Product Specification

USB 3072
basicLIN 3072

LIN Interfaces

User Manual   Version 1.2

Get the total Coverage!

GOEPEL electronic GmbH
Goeschwitzer Str. 58/60
D-07745 Jena
Phone: +49-3641-6896-597
Fax: +49-3641-6896-944
E-Mail: ats_support@goepel.com
http://www.goepel.com

Printed: 10.06.2010

Issue: June 2010

1 Installation

1.1 Hardware Installation

Generally hardware installation for USB 3072/ basicLIN 3072 means exchanging the transceiver modules.

Warning

Please make absolutely certain that all of the installation procedures described below are carried out with your system **switched off**.

If it is necessary to exchange transceiver modules, the corresponding device is to be opened according to its conditions.

Doing this, pay attention to the general rules to avoid electrostatic charging. Transceiver modules must never be removed or mounted with the power switched on! In addition, the right alignment is absolutely required (see Assembly).

1.2 Driver Installation

For proper installation of the GOEPEL electronic USB drivers on your system, we recommend to execute the GUSB driver setup.
To do that, start the *GUSB-Setup-*.exe* setup program (of the supplied CD, "*" stands for the version number) and follow the instructions.

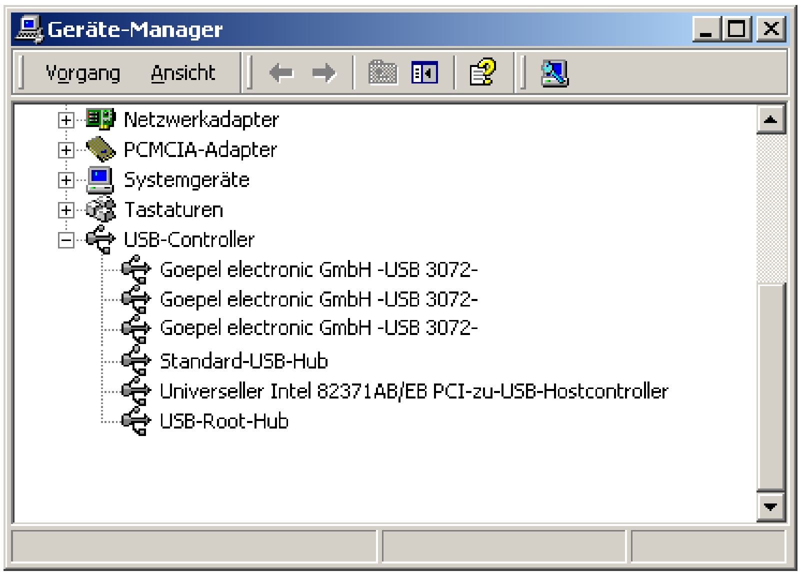
At present, the available device driver only supports Windows® 2000/ XP systems.

If you want to create your own software for the boards, you possibly need additional files for user specific programming (**.LLB*, **.H*).
These files are not automatically copied to the computer and have to be transferred individually from the supplied CD to your development directory.

The USB interface uses the high-speed data rate according to the USB2.0 specification (if possible, otherwise full-speed).

After driver installation, you can check whether the modules are properly embedded by the system.

The following picture shows the successful embedding of one USB 3072/ basicLIN 3072 device with three controllers:



Netzwerkadapter
PCMCIA-Adapter
Systemgeräte
Tastaturen
USB-Controller
Goepel electronic GmbH -USB 3072-
Goepel electronic GmbH -USB 3072-
Goepel electronic GmbH -USB 3072-
Standard-USB-Hub
Universeller Intel 82371AB/EB PCI-zu-USB-Hostcontroller
USB-Root-Hub

Figure 1-1: Display of Device Manager

Please note that the Device Manager shows ALL USB controllers.

2 Hardware

2.1 Definition

USB 3072 LIN boards are GOEPEL electronic GmbH communication boards with USB 2.0 interface.

These boards are used in general control technology, e.g. for applications in automotive technology.

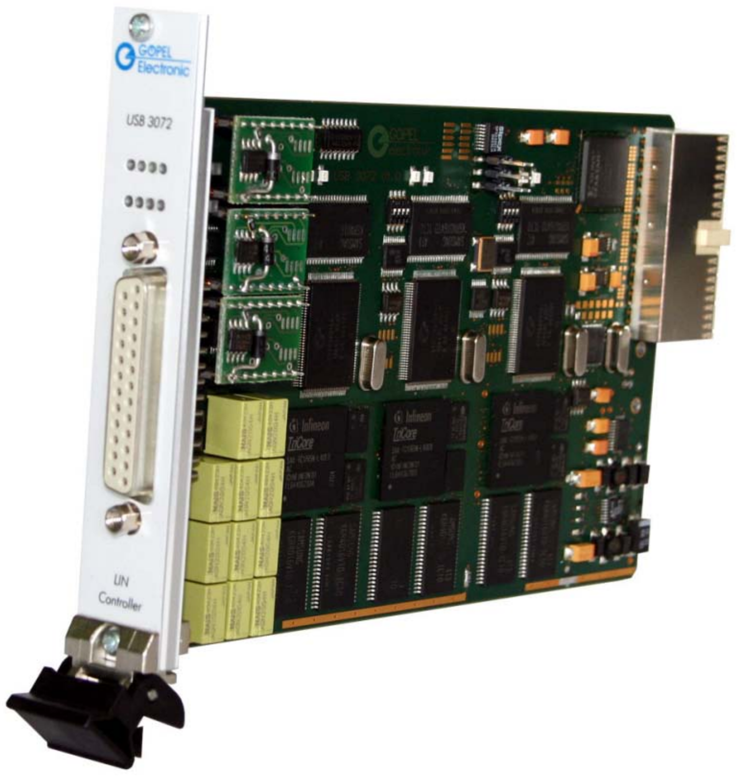


Figure 2-1: USB 3072

Please note: Downloading the Xilinx FPGA is absolutely required for operating the USB 3072 board (see Xilinx_Download in the Windows Device Driver section).

For operating USB 3072 you need the GOEPEL electronic USB rack which can cover up to 16 GOEPEL electronic USB boards.

In this case, power supply comes from the built-in power supply unit.

basicLIN 3072 is a GOEPEL electronic GmbH stand-alone device based on a USB 3072 communication board to be connected to a PC or laptop. It was in particular developed for applications out of complex test systems.

The external power supply allows the use of this device for data acquisition and the inspection of signals in motor vehicles.

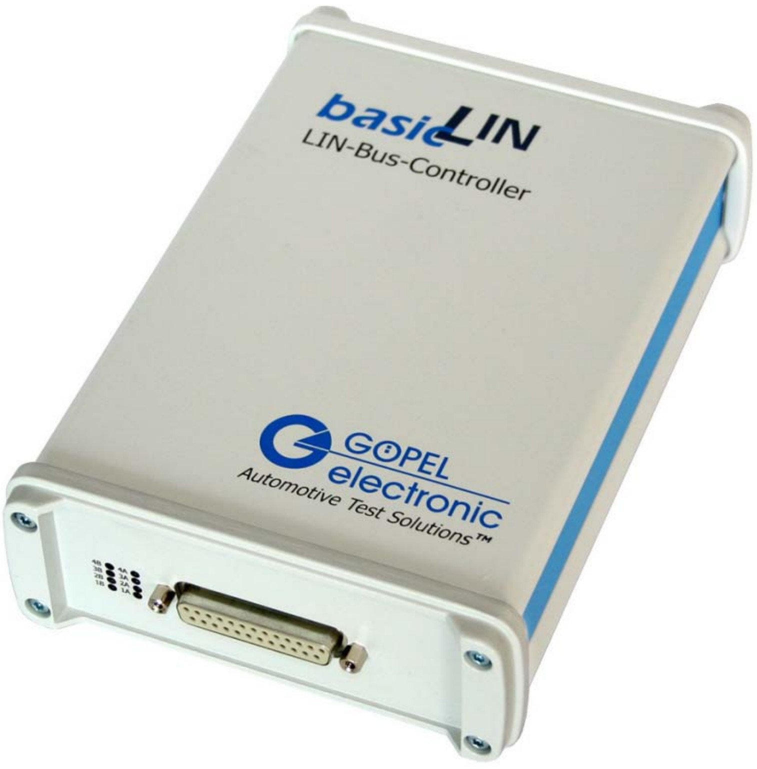


Figure 2-2: basicLIN 3072

Power supply with 8..25 VDC (and approx. 500 mA at 12 V) is effected via the two ext. Power Supply females (red = plus/ blue= minus) at the device's rear side (opposite to the LIN interfaces connector).

These females are used to supply the internal logic. In addition, the blue female is connected with the GND connections of the USB and LIN interfaces.

Resources of USB 3072/ basicLIN 3072:

- 3 LIN interfaces of Version 2.0b
- As an option, every interface can also be designed as a K-Line interface
- Extended trigger functions with one trigger input and output per interface to the frontal plug connector or the backplane
- For each LIN interface USB 3072/ basicLIN 3072 has an own 32 bit microcontroller (TriCore TC1765, 40MHz)
- Visualisation of the controller states by LEDs arranged at the front panel (two LEDs per controller, see LED Indication)
- High flexibility through pluggable transceiver modules

In this User Manual, Controller means ALWAYS one of the microcontrollers assigned to each LIN interface
(with the exception of the “LIN Controller” designation on the front panel of a USB 3072 board, which refers to the entire board).

In case a basicLIN 3072 device does not provide enough resources for your applications, there is a GOEPEL electronic USB Rack available to cover up to 16 GOEPEL electronic USB boards.
Then the power supply comes from a built-in power supply unit with 230V or 115V connector at the rack’s rear side.

2.2 Technical Specification

2.2.1 Dimensions

(width x height x depth):

- USB 3072: 4 HP mm x 130 mm x 185 mm
- basicLIN 3072: 145 mm x 70 mm x 220 mm

The dimensions stated for USB 3072 refer to an installed board.

2.2.2 Properties

The characteristics of USB 3072/ basicLIN 3072 are as follows:

Symbol	**Parameter**	**Min.**	**Typ.**	**Max.**	**Unit**	**Remarks**
Ext. Power Supply	Power supply for internal logic	8		25	V	
V_{BAT}	Battery voltage		12	27	V	for LIN
	Transmission rate			22	kBaud	
	External trigger input	3.3		50	V	
	External trigger output		5		V	

The external trigger inputs can also be used to feed in an external clock signal (see also the 0x51 LIN Monitor – Time Stamp Configuration Firmware command).

2.3 Construction

2.3.1 General

USB 3072 boards or basicLIN 3072 devices have three LIN interfaces of version 2.0b. (As an option, also K-Line is possible).

Each LIN interface is supported by an own microcontroller.

On the board the USB information is distributed among the controllers.

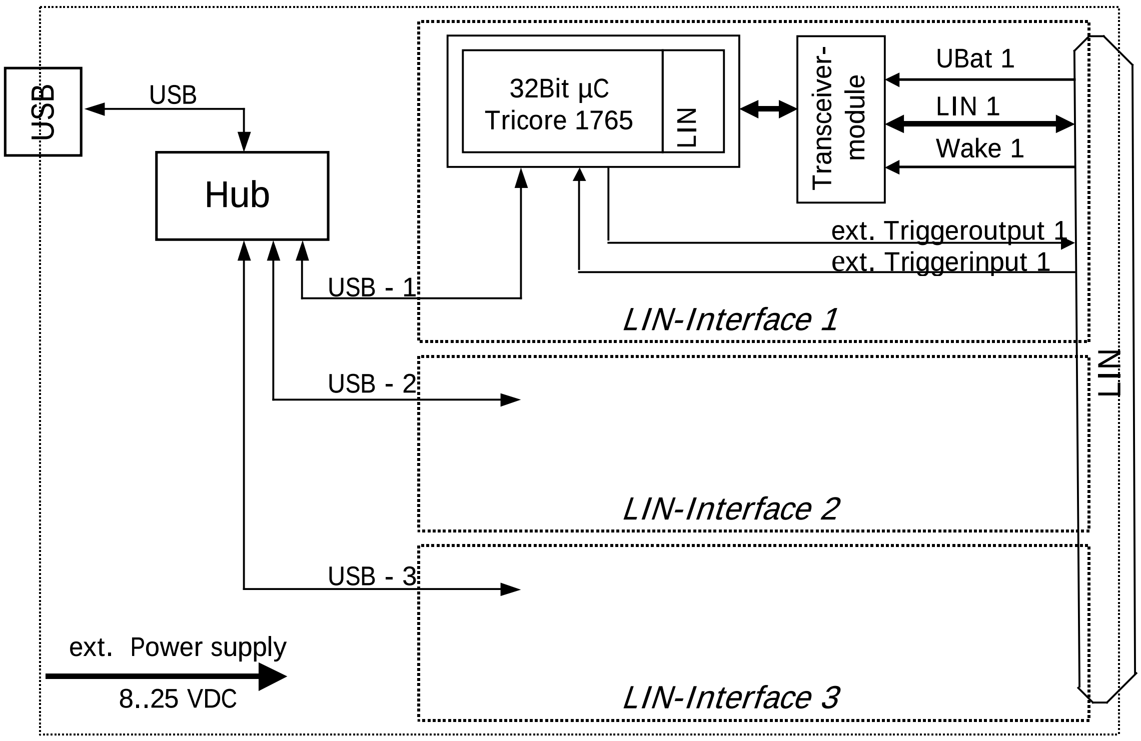


Figure 2-3: Block diagram USB 3072

Warning

Please use the delivered USB cables to connect USB 3072/ basicLIN 3072 devices to the PC's USB interface.

Other cables may be inapplicable.

2.3.2 Addressing

Addressing an individual USB 3072/ basicLIN 3072 device when operating several USB 3072/ basicLIN 3072 at the same computer takes place exclusively according to the serial numbers of their LIN controllers (see Control Software): The LIN controller with the LEAST serial number is always the device with the number 1.

Tip

To improve clarity, we recommend to arrange the individual USB 3072 devices in the USB rack in the order of ascending serial numbers of their LIN controllers (or to connect the individual basicLIN 3072 devices in the same order to the computer)

2.3.3 Communication Interfaces

Up to <u>3 x LIN-Interface Version 2.0</u> or
Up to <u>3 x K-Line Interface (ISO 9141)</u>

The following figure shows the output circuitry of a USB 3072 board between the transceiver modules and the frontal plug connector:

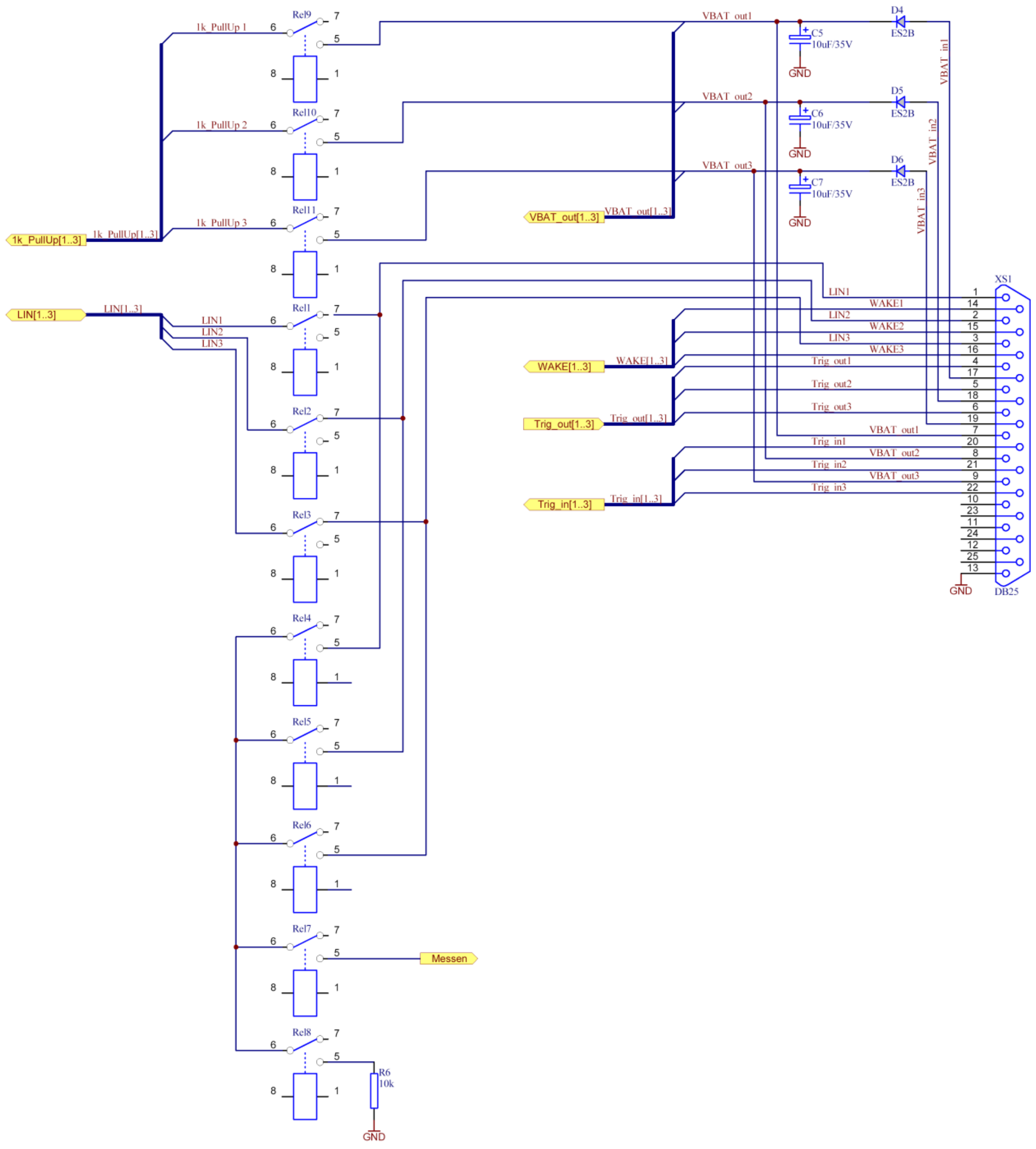


Figure 2-4: Detail of USB 3072 Output Circuit Diagram

If it is necessary to change transceiver modules, take care that the square pins (Pin1) of the socket and the transceiver module lie about each other.

LIN:

The transceivers are designed as plug-in modules. Generally, the TJA1020 is used for this type of transceicer.

For the standard design of the transceiver modules, it is possible to change over between Master and Slave configuration per software using the relays Rel9 for LIN1, Rel10 for LIN2 or Rel11 for LIN3.

The pull-up resistors for LIN are located on the transceiver modules.

Via the V_{Bat} contacts the power supply of the transceiver modules is connected. According to the LIN specification, this power supply is to be carried out via a reverse-connect protection diode and a support capacitor (so the V_{Bat} voltage is fed by VBAT in 1..VBAT in 3).
The voltage fed at the power supply (voltage) pin should not exceed the transceiver's scope (for example, the upper V_{BAT} voltage limit is 27 V for TJA1020).

The VBAT out 1..VBAT out 3 connections are monitoring outputs to measure the real voltage at the transceiver, possibly to compare this voltage with the LIN signal level.

For special control and measuring tasks, the communication boards are provided with a hardware trigger output and input for each LIN interface.
Their function description can be taken from the LIN Commands section of the GOEPEL Firmware documentation.

In addition, USB 3072 boards offer the possibility to separate the LIN communication bus from the corresponding test object (device or unit under test) via the relays 1..3.

The relays 4..6 allow the interconnection of the three LIN interfaces to a common bus on the board (see Figure 2-4).

K-Line:

The transceivers are designed as plug-in modules. Generally, the L9637 of ST is used for this type of transceicer.

Via the VBAT in1..VBAT in3 contacts, the supply voltage of the transceiver modules is connected. The voltage fed at the power supply (voltage) pin should not exceed the transceiver's scope (for example for L9637, the upper V_{BAT} voltage limit is 36 V).

To bridge the reverse polarity diode for V_{Bat} for LIN, the V_{BAT} voltage can be fed via VBAT out 1..VBAT out 3.

2.3.4 Assembly

Figure 2-5 shows schematically the component side of a USB 3072 board.

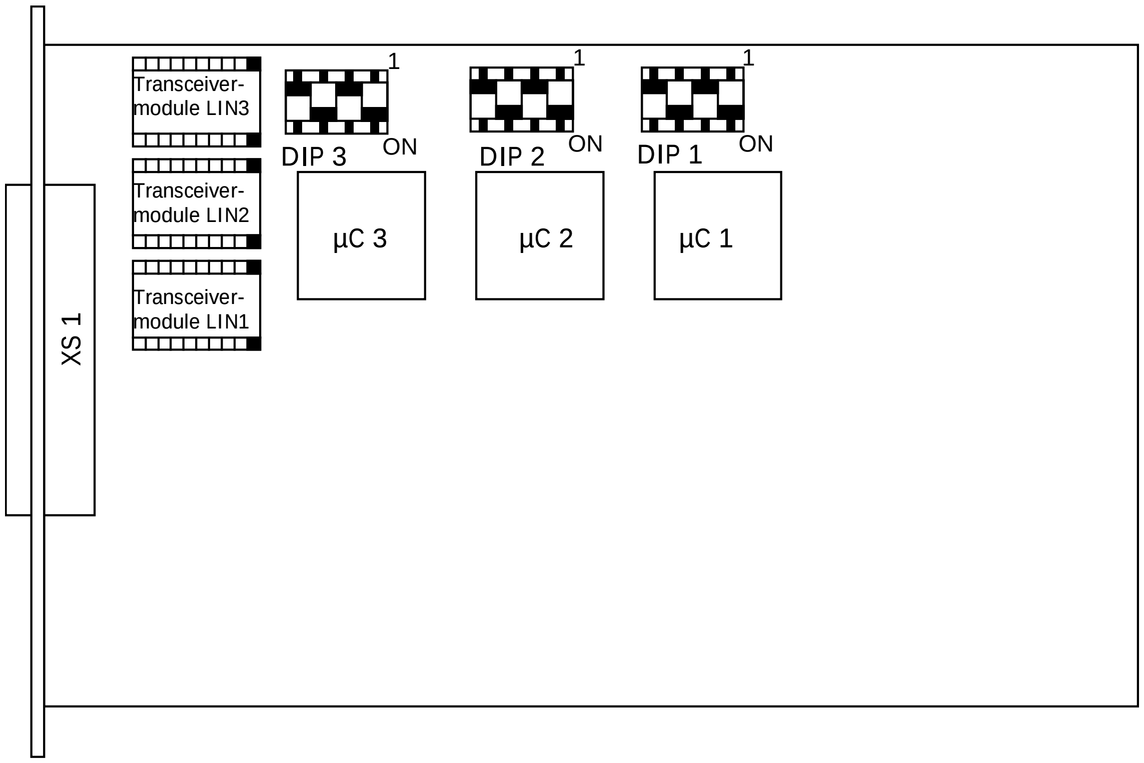


Figure 2-5: Component side of USB 3072

The configuration elements of Figure 2-5 are explained in the following table:

LIN1	Transceiver module for LIN1/K-Line1
LIN2	Transceiver module for LIN2/K-Line2
LIN3	Transceiver module for LIN3/K-Line3
DIP1...3	DIP switches of USB 3072 boards for configurating the microcontrollers **Do not change the settings!**

2.3.5 Connector Assignments

Type: DSub 25 poles socket

The signals of the LIN interfaces can be accessed via this connector at the communication board's front side with the following assignment:

No.	XS1 pin	Signals name	Remarks
1	14	LIN1/ K-Line1	LIN-(K-Line) Bus line1
2	2	LIN2/ K-Line2	LIN-(K-Line) Bus line2
3	15	LIN3/ K-Line3	LIN-(K-Line) Bus line3
4	1	Trig out1	Trigger output 1
5	17	Trig out2	Trigger output 2
6	5	Trig out3	Trigger output 3
7	18	VBAT out1	Measuring output for supply voltage of Transceiver 1 OR Supply voltage input for Transceiver 1 without reverse polarity diode
8	4	VBAT out2	Measuring output for supply voltage of Transceiver 2 OR Supply voltage input for Transceiver 2 without reverse polarity diode
9	20	VBAT out3	Measuring output for supply voltage of Transceiver 3 OR Supply voltage input for Transceiver 3 without reverse polarity diode
10	8	-	not assigned
11	21	-	not assigned
12	7	-	not assigned
13	23	Gnd	Ground potential
14	11	Wake1	Transceiver Wake input 1
15	24	Wake2	Transceiver Wake input 2
16	10	Wake3	Transceiver Wake input 3
17	3	VBAT in1	Supply voltage input for Transceiver 1 via reverse polarity diode
18	16	VBAT in2	Supply voltage input for Transceiver 2 via reverse polarity diode
19	6	VBAT in3	Supply voltage input for Transceiver 3 via reverse polarity diode
20	19	Trig in1	Trigger input 1
21	9	Trig in2	Trigger input 2
22	22	Trig in3	Trigger input 3
23	12	-	not assigned
24	25	-	not assigned
25	13	-	not assigned

For LIN, the PINs 14/ 15/ 16 may be connected with the WAKE lines (depending on the transceiver).

USB Interface

You find the USB-B-Socket (with USB standard assignment) for the USB 2.0 interface opposite to the LIN interfaces connector side of USB 3072.

2.3.6 LED Indication

The LEDs arranged at the front panel of a USB 3072 board indicate the current operating state of the controllers assigned to the LIN interfaces (also called “LIN Ports”).
One green LED and one red LED belong to each LIN interface.
The arrangement is shown in the following figure:

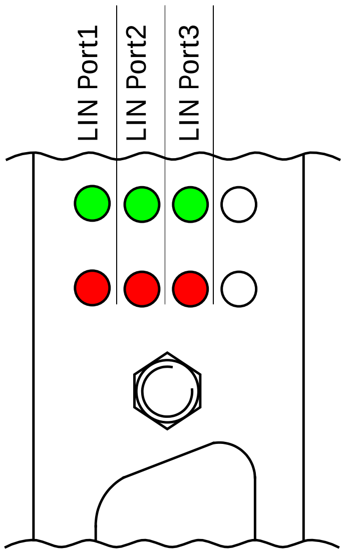


Figure 2-6: LED Indication

The LED states are explained in the table:

green LED	red LED	Remarks
Permanently ON		Controller not running Error cause (probably): Xilinx download not executed
Alternately blinking		Bootloader software runs Error cause (probably): Software reset not executed
OFF		Firmware runs
ON (shortly)	OFF	Firmware runs executing firmware commands

2.4 Delivery Notes

USB 3072/ basicLIN 3072 devices are delivered as follows:

- 3x LIN interface

As an option, each interface can also be designed as a K-Line interface.

In addition to the interface, the type of the corresponding LIN/ K-Line Transceiver as well as the required Functionalities for each interface must be selected.

For operating USB 3072 boards you need the GOEPEL electronic USB rack which can cover up to 16 GOEPEL electronic USB boards.

In this case, power supply comes from the built-in power supply unit.

3 Control Software

There are three ways to integrate USB 3072/ basicLIN 3072 hardware in your own applications:

- Programming via G-API
- Programming via DLL Functions
- Programming with LabVIEW

3.1 Programming via G-API

The G_API (GOEPEL-API) is the favored user interface for this GOEPEL hardware.

You can find all necessary information in the *G-API* folder of the delivered CD.

3.2 Programming via DLL Functions

Programming via DLL Functions is possible also in future for existing projects which can not be processed with the GOEPEL electronic programming interface G-API.

We would be pleased to send the GOEPEL Firmware documentation to you on your request. Please get in touch with our sales department in case you need that.

The GUSB_Platform expression used in the following function description stands for the name of a GOEPEL electronic USB driver.

For the used structures, data types and error codes refer to the headers – you find the corresponding files on the supplied CD.

In this User Manual, Controller means always the microcontroller assigned to the corresponding LIN/ K-Line interface of a USB 3072/ basicLIN 3072 device. An own USB Controller providing the USB 2.0 interface is assigned to each of these Controllers.

On the other hand, USB Controller means ALWAYS the controller providing the USB 2.0 interface of the USB 3072/ basicLIN 3072 device.

3.2.1 Windows Device Driver

The DLL functions for programming using the Windows device driver are described in the following sections:

- Driver_Info
- DLL_Info
- Write_FIFO
- Read_FIFO
- Read_FIFO_Timeout
- Write_COMMAND
- Read_COMMAND
- Xilinx_Download
- Xilinx_Version

Assignment of the USB controllers to a USB 3072/ basicLIN 3072 device

A USB 3072/ basicLIN 3072 device appears with three USB devices in the Windows Device Manager, as each Controller has an own USB controller (that means each LIN node, too.
See Figure 1-1 in the Driver Installation chapter).

To be able to assign these USB devices to the USB 3072/ basicLIN 3072 device(s) and their Controllers, first find out the serial numbers by the Driver Info command.
The assignment is defined by the remainder of the integer division of the serial number by the number 4
(modulo, mathematic formula symbol mod).

The following rule is valid:

Serial number mod 4	Controller
0	1
1	2
2	3

Example: Two USB 3072/ basicLIN 3072 devices with three Controllers each:

Serial number	Controller	Device	DeviceNumber
20070060	1	1	1
20070061	2	1	2
20070062	3	1	3
20070064	1	2	4
20070065	2	2	5
20070066	3	2	6

3.2.1.1 *Driver_Info*

The GUSB_Platform_Driver_Info function is for the status query of the hardware driver and for the internal initialization of the required handles.

Executing this function at least once is obligatory before calling any other function of the GUSB_Platform driver.

Format:

```
int GUSB_Platform_Driver_Info(GUSB_Platform_DriverInfo *pDriverInfo,
                              unsigned int LengthInByte)
```

Parameter:

Pointer, for example pDriverInfo
to a data structure
For the structure, see the *GUSB_Platform.h* file on the delivered CD

LengthInByte
Size of the storage area pDriverInfo is pointing to, in bytes

Description:

The GUSB_Platform_Driver_Info function returns information regarding the status of the hardware driver.

For this reason, the address of the pDriverInfo pointer has to be transferred to the function. By means of the LengthInByte parameter the function checks internally if the user memory is initialized correctly.

The function fills the structure pDriverInfo is pointing to with statements regarding the driver version, the number of all involved USB controllers (supported by this driver) and additional information, e.g. the serial number(s).

Making the hardware information available
as well as initializing the belonging handles is obligatory for the further use of the USB hardware.

3.2.1.2 *DLL_Info*

The GUSB_Platform_DLL_Info function is for the version number query of the DLL.

Format:

```
int GUSB_Platform_DLL_Info(GUSB_Platform_DLLinfo *DLLinformation)
```

Parameter

Pointer, for example DLLinformation
to a data structure
For the structure, see the *GUSB_Platform.h* file on the delivered CD

Description:

The GUSB_Platform_DLL_Info function returns the DLLinfo structure. The first integer value contains the version number of the *GUSB_Platform.dll*.

Example:

Version number **`1.23`** is returned as **`123`**,
and version number **`1.60`** as **`160`**.

3.2.1.3 *Write_FIFO*

With the GUSB_Platform_Write_FIFO function a command is sent to the Controller.

Format:

```
int GUSB_Platform_Write_FIFO(unsigned int DeviceName,
                             unsigned int DeviceNumber,
                             t_USB_FIFO_Interface_Buffer *pWrite,
                             unsigned int DataLength)
```

Parameters:

DeviceName

Type of the addressed device (number declared in *GUSB_Platform_def.h*, for USB 3072/ basicLIN 3072 = 5)

DeviceNumber

Number of the addressed device. In the case several devices of the same type are connected, numbering is carried out according to their serial numbers in ascending order (the device with the LEAST serial number has always the DeviceNumber 1).

Please respect the Assignment of the USB controllers ... notes given in the Windows Device Driver chapter: The quantity of “devices” and so of DeviceNumbers corresponds to the quantity of available LIN Interfaces.

Pointer, for example pWrite
to the write data area

DataLength

Size of the storage area pWrite is pointing to, in bytes
Data is consisting of Command Header and Command Bytes
(Currently max. 1024 bytes per command)

Description:

The GUSB_Platform_Write_FIFO function sends a command to the Controller.

For the general structure, see the General Firmware Notes section of the GOEPEL Firmware document.

3.2.1.4 *Read_FIFO*

The GUSB_Platform_Read_FIFO function is for reading a response from the Controller.

Format:

```
int GUSB_Platform_Read_FIFO(unsigned int DeviceName,
                            unsigned int DeviceNumber,
                            t_USB_FIFO_Interface_Buffer *pRead,
                            unsigned int *DataLength)
```

Parameters:

DeviceName

Type of the addressed device (number declared in *GUSB_Platform_def.h*, for USB 3072/ basicLIN 3072 = 5)

DeviceNumber

Number of the addressed device. In the case several devices of the same type are connected, numbering is carried out according to their serial numbers in ascending order (the device with the LEAST serial number has <u>always</u> the DeviceNumber 1).

Please respect the Assignment of the USB controllers ... notes given in the Windows Device Driver chapter: The quantity of “devices” and so of DeviceNumbers corresponds to the quantity of available LIN Interfaces.

Pointer, for example pRead
to the reading buffer
After successful execution of the function, there is the data in this reading buffer, consisting of Response Header and Response Bytes (Currently max. 1024 bytes per response)

DataLength

Prior to function call: Size of the reading buffer in bytes (to be given)

After function execution: Number of bytes actually read

Description:

The GUSB_Platform_Read_FIFO function reads back the oldest response written by the Controller. In the case no response was received within the fixed Timeout of 100 ms, the function returns NO error, but the Number of bytes actually read is 0 !!!

3.2.1.5 *Read_ FIFO_ Timeout*

The GUSB_Platform_Read_FIFO_Timeout function is for reading a response from the Controller within the Timeout to be given.

Format:

```
int GUSB_Platform_Read_FIFO_Timeout(unsigned int DeviceName,
                                    unsigned int DeviceNumber,
                                    t_USB_FIFO_Interface_Buffer *pRead,
                                    unsigned int *DataLength,
                                    unsigned int Timeout)
```

Parameters:

DeviceName

Type of the addressed device (number declared in *GUSB_Platform_def.h*, for USB 3072/ basicLIN 3072 = 5)

DeviceNumber

Number of the addressed device. In the case several devices of the same type are connected, numbering is carried out according to their serial numbers in ascending order (the device with the LEAST serial number has always the DeviceNumber 1).

Please respect the Assignment of the USB controllers ... notes given in the Windows Device Driver chapter: The quantity of "devices" and so of DeviceNumbers corresponds to the quantity of available LIN Interfaces.

Pointer, for example pRead
to the reading buffer
After successful execution of the function, there is the data in this reading buffer, consisting of Response Header and Response Bytes (Currently max. 1024 bytes per response)

DataLength

Prior to function call: Size of the reading buffer in bytes (to be given)

After function execution: Number of bytes actually read

Timeout

To be given in milliseconds (500 as a standard value)

Description:

The GUSB_Platform_Read_FIFO_timeout function reads back the oldest response written by the Controller. In the case no response was received within the Timeout to be given, the function returns NO error, but the Number of bytes actually read is 0 !!!

3.2.1.6 Write_ COMMAND

With the GUSB_Platform_Write_COMMAND a configuration command is sent to the USB Controller.

Format:

```
int GUSB_Platform_Write_COMMAND(unsigned int DeviceName,
                                unsigned int DeviceNumber,
                                t_USB_COMMAND_Interface_Buffer *pWrite,
                                unsigned int DataLength)
```

Parameters:

DeviceName

Type of the addressed device (number declared in *GUSB_Platform_def.h*, for USB 3072/ basicLIN 3072 = 5)

DeviceNumber

Number of the addressed device. In the case several devices of the same type are connected, numbering is carried out according to their serial numbers in ascending order (the device with the LEAST serial number has always the DeviceNumber 1).

Please respect the Assignment of the USB controllers ... notes given in the Windows Device Driver chapter: The quantity of “devices” and so of DeviceNumbers corresponds to the quantity of available LIN Interfaces.

Pointer, for example pWrite
to the write data area

DataLength

Size of the storage area pWrite is pointing to, in bytes
See also USB Controller Control Commands
(Currently max. 64 bytes per command)

Description:

The GUSB_Platform_Write_COMMAND function sends a command to the USB Controller.

For the general structure, see the USB Controller Control Commands section.

3.2.1.7 Read_ COMMAND

The GUSB_Platform_Read_COMMAND function is for reading a response from the USB Controller.

Format:

```
int GUSB_Platform_Read_COMMAND(unsigned int DeviceName,
                               unsigned int DeviceNumber,
                               t_USB_COMMAND_Interface_Buffer *pRead,
                               unsigned int *DataLength)
```

Parameters:

DeviceName

Type of the addressed device (number declared in *GUSB_Platform_def.h*, for USB 3072/ basicLIN 3072 = 5)

DeviceNumber

Number of the addressed device. In the case several devices of the same type are connected, numbering is carried out according to their serial numbers in ascending order (the device with the LEAST serial number has <u>always</u> the DeviceNumber 1).

Please respect the Assignment of the USB controllers ... notes given in the Windows Device Driver chapter: The quantity of "devices" and so of DeviceNumbers corresponds to the quantity of available LIN Interfaces.

Pointer, for example pRead
to the reading buffer
After successful execution of the function, there is the data in this reading buffer, consisting of Response Header and Response Bytes
See also USB Controller Control Commands
(Currently min. 64 bytes per response)

DataLength

Prior to function call: Size of the reading buffer in bytes (to be given)

After function execution: Number of bytes actually read

Description:

The GUSB_Platform_Read_COMMAND function reads back the oldest response written by the USB Controller.

If several responses were provided by the USB Controller, up to two of these responses are written into the buffer of the USB Controller. More possibly provided responses get lost!

3.2.1.8 *Xilinx_ Download*

The GUSB_Platform_Xilinx_Download function is to load an FPGA file to the XILINX.

This function can only be executed on the USB Controller of the FIRST Controller of a USB 3072/ basicLIN 3072 device.

Format:

```
int GUSB_Platform_Xilinx_Download(unsigned int DeviceName,
                                  unsigned int DeviceNumber,
                                  char *pFileName,
                                  unsigned char *pFirmwareErrorCode)
```

Parameters:

DeviceName

Type of the addressed device (number declared in *GUSB_Platform_def.h*, for USB 3072/ basicLIN 3072 = 5)

DeviceNumber

Number of the addressed device. In the case several devices of the same type are connected, numbering is carried out according to their serial numbers in ascending order (the device with the LEAST serial number has always the DeviceNumber 1).

Please respect the Assignment of the USB controllers ... notes given in the Windows Device Driver chapter: The quantity of "devices" and so of DeviceNumbers corresponds to the quantity of available LIN Interfaces.

pFileName

Path of the FPGA file to be loaded

pFirmwareErrorCode

Error code occurring during executing this DLL function
(error code 0 means no error occurred)

(error codes -> card firmware see *GUSB_Platform_def.h*)

Description:

The GUSB_Platform_Xilinx_Download function allows to load an FPGA file to the XILINX (extension **.cfd*).
The loaded data is volatile. Therefore the function has to be executed again after switching off power.

After Xilinx_Download, a delay of about 500 ms is required (as the controllers execute a power-on reset).
Then, carry out the 0x10 Software Reset firmware command to come into the normal operating mode from bootloader mode.

3.2.1.9 Xilinx_ Version

The GUSB_Platform_Xilinx_Version function allows reading out the version of the loaded XILINX firmware.

Format:

```
int GUSB_Platform_Xilinx_Version(unsigned int DeviceName,
                                 unsigned int DeviceNumber,
                                 unsigned int *Version)
```

Parameters:

DeviceName

Type of the addressed device (number declared in *GUSB_Platform_def.h*, for USB 3072/ basicCAN 3072 = 5)

DeviceNumber

Number of the addressed device. In the case several devices of the same type are connected, numbering is carried out according to their serial numbers in ascending order (the device with the LEAST serial number has always the DeviceNumber 1).

Version

XILINX software version

Description:

The GUSB_Platform_Xilinx_Version function can be used to read out the version number of the software loaded to the FPGA.

Example:

Version number **`2.34`** is returned as **`234`**, version **`2.60`** as **`260`**.

3.3 Programming with LabVIEW

3.3.1 LabVIEW via G-API

On the delivered CD there is a folder with VIs to call USB 3072/ basicLIN 3072 devices under LabVIEW.

The LabVIEW VIs use the functions of the GOEPEL G-API for this.

3.3.2 LLB using the Windows Device Driver

On the delivered CD there is a folder with VIs to call USB 3072/ basicLIN 3072 devices under LabVIEW.

The functions described in the Windows Device Driver section are used for this.

3.4 Further GOEPEL Software

PROGRESS, Program Generator and myCAR of GOEPEL electronic are comfortable programs for testing with GOEPEL hardware.

Please refer to the corresponding Software Manuals to get more information regarding these programs.

3.5 USB Controller Control Commands

The USB Controllers are responsible for connecting the USB 3072/ basicLIN 3072 device to the PC via USB 2.0.

Messages (generally USB commands) required for configuration can be sent to these USB Controllers.

3.5.1 USB Command Structure

A USB command consists of four bytes Header and the Data (but Data is NOT required for all USB commands!).

The header of a USB command has the following structure:

Byte number	Indication	Contents
0	StartByte	0x23 (“#” ASCII character)
1	Command	(0x..) used codes according to USB Commands
2	reserved	0x00
3	reserved	0x00

3.5.2 USB Response Structure

Same as a USB command, also the USB response consists of four bytes Header and the Data (but Data is NOT returned by all USB commands!).

The header of a USB response has the following structure:

Byte number	Indication	Contents
0	StartByte	0x24
1	Command	(0x..) used codes according to USB Commands
2	Length	Length depending on the command
3	ErrorCode	Returns the error code of the command

3.5.3 USB Commands

At present there is only the READ_SW_VERSION USB command available.

Command	Indication	Description
0x04	`READ_SW_VERSION`	Provides the firmware version of the USB Controller Response: Byte 4: low byte of generic software version Byte 5: high byte of generic software version Byte 6: low byte of software version of functional part Byte 7: high byte of software version of functional part