Product Specification

# USB 4120
# basicCON 4120

## LVDS Framegrabber
## User Manual Version 1.4

GOPEL electronic
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: 23.04.2013



**Issue: April 2013**

# 1 Installation

## 1.1 USB 4120/ basicCON 4120 Hardware Installation

The USB 4120 LVDS Framegrabber board is installed in a USB Rack (either USB 1004, USB 1008 or USB 1016) of GOEPEL electronic GmbH.


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

To install a USB 4120 board, open your USB system as described in the corresponding manual and select a free slot. Insert the board carefully into the prepared slot. Use the lever at the front plate in order to push in the board finally.


When installing the board, touch it at its edges only.
Never touch the surface of the board, because otherwise it may be destroyed by electrostatic discharge.

For the hardware installation of a basicCON 4120, only the cables for USB, LVDS and Power supply (if necessary) must be connected (see Hardware).

Please refer to the notes in the Addressing chapter regarding the installation of several USB 4120 boards in a USB Rack or basicCON 4120 devices at the PC/ Laptop.

## 1.2 Driver Installation

**Windows Device Drivers**

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.

The available device driver supports Windows® 2000/ XP as well as Windows® 7 (32Bit and 64Bit) systems.

If you want to create own software for USB 4120 boards or basicCON 4120 devices, you need the **.VI* or **.LLB* files for user specific programming. These files are not automatically copied to the PC/ laptop 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 by the Device Manager whether the devices are properly embedded by the system.

The following figure shows the successful embedding of four USB 4120 boards or basicCON 4120 devices (USB 4120):


*Figure 1-1: Device Manager Display*

Please note that the Device manager shows ALL USB controllers supported by the GUSB driver.

# 2 Hardware

## 2.1 Definition

The USB 4120 LVDS Framegrabber board is a signal processing board of GOEPEL electronic GmbH with USB 2.0 interface.

This board is used particularly in automotive technology for recording and evaluating picture data transferred via LVDS systems.
The LVDS Framegrabber has the following characteristics:

- LVDS data rates up to 1,200Mbits/s
- Pixel clock up to 50MHz with a color depth of 24 bits
- 1 Mega pixels maximum resolution
- USB 2.0 interface
- Digital I/O interface up to 24V for real time triggering
- 8 MByte SRAM onboard
- Reference picture memory for onboard picture comparison
- Changeable LVDS receiver top boards
  Available types: DS90UR124, DS90CF364, DS90UR906, DS90UB926, INAP125R24APIX, MAX9248, MAX9260 and CXB1458R
- Fully flexibility regarding resolution, color depth and synchronization for exact matching of the framegrabber to the source
- Display of the controller states by four LEDs
  (see LED Indication) on the front panel

The DS90UR906 and DS90UB926 LVDS receiver top boards provide a second LVDS connection to output the LVDS input signal again after processing it by a repeater. This signal can be used for a display, for example.

USB 4120 is a signal processing board for the use in a USB-Rack of GOEPEL electronic GmbH. In this case, power supply comes from the built-in power supply unit.

basicCON 4120 is a GOEPEL electronic GmbH stand-alone device based on a USB 4120 signal processing board to be connected to a PC or laptop. It was in particular developed for applications out of complex test systems.

Power supply for basicCON 4120 can be effected via the USB interface or the ext. Power Supply females (7..25VDC, see Power Supply).

After corresponding configuration, the external power supply and the digital I/O interface allow operating the basicCON 4120 for picture comparison without USB connection (see Digital I/O Interface).

For operating USB 4120 boards you need a GOEPEL USB Rack (USB 1004, USB 1008 or USB 1016) which can cover up to 16 GOEPEL USB boards.

*Figure 2-1: USB 4120*


*Figure 2-2: basicCON 4120*

There is a different deserializer mounted in the basicCON 4120 of Figure compared with the USB 4120 board of Figure 2-1Figure . This is the reason for the different number of LVDS connectors in both figures (see also LVDS Interface and Receiver Top Boards).

At basicCON 4120's rear side there are the following connections:


*Figure 2-3: basicCON 4120 – rear side*

- USB-B female for the USB 2.0 interface with USB standard assignment
- DC female for the AC adaptor plug (part of delivery)
- Banana females for power supply

Warning

Please use for external power supply either the banana females OR the DC female for the AC adaptor plug.

Regarding Power Supply, please refer to the corresponding section.

# 2.2 Technical Data

## 2.2.1 Dimensions (W x H x D):

- USB 4120: 4 HP x 130 mm x 185 mm
- basicCON 4120: 130 mm x 55 mm x 200 mm

The dimensions given for USB 4120 refer to a board inside a GOEPEL electronic USB Rack.

## 2.2.2 Properties

| Symbol | Parameter | Min. | Typ. | Max. | Unit | Remarks |
|---|---|---|---|---|---|---|
| $V_{SS}$ | Operating voltage | | 5 | | V | Supply via USB (basicCON 4120) |
| $I_{SS}$ | Operating current | | 300 | 400 | mA | |
| $V_{SExt}$ | External power supply voltage | 8 | 12 | 25 | V | $I_{Max}$ = 200mA at 12V Optional for basicCON 4120 |
| $V_{SDIO}$ | Power supply voltage of the digital I/O interface | 5 | | 27 | V | Optional (see Digital I/O Interface) |
| $V_{LVDS}$ | Differential LVDS input voltage | ±50 | ±500 | | mV | Depending on the receiver top board |
| $I_{LVDS}$ | Differential LVDS input current | | ±40 | | µA | Depending on the receiver top board |
| $Z_{LVDS}$ | LVDS input impedance | | 100 | | Ω | Ohms |
| $f_{LVDS}$ | LVDS input frequency | | | 1,200 | MHz | Depending on the receiver top board |
| $V_{DIn}$ | Input voltage of the digital I/O | 5 | | 27 | V | 5V without supply via the DIO interface |
| $I_{DIn}$ | Input current of the digital I/O | | | 12 | mA | At 27V |
| $V_{DOut}$ | Output voltage of the digital I/O | 5 | | 27 | V | |
| $I_{DOut}$ | Output current of the digital I/O | | | 250 | mA | At 27V |

Warning

Please use the delivered USB cables to connect USB 4120/ basicCON 4120 devices to the PC's or laptop's USB interface.
Other cables may be inapplicable.

# 2.3 Construction

## 2.3.1 General

The devices are delivered completely including the LVDS receiver top board.

The following receiver top boards are currently available:

| LVDS Deserializer | Properties | Supported LVDS Serializer |
|---|---|---|
| CXB1458R | 24 bits color depth<br>4 bits control<br>7.6-42MHz pixel clock | CXB1457R |
| DS90CF364 | 18 bits color depth<br>3 bits control<br>20-50MHz pixel clock | DS90C363 |
| DS90UB926 | 24 bits color depth<br>3 bits control<br>5-50MHz pixel clock | DS90UB925<br>DS90UH925 (without HDCP encoding)<br>DS90UR905<br>DS90UR907 |
| DS90UR124 | 21 bits color depth<br>3 bits control<br>18-43MHz pixel clock | DS90UR241<br>(DS90C241) |
| DS90UR906 | 24 bits color depth<br>3 bits control<br>5-50MHz pixel clock | DS90UR905<br>(DS90UR124<br>DS90C241) |
| INAP125R24<br>APIX | 24 bits color depth<br>3 bits control<br>6-32MHz pixel clock | INAP125T24 |
| MAX9248 | 18 bits color depth<br>9 bits control<br>18-42MHz pixel clock | MAX9247<br>(MAX9217) |
| MAX9260 | 24 bits color depth<br>8 bits control<br>12.5-50MHz pixel clock | MAX9259 |

Figure 2-4 shows the schematic structure of the devices as a block diagram.

For basicCON 4120 devices/ USB 4120 boards, a USB2.0 controller is used as the interface to the USB bus. It includes all function blocks required for the communication with the bus.

The USB 4120 board was developed for the use in one of the GOEPEL electronic USB Racks (USB 1004, USB 1008, USB 1016). Unlike the basicCON 4120, USB 4120 has additional signal lines (GPIO) via the USB connector, that can be interconnected for the communication of several boards with each other.

For the basic functions, these lines are not required.

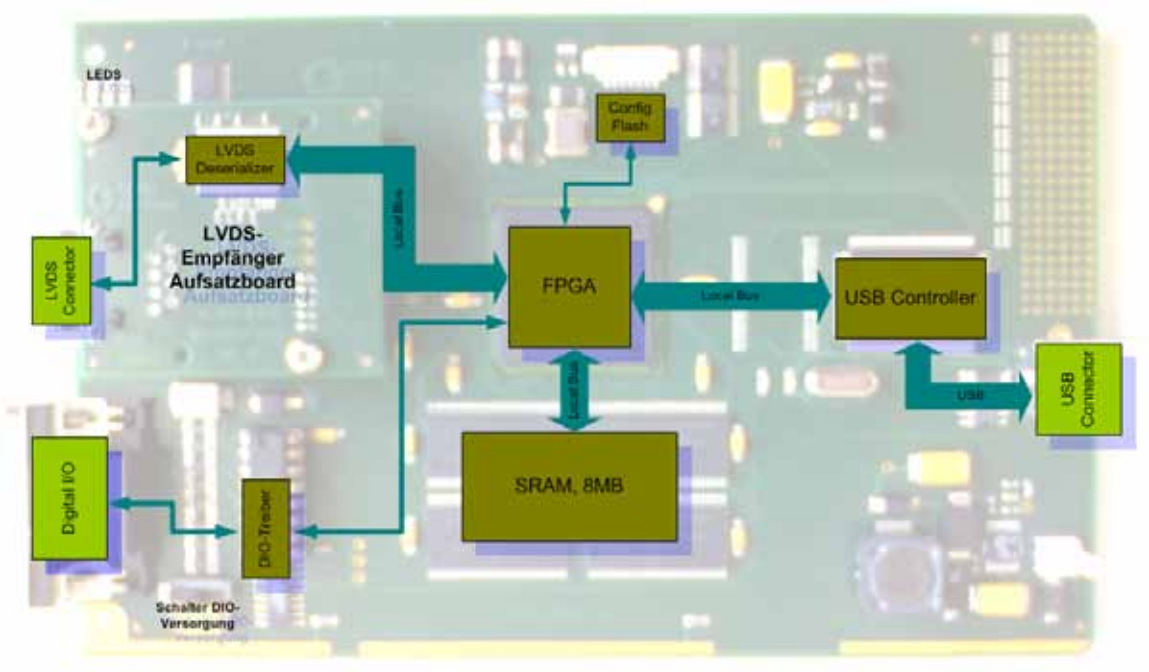



*Figure 2-4: Block diagram of a USB 4120 signal processing board*

All boards have an HF connector available to connect the LVDS input signals, a 9-pole DSub male for the digital input/ output interface (DIO) as well as four LEDs for status display.

The USB 4120 board has a 132-pole backplane connector for the connection in a GOEPEL electronic USB Rack.

The basicCON 4120 has a USB-B connection as well as two 4mm banana females for the external power supply (or the female for the supplied AC adaptor plug.

### 2.3.2 Power Supply

The USB 4120 is supplied via the USB Rack in that is has been installed.

Power supply for basicCON 4120 can be effected via the USB interface (as far as it is configured for the required current of about 500mA).

For operating the basicCON 4120 at passive USB hubs or laptops with lower power supply, use the two banana females for external Power Supply (red = plus, blue = minus) for supplying the device with 7-25VDC (and approx. 200mA at 12V).
Alternatively, you may use the female for the supplied AC adaptor plug with coaxial power plug 2.1 x 5.5mm/ plus polarity inside (see Figure 2-3).

### 2.3.3 Addressing

Addressing an individual USB 4120/ basicCON 4120 device when operating several USB 4120/ basicCON 4120 in the same system, for example in the GOEPEL electronic USB Rack, takes place exclusively according to the serial number (see also Control Software):
The device with the least serial number is always the device with the number 1.


To improve clarity, we recommend to arrange several USB 4120 boards in the USB Rack in the order of ascending serial numbers (or to connect several basicCON 4120 devices in the same order to the PC/ laptop).

### 2.3.4 LVDS Interface

LVDS is a serial broadband transmission standard, becoming more and more important in automotive technology.

Data is transmitted differential on two signal lines with very low voltage and current, and frequencies more than 1 GHz.
Therefore, appropriate connectors and cables are strongly required for failure-free transmission.

For the DS90UR124, DS90UR906, DS90UB926, INAP125R24 APIX MAX9248, MAX9260 and CXB1458R receiver top boards, the following connector is used:

D4S20A-40ML5-Z of Rosenberger company

LVDS In pin assignment:

| | | | | |
|---|---|---|---|---|
| 1 – GND | 2 – LVDS+ | 3 – NC | 4 – LVDS- | Shield – GND |

The DS90UR906 and DS90UB926 LVDS receiver top boards provide a second LVDS connection (LVDS Out) to output the LVDS input signal again after processing it by a repeater. This signal can be used for a display, for example.

LVDS Out pin assignment (for DS90UR906 and DS90UB926 only):

| | | | | |
|---|---|---|---|---|
| 1 – LVDS- | 2 – NC | 3 – LVDS+ | 4 – GND | Shield – GND |

The DS90CF364 receiver top board is operated with 4 LVDS channels (3x data, 1x clock).
It has the following 8-pole connector:

RJ45 – female Kat6

LVDS In pin assignment:

| | | | |
|---|---|---|---|
| 1 – LVDS In0- | 2 – LVDS In0+ | 3 – LVDS In1- | 4 – LVDS In2+ |
| 5 – LVDS In2- | 6 – LVDS In1+ | 7 – Clk In- | 8 – Clk In+ |
| Shield – GND | | | |

For connecting LVDS signals, only STP cables with 100Ω impedance and appropriate connectors should be used.
These cables can be delivered by GOEPEL electronic.

### 2.3.5 Digital I/O Interface

By the frontal connector Digital I/O (DIO) there is a digital interface available. With the toggle on the corresponding board you define the power supply for the output driver: Via the DIO (5 to 27V, Pos DIO) or internal via 5V (Pos 5V).

Factory setting is the toggle in Pos DIO (see following figure).

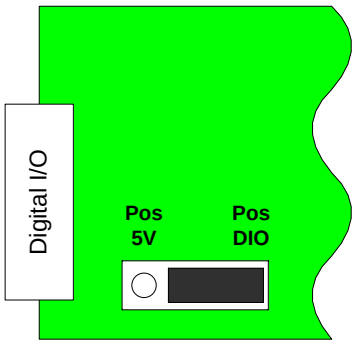



*Figure 2-5: Toggle DIO power supply*

By means of the DIO the board can be operated without PC or laptop (after executing the configuration via USB). In this case, power supply must be continuously assured (see Power Supply).

The DIO can be used to release picture scans to be automatically compared onboard with the reference picture. But this interface does not allow any configuration or specific picture evaluation.

An advantage of the DIO is the possibility of real time triggering of the framegrabber. Only by using the DIO it can be assured that a specific picture can be gripped in the video stream.

The following table shows the pin assignment of the Digital I/O Interface:

| Pin No. | Signal name | Description |
|---|---|---|
| 1 | FG_Ready_Out | Framegrabber output ready for DIO control |
| 2 | Comp_Fail_Out | Picture comparator has detected at least one failure |
| 3 | Ext_Trigger_In | The external trigger input starts a picture grip followed by the comparison with the reference picture |
| 4 | Ext_Res1_In | Reserve 1 input |
| 5 | DIO_VCC | DIO power supply (+5 .. +27V),<br>in case of Pos DIO setting (Figure 2-5) |
| 6 | FG_Pass_Out | The Framegrabber Pass output indicates that triggering has been finished and the picture has been compared failure-free |
| 7 | FG_Lock_Out | The Framegrabber Lock output indicates that the LVDS signal is locked |
| 8 | Ext_Res0_In | Reserve 0 input |
| 9 | DIO_GND | DIO Ground Pin |

## 2.3.6 LED Indication

The LEDs arranged at the front panel of a USB 4120 board indicate the current operating state of the framegrabber.

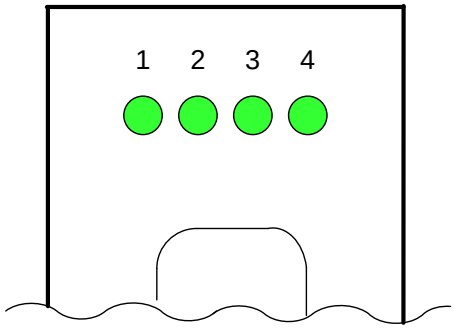



*Figure 2-6: LED Indication*

The states displayed by the LEDs are described in the following table:

| LED | Description |
|---|---|
| LED 1 ON | Pixel error during the onboard picture comparison |
| LED 2 ON | Failure at picture scan (Sync or resolution error) |
| LED 3 ON | External trigger mode ON |
| LED 4 ON | LVDS Lock active |

## 2.3.7 Receiver Top Boards

The LVDS frame grabber can be attuned to the signal source by means of several receiver top boards.

The arbitrary assignment of the LVDS bits to the corresponding color or control bits of the picture is also decisive. For that, the framegrabber must be configured appropriate to the assignment of the LVDS source.

The following figures show the assignment of the corresponding LVDS receiver (deserializer) to be observed during configuration or initializing the framegrabber.

**DS90UR124**

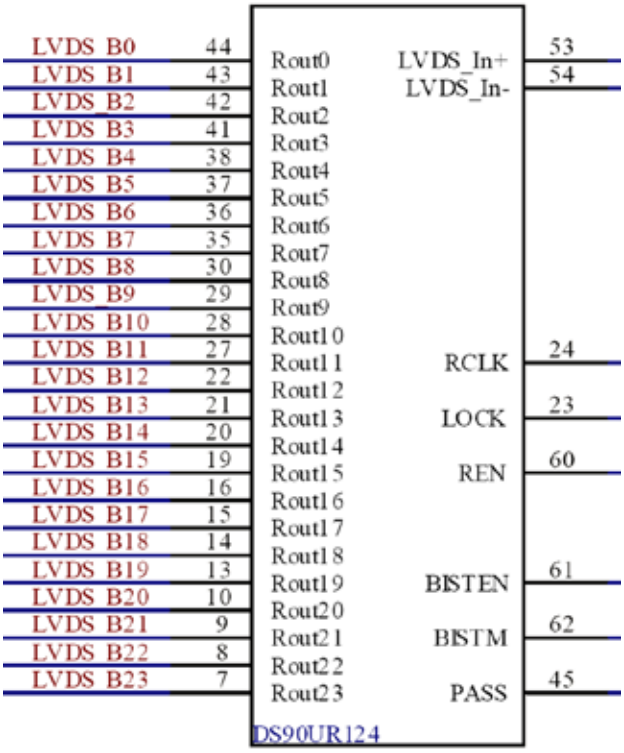



*Figure 2-7: DS90UR124 Pin assignment*

This top board has a toggle/ jumper to make the USB 4120 board compatible either to the DS90UR241 or the DS90C241 serializers.

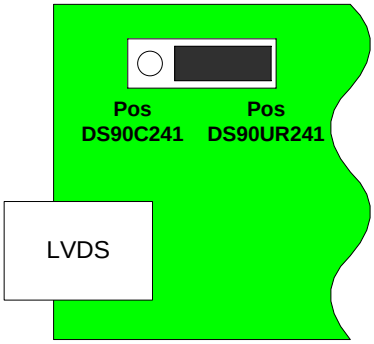



*Figure 2-8: Serializer-toggle*

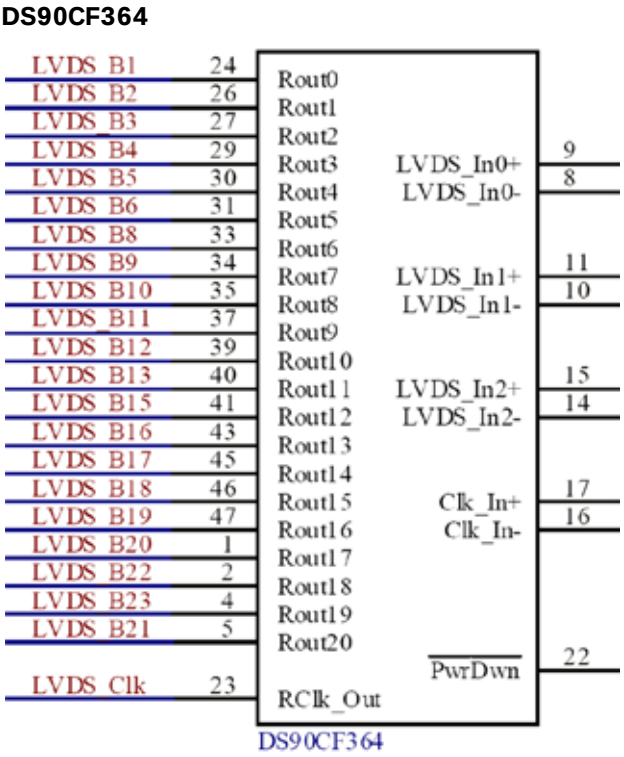



*Figure 2-9:*
*DS90CF364 Pin assignment*

**DS90UR906**

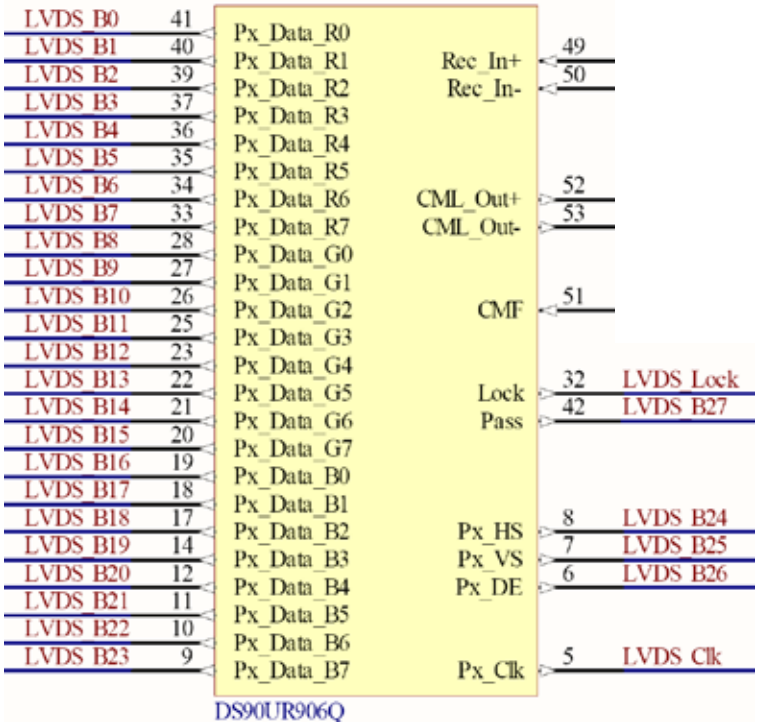



*Figure 2-10: DS90UR906 Pin assignment*

For this top board the deserializer must be configured by an I2C interface. This configuration is carried out by the framegrabber. If necessary, the configuration can be changed by the user at any time.

By configuration this top board can be made compatible to the DS90UR905 and DS90UR241 serializers as well as to the DS90C241 serializer.

For the compatibility to the DS90UR241 and DS90C241 serializers, the assignment of the serializers must correspond to the datasheet (DS90UR906).

In addition, this top board has a repeater. It processes the LVDS input signal and outputs it again on a second LVDS connector.

**DS90UB926**

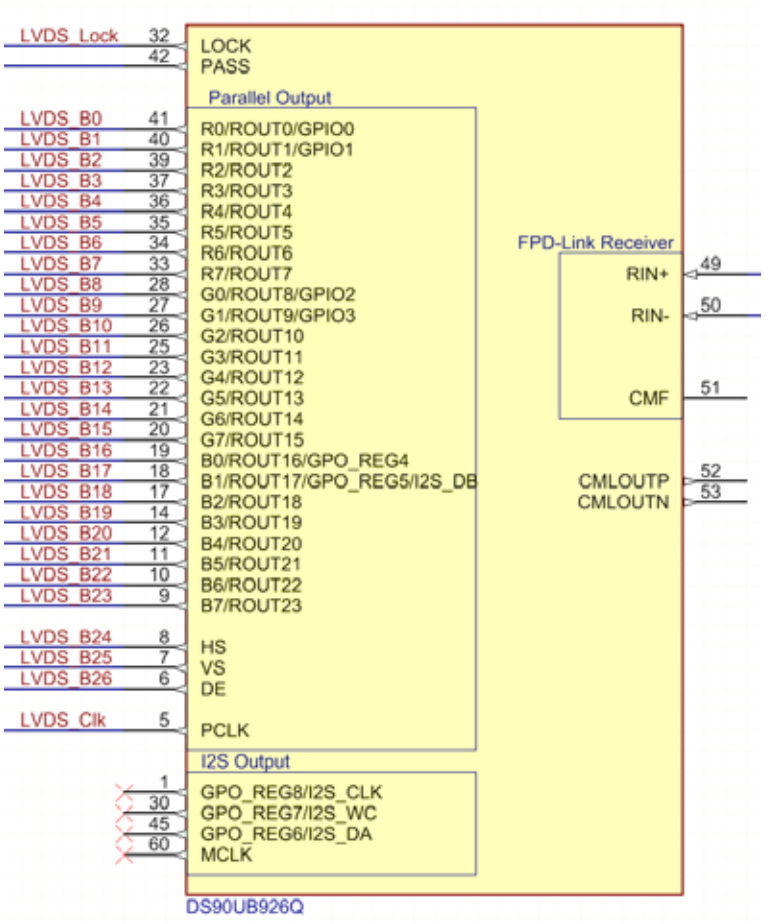



*Figure 2-11:*
*DS90UB926 Pin assignment*

For this top board the deserializer must be configured by an I2C interface. This configuration is carried out by the framegrabber. If necessary, the configuration can be changed by the user at any time.

By different configurations, this top board can be made compatible to the DS90UB925 and DS90UH926 as well as DS90UR905 and DS90UR907 serializers.
Please note: The compatibility to the DS90UH926 does not include HDCP encoding of the DS90UB926.

In addition, this top board has a repeater. It processes the LVDS input signal and outputs it again on a second LVDS connector.

**INAP125R24 APIX**

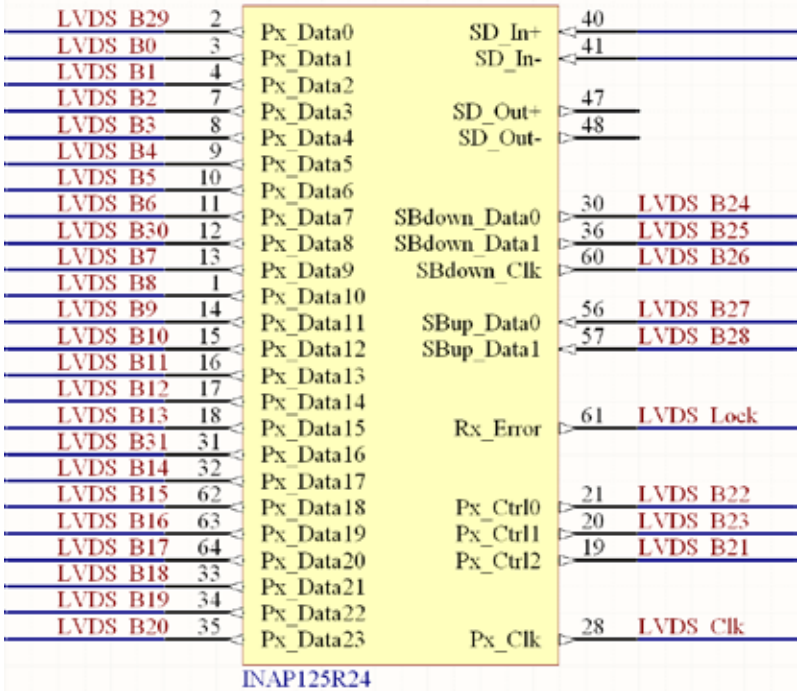



*Figure 2-12: INAP125R24 APIX Pin assignment*

Please pay attention to configure the deserializer accurately for this receiver top board

This configuration is carried out by the framegrabber. If necessary, the configuration can be changed by the user at any time.

Upstream and sideband channels are not used.

**MAX9248**

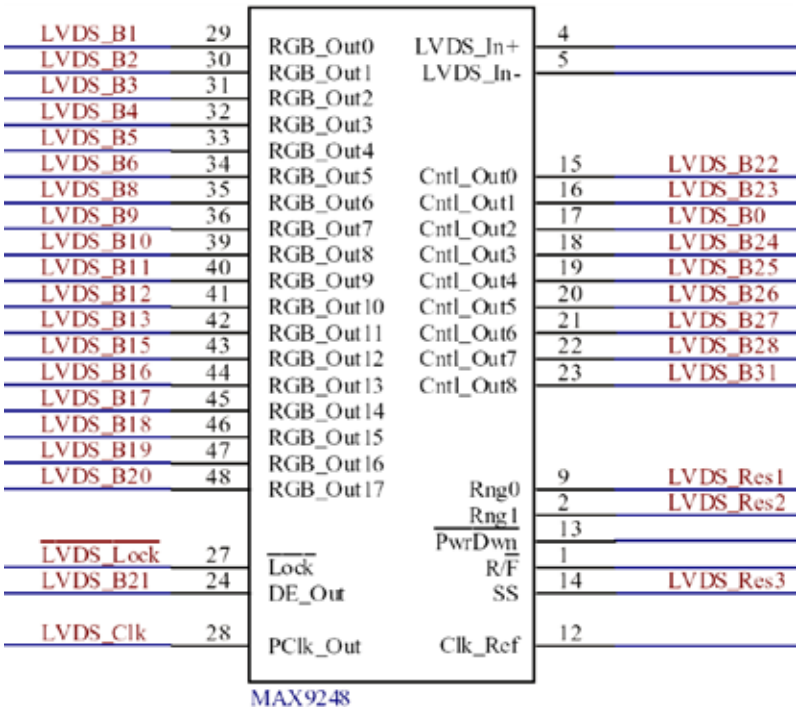



*Figure 2-13:*
*MAX9248 Pin assignment*

For this receiver top board the crystal frequency must be matched to the transmitter. For this reason, please indicate the pixel frequency of the transmitter when ordering.

**MAX9260**

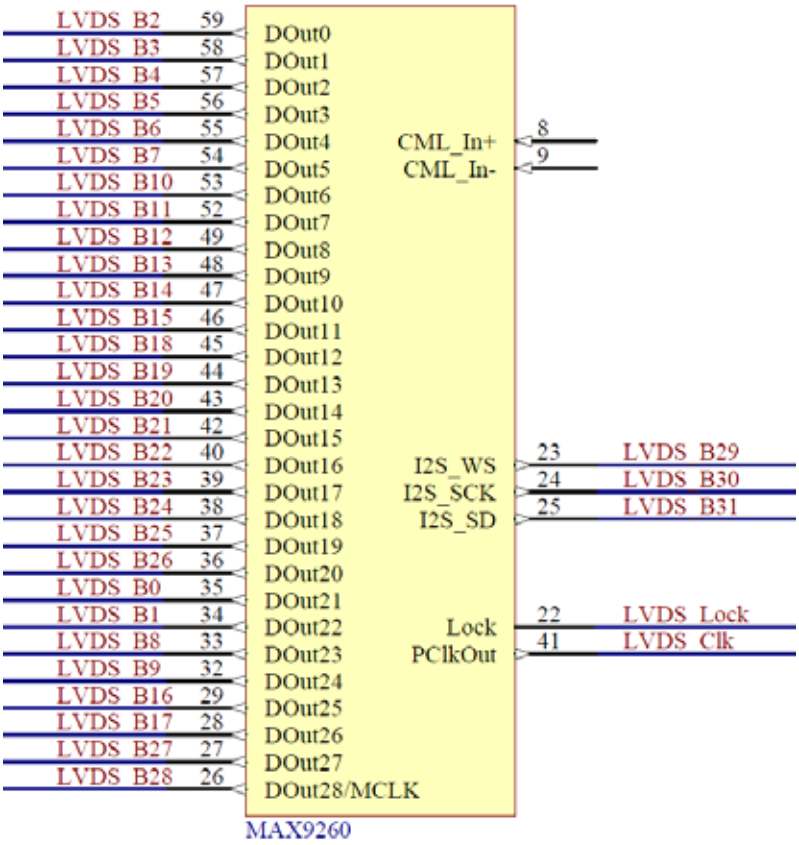



*Figure 2-14:*
*MAX9260 Pin assignment*

**CXB1458R**

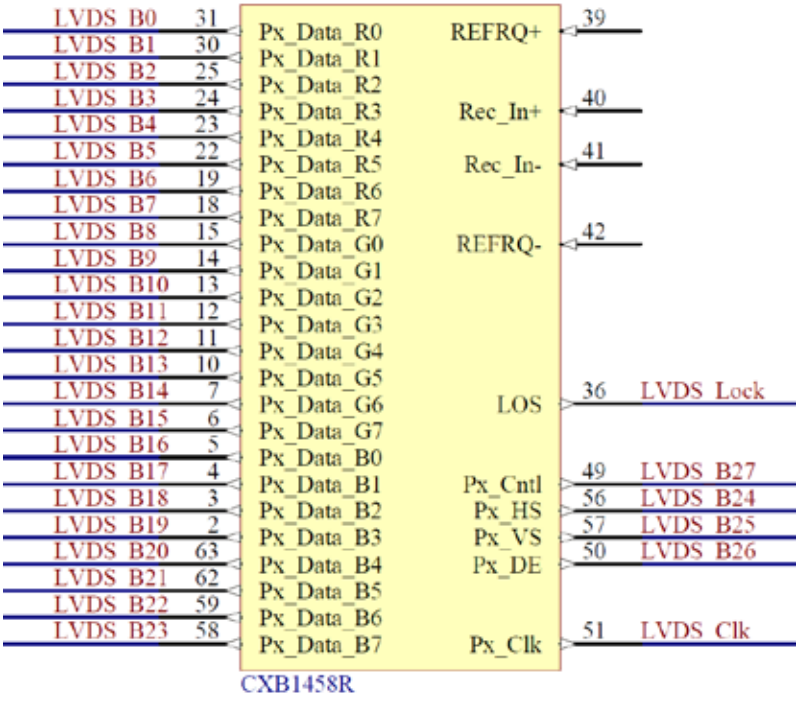



*Figure 2-15:*
*PCXB1458R Pin assignment*

# 2.4 Delivery Notes

USB 4120 boards/ basicCON 4120 devices are available in the following variants with the following accessories:

- USB 4120.10 with DS90UR124 receiver top board
- USB 4120.20 with DS90CF364 receiver top board
- USB 4120.30 with DS90CF906 receiver top board
- USB 4120.40 with INAP125R24APIX receiver top board
- USB 4120.50 with MAX9248 receiver top board (please indicate the pixel frequency when ordering)
- USB 4120.60 with MAX9260 receiver top board
- USB 4120.70 with CXB1458R receiver top board
- USB 4120.80 with DS90UB926 receiver top board

- basicCON 4120.10 with DS90UR124 receiver top board
- basicCON 4120.20 with DS90CF364 receiver top board
- basicCON 4120.30 with DS90CF906 receiver top board
- basicCON 4120.40 with INAP125R24APIX receiver top board
- basicCON 4120.50 with MAX9248 receiver top board (please indicate the pixel frequency when ordering)
- basicCON 4120.60 with MAX9260 receiver top board
- basicCON 4120.70 with CXB1458R receiver top board
- basicCON 4120.80 with DS90UB926 receiver top board

- ST 4120.10 DS90UR124 receiver top board
- ST 4120.20 DS90CF364 receiver top board
- ST 4120.30 DS90CF906 receiver top board
- ST 4120.40 INAP125R24APIX receiver top board
- ST 4120.50 MAX9248 receiver top board (please indicate the pixel frequency when ordering)
- ST 4120.60 MAX9260 receiver top board
- ST 4120.70 CXV1458 receiver top board
- ST 4120.80 DS90UB926 receiver top board

# 3 Control Software

There are two ways to integrate USB 4120/ basicCON 4120 hardware in your own applications:

- Programming via G-API
- Programming via DLL Functions

## 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 only required if projects are not processed by the GOEPEL G-API.

With the function calls described in this section, USB 4120 boards or basicCON 4120 devices can be addressed directly from several standard languages (VisualC++, CVI).

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

In the following, all information given for USB 4120 boards is also valid for basicCON 4120 devices.

For the used structures, data types and error codes refer to the *GUSB_Platform.h* C Header file – you find it on the delivered CD.

### Windows Device Driver

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

### 3.2.1 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 *pVersion
                              unsigned int             LengthInByte)
```

**Parameters:**

pVersion

Pointer to a data structure (storage area)
For the structure, see the *GUSB_Platform.h* file on the delivered CD

LengthInByte

Size of the storage area pVersion 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 pVersion 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 pVersion 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.

### 3.2.2 DLL_Info

The GUSB_Platform_DLL_Info is used to query information regarding the DLL.

**Format:**

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

**Parameter**

DLLinformation

Pointer 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*.

**Examples:**

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

### 3.2.3 Xilinx_ Version

The GUSB_Platform_Xilinx_Version function allows reading out the version number 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 4120 = 19)

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.

**Examples:**

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

### 3.2.4 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 4120 = 19)

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).

pWrite

Pointer to the write data area
(currently max. 512 byte per command,
see also USB Controller Control Commands)

DataLength

Number of bytes of the data to be written in pWrite (to be given)

**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.

### 3.2.5 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 4120 = 19)

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).

pRead

Pointer 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. 512 bytes per response,
see also USB Controller Control Commands.)

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!

## 3.2.6 IdentInfo

The GUSB_Platform_4120_IdentInfo function provides general information of the USB 4120 board indicated by DeviceNumber.

**Format:**

```
int GUSB_Platform_4120_IdentInfo(unsigned int                    DeviceNumber,
                                 t_GUSB_Platform_4120_IdentInfo *IdentData,
                                 unsigned int                   *LengthInByte,
                                 unsigned int                   *State)
```

**Parameters:**

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).

IdentData

Pointer to a data structure (storage area)

For the structure, see the *GUSB_Platform.h* file on the delivered CD

LengthInByte

Size of the storage area IdentData is pointing to, in bytes

After function execution: Number of bytes actually read

State

Bit selective status information of the board

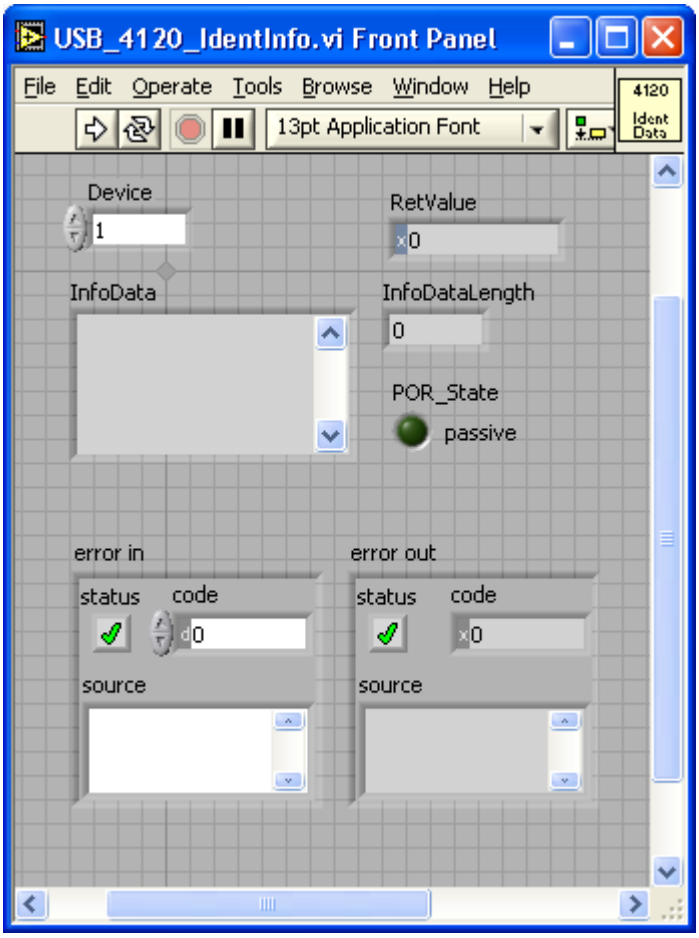
Bit 0: PowerOnReset
0 -> passive
1 -> active

Bits 1..31: not used

**Description:**

The function provides general information as well as configuration states of the device.

Belonging LabVIEW – VI:



### 3.2.7 Reset

The GUSB_Platform_4120_Reset function resets all FPGA registers of the USB 4120 board indicated by DeviceNumber to their default values.

**Format:**

```
int GUSB_Platform_4120_Reset(unsigned int DeviceNumber)
```

**Parameter:**

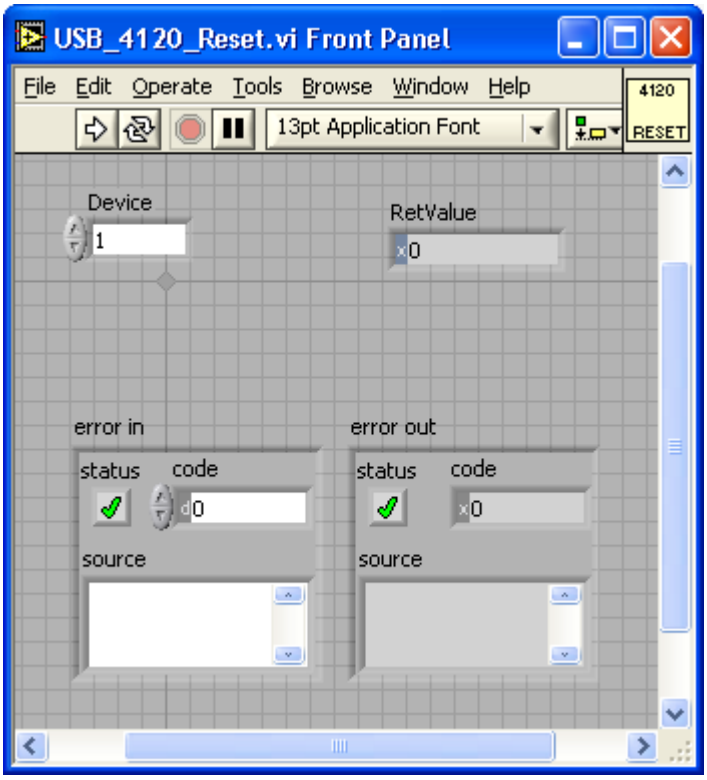
DeviceNumber

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).

**Description:**

All register values of the FPGA are reset.

Belonging LabVIEW – VI:



### 3.2.8 Get HardwareInfo

The GUSB_Platform_4120_GetHardwareInfo function provides general information and states regarding the USB 4120 board indicated by DeviceNumber.

**Format:**

```
int USB_Platform_4120_GetHardwareInfo(unsigned int  DeviceNumber,
                                      unsigned int  *VHDL_Version,
                                      unsigned int  *HW_Version,
                                      unsigned char *Deserializer,
                                      unsigned char *LockState,
                                      unsigned char *ConfigError)
```

**Parameters:**

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).

VHDL_Version

Indicates the VHDL design version of the FPGA
(Example: value 120 -> version 1.20)

HW_Version

Indicates the version of the board layout
(Example: value 11 -> version 1.1)

Deserializer

Indicates the type of the receiver IC on the receiver top board
(see *GUSB_Platform.h* file)

LockState

Indicates whether an LVDS signal is on and whether the deserializer is locked on this signal
(see *GUSB_Platform.h* file)

ConfigError

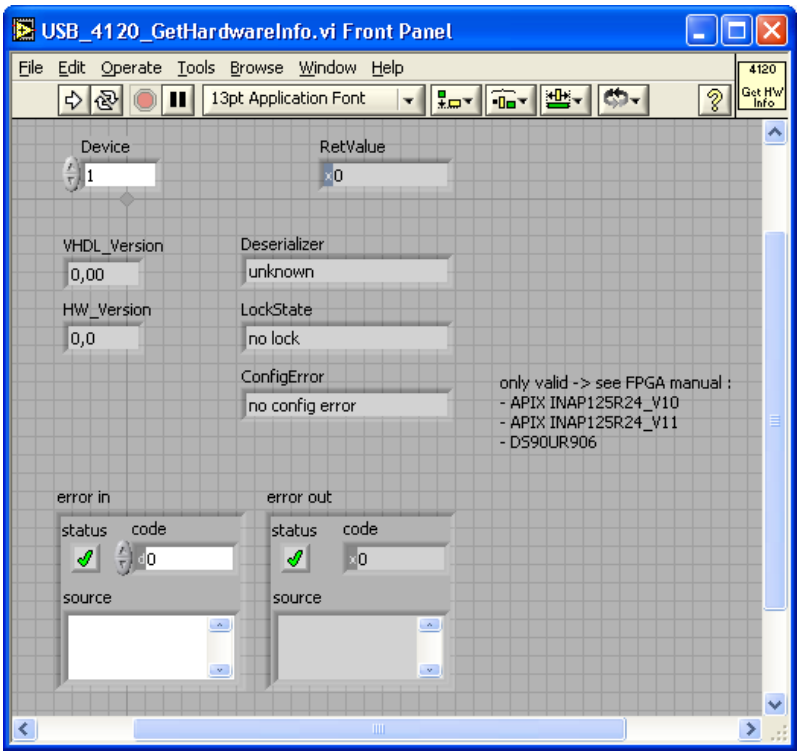
Indicates whether an error occurred when configuring the deserializer (GUSB_Platform_4120_DeserializerConfigurationVector())
(only valid for deserializer types APIX INAP125R24_V10, APIX INAP125R24_V11 and DS90UR906)
(see *GUSB_Platform.h* file)

**Description:**

At present the following types of deserializers are available:

- DS90UR124
- DS90CF364
- INAP125R24_V10
- INAP125R24_V11
- MAX9248
- DS90UR906
- CXB1458R
- MAX9260

Belonging LabVIEW – VI:



### 3.2.9 SyncInfo

The GUSB_Platform_4120_SyncInfo function provides information regarding the die synchronization characteristics of the LVDS data stream of the USB 4120 board indicated by DeviceNumber.

**Format:**

```
int GUSB_Platform_4120_SyncInfo(unsigned int   DeviceNumber,
                                unsigned int   *PixelClock,
                                unsigned short *DataEnableNumberOfColumn,
                                unsigned short *DataEnableNumberOfLines,
                                unsigned short *HSyncWidth,
                                unsigned short *VSyncWidth,
                                unsigned short *HSyncPeriod,
                                unsigned int   *VSyncPeriod)
```

**Parameters:**

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).

PixelClock

Indicates the clock frequency in kHz of the pixels transfer (precision at least 50kHz)

DataEnableNumberOfColumn

Indicates the number of columns for an active DataEnable signal (corresponds to horizontal resolution)

The definition of the active DataEnable signal is set in GUSB_Platform_4120_Init() (see Init) with the SignalLevel.DataControls parameter (low-active or high-active).

(only valid in case the RoutingControls.DataEnable parameter was set unequal to K_4120_LVDS_BIT_DISABLE in GUSB_Platform_4120_Init())

DataEnableNumberOfLines

Indicates the number of lines for an active DataEnable signal (corresponds to vertical resolution).

The definition of the active DataEnable signals is set in GUSB_Platform_4120_Init() (see Init) with the SignalLevel.DataControls parameter (low-active or high-active).

(only valid in case the RoutingControls.DataEnable parameter was set unequal to K_4120_LVDS_BIT_DISABLE in GUSB_Platform_4120_Init())

HSyncWidth

Indicates the number of clocks during that the horizontal synchronization signal is active

The definition of the active Hsync signal is set in GUSB_Platform_4120_Init() (see Init) with the SignalLevel.HSync parameter (low-active or high-active).

VSyncWidth

Indicates the number of clocks during that the vertical synchronization signal is active

The definition of the active Vsync signals is set in GUSB_Platform_4120_Init() (see Init) with the SignalLevel.VSync parameter (low-active or high-active).

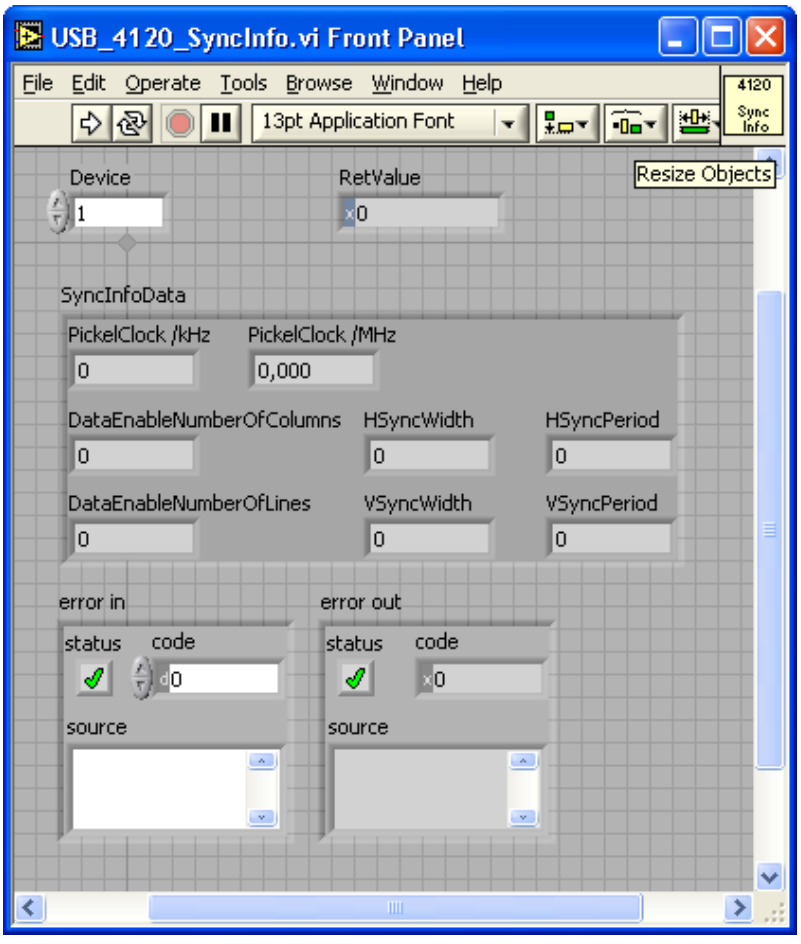
HSyncPeriod

Indicates the number of clocks between two equal horizontal synchronization edges

VSyncPeriod

Indicates the number of clocks between two equal vertical synchronization edges

Belonging LabVIEW – VI:



## 3.2.10 InitInfo

The GUSB_Platform_4120_InitInfo function provides information regarding the initialization state of the USB 4120 board indicated by DeviceNumber.

**Format:**

```
int GUSB_Platform_4120_InitInfo(unsigned int                DeviceNumber,
                  t_GUSB_Platform_4120_Init_Resolution      *InitResolution,
                  t_GUSB_Platform_4120_Init_SyncWidth       *SyncWidth,
                  t_GUSB_Platform_4120_Init_SignalLevel     *SignalLevel,
                  t_GUSB_Platform_4120_Init_EdgeSelection   *EdgeSelection,
                  t_GUSB_Platform_4120_Init_RoutingColours  *RoutingColors,
                  t_GUSB_Platform_4120_Init_RoutingControls *RoutingControls)
```

**Parameters:**

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).

InitResolution.NumberOfColumns

Indicates the set number of picture columns (horizontal resolution)

InitResolution.NumberOfLines

Indicates the set number of picture lines (vertical resolution)

SyncWidth.NumberOfColumnsSync

Indicates the set number of empty columns in the horizontal blanking following the active horizontal synchronization signal
(only valid in case the RoutingControls.DataEnable parameter was set equal to K_4120_LVDS_BIT_DISABLE in GUSB_Platform_4120_Init() (see Init)

SyncWidth.NumberOfLinesSync

Indicates the set number of empty lines in the vertical blanking following the active vertical synchronization signal
(only valid in case the RoutingControls.DataEnable parameter was set equal to K_4120_LVDS_BIT_DISABLE in GUSB_Platform_4120_Init() (see Init)

SignalLevel.HSync

Indicates the set polarity of the horizontal synchronization signal

0 -> low-active

1 -> high-active

SignalLevel.VSync

Indicates the set polarity of the vertical synchronization signal

0 -> low-active
1 -> high-active

SignalLevel.DataEnable

Indicates the set polarity of the DataEnable signal
0 -> low-active
1 -> high-active

EdgesSelection.Clock

Indicates the set edge of the clock for reading-in data
0 -> falling edge
1 -> rising edge

RoutingColors.B0...RoutingColors.B7

Indicates the set source (LVDS bit number)
of color bits Blue 0..7

RoutingColors.G0...RoutingColors.G7

Indicates the set source (LVDS bit number)
of the color bits Green 0..7

RoutingColors.R0...RoutingColors.R7

Indicates the set source (LVDS bit number)
of the color bits Red 0..7

RoutingControls.VSync

Indicates the set source (LVDS bit number)
of the vertical synchronization signal

RoutingControls.HSync

Indicates the set source (LVDS bit number)
of the horizontal synchronization signal

RoutingControls.DataEnable

Indicates the set source (LVDS bit number)
of the DataEnable signal

RoutingControls.C3...RoutingControls.C7

Indicates the set source (LVDS bit number)
of the control bits 3..7 (optional control bits) an

**Description:**

The definitions of the applied structures can be found in the *GUSB_Platform.h* file on the supplied CD.

The elements of the structures RoutingColors and RoutingControl indicate the current assignment of the LVDS bits:

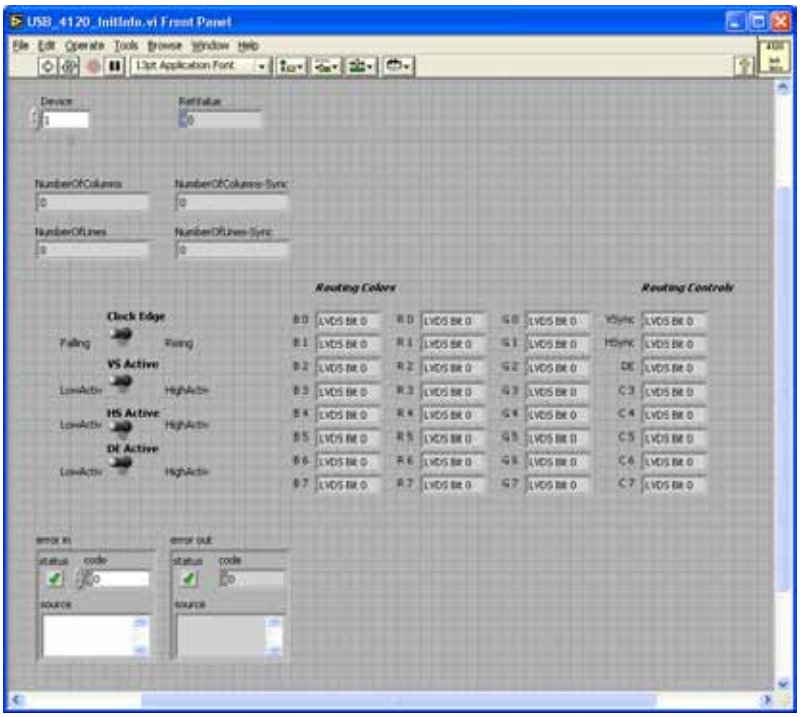
0 -> LVDS-Bit 0
1 -> LVDS-Bit 1

31 -> LVDS-Bit 31
32 -> not used

254 -> not used
255 -> disable

Please refer to the hardware description, section Receiver Top Boards, regarding the assignment of the LVDS bits

Belonging LabVIEW – VI:



### 3.2.11 Init

The GUSB_Platform_4120_Init function releases a RESET and configures the USB 4120 board indicated by DeviceNumber for matching with the picture source.

**Format:**

```
int GUSB_Platform_4120_Init(unsigned int                    DeviceNumber,
                  t_GUSB_Platform_4120_Init_Resolution      *InitResolution,
                  t_GUSB_Platform_4120_Init_SyncWidth       *SyncWidth,
                  t_GUSB_Platform_4120_Init_SignalLevel     *SignalLevel,
                  t_GUSB_Platform_4120_Init_EdgeSelection   *EdgeSelection,
                  t_GUSB_Platform_4120_Init_RoutingColours  *RoutingColors,
                  t_GUSB_Platform_4120_Init_RoutingControls *RoutingControls)
```

**Parameters:**

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).

InitResolution.NumberOfColumns

Defines the number of picture columns (horizontal resolution)

InitResolution.NumberOfLines

Defines the number of picture lines (vertical resolution)

SyncWidth.NumberOfColumnsSync

Defines the number of empty columns in the horizontal blanking after the active horizontal synchronization signal
(only valid if the RoutingControls.DataEnable parameter was set equal to K_4120_LVDS_BIT_DISABLE)

SyncWidth.NumberOfLinesSync

Defines the number of empty columns in the vertical blanking after the active vertical synchronization signal
(only valid if the RoutingControls.DataEnable parameter was set equal to K_4120_LVDS_BIT_DISABLE)

SignalLevel.HSync

Sets the polarity of the horizontal synchronization signal
0 -> low-active
1 -> high-active

SignalLevel.VSync

Sets the polarity of the vertical synchronization signal
0 -> low-active
1 -> high-active

SignalLevel.DataEnable

Sets the polarity of the DataEnable signal
0 -> low-active
1 -> high-active

EdgeSelection.Clock

Indicates the set edge of the clock for reading-in data

Sets the edge of the clock for reading-in data
0 -> falling edge
1 -> rising edge

RoutingColors.B0...RoutingColors.B7

Sets the source (LVDS bit number) of the Blue 0...7 color bits

RoutingColors.G0...RoutingColors.G7

Sets the source (LVDS bit number) of the Green 0...7 color bits

RoutingColors.R0...RoutingColors.R7

Sets the source (LVDS bit number) of the Red 0...7 color bits

RoutingControls.VSync

Sets the source (LVDS bit number)
of the vertical synchronization signal

RoutingControls.HSync

Sets the source (LVDS bit number)
of the horizontal synchronization signal

RoutingControls.DataEnable

Sets the source (LVDS bit number) of the DataEnable signal

RoutingControls.C3...RoutingControls.C7

Sets the source (LVDS bit number) of the Control bits 3...7
(optional control bits)

**Description:**

The definitions of the applied structures can be found in the *GUSB_Platform.h* file on the supplied CD.

The elements of the structures RoutingColors and RoutingControl indicate the current assignment of the LVDS bits:

0 -> LVDS-Bit 0
1 -> LVDS-Bit 1

31 -> LVDS-Bit 31
32 -> not used

254 -> not used
255 -> disable

Please refer to the hardware description, section Receiver Top Boards, regarding the assignment of the LVDS bits.

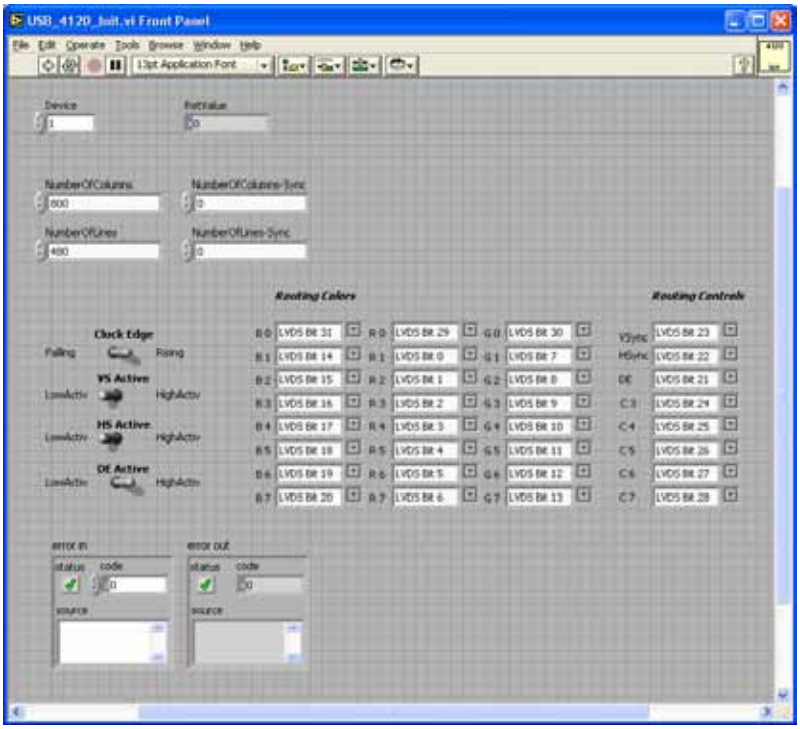
In the case a bit assignment for the RoutingColors (RGB) is initialized with disable, the frame grabber sets this bit for the grabbed picture data to the fixed value of 1.

In the case a bit assignment for the RoutingControls is initialized with disable, the frame grabber sets this bit for the grabbed picture data to the fixed value of 0.

Structure of the 32 data bits of a pixel:

| Bits | 31..27 | 26 | 25 | 24 | 23..16 | 15..8 | 7..0 |
|---|---|---|---|---|---|---|---|
| Content | C7..C3 | DE | HS | VS | R7..R0 | G7..G0 | B7..B0 |
| | Control signals | | | | Color information | | |

Belonging LabVIEW – VI:



## 3.2.12 Set ClockSource

The GUSB_Platform_4120_SetClockSource function sets the LVDS clock source of the USB 4120 board indicated by DeviceNumber.

**Format:**

```
int GUSB_Platform_4120_SetClockSource(unsigned int  DeviceNumber,
                                      unsigned char Mode)
```

**Parameters:**

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).

Mode

Source of the LVDS clock used by the frame grabber:

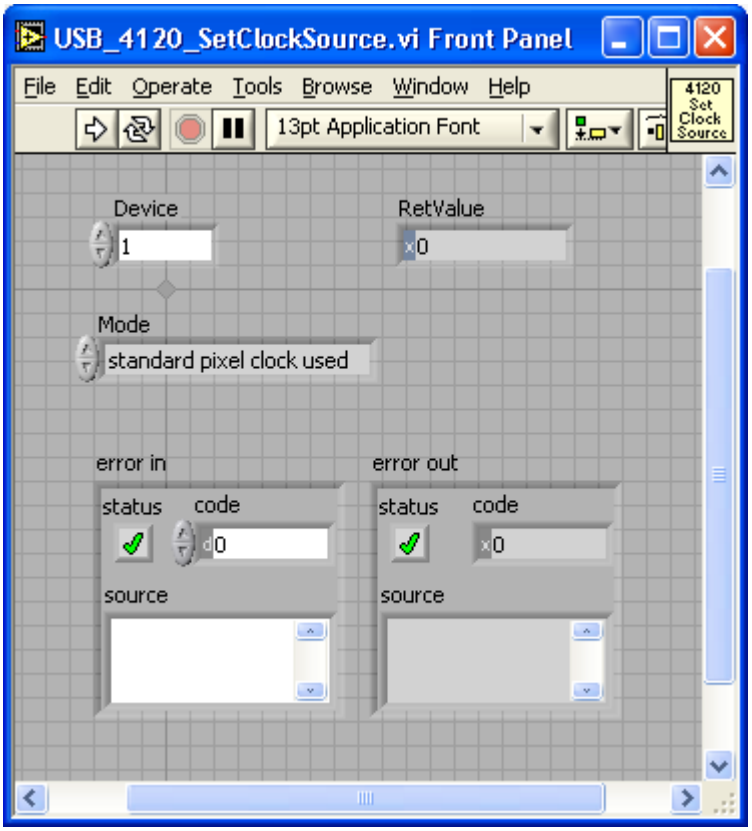
0 -> Standard Pixel Clock

1 -> LVDS-Bit 19

**Description:**

Switching the source of the LDVS clock is a special board function. Basic setting is using the Standard Pixel Clock.

Belonging LabVIEW – VI:



## 3.2.13 Set BitMask

The GUSB_Platform_4120_SetBitMask function defines the mask used for the internal picture comparison of the USB 4120 board indicated by DeviceNumber.

**Format:**

```
int GUSB_Platform_4120_SetBitMask(unsigned int  DeviceNumber,
                                  unsigned char BitMask)
```

**Parameters:**

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).

BitMask

Bits set to “0” in the mask are not considered for the pixel comparison.

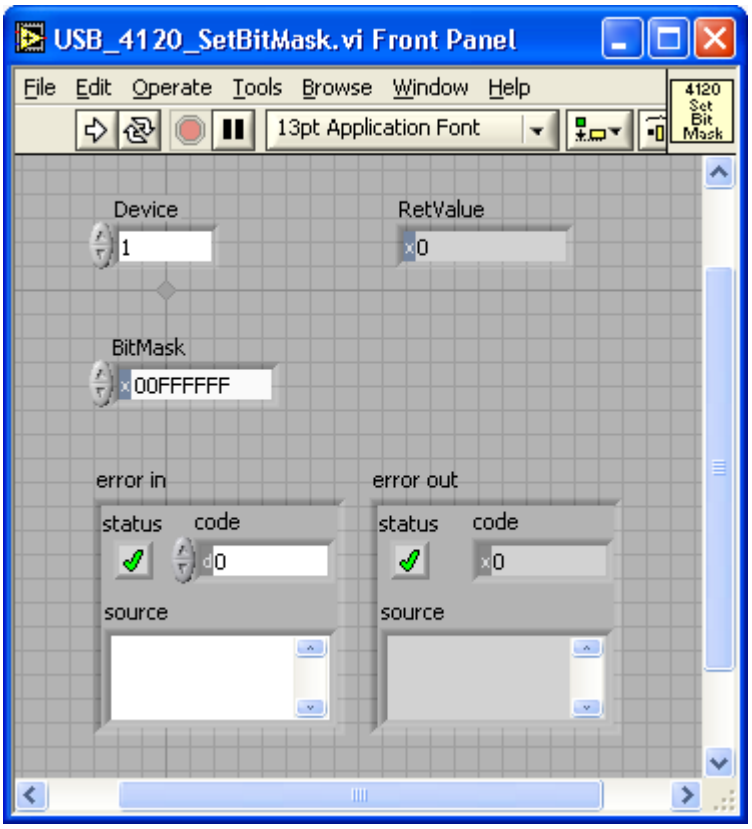
**Description:**

The function masks bits for the internal picture comparison.

Meaning of the individual bits:

| Bits | 31..27 | 26 | 25 | 24 | 23..16 | 15..8 | 7..0 |
|---|---|---|---|---|---|---|---|
| Content | C7..C3 | DE | HS | VS | R7..R0 | G7..G0 | B7..B0 |
| | Control signals | | | | Color information | | |

Belonging LabVIEW – VI:



## 3.2.14 External TriggerMode_Start

The GUSB_Platform_4120_ExternalTriggerMode_Start function activates the external trigger mode of the USB 4120 board indicated by DeviceNumber.

**Format:**

```
int GUSB_Platform_4120_ExternalTriggerMode_Start(unsigned int  DeviceNumber,
                                                 unsigned char TriggerLevel)
```

**Parameters:**

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).

TriggerLevel

Sets the polarity of the trigger signal

0 -> low-active
1 -> high-active

**Description:**

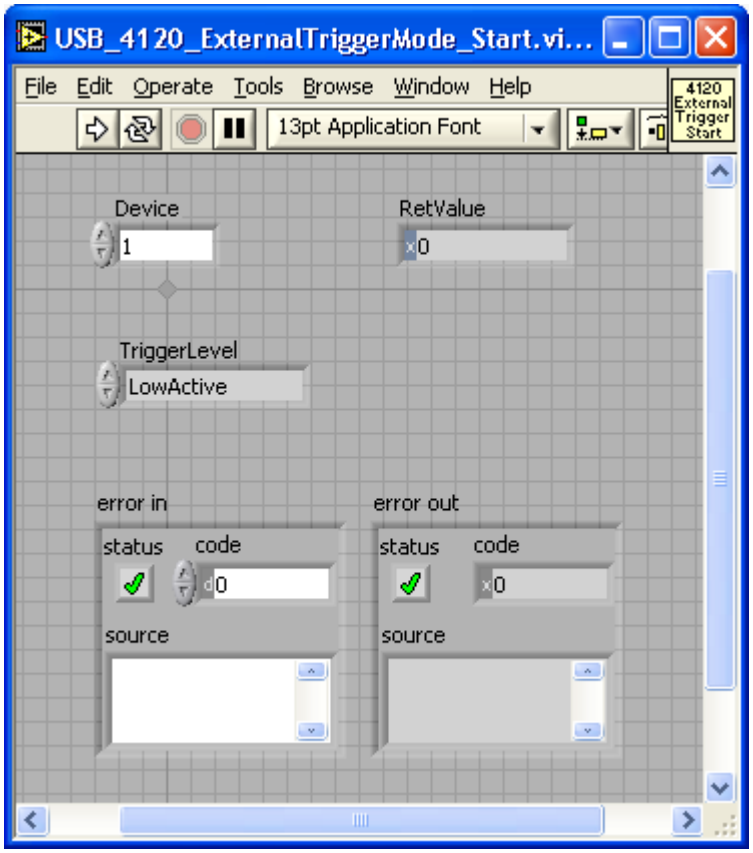
The frame grabber can be triggered external via D-Sub pin 3.

After the external triggering, the next picture is grabbed, stored on the board and compared with the also stored reference picture, considering the configured bit mask (see Set BitMask).

Prerequisite is the error-free configuration (GUSB_Platform_4120_Init()) and storing a reference picture on the board (GUSB_Platform_4120_LoadReferenceFromBuffer(), GUSB_Platform_4120_LoadReferenceFromFile()).

The advantage of this mode is real time triggering and the possibility to operate the board without PC or Laptop. In this case power supply must be assured further on (see Hardware/ Digital I/O Interface chapter).

Belonging LabVIEW – VI:



## 3.2.15 External TriggerMode_Stop

The GUSB_Platform_4120_ExternalTriggerMode_Stop function deactivates the external trigger mode of the USB 4120 board indicated by DeviceNumber.

**Format:**

```
int GUSB_Platform_4120_ExternalTriggerMode_Stop(unsigned int DeviceNumber)
```

**Parameter:**

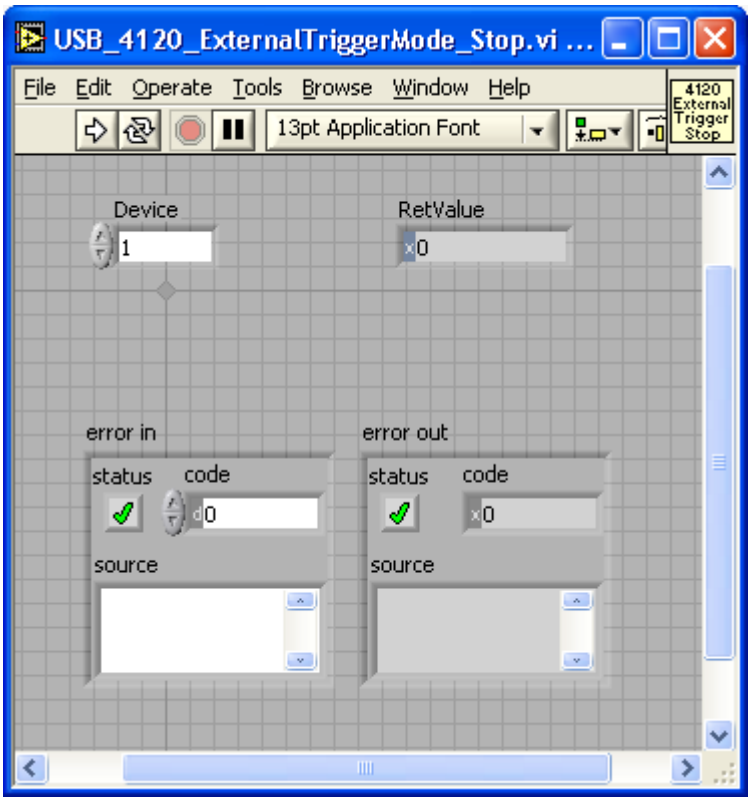
DeviceNumber

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).

**Description:**

Stops the external trigger mode

Belonging LabVIEW – VI:



## 3.2.16 External TriggerMode_ GetState

The GUSB_Platform_4120_ExternalTriggerMode_GetState indicates the state of the external trigger functionality of the USB 4120 board indicated by DeviceNumber.

**Format:**

```
int GUSB_Platform_4120_ExternalTriggerMode_GetState(unsigned int DeviceNumber,
                                                    unsigned int *RspFlags)
```

**Parameters:**

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).

RspFlags

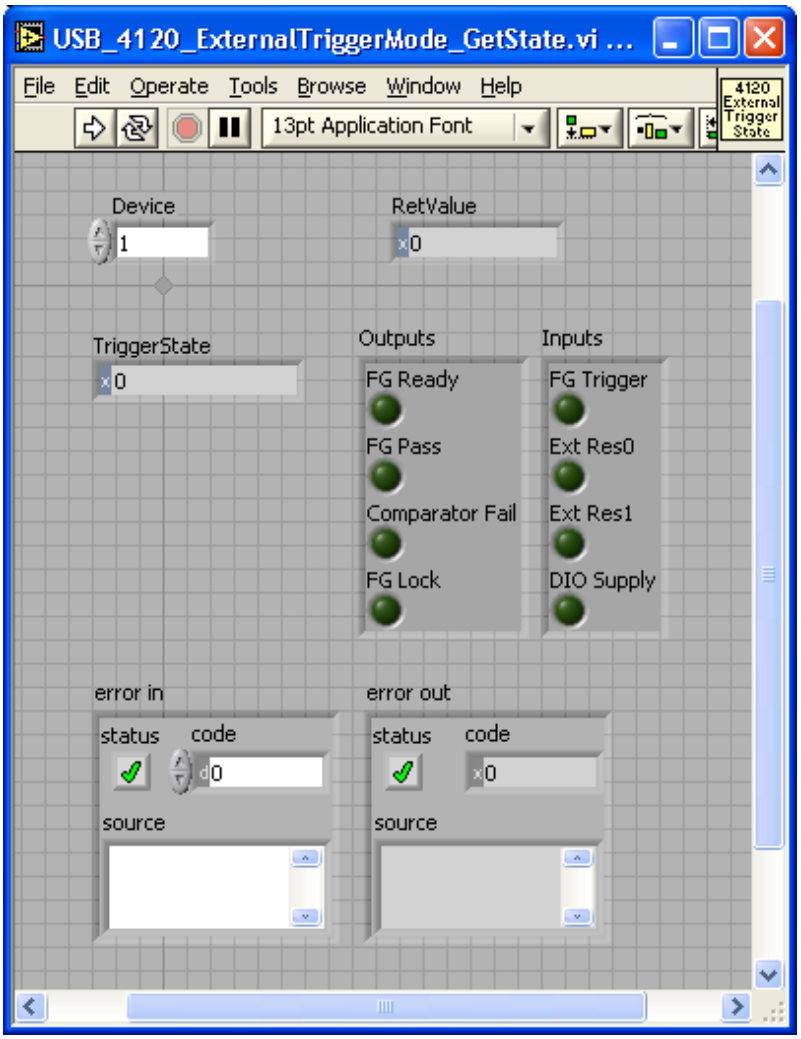
Indicates the current state of the eight DIO pins of the D-Sub connector:

| Bit | Indication | Meaning |
|---|---|---|
| 0 | Frame grabber Ready Output (D-Sub Pin 1) | Bit 0 = 1 -> Frame grabber ready (External Trigger Mode active, Reference picture loaded) |
| 1 | Frame grabber Pass Output (D-Sub Pin 6) | Bit 1= 1 -> Picture is error free grabbed and compared |
| 2 | Comparator Failure Output (D-Sub Pin 2) | Bit 2= 1 -> Comparator has detected at least one picture error |
| 3 | Frame grabber Lock Output (D-Sub Pin 7) | Bit 3= 1 -> Frame grabber locked to LVDS signal |
| 4 | External Trigger Input (D-Sub Pin 3) | Bit 4= 1 -> External Trigger active (picture reception and automatic comparison are started) |
| 5 | External Input Reserve 0 (D-Sub Pin 8) | Reserve 0 |
| 6 | External Input Reserve 1 (D-Sub Pin 4) | Reserve 1 |
| 7 | External Vcc Input (D-Sub Pin 5) | Bit 7= 1 -> External power supply active, required if the corresponding toggle is set to DIO (see Figure 2-5Figure ) |
| 8..31 | not used | Not assigned |

**Description:**

The function indicates the current state of the external trigger functionality.

Belonging LabVIEW – VI:



### 3.2.17 Capture

The GUSB_Platform_4120_Capture releases a picture grabbing by the USB 4120 board indicated by DeviceNumber.

**Format:**

```
int GUSB_Platform_4120_Capture(unsigned int   DeviceNumber,
                               unsigned short *ErrorLine,
                               unsigned short *ErrorColumn)
```

**Parameters:**

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).

ErrorLine

Indicates the number of the picture line in that a synchronization error has been occurred

0 -> no error

Synchronization errors can occur by wrong initialization or faulty HSync, VSync and DE signals.

ErrorColumn

Indicates the number of the picture column in that a synchronization error has been occurred

0 -> no error

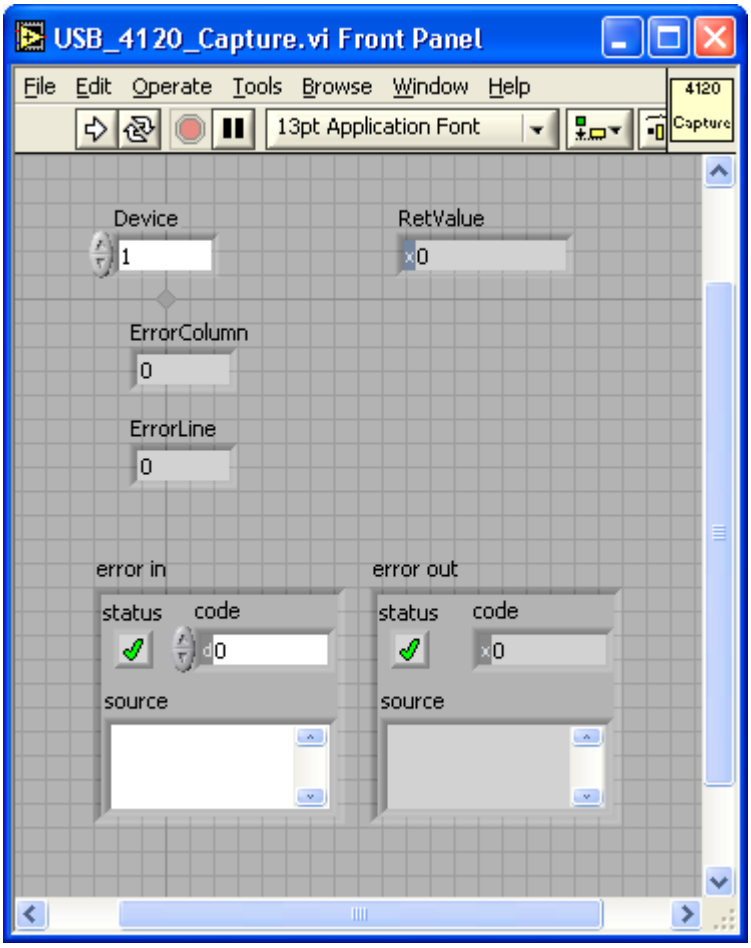
Synchronization errors can occur by wrong initialization or faulty HSync, VSync and DE signals.

**Description:**

This function releases a picture grabbing.

Picture data is stored in the frame grabber.
This way it is available for the internal comparison with a reference picture (function GUSB_Platform_4120_Compare())
or can be read out by GUSB_Platform_4120_ReadCapturedFrameToBuffer() or GUSB_Platform_4120_ReadCapturedFrameToFile() (see also Compare, Read CapturedFrame ToBuffer and Read CapturedFrame ToFile).

Belonging LabVIEW – VI:



### 3.2.18 Compare

The GUSB_Platform_4120_Compare function compares the picture grabbed last with the reference picture in the USB 4120 board indicated by DeviceNumber.

**Format:**

```
int GUSB_Platform_4120_Compare(unsigned int   DeviceNumber,
                               unsigned short *ErrorCount)
```

**Parameters:**

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).

ErrorCount

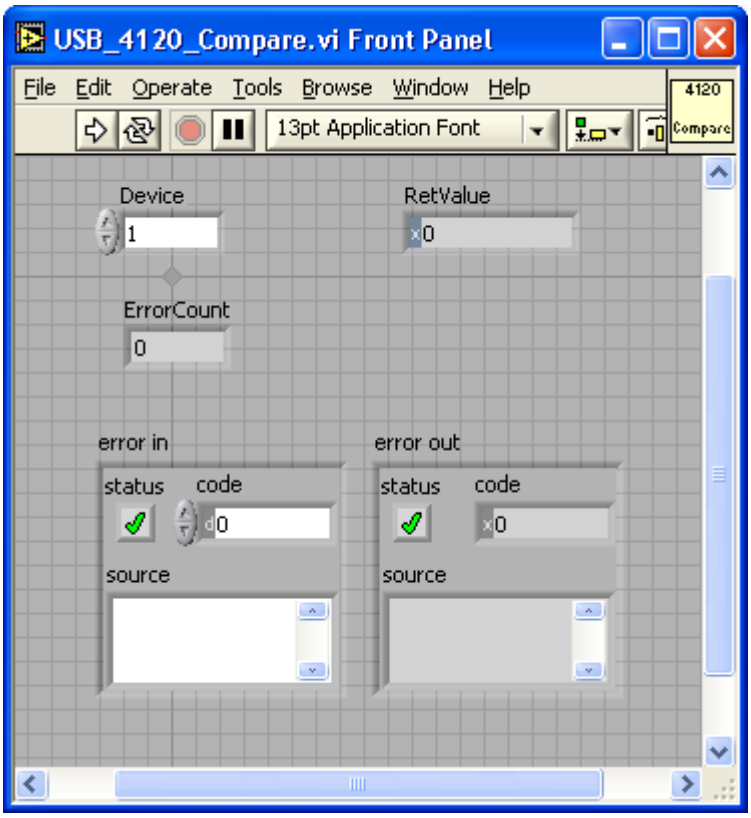
Indicates the number of detected pixel errors

0 -> no error occurred

**Description:**

The function compares the last error-free grabbed picture with the picture stored on the frame grabber, considering the bit mask loaded by GUSB_Platform_4120_SetBitMask() (see Set BitMask).

Belonging LabVIEW – VI:



## 3.2.19 Capture ToBuffer

The GUSB_Platform_4120_CaptureToBuffer function releases picture grabbing with the USB 4120 board indicated by DeviceNumber and transfers the grabbed picture data to the provided data buffer.

**Format:**

```
int GUSB_Platform_4120_CaptureToBuffer(unsigned int   DeviceNumber,
                                       unsigned char  *Data,
                                       unsigned int   *Length,
                                       unsigned short *ErrorLine,
                                       unsigned short *ErrorColumn)
```

**Parameters:**

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).

Data

Pointer to a data buffer

The size of the data buffer provided by the application must be at least K_MAX_BYTE_RD_EXTENDED_4120 (see *GUSB_Platform* file).

One pixel consists of 32 bits. The frame grabber grabs line by line from the upper left to the lower right picture corner.
Correspondingly, data is stored in the buffer.

Structure of the 32 data bits of a pixel:

| Bits | 31..27 | 26 | 25 | 24 | 23..16 | 15..8 | 7..0 |
|---|---|---|---|---|---|---|---|
| Content | C7..C3 | DE | HS | VS | R7..R0 | G7..G0 | B7..B0 |
| | Control signals | | | | Color information | | |

Length

Size of the data buffer Data is pointing to, in bytes

After executing the function: Number of bytes actually written to the data buffer

ErrorLine

Indicates the number of the picture line in that a synchronization error has been occurred

0 -> no error

Synchronization errors can occur by wrong initialization or faulty HSync, VSync and DE signals.

ErrorColumn

Indicates the number of the picture column in that a synchronization error has been occurred

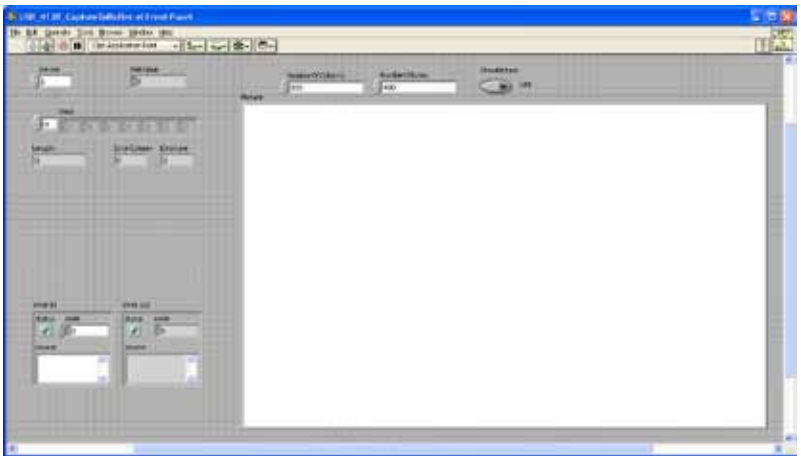
0 -> no error

Synchronization errors can occur by wrong initialization or faulty HSync, VSync and DE signals.

**Description:**

This function releases a picture grabbing.
Data is stored in the provided data buffer.

Belonging LabVIEW – VI:



Here there is the additional possibility to represent the data provided by the frame grabber as a Picture in the corresponding field.

For ShowPicture = ON, the correct values of the grabbed picture must be entered in the NumberOfColums and NumberOfLines fields.

## 3.2.20 Capture ToFile

The GUSB_Platform_4120_CaptureToFile function releases picture grabbing with the USB 4120 board indicated by DeviceNumber and returns the picture data as a BMP file.

**Format:**

```
int GUSB_Platform_4120_CaptureToFile(unsigned int   DeviceNumber,
                                     unsigned short Width,
                                     unsigned short Height,
                                     char           *Path,
                                     unsigned short *ErrorLine,
                                     unsigned short *ErrorColumn)
```

**Parameters:**

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).

Width

Number of picture columns (horizontal resolution)

(required for the internal conversion of the grabbed picture data to a BMP file)

Height

Number of picture lines (vertical resolution)

(required for the internal conversion of the grabbed picture data to a BMP file)

Path

Path for storing picture data

ErrorLine

Indicates the number of the picture line in that a synchronization error has been occurred

0 -> no error

Synchronization errors can occur by wrong initialization or faulty HSync, VSync and DE signals.

ErrorColumn

Indicates the number of the picture column in that a synchronization error has been occurred

0 -> no error

Synchronization errors can occur by wrong initialization or faulty HSync, VSync and DE signals.

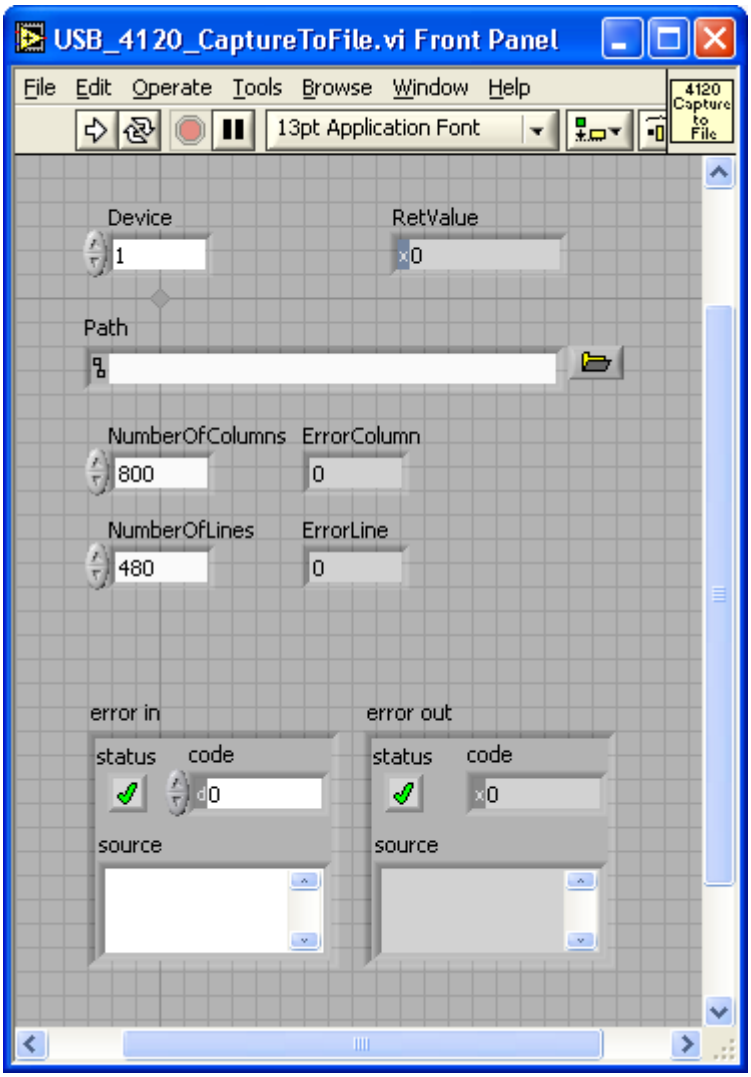
**Description:**

This function releases a picture grabbing.
Picture data is stored in the provided path as a BMP file.

Format of the BMP file:

- 32 bits per pixel
- 24 bits color depth

Belonging LabVIEW – VI:



## 3.2.21 Read CapturedFrame ToBuffer

The GUSB_Platform_4120_ReadCapturedFrameToBuffer function provides the data of the picture grabbed last of the USB 4120 board indicated by DeviceNumber and stores the date in a buffer.

**Format:**

```
int GUSB_Platform_4120_ReadCapturedFrameToBuffer(unsigned int  DeviceNumber,
                                                 unsigned char *Data,
                                                 unsigned int  *Length)
```

**Parameters:**

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).

Data

Pointer to a data buffer

The size of the data buffer provided by the application must be at least K_MAX_BYTE_RD_EXTENDED_4120 (see *GUSB_Platform* file).

One pixel consists of 32 bits. The frame grabber grabs line by line from the upper left to the lower right picture corner.
Correspondingly, data is stored in the buffer.

Structure of the 32 data bits of a pixel:

| Bits | 31..27 | 26 | 25 | 24 | 23..16 | 15..8 | 7..0 |
|---|---|---|---|---|---|---|---|
| Content | C7..C3 | DE | HS | VS | R7..R0 | G7..G0 | B7..B0 |
| | Control signals | | | | Color information | | |

Length

Size of the data buffer Data is pointing to, in Bytes

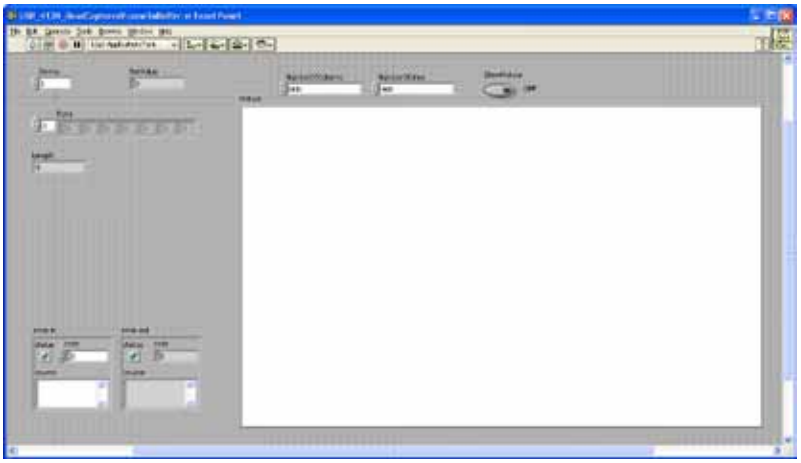
After executing the function: Number of bytes actually written to the data buffer

**Description:**

The function stores the data of the picture grabbed last in the provided path.

To get up-to-date picture date, before a picture grabbing must be released with GUSB_Platform_4120_Capture().

Belonging LabVIEW – VI:



Here there is the additional possibility to represent the data provided by the frame grabber as a Picture in the corresponding field.

For ShowPicture = ON, the correct values of the grabbed picture must be entered in the NumberOfColums and NumberOfLines fields.

## 3.2.22 Read CapturedFrame ToFile

The GUSB_Platform_4120_ReadCapturedFrameToFile function provides the data of the picture grabbed last of the USB 4120 board indicated by DeviceNumber and stores the date as a BMP file.

**Format:**

```
int GUSB_Platform_4120_ReadCapturedFrameToFile(unsigned int   DeviceNumber,
                                               unsigned short Width,
                                               unsigned short Height,
                                               char           *Path)
```

**Parameters:**

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).

Width

Number of picture columns (horizontal resolution)

(required for the internal conversion of the grabbed picture data to a BMP file)

Height

Number of picture lines (vertical resolution)

(required for the internal conversion of the grabbed picture data to a BMP file)

Path

Path the picture data is stored in

**Description:**

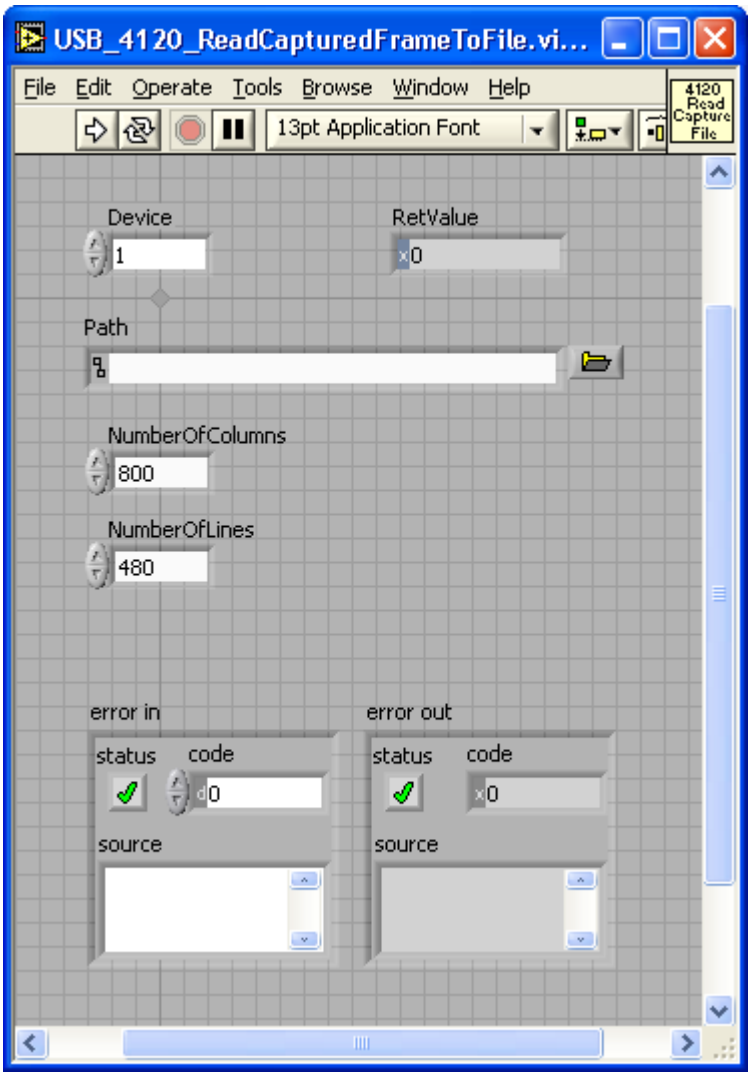
The function stores the data of the picture grabbed last as BMP file in the provided path.

To get up-to-date picture date, before a picture grabbing must be released with GUSB_Platform_4120_Capture().

Format of the BMP file:

- 32 bits per pixel
- 24 bits color depth

Belonging LabVIEW – VI:



## 3.2.23 Load Reference FromBuffer

The GUSB_Platform_4120_LoadReferenceFromBuffer function loads data of a reference picture in the USB 4120 board indicated by DeviceNumber.

**Format:**

```
int GUSB_Platform_4120_LoadReferenceFromBuffer(unsigned int  DeviceNumber,
                                               unsigned char *Data,
                                               unsigned int  Length)
```

**Parameters:**

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).

Data

Pointer to a data buffer

Picture data must not be greater sized than K_MAX_BYTE_RD_EXTENDED_4120 – 12 (see *GUSB_Platform.h* file).

One pixel consists of 32 bits. The frame grabber grabs line by line from the upper left to the lower right picture corner.
Correspondingly, data is stored in the buffer.

Structure of the 32 data bits of a pixel:

| Bits | 31..27 | 26 | 25 | 24 | 23..16 | 15..8 | 7..0 |
|---|---|---|---|---|---|---|---|
| Content | C7..C3 | DE | HS | VS | R7..R0 | G7..G0 | B7..B0 |
| | Control signals | | | | Color information | | |

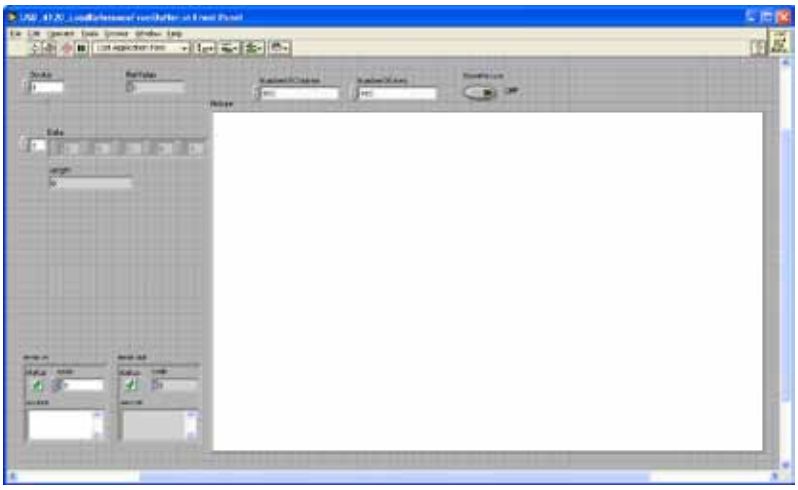
Length

Size of the data buffer Data is pointing to, in Bytes

**Description:**

The function loads picture data from a BMP file in the frame grabber, which deposits the data in an internal memory area.
This way this reference picture can be compared with a grabbed picture direct on the board.

Belonging LabVIEW – VI:



Here there is the additional possibility to represent the data provided by the frame grabber as a Picture in the corresponding field.

For ShowPicture = ON, the correct values of the grabbed picture must be entered in the NumberOfColums and NumberOfLines fields.

## 3.2.24 Load Reference FromFile

The GUSB_Platform_4120_LoadReferenceFromFile function loads data of a reference picture from a BMP file to the USB 4120 board indicated by DeviceNumber.

**Format:**

```
int GUSB_Platform_4120_LoadReferenceFromFile(unsigned int DeviceNumber,
                                             char         *Path)
```

**Parameters:**

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).

Path

Path the BMP picture data is stored in

Picture data must not be sized greater than K_MAX_BYTE_RD_EXTENDED_4120 – 12 (see *GUSB_Platform.h* file).

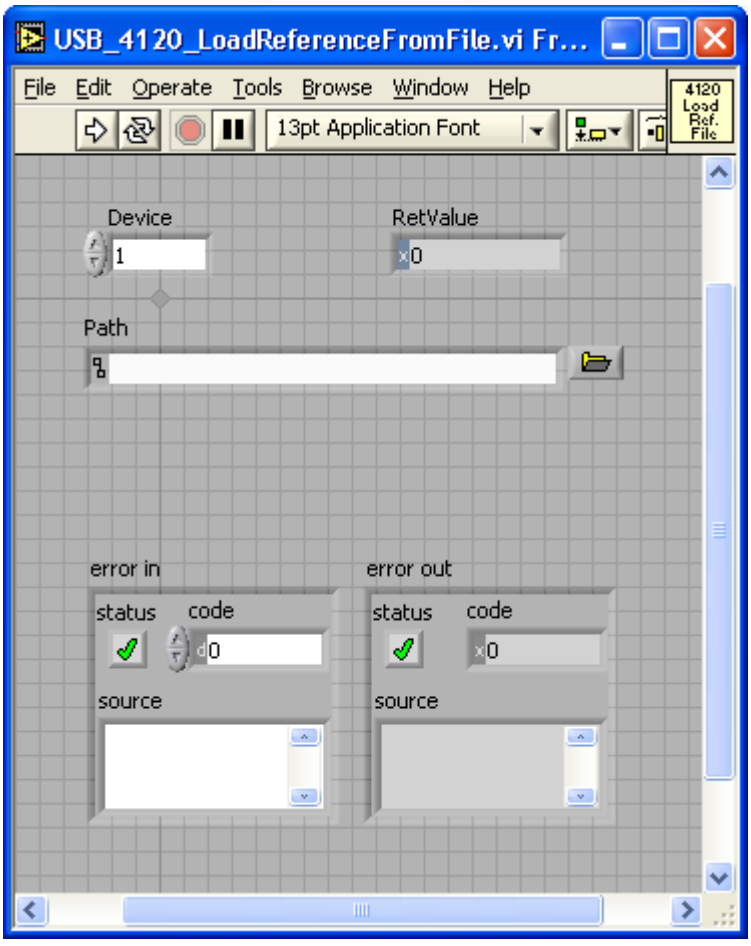
**Description:**

The function loads picture data from a BMP file in the frame grabber, which deposits the data in an internal memory area.
This way this reference picture can be compared with a grabbed picture direct on the board.

Format of the BMP file:

- 32 bits per pixel
- 24 bits color depth

Belonging LabVIEW – VI:



## 3.2.25 Read Reference ToBuffer

The GUSB_Platform_4120_ReadReferenceToBuffer function provides the stored reference picture data of the USB 4120 board indicated by DeviceNumber.

**Format:**

```
int GUSB_Platform_4120_ReadReferenceToBuffer(unsigned int  DeviceNumber,
                                             unsigned char *Data,
                                             unsigned int  *Length)
```

**Parameters:**

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).

Data

Pointer to a data buffer

The size of the data buffer provided by the application must be at least K_MAX_BYTE_RD_EXTENDED_4120 (see *GUSB_Platform* file).

One pixel consists of 32 bits. The frame grabber grabs line by line from the upper left to the lower right picture corner.
Correspondingly, data is stored in the buffer.

Structure of the 32 data bits of a pixel:

| Bits | 31..27 | 26 | 25 | 24 | 23..16 | 15..8 | 7..0 |
|---|---|---|---|---|---|---|---|
| Content | C7..C3 | DE | HS | VS | R7..R0 | G7..G0 | B7..B0 |
| | Control signals | | | | Color information | | |

Length

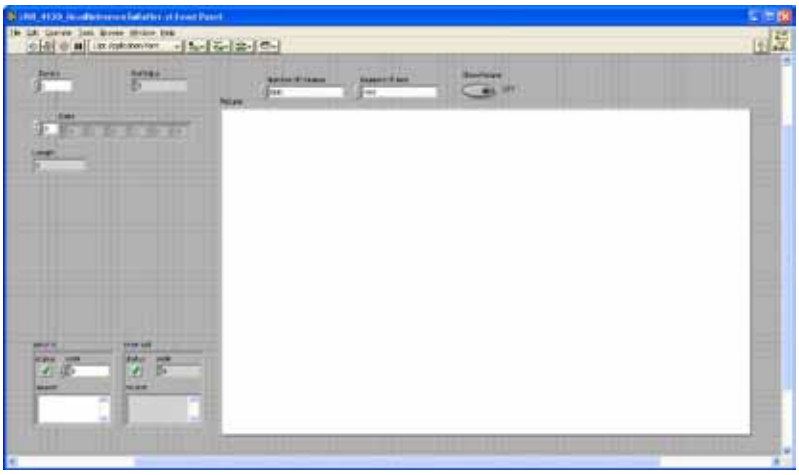
Size of the data buffer Data is pointing to, in Bytes

After executing the function: Number of bytes actually written to the data buffer

**Description:**

The function stores the reference data of a picture deposited in the frame grabber in the provided data buffer.

Belonging LabVIEW – VI:



Here there is the additional possibility to represent the data provided by the frame grabber as a Picture in the corresponding field.

For ShowPicture = ON, the correct values of the grabbed picture must be entered in the NumberOfColums and NumberOfLines fields.

## 3.2.26 Read Reference ToFile

The GUSB_Platform_4120_ReadReferenceToFile function returns the stored reference picture data of the USB 4120 board indicated by DeviceNumber as a BMP file.

**Format:**

```
int GUSB_Platform_4120_ReadReferenceToFile(unsigned int   DeviceNumber,
                                           unsigned short Width,
                                           unsigned short Height,
                                           char           *Path)
```

**Parameters:**

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).

Width

Number of picture columns (horizontal resolution)

(required for the internal conversion of the grabbed picture data to a BMP file)

Height

Number of picture lines (vertical resolution)

(required for the internal conversion of the grabbed picture data to a BMP file)

Path

Path for storing picture data

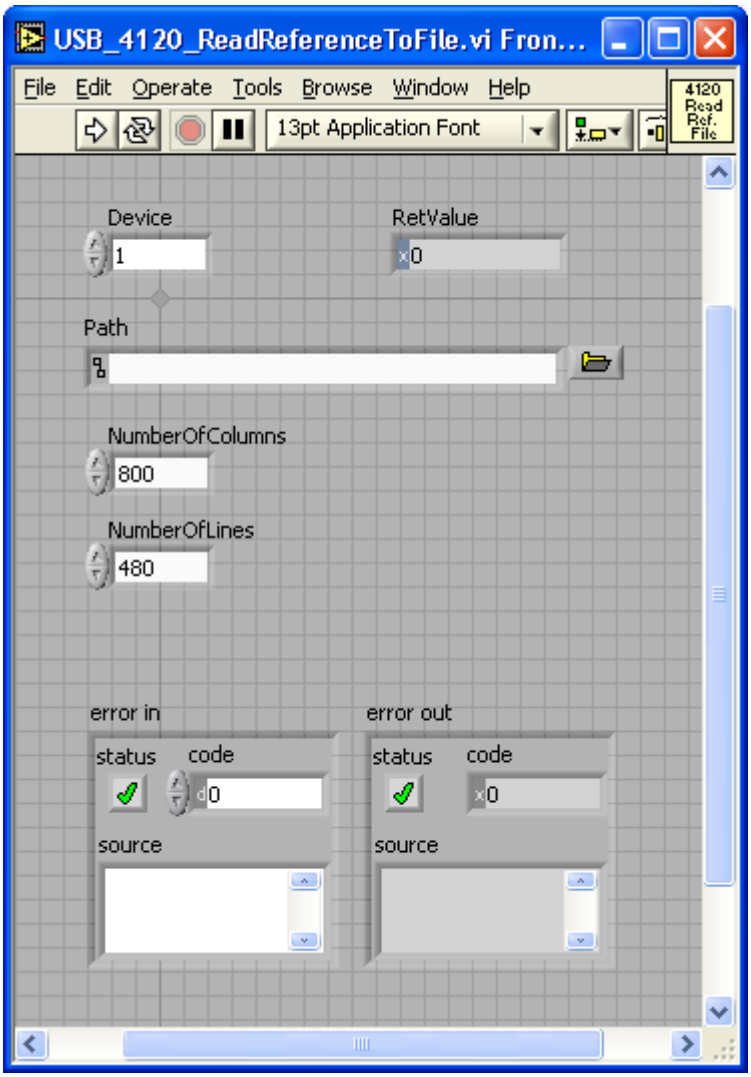
**Description:**

The function stores the reference data of a picture deposited in the frame grabber as a BMP file in the provided path.

Format of the BMP file:

- 32 bits per pixel
- 24 bits color depth

Belonging LabVIEW – VI:



## 3.2.27 Deserializer Configuration Vector

The GUSB_Platform_4120_DeserializerConfigurationVector function configures the deserializer of the USB 4120 board indicated by DeviceNumber.

**Format:**

```
int GUSB_Platform_4120_DeserializerConfigurationVector(
                                                unsigned int  DeviceNumber,
                                                unsigned char Mode,
                                                unsigned char *Byte1,
                                                unsigned char *Byte2,
                                                unsigned char *Byte3,
                                                unsigned char *Byte4)
```

**Parameters:**

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).

Mode

Defines the access mode to the parameters Byte1 ... Byte4
(see *GUSB_Platform.h* file)

K_4120_READ_DESERIALIZER_VECTOR - > reading
K_4120_WRITE_DESERIALIZER_VECTOR -> writing

Byte1

Value for the configuration register of the deserializer
INAP -> ADR 1; DS90 -> ADR 0

Byte2

Value for the configuration register of the deserializer
INAP -> ADR 2; DS90 -> ADR 1

Byte3

Value for the configuration register of the deserializer
INAP -> ADR 3; DS90 -> ADR 2

Byte4

Value for the configuration register of the deserializer
INAP -> ADR 4; DS90 -> ADR 3

**Description:**

Configuration is only required for the following types of deserializers:

- APIX INAP125R24_V10
- APIX INAP125R24_V11
- DS90UR906

The exact description of the configuration registers can be found in the data sheets of the corresponding deserializers.

Example 1: INAP125R24:

Byte1: C5 (dedicated upstream=disabled, rest=default)

Byte2: 7F (pixel date width=24, rest=default)

Byte3: 28 (Reserved=2, rest=default)

Byte4: 0C (Default)

The bits 7..4 in byte 3 must be set to 0x2; otherwise the LVDS lock is not displayed correctly.

Example 2: DS90UR906:

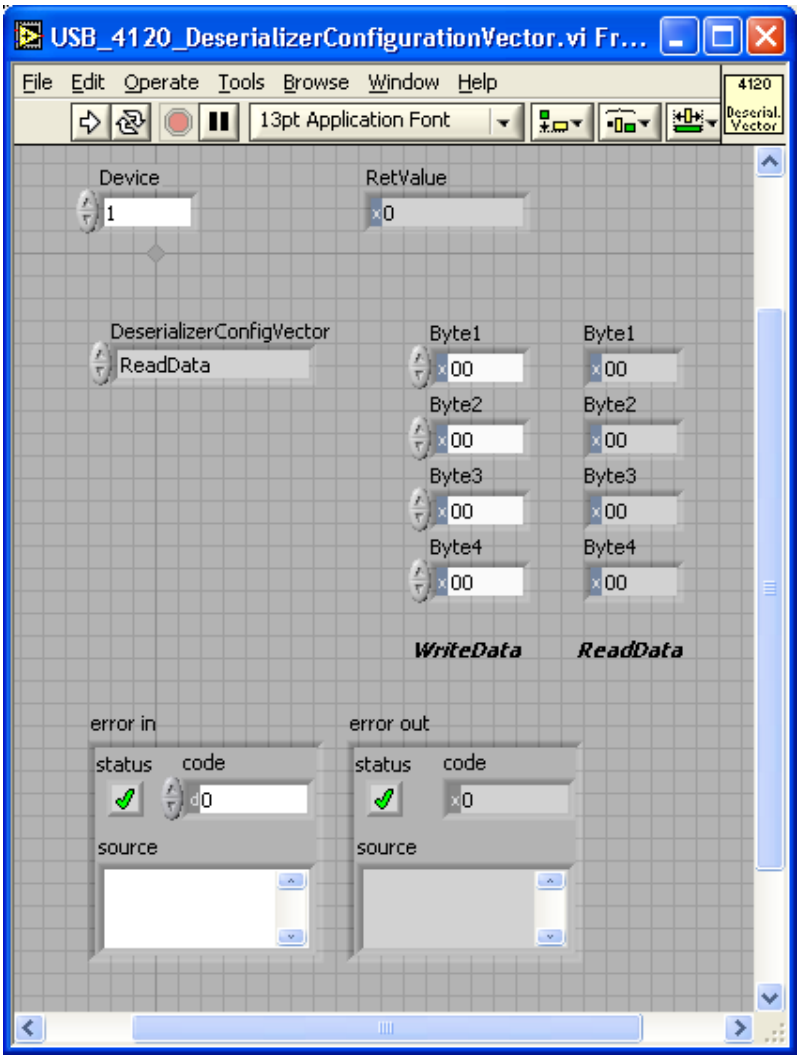
Byte1: 51 (OSSel=1, rfb=1, UserReg1=1)

Byte2: 70 (Default)

Byte3: 00 (Default)

Byte4: 00 (Default)

Belonging LabVIEW – VI:



## 3.3 Programming by LabVIEW

**LLB using the Windows Device Driver**

On the delivered CD there are VIs to call USB 4120 boards or basicCON 4120 devices directly under LabVIEW.

The functions described in the Programming via DLL Functions section are used for this.

## 3.4 Using 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 Controller is responsible for connecting the USB 4120 board/ basicCON 4120 device to the PC via USB 2.0.

Messages (generally USB commands) required for configuration can be sent to this USB Controller.

## 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..)<br>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..)<br>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<br><br>Response:<br>Byte 4: low byte of generic software version<br>Byte 5: high byte of generic software version<br>Byte 6: low byte of software version of functional part<br>Byte 7: high byte of software version of functional part |