debian

User Guide for the Library “LibIOBOARD” for Debian

The Library offers high-level and convenient APIs for the analog and digital input/output boards Application development

Revision	Datum	Autor	Modifikation / Kommentar
1.0	10.01.2018	S. Mutter	First Release
1.1	09.10.2020	S. Mutter	Added new Library Version and example for Board IPC/AI4IX-xxxE

Content

1 Introduction **3**

1.1 Overview 3

1.2 Scope 4

2 Connection of I/O Boards **5**

3 Getting started **6**

Example for one connected Digital I/O Board 6

Example for one connected Analog I/O Board 8

4 Application Programming Interface (API) **11**

4.1 IOBOARD general functions 11

4.1.1 IOBOARD_Init 11

4.1.2 IOBOARD_Select 11

4.1.3 IOBOARD_PrintInfo 12

4.2 Digital I/O Board Functions 13

4.2.1 IOBOARD_SetOutputs 13

4.2.2 IOBOARD_ReadOutputs 13

4.2.3 IOBOARD_ReadInputs 14

4.3 Analog I/O Board Functions 15

4.3.1 IOBOARD_ReadAnalogInput 15

4.3.2 IOBOARD_SetAnalogInputRange 15

4.3.3 IOBOARD_ReadTempSensor 16

4.4 IOBOARD_STATUS 17

Contact Information / Disclaimer **19**

1 Introduction

1.1 Overview

The I/O Boards of Syslogic are full USB devices which can be used through standard USB HID interfaces. Most operating systems natively support the USB HID class. In Linux, it is not required to install a custom driver for the boards.

The USB HID class exchanges data between a host and a device by reports. There are three types of reports in USB HID:

1. Feature report: Configuration data are exchanged between the host and the HID device through a control pipe. The feature report is usually used to turn on/off device functions.
2. Input report: Data content that is sent from the HID device to the host.
3. Output report: Data content that is sent from the host to the HID device.

The I/O Boards receives output reports from the HID application, decodes the request and configures, sets, and reads the values of the I/Os accordingly. Data requested of the I/Os is sent to the host by input reports.

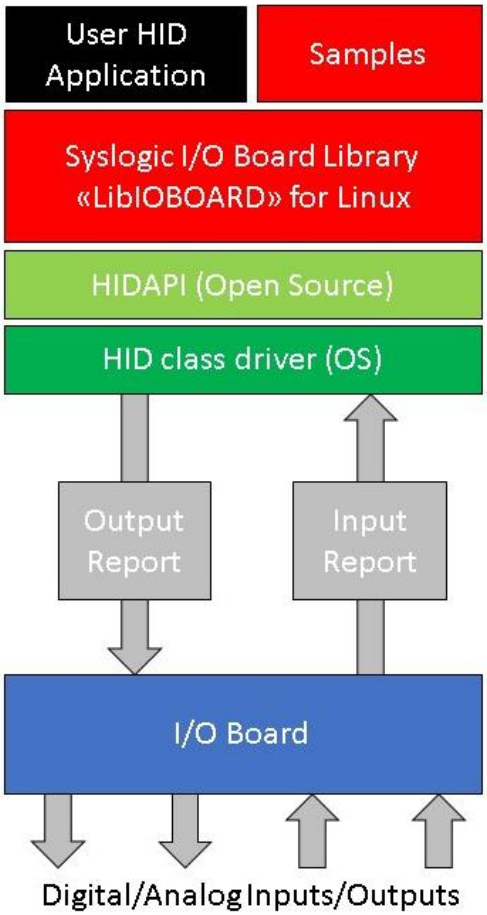

Fig. 1 The I/O Board System Block Diagram

1.2 Scope

This guide is intended for developers who are creating applications, extending Syslogic provided applications or implementing Syslogic's applications for the I/O Boards.
The support library LibIOBOARD hides the details of communicating by HID protocol with the I/O Board and provides simple APIs for developers to create their own applications. The library makes use of HIDAPI, an open source Library for communication with hid devices as the FT260 of FTDI. The device FT260 is part of the I/O Boards. For Application development, the LibIOBOARD and the HIDAPI libraries have to be included into the project.
The sample source code contained in this application note is provided as an example and is neither guaranteed nor supported by Syslogic.

2 Connection of I/O Boards

The different I/O boards can be connected directly over the USB connector, but because the peripherals on the I/O boards are accessed over a local I2C bus, it is possible to connect other I/O Boards over an onboard I2C connector. A possible connecting scheme is shown in Fig.

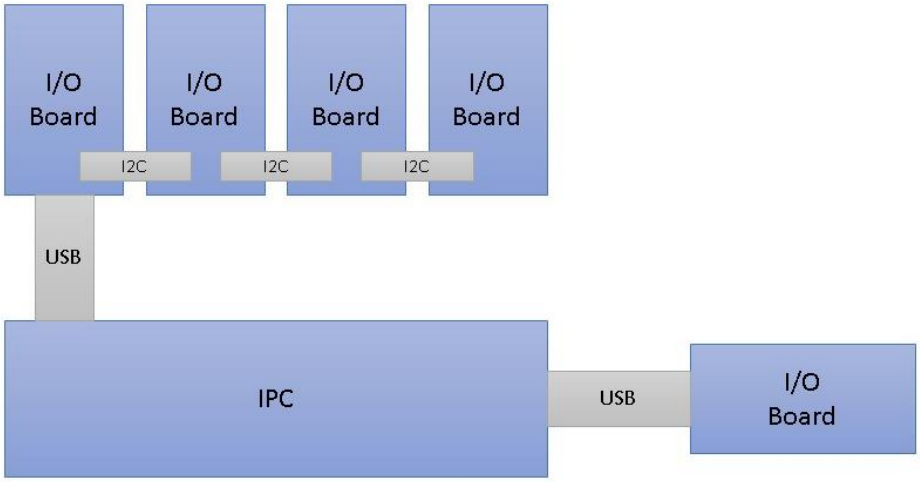

Fig. 2 Possible connections of I/O Boards to an industrial PC of Syslogic

It is possible to connect up to two I/O boards to an IPC over direct connection to an internal USB connector. If more I/O boards are needed, up to 4 boards can be connected to one USB connection by cascading the boards on the local I2C connector on the boards. With this setup, it is possible to use up to 8 I/O boards with the same industrial PC of Syslogic. Different types and versions of I/O boards can be mixed.

3 Getting started

The Library and examples can be downloaded from the Syslogic webpage. A description of the files in the directory is in the README.txt.
This is an example which shows how to initialize a digital I/O Board with the LibIOBOARD support library. After initializing the device, the functions to control the I/O Board are ready to use. Different I/O board types require different configurations and offer different kind of functions. For more details, refer to chapter 4.

Example for one connected Digital I/O Board

```
#include "LibIOBOARD.h"
#include <unistd.h>

int main(int argc, char const* argv[])
{
	// only one I/O Board connected
	uint8 numIOBoards[1] = { 1 };
	// transfer speed of the I2C communication
	uint32 kbps = 400;

	unsigned char value = 0x00, realvalue = 0x00;
	IOBOARD_STATUS ioboard_status;

	// initialize the IOBOARD
	ioboard_status = IOBOARD_Init(kbps, numIOBoards);

	if (ioboard_status != IOBOARD_OK)
	{
		printf("Initialization of IO Board failed");
		return 0;
	}
	else
	{
		printf("Initialization of IO Board OK");
	}

	// Print Info of connected IO Board to console
	const char* info = IOBOARD_PrintInfo(numIOBoards);
	printf(info);

	// while status is ok, set outputs and read inputs
	// every second and print values to console
	while (ioboard_status == IOBOARD_OK)
	{
		value++;
		// set outputs of board 0
		ioboard_status = IOBOARD_SetOutputs(0, value);
		// read inputs of board 0
		ioboard_status = IOBOARD_ReadInputs(0, &realvalue);
		printf("set value: %02x read value: %02x \n", value, realvalue);
		sleep(1);
	}
```

```
        printf("Error occurred");
}
```

Example for one connected Analog I/O Board

```
#include "LibIOBOARD.h"
#include <math.h>
#include <unistd.h>

int main(int argc, char const* argv[])
{
	// only one I/O Board connected
	uint8 numIOBoards[1] = { 1 };
	// transfer speed of the I2C communication
	uint32 kbps = 400;

	// Pointer to the two bytes of the ADC
	unsigned char *value;
	value = (unsigned char *)malloc(2);

	IOBOARD_STATUS ioboard_status;

	// initialize the IOBOARD
	ioboard_status = IOBOARD_Init(kbps, numIOBoards);

	if (ioboard_status != IOBOARD_OK)
	{
		printf("Initialization of IO Board failed");
		return 0;
	}
	else
	{
		printf("Initialization of IO Board OK");
	}

	// Print Info of connected IO Board to console
	const char* info = IOBOARD_PrintInfo(numIOBoards);
	printf(info);

	// Set the range of the Analog Input Channel 0 to the range 0 to 2.5 x 4.096V = 10.24 V
	ioboard_status = IOBOARD_SetAnalogInputRange(0, 0, 5);

	// while status is ok, read the analog input channel 0
	// every second and print value to console
	while (ioboard_status == IOBOARD_OK)
	{
		// read inputs of board 0
		ioboard_status = IOBOARD_ReadAnalogInput(0, 0, value);
		printf("Voltage: %f V \n", (double)(((value[0] << 8) | value[1]) / pow(2,16)*10.24f ) );
		sleep(1);
	}
	printf("Error occurred");
}
```

Example for one connected Analog I/O Board IPC/AI4IX-xxxE – current measure (100 Ohm Resistor)

```
#include "LibIOBOARD.h"
#include <math.h>
#include <unistd.h>

int main(int argc, char const* argv[])
{
	// only one I/O Board connected
	uint8 numIOBoards[1] = { 1 };
	// transfer speed of the I2C communication
	uint32 kbps = 400;

	// Pointer to the two bytes of the ADC
	unsigned char *value;
	value = (unsigned char *)malloc(2);

	IOBOARD_STATUS ioboard_status;

	// initialize the IOBOARD
	ioboard_status = IOBOARD_Init(kbps, numIOBoards);

	if (ioboard_status != IOBOARD_OK)
	{
		printf("Initialization of IO Board failed\n");
		return 0;
	}
	else
	{
		printf("Initialization of IO Board OK\n");
	}

	// Print Info of connected IO Board to console
	const char* info = IOBOARD_PrintInfo(numIOBoards);
	printf(info);

	// Set Range of Input 0 to +-2.56 V / 100 Ohm = +-25.6 mA
	ioboard_status = IOBOARD_SetAnalogInputRange (0, 0, 2);
	// Set Range of Input 1 from 0 to 2.56 V / 100 Ohm = 25.6 mA
	ioboard_status = IOBOARD_SetAnalogInputRange (0, 1, 7);
	// while status is ok, read the analog input channels 0 and 1
	// every second and print value to console
	while (ioboard_status == IOBOARD_OK)
	{
		// read input 0 of board 0
```

```
                ioboard_status = IOBOARD_ReadAnalogInput(0, 0, value);
                // Resolution 2^16, value range 5.12, +- 2.56, 100 Ohm resistor, result in mA
                printf("Current: %f mA \n", (double)(((value[0] << 8) | value[1])/pow(2,16)*5.12f -
2.56f)*10);

                // read input 1 of board 1
                ioboard_status = IOBOARD_ReadAnalogInput(0, 1, value);
                // Resolution 2^16, value range 2.56, 100 Ohm resistor, result in mA
                printf("Current: %f mA \n", (double) (((value[0] << 8) | value[1]) /pow(2,16)*2.56f)*10);

                sleep(1);
        }
        printf("Error occurred");
}
```

4 Application Programming Interface (API)

LibIOBOARD supports the communication with analog and digital I/O boards by using high-level APIs. The initialization function sets up the IO boards for the subsequent operations.

4.1 IOBOARD general functions

The funtions listed in this section can be used with every I/O board by Syslogic.

4.1.1 IOBOARD_Init

IOBOARD_STATUS IOBOARD_Init(uint32 kbps, uint8* numIOBoards)

Summary:

Initializes the connected I/O Boards.
Note: The user must know how many boards are connected to each USB connection

Parameters

Name	Description	Minimal Value	Maximal Value
kbps	Speed of the I2C transfer	60	400
numIOBoards	Array of connected boards to each USB connection	1	4

Return Value

IOBOARD_OK if successful, otherwise the return value is an error code

4.1.2 IOBOARD_Select

IOBOARD_STATUS IOBOARD_Select(uint8 FTChip)

Summary:

Function to select the FT260 Chip
Note: If only one IO board is connected to an USB, this function is of no use

Parameters

Name	Description	Minimal Value	Maximal Value
FTChip	The chip to select	0	1

Return Value

IOBOARD_OK if successful, otherwise the return value is an error code

4.1.3 IOBOARD_PrintInfo

const char* IOBOARD_PrintInfo(uint8* numIOBoards)

Summary:

Prints Information about the connected boards to a string
Note: The description of the information contained in the string can be seen in Tab.1.

Parameters

Name	Description	Minimal Value	Maximal Value
numIOBoards	Array of connected boards to each USB connection	1	4

Return Value

Constant string containing the information of all connected boards or containing "ERROR"

Name	Description
General information about the Library	
Driver Version:	The version number of the library
For each USB connection	
Device Index:	The index of the I/O Board (HID device)
For each connected board	
IOBOARD Number:	The number associated to the specific board
FID:	Function ID of the board
OID:	Option ID of the board
RID:	Revision ID of the board
Serial Number:	Serial Number of the board
Type:	Type of the board, digital or analog
Number of Inputs:	The number of inputs that the board has
Number of Outputs:	The number of outputs that the board has

Tab. 1 Information provided by the Function about the library and the boards

4.2 Digital I/O Board Functions

The digital I/O board provides functions to read and set its outputs and to read its inputs.

4.2.1 IOBOARD_SetOutputs

IOBOARD_STATUS IOBOARD_SetOutputs(uint8 boardNumber, unsigned char value)

Summary:

Function to set the outputs of the digital I/O board

Parameters

Name	Description	Minimal Value	Maximal Value
boardNumber	The number of the board to set the outputs	0	Boards on USB -1
value	The value with which the outputs are set	0	255

Return Value

IOBOARD_OK if successful, otherwise the return value is an error code

4.2.2 IOBOARD_ReadOutputs

IOBOARD_STATUS IOBOARD_ReadOutputs(uint8 boardNumber, unsigned char* value)

Summary:

Function to read the outputs of the digital I/O board

Parameters

Name	Description	Minimal Value	Maximal Value
boardNumber	The number of the board to read the outputs	0	Boards on USB -1
value	Pointer to the buffer that receives the value read of the outputs	0	255

Return Value

IOBOARD_OK if successful, otherwise the return value is an error code

4.2.3 IOBOARD_ReadInputs

IOBOARD_STATUS IOBOARD_ReadInputs(uint8 boardNumber, unsigned char* value)

Summary:

Function to read the inputs of the digital I/O board

Parameters

Name	Description	Minimal Value	Maximal Value
boardNumber	The number of the board to read the inputs	0	Boards on USB -1
value	Pointer to the buffer that receives the value read of the inputs	0	255

Return Value

IOBOARD_OK if successful, otherwise the return value is an error code

4.3 Analog I/O Board Functions

4.3.1 IOBOARD_ReadAnalogInput

IOBOARD_STATUS IOBOARD_ReadAnalogInput(uint8 boardNumber, uint8 channelNumber unsigned char* value)

Summary:

Function to read an input of the analog I/O Board

Parameters

Name	Description	Minimal Value	Maximal Value
boardNumber	The number of the board to read an analog Input	0	Boards on USB -1
channelNumber	The channel which is sampled	0	3
value	Pointer to the buffer of size 2 that receives the value read of the analog input	0	2^{16}

Return Value

IOBOARD_OK if successful, otherwise the return value is an error code

4.3.2 IOBOARD_SetAnalogInputRange

IOBOARD_STATUS IOBOARD_SetAnalogInputRange(uint8 boardNumber, uint8 channelNumber unsigned char* range)

Summary:

Function to set the range of an input of the analog I/O Board
Note: For the possible ranges, see Tab. 2.

Parameters

Name	Description	Minimal Value	Maximal Value
boardNumber	The number of the board to set the range of an analog input	0	Boards on USB -1
channelNumber	The channel to set to a range	0	3
range	Range in which the channel is set to	0	15

Return Value

IOBOARD_OK if successful, otherwise the return value is an error code

Value of range	Selected Input range
0	+-2.5 x V_{REF}
1	+-1.25 x V_{REF}
2	+-0.625 x V_{REF}
3	+-0.3125 x V_{REF}
11	+-0.15625 x V_{REF}
5	0 to 2.5 x V_{REF}
6	0 to 1.25 x V_{REF}
7*	0 to 0.625 x V_{REF}
15	0 to 0.3125 x V_{REF}

* default setting for IPC/AI4Ix Boards after initialization

Tab. 2 Value for range to select the Input Range

4.3.3 IOBOARD_ReadTempSensor

IOBOARD_STATUS IOBOARD_ReadTempSensor(uint8 boardNumber, float* temp)

Summary:

Function to read the temperature sensor of the analog I/O Board

Parameters

Name	Description	Minimal Value	Maximal Value
boardNumber	The number of the board to set the range of an analog input	0	Boards on USB -1
temp	Pointer to temperature value as float	-55.0	128.0

Return Value

IOBOARD_OK if successful, otherwise the return value is an error code

4.4 IOBOARD_STATUS

For indication if the device is working, the Functions return a status. The status can have the values listed in Tab. 3.

IOBOARD_STATUS	Description
IOBOARD_OK	Board is working fine
IOBOARD_NOT_RECOGNIZED	The board is not recognized by its FID and OID
IOBOARD_DEVICE_NOT_FOUND	No connected I/O board was found
IOBOARD_EEPROM_READ_ERROR	The read of the EEPROM of the board went wrong
IOBOARD_INVALID_NUMBER_OF_BOARDS	Wrong parameter for the number of boards
IOBOARD_INVALID_CHIP_NUMBER	Wrong parameter for the select function
IOBOARD_SELECTED_CHIP_NOT_FOUND	Selected chip not found
IOBOARD_INVALID_SPEED	Speed for I2C communication not allowed
IOBOARD_SET_OUTPUT_ERROR	Setting output not successful
IOBOARD_READ_OUTPUT_ERROR	Read output not successful
IOBOARD_READ_INPUT_ERROR	Read input not successful
IOBOARD_INIT_OUTPUTS_ERROR	Initialization of outputs not successful
IOBOARD_INIT_INPUTS_ERROR	Initialization of inputs not successful

Tab. 3 Possible values of IOBOARD_STATUS

5 Software Revision History

This paragraph lists the different software revisions of the LibIOBOARD for Debian beginning with the first released versions. Note that Beta-Versions of the Library are not included.

Software	Product Revision	Remarks
LibIOBoard	v1.0	Original Release
LibIOBoard	v1.1	Adapted the Library to read an analog input with two I2C commands (start conversion, read value) instead of three due to speed up of the communication
LibIOBoard	V1.2	Added new Analog Board IPC/AI4Ix-xxxE to Library

Tab. 4 Software Revision State

Contact Information / Disclaimer

Our distributors and system integrators will gladly give you any information about our products and their use. If you want to contact the manufacturer directly, please send a fax or email message containing a short description of your application and your request to the following address or use one of the information or technical support request forms on our internet homepage. Syslogic is grateful for any help referring to errors or suggestions for improvements.

The following registered trademarks/licences are used:

- FTDI Trademark of Future Technology Device International Limited

The content and presentation of this document has been carefully checked. No responsibility is accepted for any errors or omissions in the documentation. Note that this application note is constantly revised and improved. The right to change this documentation at any time without notice is therefore reserved.

Syslogic Datentechnik AG
Täfernstrasse 28
CH-5405 Baden-Dättwil / Switzerland

support@syslogic.com
http://www.syslogic.com
T +41 56 200 90 40
F +41 56 200 90 50