL53L0X\_REG\_SYSTEM\_INTERRUPT\_GPIO\_DISABLED  
DISABLED  
Define Registers, 79

VL53L0X\_REG\_SYSTEM\_INTERRUPT\_GPIO\_LEVEL\_HIGH  
LEVEL\_HIGH  
Define Registers, 79

VL53L0X\_REG\_SYSTEM\_INTERRUPT\_GPIO\_LEVEL\_LOW  
LEVEL\_LOW  
Define Registers, 79

VL53L0X\_REG\_SYSTEM\_INTERRUPT\_GPIO\_NEW\_SAMPLE\_READY  
NEW\_SAMPLE\_READY  
Define Registers, 79

VL53L0X\_REG\_SYSTEM\_INTERRUPT\_GPIO\_OUT\_OF\_WINDOW  
OUT\_OF\_WINDOW  
Define Registers, 79

VL53L0X\_REG\_SYSTEM\_RANGE\_CONFIG  
Define Registers, 79

VL53L0X\_REG\_SYSTEM\_SEQUENCE\_CONFIG  
Define Registers, 79

VL53L0X\_REG\_SYSTEM\_THRESH\_HIGH  
Define Registers, 78

VL53L0X\_REG\_SYSTEM\_THRESH\_LOW  
Define Registers, 79

VL53L0X\_REG\_VHV\_CONFIG\_PAD\_SCL\_SD\_A\_EXTSUP\_HV  
Define Registers, 84

VL53L0X\_release\_gpio  
vl53l0x\_i2c\_platform.h, 130

VL53L0X\_ResetDevice  
VL53L0X Init Functions, 22

VL53L0X\_reverse\_bytes  
vl53l0x\_api\_core.h, 109

VL53L0X\_SchedulerSequenceSteps\_t, 97  
DssOn, 98  
FinalRangeOn, 98  
MsrcOn, 98  
PreRangeOn, 98  
TccOn, 98

VL53L0X\_SEQUENCESTEP\_DSS  
Defines the Polarity, 68

VL53L0X\_SEQUENCESTEP\_FINAL\_RANGE  
Defines the Polarity, 68

VL53L0X\_SEQUENCESTEP\_MSRC  
Defines the Polarity, 68

VL53L0X\_SEQUENCESTEP\_NUMBER\_OF\_CHECKS  
Defines the Polarity, 68

VL53L0X\_SEQUENCESTEP\_PRE\_RANGE  
Defines the Polarity, 68

VL53L0X\_SEQUENCESTEP\_TCC  
Defines the Polarity, 68

VL53L0X\_SequenceStepId  
Defines the Polarity, 68

VL53L0X\_set\_gpio  
vl53l0x\_i2c\_platform.h, 129

VL53L0X\_set\_measurement\_timing\_budget\_micro\_seconds  
vl53l0x\_api\_core.h, 109

VL53L0X\_set\_offset\_calibration\_data\_micro\_meter  
vl53l0x\_api\_calibration.h, 107

VL53L0X\_set\_ref\_calibration  
vl53l0x\_api\_calibration.h, 107

VL53L0X\_set\_reference\_spads  
vl53l0x\_api\_calibration.h, 107

VL53L0X\_set\_vcsel\_pulse\_period  
vl53l0x\_api\_core.h, 109

VL53L0X\_SETARRAYPARAMETERFIELD  
General Macro Defines, 69

VL53L0X\_SetDeviceAddress  
VL53L0X Init Functions, 20

VL53L0X\_SetDeviceMode  
VL53L0X Parameters Functions, 26

VL53L0X\_SetDeviceParameters  
VL53L0X Parameters Functions, 25

VL53L0X\_SETDEVICESPECIFICPARAMETER  
General Macro Defines, 70

VL53L0X\_SetDmaxCalParameters  
VL53L0X Parameters Functions, 39

VL53L0X\_SetGpioConfig  
VL53L0X Interrupt Functions, 48

VL53L0X\_SetGroupParamHold  
VL53L0X General Functions, 18

VL53L0X\_SetHistogramMode  
VL53L0X Parameters Functions, 27

VL53L0X\_SetInterMeasurementPeriodMilliseconds  
VL53L0X Parameters Functions, 32

VL53L0X\_SetInterruptThresholds  
VL53L0X Interrupt Functions, 49

VL53L0X\_SetLimitCheckEnable  
VL53L0X Parameters Functions, 36

VL53L0X\_SetLimitCheckValue  
VL53L0X Parameters Functions, 37

VL53L0X\_SetLinearityCorrectiveGain  
VL53L0X General Functions, 18

VL53L0X\_SetMeasurementTimingBudgetMicroSeconds  
VL53L0X Parameters Functions, 28

VL53L0X\_SetNumberOfROI Zones  
VL53L0X Measurement Functions, 46

VL53L0X\_SetOffsetCalibrationDataMicroMeter  
VL53L0X General Functions, 17

VL53L0X\_SETPARAMETERFIELD  
General Macro Defines, 69

VL53L0X\_SetPowerMode  
VL53L0X General Functions, 16

VL53L0X\_SetRangeFractionEnable  
VL53L0X Parameters Functions, 27

VL53L0X\_SetRefCalibration  
VL53L0X Parameters Functions, 34

VL53L0X\_SetReferenceSpads  
VL53L0X SPAD Functions, 54

VL53L0X\_SetSequenceStepEnable  
VL53L0X Parameters Functions, 30

VL53L0X\_SetSequenceStepTimeout  
VL53L0X Parameters Functions, 31

VL53L0X\_SetSpadAmbientDamperFactor  
VL53L0X SPAD Functions, 53

VL53L0X\_SetSpadAmbientDamperThreshold  
VL53L0X SPAD Functions, 52

VL53L0X\_SetTuningSettingBuffer  
VL53L0X Init Functions, 21

VL53L0X\_SetVcselPulsePeriod  
VL53L0X Parameters Functions, 29

VL53L0X\_SetWrapAroundCheckEnable  
VL53L0X Parameters Functions, 38

VL53L0X\_SetXTalkCompensationEnable  
VL53L0X Parameters Functions, 33

VL53L0X\_SetXTalkCompensationRateMegaCps  
VL53L0X Parameters Functions, 34

VL53L0X\_SIGMA\_ESTIMATE\_MAX\_VALUE  
Define Registers, 83

VL53L0X\_SpadData\_t, 98  
RefGoodSpadMap, 99  
RefSpadEnables, 99

VL53L0X\_SPECIFICATION\_VER\_MAJOR  
VL53L0X Defines, 57

VL53L0X\_SPECIFICATION\_VER\_MINOR  
VL53L0X Defines, 57

VL53L0X\_SPECIFICATION\_VER\_REVISION  
VL53L0X Defines, 58

VL53L0X\_SPECIFICATION\_VER\_SUB  
VL53L0X Defines, 57

VL53L0X\_SPEED\_OF\_LIGHT\_IN\_AIR  
Define Registers, 84

VL53L0X\_StartMeasurement  
VL53L0X Measurement Functions, 43

VL53L0X\_State  
Defines the current status of the device, 65

VL53L0X\_STATE\_ERROR  
Defines the current status of the device, 65

VL53L0X\_STATE\_IDLE  
Defines the current status of the device, 65

VL53L0X\_STATE\_POWERDOWN  
Defines the current status of the device, 65

VL53L0X\_STATE\_RUNNING  
Defines the current status of the device, 65

VL53L0X\_STATE\_STANDBY  
Defines the current status of the device, 65

VL53L0X\_STATE\_UNKNOWN  
Defines the current status of the device, 65

VL53L0X\_STATE\_WAIT\_STATICINIT  
Defines the current status of the device, 65

VL53L0X\_StaticInit  
VL53L0X Init Functions, 22

VL53L0X\_StopMeasurement  
VL53L0X Measurement Functions, 43

VL53L0X\_STRING\_CHECKENABLE\_RANGE\_IGNORE\_THRESHOLD  
vl53l0x\_api\_strings.h, 117

VL53L0X\_STRING\_CHECKENABLE\_SIGMA\_FINAL\_RANGE  
vl53l0x\_api\_strings.h, 116

VL53L0X\_STRING\_CHECKENABLE\_SIGNAL\_RATE\_FINAL\_RANGE  
vl53l0x\_api\_strings.h, 116

VL53L0X\_STRING\_CHECKENABLE\_SIGNAL\_RATE\_MSRC  
vl53l0x\_api\_strings.h, 117

VL53L0X\_STRING\_CHECKENABLE\_SIGNAL\_RATE\_PRE\_RANGE  
vl53l0x\_api\_strings.h, 117

VL53L0X\_STRING\_CHECKENABLE\_SIGNAL\_REF\_CLIP  
vl53l0x\_api\_strings.h, 117

VL53L0X\_STRING\_DEVICE\_INFO\_NAME  
vl53l0x\_api\_strings.h, 112

VL53L0X\_STRING\_DEVICE\_INFO\_NAME\_ES1  
vl53l0x\_api\_strings.h, 112

VL53L0X\_STRING\_DEVICE\_INFO\_NAME\_TS0  
vl53l0x\_api\_strings.h, 112

VL53L0X\_STRING\_DEVICE\_INFO\_NAME\_TS1  
vl53l0x\_api\_strings.h, 112

VL53L0X\_STRING\_DEVICE\_INFO\_NAME\_TS2  
vl53l0x\_api\_strings.h, 112

VL53L0X\_STRING\_DEVICE\_INFO\_TYPE  
vl53l0x\_api\_strings.h, 112

VL53L0X\_STRING\_DEVICEERROR\_ALGOOVERFLOW  
vl53l0x\_api\_strings.h, 116

VL53L0X\_STRING\_DEVICEERROR\_ALGOUNDERFLOW  
vl53l0x\_api\_strings.h, 116

VL53L0X\_STRING\_DEVICEERROR\_MINCLIP  
vl53l0x\_api\_strings.h, 116

VL53L0X\_STRING\_DEVICEERROR\_MSRCNOTARGET  
vl53l0x\_api\_strings.h, 115

VL53L0X\_STRING\_DEVICEERROR\_NONE  
vl53l0x\_api\_strings.h, 115

VL53L0X\_STRING\_DEVICEERROR\_NOVHVVALUEFOUND  
vl53l0x\_api\_strings.h, 115

VL53L0X\_STRING\_DEVICEERROR\_PHASECONSISTENCY  
vl53l0x\_api\_strings.h, 116

VL53L0X\_STRING\_DEVICEERROR\_RANGECOMPLETE  
vl53l0x\_api\_strings.h, 116

VL53L0X\_STRING\_DEVICEERROR\_RANGEIGNORETHRESHOLD  
vl53l0x\_api\_strings.h, 116

VL53L0X\_STRING\_DEVICEERROR\_RANGEHASECHECK  
vl53l0x\_api\_strings.h, 115

VL53L0X\_STRING\_DEVICEERROR\_SIGMAHRESHOLDCHECK  
vl53l0x\_api\_strings.h, 116

VL53L0X\_STRING\_DEVICEERROR\_SNRCHECK  
vl53l0x\_api\_strings.h, 115

VL53L0X\_STRING\_DEVICEERROR\_TCC  
vl53l0x\_api\_strings.h, 116

VL53L0X\_STRING\_DEVICEERROR\_UNKNOWN  
vl53l0x\_api\_strings.h, 116

VL53L0X\_STRING\_DEVICEERROR\_VCSELCONTINUITYTESTFAILURE  
vl53l0x\_api\_strings.h, 115

VL53L0X\_STRING\_DEVICEERROR\_VCSELWATCHDOGTESTFAILURE  
vl53l0x\_api\_strings.h, 115

VL53L0X\_STRING\_ERROR\_BUFFER\_TOO\_SMALL  
vl53l0x\_api\_strings.h, 113

VL53L0X\_STRING\_ERROR\_CALIBRATIONWARNING  
vl53l0x\_api\_strings.h, 112

VL53L0X\_STRING\_ERROR\_CONTROL\_INTERFACE  
vl53l0x\_api\_strings.h, 113

VL53L0X\_STRING\_ERROR\_DIVISION\_BY\_ZERO  
 vl53l0x\_api\_strings.h, 113

VL53L0X\_STRING\_ERROR\_GPIO\_FUNCTIONALITY\_NOT\_SUPPORTED  
 vl53l0x\_api\_strings.h, 113

VL53L0X\_STRING\_ERROR\_GPIO\_NOT\_EXISTING  
 vl53l0x\_api\_strings.h, 113

VL53L0X\_STRING\_ERROR\_INTERRUPT\_NOT\_CLEARED  
 vl53l0x\_api\_strings.h, 113

VL53L0X\_STRING\_ERROR\_INVALID\_COMMAND  
 vl53l0x\_api\_strings.h, 113

VL53L0X\_STRING\_ERROR\_INVALID\_PARAMETER  
 vl53l0x\_api\_strings.h, 112

VL53L0X\_STRING\_ERROR\_MIN\_CLIPPED  
 vl53l0x\_api\_strings.h, 112

VL53L0X\_STRING\_ERROR\_MODE\_NOT\_SUPPORTED  
 vl53l0x\_api\_strings.h, 113

VL53L0X\_STRING\_ERROR\_NONE  
 vl53l0x\_api\_strings.h, 112

VL53L0X\_STRING\_ERROR\_NOT\_IMPLEMENTED  
 vl53l0x\_api\_strings.h, 114

VL53L0X\_STRING\_ERROR\_NOT\_SUPPORTED  
 vl53l0x\_api\_strings.h, 113

VL53L0X\_STRING\_ERROR\_RANGE\_ERROR  
 vl53l0x\_api\_strings.h, 113

VL53L0X\_STRING\_ERROR\_REF\_SPAD\_INIT  
 vl53l0x\_api\_strings.h, 113

VL53L0X\_STRING\_ERROR\_TIME\_OUT  
 vl53l0x\_api\_strings.h, 113

VL53L0X\_STRING\_ERROR\_UNDEFINED  
 vl53l0x\_api\_strings.h, 112

VL53L0X\_STRING\_RANGESTATUS\_HW  
 vl53l0x\_api\_strings.h, 114

VL53L0X\_STRING\_RANGESTATUS\_MINRANGE  
 vl53l0x\_api\_strings.h, 114

VL53L0X\_STRING\_RANGESTATUS\_NONE  
 vl53l0x\_api\_strings.h, 114

VL53L0X\_STRING\_RANGESTATUS\_PHASE  
 vl53l0x\_api\_strings.h, 114

VL53L0X\_STRING\_RANGESTATUS\_RANGEVALID  
 vl53l0x\_api\_strings.h, 114

VL53L0X\_STRING\_RANGESTATUS\_SIGMA  
 vl53l0x\_api\_strings.h, 114

VL53L0X\_STRING\_RANGESTATUS\_SIGNAL  
 vl53l0x\_api\_strings.h, 114

VL53L0X\_STRING\_SEQUENCESTEP\_DSS  
 vl53l0x\_api\_strings.h, 117

VL53L0X\_STRING\_SEQUENCESTEP\_FINAL\_RANGE  
 vl53l0x\_api\_strings.h, 117

VL53L0X\_STRING\_SEQUENCESTEP\_MSRC  
 vl53l0x\_api\_strings.h, 117

VL53L0X\_STRING\_SEQUENCESTEP\_PRE\_RANGE  
 vl53l0x\_api\_strings.h, 117

VL53L0X\_STRING\_SEQUENCESTEP\_TCC  
 vl53l0x\_api\_strings.h, 117

VL53L0X\_STRING\_STATE\_ERROR  
 vl53l0x\_api\_strings.h, 115

VL53L0X\_STRING\_STATE\_IDLE  
 vl53l0x\_api\_strings.h, 115

VL53L0X\_STRING\_STATE\_POWERDOWN  
 vl53l0x\_api\_strings.h, 114

VL53L0X\_STRING\_STATE\_RUNNING  
 vl53l0x\_api\_strings.h, 115

VL53L0X\_STRING\_STATE\_STANDBY  
 vl53l0x\_api\_strings.h, 114

VL53L0X\_STRING\_STATE\_UNKNOWN  
 vl53l0x\_api\_strings.h, 115

VL53L0X\_STRING\_STATE\_WAIT\_STATICINIT  
 vl53l0x\_api\_strings.h, 114

VL53L0X\_STRING\_UNKNOW\_ERROR\_CODE  
 vl53l0x\_api\_strings.h, 114

vl53l0x\_tuning.h, 133  
 DefaultTuningSettings, 133

vl53l0x\_types.h, 134  
 FixPoint1616\_t, 135  
 int16\_t, 135  
 int32\_t, 134  
 int8\_t, 135  
 uint16\_t, 134  
 uint32\_t, 134  
 uint64\_t, 134  
 uint8\_t, 135

VL53L0X\_UnlockSequenceAccess  
 PAL Register Access Functions, 9

VL53L0X\_UpdateByte  
 PAL Register Access Functions, 11

VL53L0X\_VCSEL\_PERIOD\_FINAL\_RANGE  
 Vcsel Period Defines, 67

VL53L0X\_VCSEL\_PERIOD\_PRE\_RANGE  
 Vcsel Period Defines, 67

VL53L0X\_VcselPeriod  
 Vcsel Period Defines, 67

VL53L0X\_Version\_t, 99  
 build, 100  
 major, 99  
 minor, 100  
 revision, 99

VL53L0X\_wait\_ms  
 vl53l0x\_i2c\_platform.h, 129

VL53L0X\_WaitDeviceBooted  
 VL53L0X Init Functions, 22

VL53L0X\_WaitDeviceReadyForNewMeasurement  
 VL53L0X Measurement Functions, 44

VL53L0X\_WrByte  
 PAL Register Access Functions, 9

VL53L0X\_WrDWord  
 PAL Register Access Functions, 10

VL53L0X_write_byte	VL53L0X10_IMPLEMENTATION_VER_SUB
vl53l0x_i2c_platform.h, 127	VL53L0X Defines, 57
VL53L0X_write_dword	VL53L0X10_SPECIFICATION_VER_MAJOR
vl53l0x_i2c_platform.h, 127	VL53L0X Defines, 56
VL53L0X_write_multi	VL53L0X10_SPECIFICATION_VER_MINOR
vl53l0x_i2c_platform.h, 126	VL53L0X Defines, 56
VL53L0X_write_word	VL53L0X10_SPECIFICATION_VER_REVISION
vl53l0x_i2c_platform.h, 127	VL53L0X Defines, 57
VL53L0X_WriteMulti	VL53L0X10_SPECIFICATION_VER_SUB
PAL Register Access Functions, 9	VL53L0X Defines, 57
VL53L0X_WrWord	WrapAroundCheckEnable
PAL Register Access Functions, 10	VL53L0X_DeviceParameters_t, 91
VL53L0X10_IMPLEMENTATION_VER_MAJOR	XTalkCompensationEnable
R	VL53L0X_DeviceParameters_t, 90
VL53L0X Defines, 57	XTalkCompensationRangeMilliMeter
VL53L0X10_IMPLEMENTATION_VER_MINOR	VL53L0X_DeviceParameters_t, 90
R	XTalkCompensationRateMegaCps
VL53L0X Defines, 57	VL53L0X_DeviceParameters_t, 90
VL53L0X10_IMPLEMENTATION_VER_REVISION	ZoneId
ION	VL53L0X_RangingMeasurementData_t, 97
VL53L0X Defines, 57	