

VL53L0X API Specification

Version 1.0.2.4823
10/13/2016 11:11:00 AM

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	130
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 `VL53L0X 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** 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:

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 Index

Data Structures

Here are the data structures with brief descriptions:

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

File Index

File List

Here is a list of all files with brief descriptions:

<u>PAL_disclaimer.c</u> (No code doxygen doc only)	100
<u>vl53l0x_api.h</u>	100

vl53l0x_api_calibration.h	106
vl53l0x_api_core.h	107
vl53l0x_api_ranging.h	110
vl53l0x_api_strings.h	110
vl53l0x_def.h (Type definitions for VL53L0X API)	118
vl53l0x_device.h	121
vl53l0x_doxydoc.c	123
vl53l0x_i2c_platform.h	123
vl53l0x_interrupt_threshold_settings.h	130
vl53l0x_platform.h (Function prototype definitions for Ewok Platform layer)	130
vl53l0x_platform_log.h (Platform log function definition)	132
vl53l0x_tuning.h	133
vl53l0x_types.h (VL53L0X types definition)	134

Module Documentation

VL53L0X Platform Functions

VL53L0X Platform Functions.

Modules

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

file [vl53l0x_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

VL53L0X 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 [vl53l0x_types.h](#) files hold basic type definition that may requires porting

file [vl53l0x_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:

pVersion	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 VL53L0X_RangingMeasurementData_t
<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 VL53L0X_DeviceError
<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 VL53L0X_Error
<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 VL53L0X_PowerModes 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 VL53L0X_PowerModes 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:

<i>Dev</i>	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:

<i>Dev</i>	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:

<i>Dev</i>	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 microsecs when wraparound enabled >= 12000 microsecs 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>VcSELPeriodType</i>	VCSEL period identifier (pre-range final).
<i>VCSELPulsePeriod</i>	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_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 < VL53L0X_GetNumberOfLimitCheck()).
<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 < VL53L0X_GetNumberOfLimitCheck()).
<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 < VL53L0X_GetNumberOfLimitCheck()).
<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 < VL53L0X_GetNumberOfLimitCheck()).
<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 < VL53L0X_GetNumberOfLimitCheck()).
<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 < VL53L0X_GetNumberOfLimitCheck()).
<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 < VL53L0X_GetNumberOfLimitCheck()).
<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 VL53L0X_GpioFunctionality

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

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 VL53L0X_GpioFunctionality
<i>pPolarity</i>	Pointer to interrupt polarity. Active high or active low see VL53L0X_InterruptPolarity

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](#) 2
VL53L0X PAL IMPLEMENTATION sub version.
- #define [VL53L0X_IMPLEMENTATION_VER_REVISION](#) 4823
VL53L0X PAL IMPLEMENTATION sub version.
- #define [VL53L0X_DEFAULT_MAX_LOOP](#) 2000
- #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 2

VL53L0X PAL IMPLEMENTATION sub version.

Definition at line 83 of file vl53l0x_def.h.

#define VL53L0X_IMPLEMENTATION_VER_REVISION 4823

VL53L0X PAL IMPLEMENTATION sub version.

Definition at line 85 of file vl53l0x_def.h.

#define VL53L0X_DEFAULT_MAX_LOOP 2000

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_VcslPeriod](#)) 0)

Identifies the pre-range vcsel period.

Definition at line 512 of file vl53l0x_def.h.

#define VL53L0X_VCSEL_PERIOD_FINAL_RANGE (([VL53L0X_VcslPeriod](#)) 1)

Identifies the final range vcsel period.

Definition at line 514 of file vl53l0x_def.h.

Typedef Documentation

typedef [uint8_t VL53L0X_VcslPeriod](#)

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_VcslPeriod](#)) 0)
- #define [VL53L0X_SEQUENCESTEP_DSS](#) (([VL53L0X_VcslPeriod](#)) 1)
- #define [VL53L0X_SEQUENCESTEP_MSRC](#) (([VL53L0X_VcslPeriod](#)) 2)
- #define [VL53L0X_SEQUENCESTEP_PRE_RANGE](#) (([VL53L0X_VcslPeriod](#)) 3)
- #define [VL53L0X_SEQUENCESTEP_FINAL_RANGE](#) (([VL53L0X_VcslPeriod](#)) 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)&0xFFFF)
- #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](#) * plsApertureSpads)

[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_UNKNOW_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](#) 2
VL53L0X PAL IMPLEMENTATION sub version.
- #define [VL53L0X_IMPLEMENTATION_VER_REVISION](#) 4823
VL53L0X PAL IMPLEMENTATION sub version.
- #define [VL53L0X_DEFAULT_MAX_LOOP](#) 2000
- #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_SINGLESHOT](#) 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.
- [`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

[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.

All end user OS/platform/application porting.

vl53l0x_platform_log.h File Reference

platform log function definition

```
#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 }

Detailed Description

platform log function definition

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

VL53L0X types definition.

```
#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.

Detailed Description

VL53L0X types definition.

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

[_LOG_FUNCTION_END](#)
vl53l0x_platform_log.h, 132
[_LOG_FUNCTION_END_FMT](#)
vl53l0x_platform_log.h, 132
[_LOG_FUNCTION_START](#)
vl53l0x_platform_log.h, 132

[AmbientRateRtnMegaCps](#)
VL53L0X_RangingMeasurementData_t, 97
[AmbTuningWindowFactor_K](#)
VL53L0X_DMaxData_t, 94
[Basic type definition](#), 11
[bool_t](#)

vl53l0x_i2c_platform.h, 125
 BufferSize
 VL53L0X_HistogramMeasurementData_t, 95
 build
 VL53L0X_Version_t, 100
 BYTES_PER_DWORD
 vl53l0x_i2c_platform.h, 125
 BYTES_PER_WORD
 vl53l0x_i2c_platform.h, 125
 Check Enable list, 74
 VL53L0X_CHECKENABLE_NUMBER_OF_CHECKS, 74
 VL53L0X_CHECKENABLE_RANGE_IGNORE_THRESHOLD, 74
 VL53L0X_CHECKENABLE_SIGMA_FINAL_RANGE, 74
 VL53L0X_CHECKENABLE_SIGNAL_RATE_FINAL_RANGE, 74
 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_G01_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_SINGLESHOT, 78
 VL53L0X_REG_SYSRANGE_MODE_START_STOP, 78
 VL53L0X_REG_SYSRANGE_MODE_TIMED, 78
 VL53L0X_REG_SYSRANGE_START, 78
 VL53L0X_REG_SYSTEM_HISTOGRAM_BIN, 82
 VL53L0X_REG_SYSTEM_INTERMEASUREMENT_PERIOD, 79
 VL53L0X_REG_SYSTEM_INTERRUPT_CLEAR, 79
 VL53L0X_REG_SYSTEM_INTERRUPT_CONFIG_GPIO, 79
 VL53L0X_REG_SYSTEM_INTERRUPT_GPIO_DISABLED, 79
 VL53L0X_REG_SYSTEM_INTERRUPT_GPIO_LEVEL_HIGH, 79
 VL53L0X_REG_SYSTEM_INTERRUPT_GPIO_LEVEL_LOW, 79
 VL53L0X_REG_SYSTEM_INTERRUPT_GPIO_NEW_SAMPLE_READY, 79
 VL53L0X_REG_SYSTEM_INTERRUPT_GPIO_OUT_OF_WINDOW, 79
 VL53L0X_REG_SYSTEM_RANGE_CONFIG, 79
 VL53L0X_REG_SYSTEM_SEQUENCE_CONFIG, 79
 VL53L0X_REG_SYSTEM_THRESH_HIGH, 78
 VL53L0X_REG_SYSTEM_THRESH_LOW, 79
 VL53L0X_REG_VHV_CONFIG_PAD_SCL_SDA_EXTSUP_HV, 84
 VL53L0X_SIGMA_ESTIMATE_MAX_VALUE, 83
 VL53L0X_SPEED_OF_LIGHT_IN_AIR, 84
 Defines Device modes, 61
 VL53L0X_DEVICEMODE_CONTINUOUS_RANGING, 61
 VL53L0X_DEVICEMODE_CONTINUOUS_Timed_RANGING, 62
 VL53L0X_DEVICEMODE_GPIO_DRIVE, 62
 VL53L0X_DEVICEMODE_GPIO_OSC, 62
 VL53L0X_DEVICEMODE_SINGLE_ALS, 62
 VL53L0X_DEVICEMODE_SINGLE_HISTOGRAM, 62
 VL53L0X_DEVICEMODE_SINGLE_RANGING, 61
 VL53L0X_DeviceModes, 62
 Defines Histogram modes, 62
 VL53L0X_HISTOGRAMMODE_BOTH, 63
 VL53L0X_HISTOGRAMMODE_DISABLED, 63
 VL53L0X_HISTOGRAMMODE_REFERENCE_ONLY, 63
 VL53L0X_HISTOGRAMMODE_RETURN_ONLY, 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_ALGOOVERFLOW, 73
 VL53L0X_DEVICEERROR_ALGOUNDERFLOW, 73
 VL53L0X_DEVICEERROR_MINCLIP, 73
 VL53L0X_DEVICEERROR_MSRCNOTARGET, 72
 VL53L0X_DEVICEERROR_NONE, 72
 VL53L0X_DEVICEERROR_NOHVVALUEFOUND, 72
 VL53L0X_DEVICEERROR_PHASECONSISTENCY, 73
 VL53L0X_DEVICEERROR_RANGECOMPLETE, 73

VL53L0X_DEVICEERROR_RANGEIGNORE
 THRESHOLD, 73
 VL53L0X_DEVICEERROR_RANGEPHASEC
 HECK, 73
 VL53L0X_DEVICEERROR_SIGMATHRESH
 OLDHECK, 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_WARNI
 NG, 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_CLEA
 RED, 60
 VL53L0X_ERROR_INVALID_COMMAND,
 60
 VL53L0X_ERROR_INVALID_PARAMS, 59
 VL53L0X_ERROR_MIN_CLIPPED, 59
 VL53L0X_ERROR_MODE_NOT_SUPPORTE
 D, 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_GETDEVICESPECIFICPARAMET
 ER, 70
 VL53L0X_GETPARAMETERFIELD, 69
 VL53L0X_MAKEUINT16, 71
 VL53L0X_SETARRAYPARAMETERFIELD,
 69
 VL53L0X_SETDEVICESPECIFICPARAMET
 ER, 70
 VL53L0X_SETPARAMETERFIELD, 69
 get_sequence_step_timeout
 vl53l0x_api_core.h, 109
 Gpio Functionality, 75
 VL53L0X_GpioFunctionality, 76
 VL53L0X_GPIOFUNCTIONALITY_NEW_M
 EASURE_READY, 76
 VL53L0X_GPIOFUNCTIONALITY_OFF, 75
 VL53L0X_GPIOFUNCTIONALITY_THRESH
 OLD_CROSSED_HIGH, 75
 VL53L0X_GPIOFUNCTIONALITY_THRESH
 OLD_CROSSED_LOW, 75
 VL53L0X_GPIOFUNCTIONALITY_THRESH
 OLD_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
 - VL53L0X Platform Functions, 7
- PALDevDataSet
 - VL53L0X Platform Functions, 7
- PalState
 - VL53L0X_DevData_t, 87
- Part2PartOffsetAdjustmentNVMMicroMeter
 - VL53L0X_DevData_t, 86
- Part2PartOffsetNVMMicroMeter
 - 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
- PreRangeVcselPulsePeriod
 - 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
 VL53L0X_DeviceSpecificParameters_t, 92
 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, 135
 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
 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_MAJO
 R, 58
 VL53L0X_IMPLEMENTATION_VER_MINO
 R, 58
 VL53L0X_IMPLEMENTATION_VER_REVIS
 ION, 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_MAJ
 OR, 57
 VL53L0X10_IMPLEMENTATION_VER_MIN
 OR, 57
 VL53L0X10_IMPLEMENTATION_VER_REV
 ISION, 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 Platform Functions, 6
 - PALDevDataGet, 7
 - PALDevDataSet, 7
 - VL53L0X_DEV, 7
 - VL53L0X_PollingDelay, 7
- 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_NOVH_VVALUEFOUND, 115
 VL53L0X_STRING_DEVICEERROR_PHASE_CONSISTENCY, 116
 VL53L0X_STRING_DEVICEERROR_RANGE_COMPLETE, 116
 VL53L0X_STRING_DEVICEERROR_RANGE_IGNORETHRESHOLD, 116
 VL53L0X_STRING_DEVICEERROR_RANGE_PHASECHECK, 115
 VL53L0X_STRING_DEVICEERROR_SIGMA_THRESHOLD_CHECK, 116
 VL53L0X_STRING_DEVICEERROR_SNR_CHECK, 115
 VL53L0X_STRING_DEVICEERROR_TCC, 116
 VL53L0X_STRING_DEVICEERROR_UNKNOWN, 116
 VL53L0X_STRING_DEVICEERROR_VCSEL_CONTINUITYTESTFAILURE, 115
 VL53L0X_STRING_DEVICEERROR_VCSEL_WATCHDOGTESTFAILURE, 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_INVALID, 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_PRE_RANGE, 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_mclks
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_PRE_RANGE
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, 125

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
VL53L0X 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_NOVHVVALUEFOUND
 Device Error, 72
 VL53L0X_DEVICEERROR_PHASECONSISTENCY
 Device Error, 73
 VL53L0X_DEVICEERROR_RANGECOMPLETE
 Device Error, 73
 VL53L0X_DEVICEERROR_RANGEIGNORETHRESHOLD
 Device Error, 73
 VL53L0X_DEVICEERROR_RANGEPHASECHECK
 Device Error, 73
 VL53L0X_DEVICEERROR_SIGMATHRESHOLDCHECK
 Device Error, 73
 VL53L0X_DEVICEERROR_SNRCHECK
 Device Error, 73
 VL53L0X_DEVICEERROR_TCC
 Device Error, 73
 VL53L0X_DEVICEERROR_VCSELCONTINUITYTESTFAILURE
 Device Error, 72
 VL53L0X_DEVICEERROR_VCSELWATCHDOGTESTFAILURE
 Device Error, 72
 VL53L0X_DeviceInfo_t, 88
 Name, 88
 ProductId, 89
 ProductRevisionMajor, 89
 ProductRevisionMinor, 89
 ProductType, 89
 Type, 88
 VL53L0X_DEVICEMODE_CONTINUOUS_RANGING
 Defines Device modes, 61
 VL53L0X_DEVICEMODE_CONTINUOUS_TIMED_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_HISTOGRAM
 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
 FinalRangeVcslPulsePeriod, 92
 LastEncodedTimeout, 92
 ModuleId, 93
 OscFrequencyMHz, 92
 PartUIDLower, 93
 PartUIDUpper, 93
 Pin0GpioFunctionality, 92
 PreRangeTimeoutMicroSecs, 92
 PreRangeVcslPulsePeriod, 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_microseconds
 vl53l0x_api_core.h, 109
 VL53L0X_get_offset_calibration_data_micro_meters
 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_GetMaxNumberOfROIzones
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_GetNumberOfROIzones
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_GetVcselPulsePeriod
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
 VL53L0X_comms_close, 125
 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, 129
 VL53L0X_set_gpio, 129
 VL53L0X_wait_ms, 129
 VL53L0X_write_byte, 126
 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, 130
 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
 VL53L0X 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_VCS_EL_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_READOUT_CTRL	Define Registers, 82
VL53L0X_REG_I2C_SLAVE_DEVICE_ADDRESS	Define Registers, 80
VL53L0X_REG_IDENTIFICATION_MODEL_ID	Define Registers, 82
VL53L0X_REG_IDENTIFICATION_REVISION_ID	Define Registers, 82
VL53L0X_REG_MSRC_CONFIG_CONTROL	Define Registers, 80
VL53L0X_REG_MSRC_CONFIG_TIMEOUT_MACROP	Define Registers, 82
VL53L0X_REG_OSC_CALIBRATE_VAL	Define Registers, 82
VL53L0X_REG_POWER_MANAGEMENT_GO1_POWER_FORCE	Define Registers, 83
VL53L0X_REG_PRE_RANGE_CONFIG_MIN_SNR	Define Registers, 80
VL53L0X_REG_PRE_RANGE_CONFIG_SIGMA_THRESH_HI	Define Registers, 81
VL53L0X_REG_PRE_RANGE_CONFIG_SIGMA_THRESH_LO	Define Registers, 81
VL53L0X_REG_PRE_RANGE_CONFIG_TIMEOUT_MACROP_HI	Define Registers, 81
VL53L0X_REG_PRE_RANGE_CONFIG_TIMEOUT_MACROP_LO	Define Registers, 81
VL53L0X_REG_PRE_RANGE_CONFIG_VALID_PHASE_HIGH	Define Registers, 81
VL53L0X_REG_PRE_RANGE_CONFIG_VALID_PHASE_LOW	Define Registers, 81
VL53L0X_REG_PRE_RANGE_CONFIG_VCSEL_PERIOD	Define Registers, 81
VL53L0X_REG_PRE_RANGE_MIN_COUNT_RATE_RTN_LIMIT	Define Registers, 81
VL53L0X_REG_RESULT_CORE_AMBIENT_WINDOW_EVENTS_REF	Define Registers, 80
VL53L0X_REG_RESULT_CORE_AMBIENT_WINDOW_EVENTS_RTN	Define Registers, 80
VL53L0X_REG_RESULT_CORE_PAGE	Define Registers, 80
VL53L0X_REG_RESULT_CORE_RANGING_TOTAL_EVENTS_REF	Define Registers, 80
VL53L0X_REG_RESULT_CORE_RANGING_TOTAL_EVENTS_RTN	Define Registers, 80
VL53L0X_REG_RESULT_INTERRUPT_STATUS	Define Registers, 80
VL53L0X_REG_RESULT_PEAK_SIGNAL_RATE_REF	Define Registers, 80
VL53L0X_REG_RESULT_RANGE_STATUS	Define Registers, 80
VL53L0X_REG_SOFT_RESET_GO2_SOFT_RESET_N	Define Registers, 82
VL53L0X_REG_SYSRANGE_MODE_BACKTOBACK	Define Registers, 78
VL53L0X_REG_SYSRANGE_MODE_HISTOGRAM	Define Registers, 78
VL53L0X_REG_SYSRANGE_MODE_MASK	Define Registers, 78
VL53L0X_REG_SYSRANGE_MODE_SINGLESHOT	Define Registers, 78
VL53L0X_REG_SYSRANGE_MODE_START_STOP	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	Define Registers, 79
VL53L0X_REG_SYSTEM_INTERRUPT_CLEAR	Define Registers, 79
VL53L0X_REG_SYSTEM_INTERRUPT_CONFIG_GPIO	Define Registers, 79

Define Registers, 79

VL53L0X_REG_SYSTEM_INTERRUPT_GPIO_DISABLED

Define Registers, 79

VL53L0X_REG_SYSTEM_INTERRUPT_GPIO_LEVEL_HIGH

Define Registers, 79

VL53L0X_REG_SYSTEM_INTERRUPT_GPIO_LEVEL_LOW

Define Registers, 79

VL53L0X_REG_SYSTEM_INTERRUPT_GPIO_NEW_SAMPLE_READY

Define Registers, 79

VL53L0X_REG_SYSTEM_INTERRUPT_GPIO_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_SDA__EXTSUP_HV

Define Registers, 84

VL53L0X_release_gpio

vl53l0x_i2c_platform.h, 129

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_SetSpadAmbientDampFactor

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_RANGEPHASECHECK
 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_UNKNOW
 N
 vl53l0x_api_strings.h, 116
 VL53L0X_STRING_DEVICEERROR_VCSEL
 CONTINUITYTESTFAILURE
 vl53l0x_api_strings.h, 115
 VL53L0X_STRING_DEVICEERROR_VCSELW
 ATCHDOGTESTFAILURE
 vl53l0x_api_strings.h, 115
 VL53L0X_STRING_ERROR_BUFFER_TOO_S
 MALL
 vl53l0x_api_strings.h, 113
 VL53L0X_STRING_ERROR_CALIBRATION_
 WARNING
 vl53l0x_api_strings.h, 112
 VL53L0X_STRING_ERROR_CONTROL_INTE
 RFACE
 vl53l0x_api_strings.h, 113
 VL53L0X_STRING_ERROR_DIVISION_BY_ZE
 RO
 vl53l0x_api_strings.h, 113
 VL53L0X_STRING_ERROR_GPIO_FUNCTION
 ALITY_NOT_SUPPORTED
 vl53l0x_api_strings.h, 113
 VL53L0X_STRING_ERROR_GPIO_NOT_EXIS
 TING
 vl53l0x_api_strings.h, 113
 VL53L0X_STRING_ERROR_INTERRUPT_NOT_
 CLEARED
 vl53l0x_api_strings.h, 113
 VL53L0X_STRING_ERROR_INVALID_COMM
 AND
 vl53l0x_api_strings.h, 113
 VL53L0X_STRING_ERROR_INVALID_PARA
 MS
 vl53l0x_api_strings.h, 112
 VL53L0X_STRING_ERROR_MIN_CLIPPED
 vl53l0x_api_strings.h, 112
 VL53L0X_STRING_ERROR_MODE_NOT_SUP
 PORTED
 vl53l0x_api_strings.h, 113
 VL53L0X_STRING_ERROR_NONE
 vl53l0x_api_strings.h, 112
 VL53L0X_STRING_ERROR_NOT_IMPLEMEN
 TED
 vl53l0x_api_strings.h, 114
 VL53L0X_STRING_ERROR_NOT_SUPPORTE
 D
 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_MINRAN
 GE
 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_RANGEV
 ALID
 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_RA
 NGE
 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_STATICINI
 T
 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, 135
 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
 Vcsl 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
 vl53l0x_i2c_platform.h, 126
 VL53L0X_write_dword
 vl53l0x_i2c_platform.h, 127
 VL53L0X_write_multi
 vl53l0x_i2c_platform.h, 126
 VL53L0X_write_word
 vl53l0x_i2c_platform.h, 127
 VL53L0X_WriteMulti
 PAL Register Access Functions, 9
 VL53L0X_WrWord
 PAL Register Access Functions, 10
 VL53L0X10_IMPLEMENTATION_VER_MAJOR
 VL53L0X Defines, 57
 VL53L0X10_IMPLEMENTATION_VER_MINOR
 VL53L0X Defines, 57
 VL53L0X10_IMPLEMENTATION_VER_REVISION
 VL53L0X Defines, 57
 VL53L0X10_IMPLEMENTATION_VER_SUB
 VL53L0X Defines, 57
 VL53L0X10_SPECIFICATION_VER_MAJOR
 VL53L0X Defines, 56
 VL53L0X10_SPECIFICATION_VER_MINOR
 VL53L0X Defines, 56
 VL53L0X10_SPECIFICATION_VER_REVISION
 VL53L0X Defines, 57
 VL53L0X10_SPECIFICATION_VER_SUB
 VL53L0X Defines, 57
 WrapAroundCheckEnable
 VL53L0X_DeviceParameters_t, 91
 XTalkCompensationEnable
 VL53L0X_DeviceParameters_t, 90
 XTalkCompensationRangeMilliMeter
 VL53L0X_DeviceParameters_t, 90
 XTalkCompensationRateMegaCps
 VL53L0X_DeviceParameters_t, 90
 ZoneId
 VL53L0X_RangingMeasurementData_t, 97