LibraryLibLidarLms151

Generated by Doxygen 1.8.6

Mon Feb 1 2016 14:24:44

Contents

1	File	Index			1
	1.1	File Lis	t		1
2	File	Docume	entation		3
	2.1	Library	_LibLidarL	ms151.c File Reference	3
		2.1.1	Detailed I	Description	4
		2.1.2	Macro De	efinition Documentation	4
			2.1.2.1	K_BUFFER_SIZE_MAX	4
		2.1.3	Enumera	tion Type Documentation	4
			2.1.3.1	te_ContentOutChannel	4
			2.1.3.2	te_ReadStatus	5
			2.1.3.3	te_subState	5
		2.1.4	Function	Documentation	5
			2.1.4.1	LibLidarLms151_Close	5
			2.1.4.2	LibLidarLms151_Configure	5
			2.1.4.3	LibLidarLms151_Get_InternalState	5
			2.1.4.4	LibLidarLms151_Get_Mode	6
			2.1.4.5	LibLidarLms151_Get_Scan	6
			2.1.4.6	LibLidarLms151_Get_Status	6
			2.1.4.7	LibLidarLms151_Get_Timestamp	7
			2.1.4.8	LibLidarLms151_Initialize	7
			2.1.4.9	LibLidarLms151_Manage	7
			2.1.4.10	LibLidarLms151_Open	9
Inc	lex				11

Chapter 1

File Index

	_			
1 1	F	ile	Шi	iet

Here is a list of all documented files with brief descriptions:	
Library_LibLidarLms151.c	
Fichier Source du Service LibLidarLms151	3

2 File Index

Chapter 2

File Documentation

2.1 Library_LibLidarLms151.c File Reference

Fichier Source du Service LibLidarLms151.

```
#include "types.h"
#include "Library_StdLib.h"
#include "Library_LibLidarLms151.h"
#include "Driver_TcpSocketDriver.h"
#include <string.h>
#include <stdlib.h>
#include <stdio.h>
#include <errno.h>
Include dependency graph for Library_LibLidarLms151.c:
```



Macros

- #define K_BUFFER_SIZE_MAX 262144ul /*! buffer maximal size value */
- #define K_START 1 /*! Start value */
- #define K_AMOUNT_OF_ENCODERS_MAX 3 /*! Maximum amount of encoders */
- #define K_AMOUNT_OF_16BITS_CHANNELS_MAX 2 /*! Maximum amount of 16bits channels */
- #define K_AMOUNT_OF_8BITS_CHANNELS_MAX 2 /*! Maximum amount of 8bits channels */

Enumerations

- enum te_command_status { E_COMMAND_ERROR =0, E_COMMAND_SUCCESS }
 define the command status
- enum te_start_status { E_START_NO_ERROR =0, E_START_NOT_ALLOWED }
 define the start status
- enum te_subState { $E_IDLE = 0$, E_READ , E_ERROR }
 - define the mode substate
- enum te_ReadStatus { E_READ_BUSY =0, E_READ_FINISH, E_READ_NO_DATA }

define the read status

• enum te_ContentOutChannel { E_OUT_DIST1 = 0, E_OUT_DIST2, E_OUT_RSSI1, E_OUT_RSSI2 }

define the lidar Content Out channel

Functions

Std ReturnType LibLidarLms151 Initialize (void)

this function initialize module

- Std_ReturnType LibLidarLms151_Configure (const ts_LibLidarLms151_Configuration *const ps_LibLidarLms151_Configuration)
- Std ReturnType LibLidarLms151 Manage (void)

this function is the central point of the library, it manage the laser

• Std_ReturnType LibLidarLms151_Open (void)

this function will open Tcp Socket needed for communication with Lms151

• Std ReturnType LibLidarLms151 Close (void)

this function will close Tcp Socket needed for communication with Lms151

Std_ReturnType LibLidarLms151_Get_Timestamp (uint32 *const pu32_Timestamp)

this function will return Laser data Timestamp

Std_ReturnType LibLidarLms151_Get_Scan (ts_LibLidarLms151_scan_data *const ps_LibLidarLms151_scan_data)

this function will return scan values

Std_ReturnType LibLidarLms151_Get_InternalState (te_InternalStateMachine *const pe_State)

this function will return internal state

Std_ReturnType LibLidarLms151_Get_Status (te_LibLidarLms151_device_status *const pe_LibLidarLms151_device_status)

this function will return laser status

Std_ReturnType LibLidarLms151_Get_Mode (te_mode *const pe_Mode)

this function will return global mode

2.1.1 Detailed Description

Fichier Source du Service LibLidarLms151. \$Source: Library LibLidarLms151.c \$Author: Slo \$Date: 2016/01/14

2.1.2 Macro Definition Documentation

2.1.2.1 #define K BUFFER SIZE MAX 262144ul /*! buffer maximal size value */

Global constant Reserved value "1"

2.1.3 Enumeration Type Documentation

2.1.3.1 enum te_ContentOutChannel

define the lidar Content Out channel

Enumerator

E_OUT_DIST2 Content channel is DIST1

E_OUT_RSSI1 Content channel is DIST2

E_OUT_RSSI2 Content channel is RSSI1

Content channel is RSSI2

2.1.3.2 enum te_ReadStatus

define the read status

Enumerator

E_READ_NO_DATA Read finished

No data read

2.1.3.3 enum te subState

define the mode substate

Enumerator

E_READ Mode is in IDLE state

E_ERROR Mode is actually in Read state

Mode is in Error state

2.1.4 Function Documentation

2.1.4.1 Std_ReturnType LibLidarLms151_Close (void)

this function will close Tcp Socket needed for communication with Lms151

Parameters

none

Returns

Std_ReturnType:

- E_NOT_OK if module is not initialized or Tcp socket return an error
- else E_OK

Tcp Socket Close Error, t_Close_Status is set to E_NOT_OK

no Tcp Socket Close Error, t_Close_Status is set to E_OK

Module is not initialized, Halt

2.1.4.2 Std_ReturnType LibLidarLms151_Configure (const ts_LibLidarLms151_Configuration *const ps_LibLidarLms151_Configuration)

Set Laser configuration

Set data configuration

Set socket configuration

Module is not initialized, Halt

2.1.4.3 Std_ReturnType LibLidarLms151_Get_InternalState (te_InternalStateMachine *const pe_State)

this function will return internal state

Parameters

te_InternalState-	*pe_State : state parameter
Machine	

Returns

Std_ReturnType:

- E_NOT_OK if pe_State is null or module is not initialized
- else E OK

Module is not initialized, Halt

2.1.4.4 Std_ReturnType LibLidarLms151_Get_Mode (te_mode *const pe_Mode)

this function will return global mode

Parameters

te_mode	*pe_Mode : mode parameter
---------	---------------------------

Returns

Std_ReturnType:

- E_NOT_OK if pe_Mode is null or module is not initialized
- else E_OK

Module is not initialized, Halt

2.1.4.5 Std_ReturnType LibLidarLms151_Get_Scan (ts_LibLidarLms151_scan_data *const ps_LibLidarLms151_scan_data)

this function will return scan values

Parameters

ts_LibLidar-	*ps_LibLidarLms151_scan_data : scan parameter
Lms151_scan	
data	

Returns

Std_ReturnType:

- E_NOT_OK if ps_LibLidarLms151_scan_data is null or module is not initialized
- else E_OK

Module is not initialized, Halt

2.1.4.6 Std_ReturnType LibLidarLms151_Get_Status (te_LibLidarLms151_device_status *const pe_LibLidarLms151_device_status)

this function will return laser status

Parameters

te_LibLidar-	*pe_LibLidarLms151_device_status : device status parameter
Lms151_device-	
_status	

Returns

Std_ReturnType:

- E_NOT_OK if pe_LibLidarLms151_device_status is null or module is not initialized
- else E OK

Module is not initialized, Halt

2.1.4.7 Std_ReturnType LibLidarLms151_Get_Timestamp (uint32 *const pu32_Timestamp)

this function will return Laser data Timestamp

Parameters

uint32	? *pu32_Timestamp : timestamp parameter
--------	---

Returns

Std ReturnType:

- E_NOT_OK if pu32_Timestamp is null or module is not initialized
- else E OK

Module is not initialized, Halt

2.1.4.8 Std_ReturnType LibLidarLms151_Initialize (void)

this function initialize module

Parameters

none

Returns

Std_ReturnType:

- E_OK if SrvConf_Get_Lms151TcpSocket, LibLidarLms151_Open and SrvConf_Get_Lms151Output-Mode return functions are E_OK
- E NOT OK if not

2.1.4.9 Std_ReturnType LibLidarLms151_Manage (void)

this function is the central point of the library, it manage the laser

Parameters

none	
none	

Returns

Std_ReturnType:

- E_OK if good return function
- E_NOT_OK if not

Check Module is Initialized and configured

Module Initialized, it has to be configured

set ge_InternalStateMachine to E_LIBLMS151_STATUS_CONFIGURATION

set ge_Mode to E_MODE_LOGIN

set ge_SubState to E_IDLE

log into laser for parameters modifications

Check login command has been well sent

if check is ok, set ge_mode to E_MODE_SET_FREQUENCY_AND_RESOLUTION and ge_SubState to E_IDLE

Configure laser Frequency and resolution

Check laser frequency and resolution configuration well done

if check is ok, set ge mode to E MODE CONFIGURE SCAN DATA CONTENT and ge SubState to E IDLE

Configure scan data content

Check scan data content configuration has been well done

if check is ok, set ge_mode to E_MODE_CONFIGURE_SCAN_DATA_OUTPUT and ge_SubState to E_IDLE configure laser scan data output

check scan data output laser command has been well sent

if check is ok, set ge_mode to E_MODE_STORE_PARAMETERS and ge_SubState to E_IDLE

Store parameters in laser eeprom

if check is ok, set ge_mode to E_MODE_START_MEAS and ge_SubState to E_IDLE

Start Measurements

Check start measurements command has been well sent

if check is ok, set ge $_$ mode to E $_$ MODE $_$ ASK $_$ STATUS and ge $_$ SubState to E $_$ IDLE and ge $_$ InternalStateMachine to E $_$ READY

Ask laser state

Check laser ask state command has been well sent

if check is ok, set ge_mode to E_MODE_LOGOUT and ge_SubState to E_IDLE

log out to run laser

Check log out command has been well sent

if check is ok, set ge $_$ mode to E $_$ MODE $_$ SCAN $_$ CONTINUOUS and ge $_$ SubState to E $_$ IDLE and ge $_$ InternalState-Machine to E $_$ STATE $_$ SCAN

Configure permanent scan

if check is ok, set ge_mode to E_MODE_GET_DATA and ge_SubState to E_IDLE and gu32_FramePosition to 0

Receive data from laser

Module is not initialized, Halt

2.1.4.10 Std_ReturnType LibLidarLms151_Open (void)

this function will open Tcp Socket needed for communication with Lms151

Parameters

none

Returns

Std_ReturnType:

- E_NOT_OK if module is not initialized or Tcp socket return an error
- else E_OK

Tcp Socket Open Error, t_Open_Status is set to E_NOT_OK no Tcp Socket Open Error, t_Open_Status is set to E_OK Module is not initialized, Halt

Index

E_ERROR
Library_LibLidarLms151.c, 5
E OUT DIST2
Library_LibLidarLms151.c, 4
E OUT RSSI1
Library_LibLidarLms151.c, 4
•
E_OUT_RSSI2
Library_LibLidarLms151.c, 4
E_READ
Library_LibLidarLms151.c, 5
E_READ_FINISH
Library_LibLidarLms151.c, 5
E READ NO DATA
Library_LibLidarLms151.c, 5
171111 1 454 01
LibLidarLms151_Close
Library_LibLidarLms151.c, 5
LibLidarLms151_Configure
Library_LibLidarLms151.c, 5
LibLidarLms151_Get_InternalState
Library_LibLidarLms151.c, 5
LibLidarLms151_Get_Mode
Library_LibLidarLms151.c, 6
LibLidarLms151_Get_Scan
Library_LibLidarLms151.c, 6
LibLidarLms151_Get_Status
Library_LibLidarLms151.c, 6
LibLidarLms151_Get_Timestamp
Library_LibLidarLms151.c, 7
LibLidarLms151_Initialize
Library_LibLidarLms151.c, 7
LibLidarLms151_Manage
Library_LibLidarLms151.c, 7
LibLidarLms151_Open
Library_LibLidarLms151.c, 8
Library_LibLidarLms151.c
E_ERROR, 5
E_OUT_DIST2, 4
E_OUT_RSSI1, 4
E_OUT_RSSI2, 4
E_READ, 5
E_READ_FINISH, 5
E_READ_NO_DATA, 5
Library_LibLidarLms151.c, 3
LibLidarLms151_Close, 5
LibLidarLms151_Configure, 5
LibLidarLms151_Get_InternalState, 5
LibLidarLms151_Get_Mode, 6
LibLidarLms151_Get_Scan, 6
LibLidarLms151_Get_Status, 6

```
LibLidarLms151_Get_Timestamp, 7
LibLidarLms151_Initialize, 7
LibLidarLms151_Manage, 7
LibLidarLms151_Open, 8
te_ContentOutChannel, 4
te_ReadStatus, 4
te_subState, 5

te_ContentOutChannel
Library_LibLidarLms151.c, 4
te_ReadStatus
Library_LibLidarLms151.c, 4
te_subState
Library_LibLidarLms151.c, 5
```