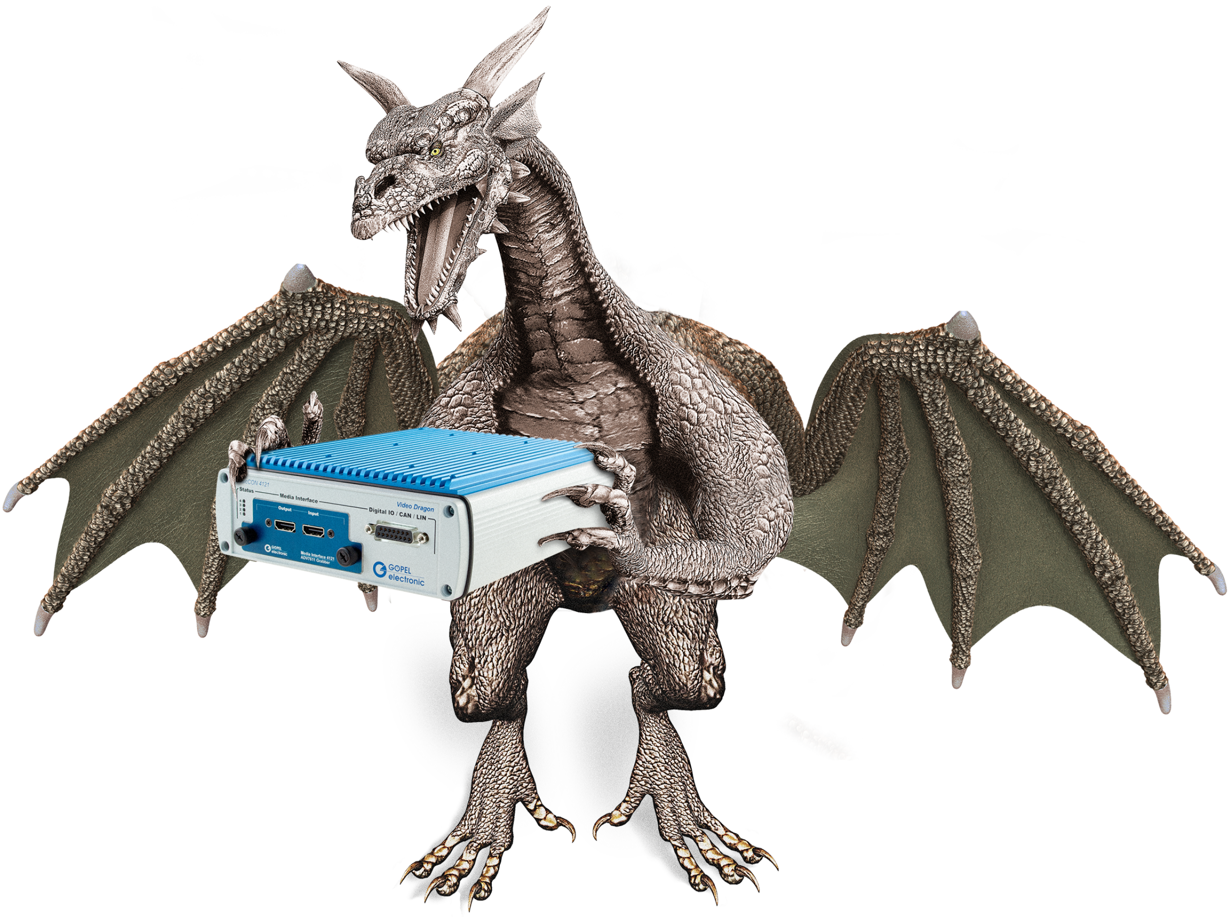
Dragon Suite

Software Documentation



DragonSuite

GÖPEL electronics
Enjoy Testing

Version: 1.2.19 / Printed: 05/11/2021

Table of Contents

List of Figures

List of Tables

1 Introduction

The **Dragon Suite** software provides a complete tool to configure the **Video Dragon** .

The **Dragon Suite** software consists of the following features:

- **Frame Grabber** configuration.
- **Frame Generator** configuration.
- **Frame Grabber** features to capture images and videos.
- **Frame Generator** features to generate images and videos.
- **Sideband** communication, to communicate with devices, connected to the **LVDS** network.
- **File System** functions to use the **Video Dragon** file system.
- IO functions to use the **Video Dragon** IO interface.
- CAN functions to use the **Video Dragon** CAN interface.
- The ability to use the **Command Line** Interface.

This documentation gives a short overview of the **Video Dragon** features.

Please check the **GOEPEL electronics** hardware documentations for more information.

1.1 Symbols

This guide highlights some important comments as follows:

Symbol	Description
	**Warning** that indicates risk situations and dangers. Disregard can lead to life-threatening situations or destruction of components.
	**Information** that indicates certain aspects or is important for a particular topic or goal.
	**Tip** that gives useful hints or recommendations.

Table 1.1 Symbols

1.2 Liability and Warranty Exclusion

This software is designed to simplify the use of our API in conjunction with our Video Dragon hardware. We do not guarantee stability, security and usability, especially when used in manufacturing processes (e.g. end-of-line). In no event shall **GOEPEL electronics** be responsible for any direct, indirect, incidental, special, exemplary, or consequential damages (including but not limited to the purchase of replacement goods or services, loss of use, loss of data or profit, breakdowns, injury, or potential death) in any way in the case of improper use of the **Dragon Suite** .

2 Installation

2.1 Supported Hardware

Supported hardware devices from **GOEPEL electronics** are:

- **basicCON 4121**
- **Video Dragon 6222**

2.2 Prerequisites

2.2.1 System Requirements

Dragon Suite is a software for Microsoft Windows operation systems. Your system must comply with the following requirements:

- CPU with at least 4 cores, 8 cores recommended at 2,9 GHz
- Windows 7 or later
- At least 200MB of free disk space
- At least 8GB RAM
- Installed **G-API** version 1.4.6325 or higher (see **G-API Manual** for reference)

2.2.2 Hardware Installation

For hardware installation please follow the steps in the hardware manual of the corresponding LVDS device.

2.3 Installation

The **Dragon Suite** comes with a setup wizard that guides you through the installation procedure. Make sure that your system meets the system requirements.

Download the Windows installer and start the execution file. Validate the integrity of the file if necessary and run the installer by following the instructions in the installation program.

Figure 2.1 **Dragon Suite** Setup Wizard

While installation you need to select the components you want to install.

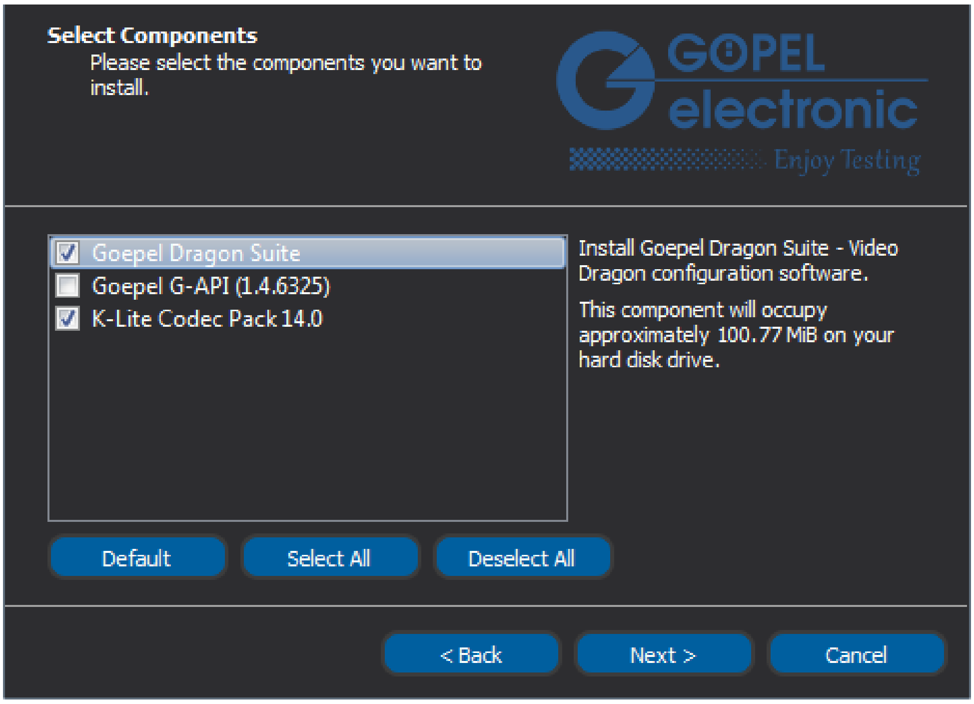


Figure 2.2 **Select Components** Window

- **Dragon Suite** always needs to be selected.
- It is indispensable having installed the **G-API** for using the **Dragon Suite** . If it is not installed yet, select this component.
- Select the **K-Lite Codec Pack** if playing videos in the **Dragon Suite** is desired.

2.4 Update Manager

The software can independently search for updates online at the **GOEPEL electronics** service homepage genesis.goepel.com.

A working internet connection is necessary for executing automatic updates.

To check for updates, use the Help menu in the software's menu bar. Use the option **Update** ( ) and the software checks if an update is available.

In case no update is available, a message appears in the Message Box.

Figure 2.3 No Update Available

When a new **Dragon Suite** version is available, a small window appears asking if you want to update now or not. Click **Yes** to close the application and start the update. Select **No** if you do not want to close the application, and then run the update later.

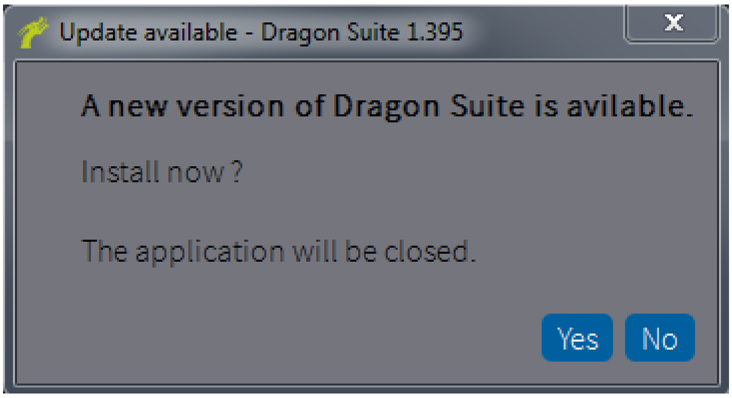


Figure 2.4 Update Available

When you start the installation, a window opens with options for adding or removing components and for updating components.

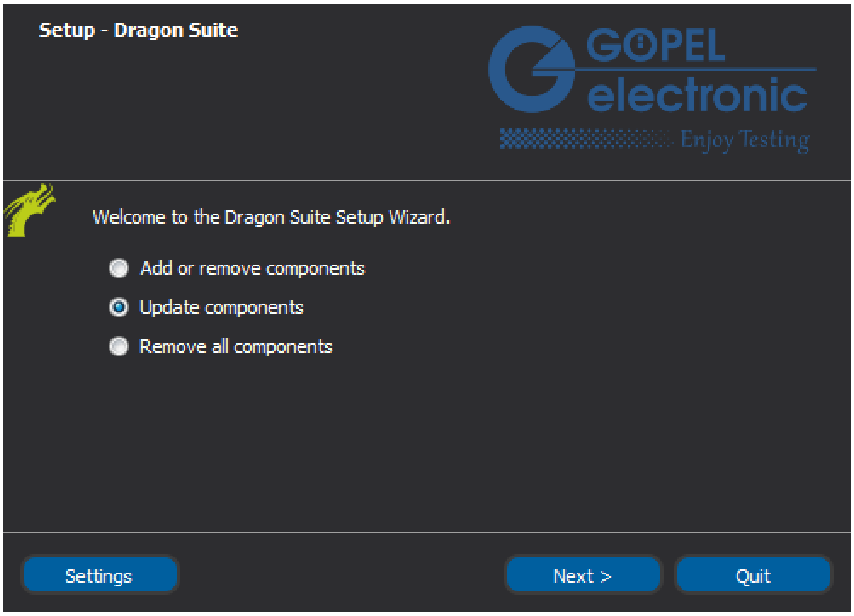


Figure 2.5 Update Manager

Select the option **update components** and click the **Next** -button.

In case an update is available, a checklist will be displayed to choose the components that could be updated. Select all components to be updated.

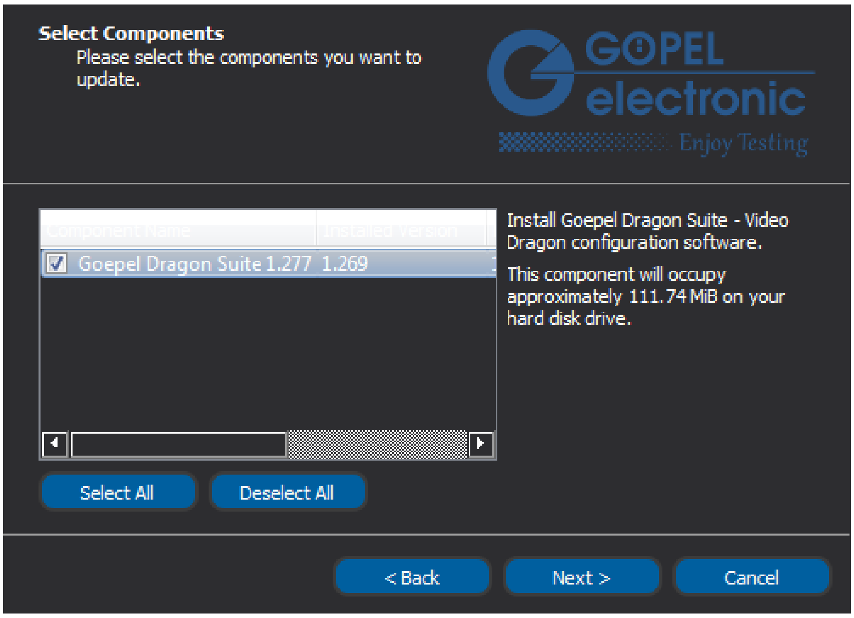


Figure 2.6 Select components

Click the button **Next** to execute the update. After the update, click **Finish** to exit the wizard.

3 Using the GUI

After starting the **Dragon Suite** the main window appears. The main window remains open during the entire run-time of the software.

Figure 3.1 **Dragon Suite** Main Window

The main window consists of the following areas:

1. The Menu Bar
2. The Control Bar
3. The Interface Tree
4. The Main Frame
5. The Message Box

3.1 Menu Bar

3.1.1 Interfaces

In this menu, you can update the interface list displayed in the Interface Tree. With **Exit** the software will be closed.

3.1.2 Settings

The Settings menu opens the **Frame Generator** or **Frame Grabber** settings window where various hard- and software settings can be made.

3.1.3 Tools

Use this menu to open the **Frame Generator** or **Frame Grabber** dialog window, the **Sideband** communication window, the **File System** manager window, the IO dialog or the CAN dialog.

3.1.4 Windows

Hide or show the Interface Tree and the Message Box in this menu.

Additionally use this menu to switch between already opened **Frame Generator** or **Frame Grabber** windows in the Main Frame.

3.1.5 Help

In the Help menu, use the option **Help** (F1) to open the **Dragon Suite** help window. The help window contains the **Dragon Suite** manual with a simplified search function. There you can search for **Contents** or by keywords in the **Search** tab.

The option **About** (F12) opens a dialog box with information about the **Dragon Suite** and how to contact **GOEPEL electronics** for support.

Use the Update option to check if an update is available.

3.2 Toolbar

The Toolbar consists of a set of icons:

Icon	Description
	Hide the Interface Tree (Ctrl + H).
	Show the Interface Tree (Ctrl + H).
	Refresh the Interface List shown in the Interface Tree (Ctrl + R).
	Open the **Frame Generator** settings window (F2).
	Open the **Frame Grabber** settings window (F3).
	Open the **Frame Generator** dialog window (F4).
	Open the **Frame Grabber** dialog window (F5).
	Open the **Sideband** communication window (F6).
	Open the **File System** manager window (F7).
	Open the IO dialog window (F8).
	Open the CAN dialog window.
	Close the **Dragon Suite** (Ctrl + Q).

Table 3.1 Toolbar icons

Arrange the Toolbar elsewhere in the Main Window by left-clicking the white dotted line and dragging it to the desired position.

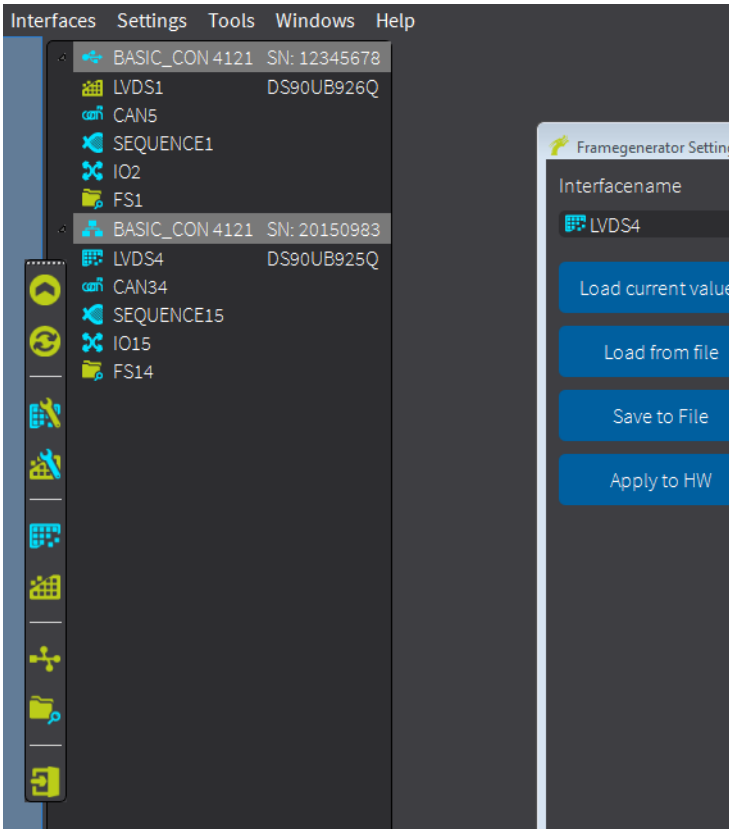


Figure 3.2 Arrange Toolbar as desired

3.3 Interface Tree

In the Interface Tree all available interfaces of the **GOEPEL electronics** devices are represented. Although you can see all existing interfaces and devices, not all are supported in the **Dragon Suite** . The interface names are assigned through the **Hardware Explorer** installed with the **G-API** . If a connected device does not appear in the Interface Tree, first update the **Hardware Explorer** . If this does not help, check the **Hardware Explorer** preferences.

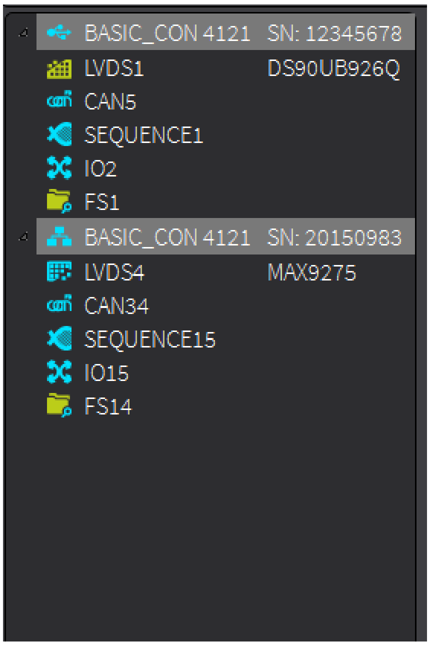


Figure 3.3 Interface Tree

To hide the Interface Tree use the Menu Bar or the Toolbar. Single devices can be hidden by clicking on the small triangle symbol on the left side of the device name.

Right-clicking on the device opens the submenu with options for resetting the entire device or displaying the **unlocked features** of the device.

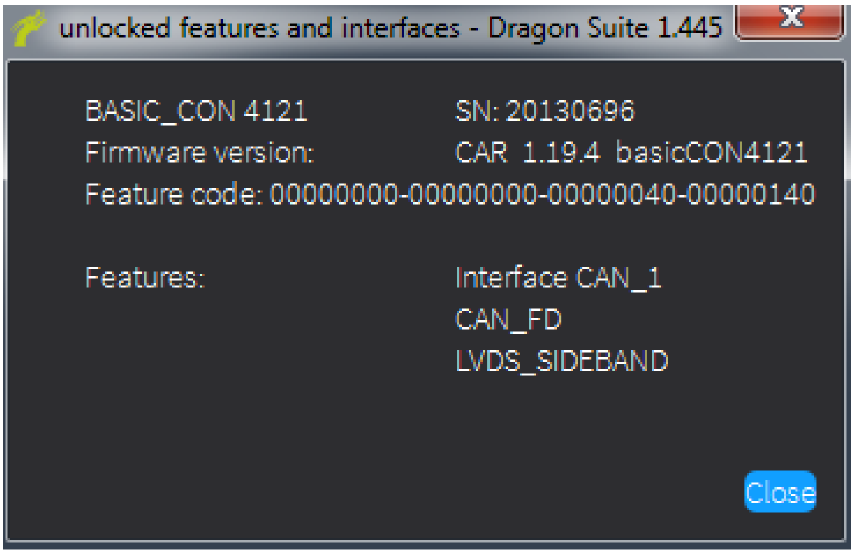


Figure 3.4 Unlocked features and interfaces of the device.

Right-clicking on the LVDS interface opens a small submenu and offers four different options. The first one is to reset the interface. Also you can open the settings window or dialog window for either **Frame Generator** or **Frame Grabber**, depending on the device. Additionally there is the possibility to open the **Sideband** communication window.

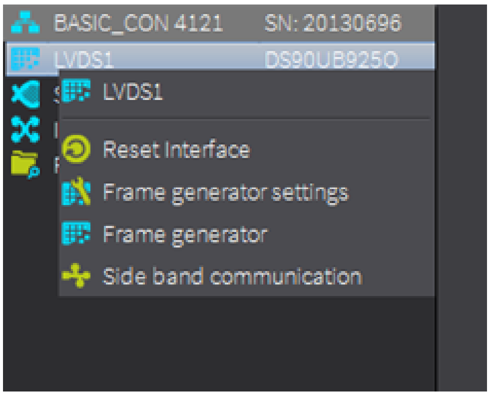


Figure 3.5 Interface submenu of Interface Tree

Any other interface can be reset by right-clicking on the interface in the Interface Tree.

3.4 Main Frame

The settings and dialog windows are displayed in the Main Frame. This windows can be arranged arbitrarily. To work with multiple windows, enlarge the Main Frame by left-clicking on the right edge of the Main Window and dragging it to the right.

Figure 3.6 Main frame

The windows can be minimized in the lower left corner of the Main Frame or maximized to fill the entire frame.

3.5 Message Box

The Message Box is used to display the **log file**, for example capturing information or error messages. If you right-click in the message box, you can save or delete the log file.

Hide the Message Box using the Menu Bar.

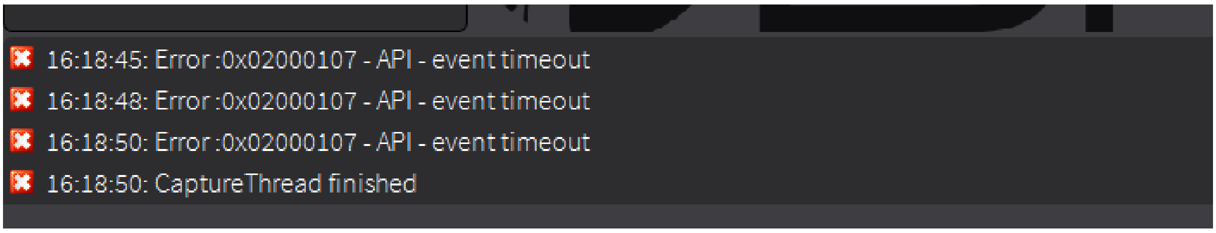


Figure 3.7 Message Box

By right clicking in the message box a small menu selection appears. The messages can be saved in a log file or copied to the clipboard. Furthermore the logging of error messages can be switched off (uncheck the box). With **Clear** all entries are deleted.

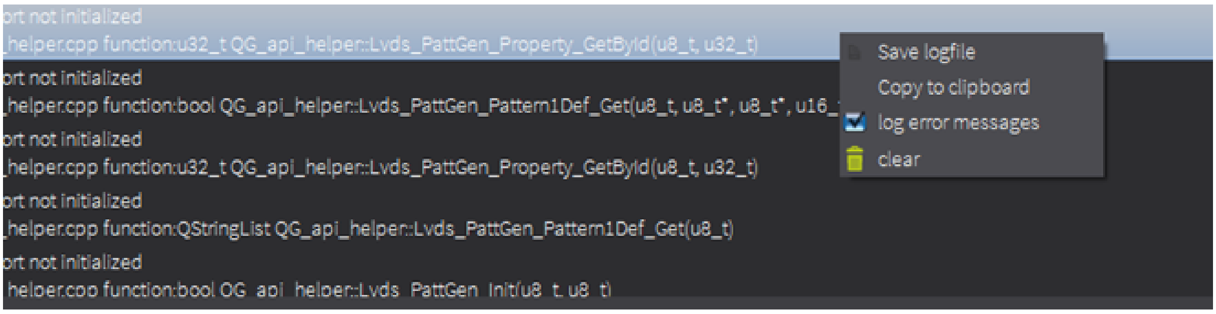


Figure 3.8 Message Box features

An overview of common mistakes can be found in chapter Common Error Messages.

4 Setting up the Frame Generator

Open the **Frame Generator** settings window by using one of the alternatives illustrated in chapter Using the GUI. The settings window shows all supported device settings. Most of the functionalities of the **Dragon Suite** depend on valid settings of the interface. Setting up the device should therefore be the first step after starting the software.

Generally parallel usage of several interfaces with the **G-API** is possible. But the **Dragon Suite** supports the usage of only one **Frame Generator** interface at a time.

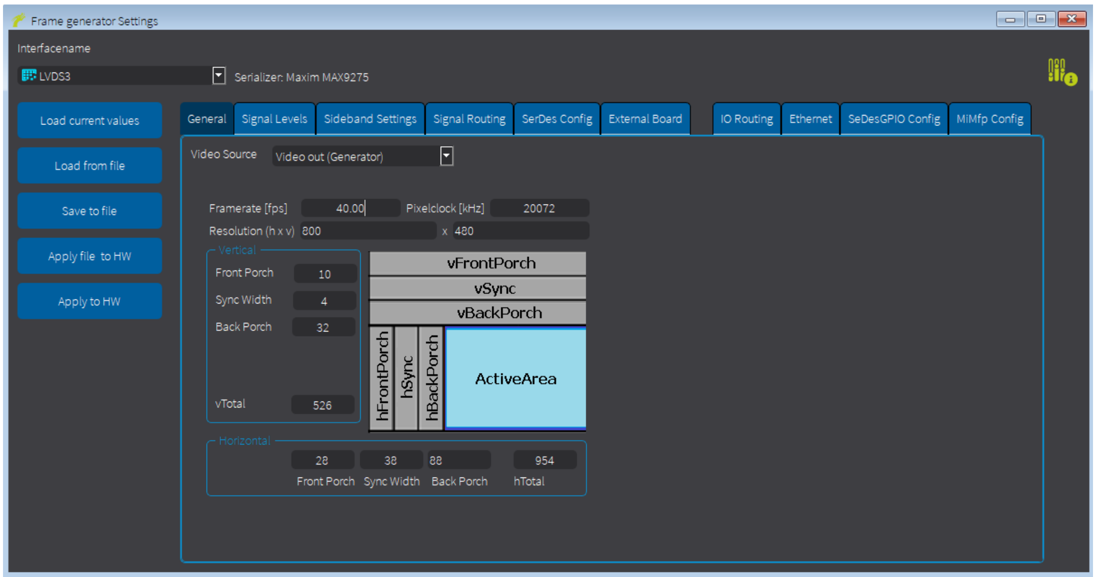


Figure 4.1 **Frame Generator** settings window

All available **Frame Generator** interfaces are listed in the drop down menu.

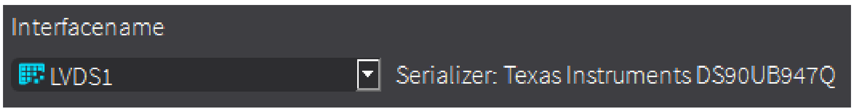


Figure 4.2 Choose interface name

The currently selected interface is shown in the text field of the drop down menu.

On the left of the settings window are four buttons:

Button	Description
Load current values	Load the current setting values of the selected interface.
Load from file	Load the values for the selected interface by importing an external XML file. Importing a settings file does not overwrites the current settings on the device. To overwrite the device settings, use the **Apply to HW** button.
Save to File	Save the current settings of the selected interface by exporting them to an external XML file.
Apply to HW	Overwrite all current settings on the device with the settings displayed on the tabs of the window.

Table 4.1 **Frame Generator** setting buttons

TIP

A right click on the **Load from file** button opens a selection of the last opened files. This facilitates the search for frequently used configurations. The history can be cleared with **clear history.**

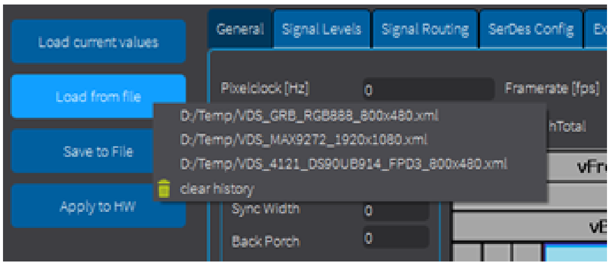


Figure 4.3 **Frame Generator** Load from file button

4.1 General

Depending on the device used, a stored image can be generated or the hardware itself generates an individual pattern. Use the dropdown menu to select the video source.

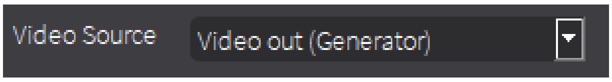


Figure 4.4 Choose video source

The general serializer settings for **video source = Video Out** are pixel clock, frame rate and image format parameters.

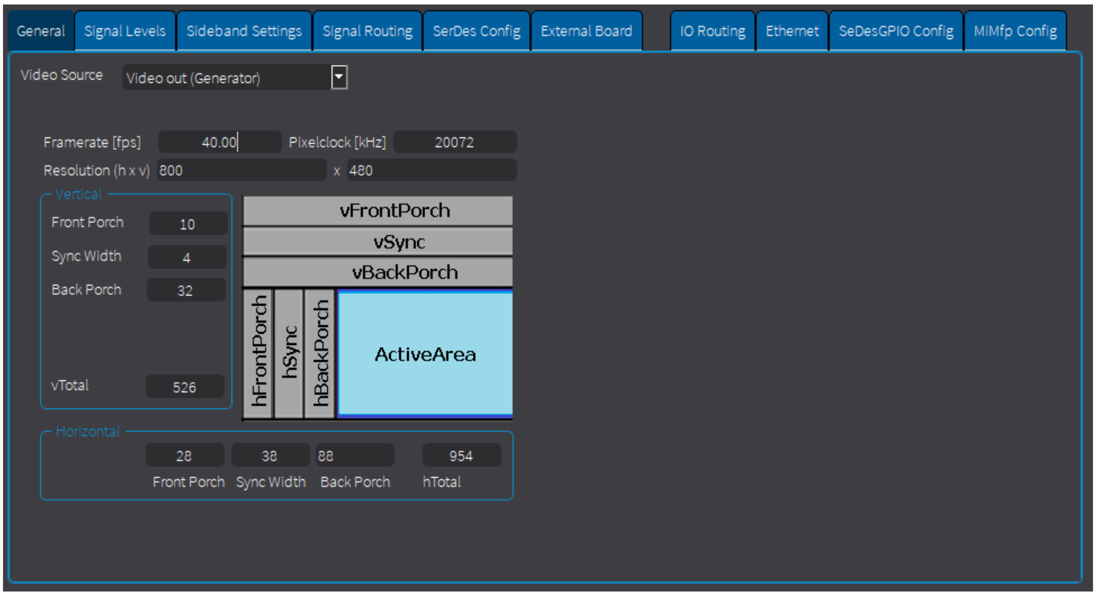


Figure 4.5 **Frame Generator** settings window - General

The **Pixel clock** value determines, how many pixels need to be sent per second.

The **Frame rate** sets the maximum rate for capturing frames in continuous mode.

Image format parameters are described by the horizontal and vertical synchronization signals, indicated in clocks. The total values consist of the **active area** (the desired file) and some previous data for synchronization. **Front Porch** describes the duration between the frame end information and the signal pulse, called **Sync Width** . The vertical Sync Width unit is lines whereas the horizontal Sync Width unit is pixels. **Back Porch** is the interval between Sync Width and the beginning of frame information. Horizontal synchronization means that the end of a single line is reached, whereas vertical synchronization indicates the end of a complete frame.

There is a dependency between pixel clock, frame rate and synchronization signal parameters, as follows:

Frame Rate [fps] = Pixel Clock [Hz] * 1000 / (hTotal * vTotal)

When changing one of the parameters, frame rate or pixel clock adapt oneself automatically in **Dragon Suite** .

TIP Changed values of pixel clock and frame rate are taken directly from the hardware, without having to use the Apply to HW button.

For **video source = Pattern Generator Out** please go to Pattern Generator chapter.

4.2 Signal Levels

The valid signal levels and edges are defined in the Signal Levels tab.

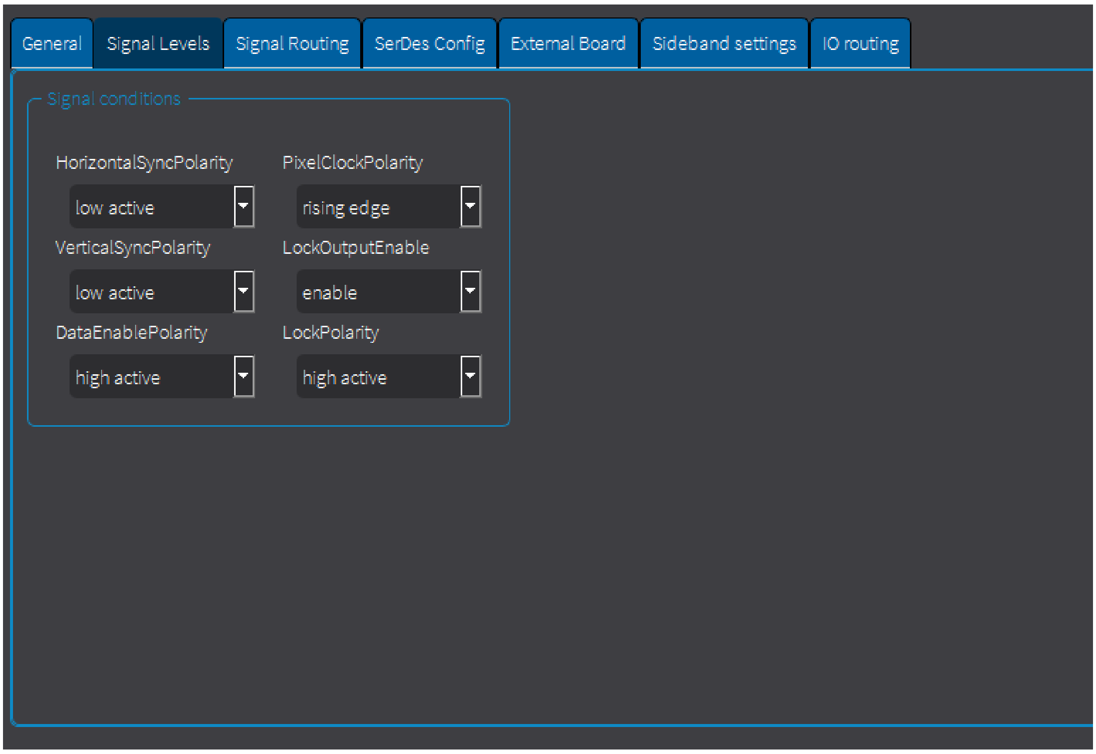


Figure 4.6 **Frame Generator** settings window - Signal Level

For the synchronization signals, the **Polarity** (low or high active) defines which signal level indicates vertical or horizontal synchronization.

The level setting of the **Data Enable** signal specifies at which level pixel data is being transmitted.

Pixel Clock Polarity describes at which edge of the pixel clock signal (rising or falling) the values of the signals are to be sampled.

To display the lock via an LED signal use the parameter **Lock Output Enable** . When this parameter is activated, LED 4 lights up for a successful lock.

Lock Polarity determines whether the signal is high or low when a lock is detected.

4.3 Signal Routing

Since there is no common standard for mapping the video signals to the 32 serialized bits in the LVDS video stream, this assignment must be defined for each device. This can be done in the Signal Routing tab.

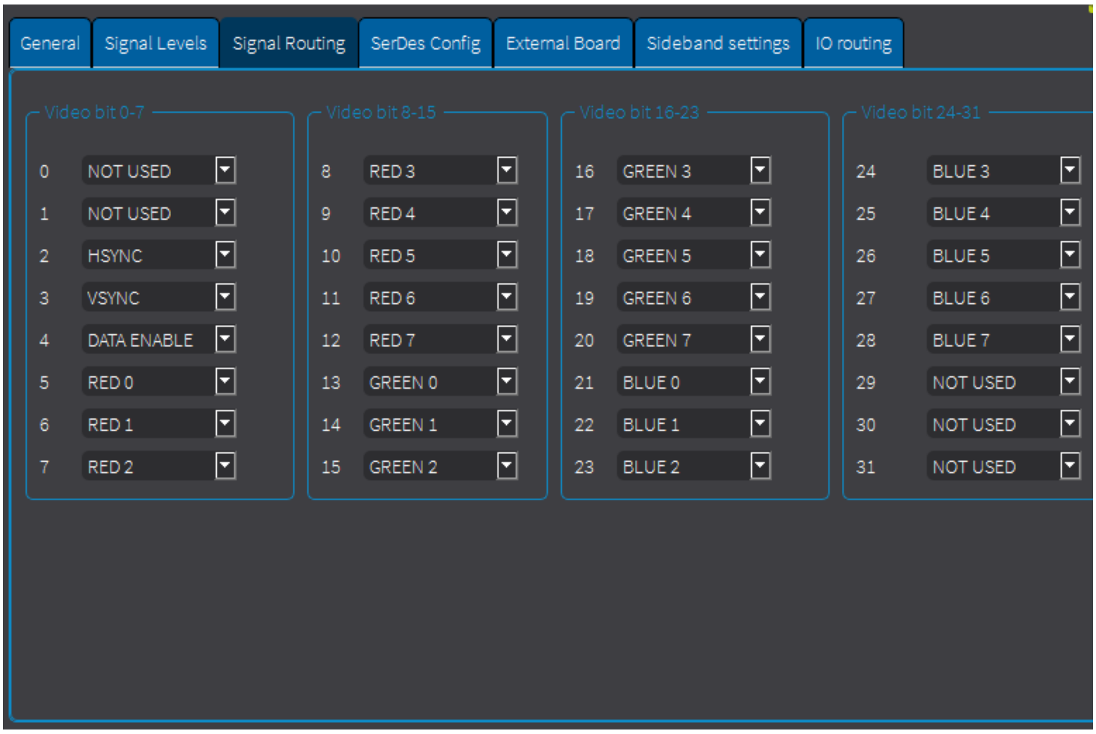


Figure 4.7 **Frame Generator** settings window - Signal Routing

On the Signal Routing tab, you can specify the mapping of each serialized bit of the LVDS stream. Therefor the combo boxes of the tab provide the option to set the color bits. Since the resulting RGB frame has a color depth of 24 bits, each color (red, green and blue) has a maximum depth of 8 bits. The selection of the corresponding bit in the video stream is made by selecting the correct value from the possibilities given by the combo box. All 32 bit of the stream and the value **not used** can be selected. **Not used** means that the signal has no correspondent in the video stream. This should be done, for example, if a video stream contains frames whose color depth is less than 24. The significance of the color bits is ascending. For example, this means for an 8-bit color value of red, **R0** is the least significant bit, and **R7** is the most significant bit.

The combo boxes are also used to assign the control signals. The vertical sync signal **VSync** notifies the end of transmission of a complete frame, whereas the horizontal sync signal **HSync** indicates the end of transmission of a line. The third control signal is the **Data Enable** signal. In a continuous stream, the video signal may contain porches in each single line and between the end and the beginning of a new frame. The data enable signal indicates whether pixel data is currently being transmitted.

4.4 SerDes Config

Dragon Suite gives the opportunity to configure the **Video Dragon** serializer by manipulating the bus register.

Manual manipulating of the configuration data list needs an extreme good knowledge of the meaning of each register value. This can only be obtained from the data sheets of the serializer. Even a single wrong setting of only one register may render the complete configuration invalid and the serializer inoperative.

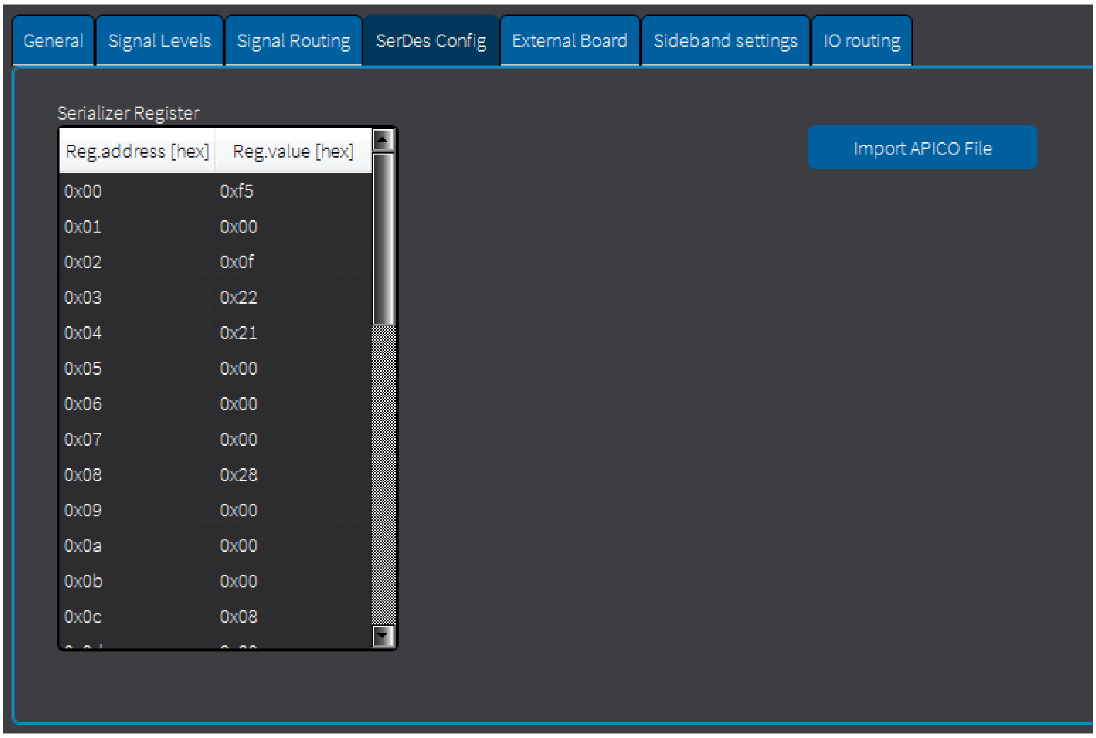


Figure 4.8 **Frame Generator** settings window - SerDes Config

To set up the serializer configuration, you must understand the structure of the configuration data: Each serializer stores its configuration in a set of one-byte register values that can be read from or written to the serializer. Some serializer types group subsets of their registers to different internal devices, others have only one internal device. The value can be presented as a decimal or hexadecimal value, which is selected by right-clicking in the list.

To add a new row to the register list, right-click in the list and select **add row** from the sub menu. A blank row appears at the bottom of the list. To delete a single row, right-click in the relevant row and select **delete row** . It is also possible to delete all rows.

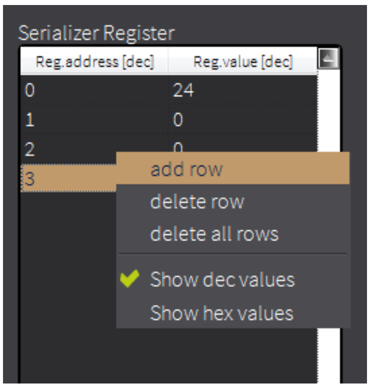


Figure 4.9 Add row to Serializer Register

To read a register, double-click an existing register address item and change its value to the requested register address. Loading the current values updates the register data from the specified address.

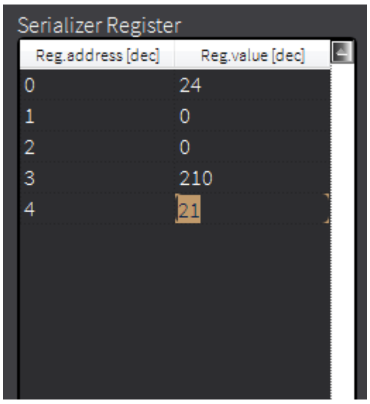


Figure 4.10 Change register address of Serializer Register

The register data can be changed by double-clicking in the corresponding entry and changing the value. The Apply to HW button immediately overwrites the current value in the serializer.

Devices with an Apix Media Interface Board require a special configuration via an Apico file. This file can be loaded via the **Import Apico File** button. The button can only be used with the appropriate hardware.

4.5 External Board

Different modes can be set for the various supported serializers. Depending on the currently selected LVDS interface, only the corresponding setting options are available.

If you set **undefined** in one of the settings, the device configuration is not taken into account.

Further information on the individual setting options can always be found in the data sheet of the respective serializer.

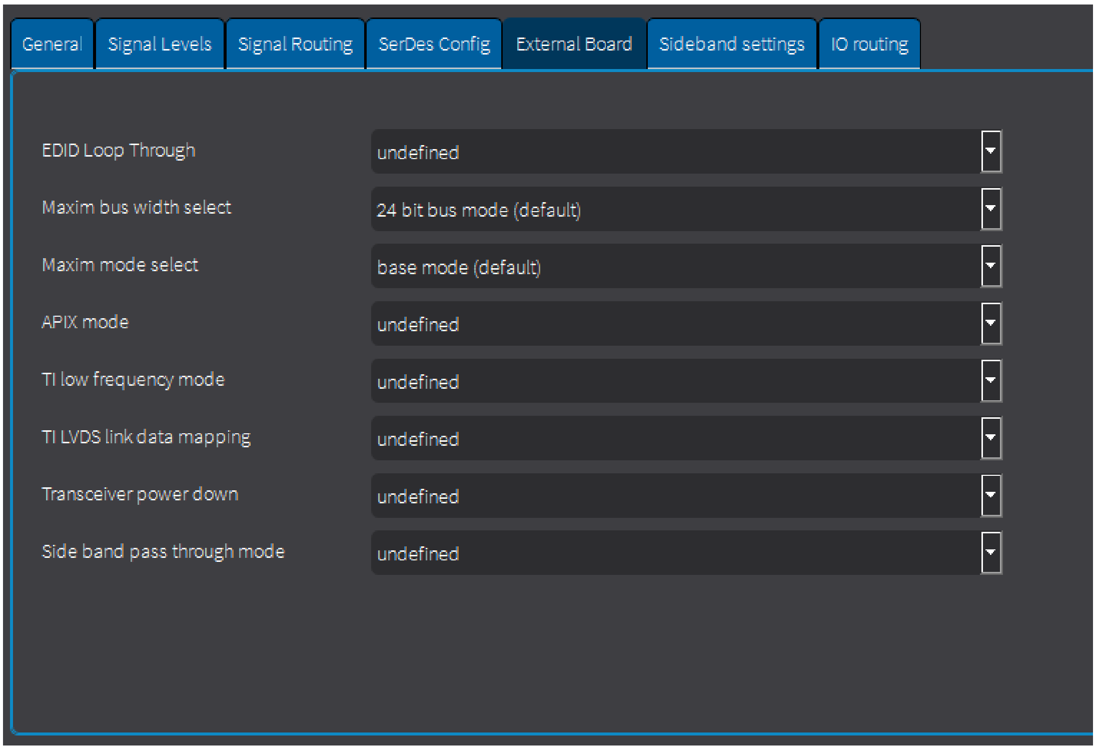


Figure 4.11 **Frame Generator** settings window - External Board

4.5.1 EDID Loop Through

Extended Display Identification Data (EDID) is a 128-byte data format for displays that describes their capabilities, such as manufacturer, date of manufacturing or display size. This parameter is only supported by ADV7611 (HDMI).

- No loop through (default): the EDID data of serdes extension board are used.
- Loop through enabled: the EDID data of output device is looped to input.

4.5.2 Maxim bus width select

This parameter is only supported by MAX9260, MAX9272, MAX9276, MAX9259, MAX9271 and MAX9275. It describes the data format of the device.

- 24 bit bus mode (default): the first 21 bits contain video data.
- 32 bit bus mode: the first 29 bits contain video data.
- High bandwidth mode (27bit; only supported by MAX9276 and MAX9275): the first 24 bits contain video data or special control signal packets.

The last 3 bits always are the embedded audio channel bit, the forward control channel bit and the parity bit of the serial word.

4.5.3 Maxim mode select

This parameter is only supported by MAX9260, MAX9272, MAX9276, MAX9259, MAX9271 and MAX9275. Two modes of control-channel operation are available for this device:

- Base mode (default): use either I^2C (half-duplex) or GMSL UART protocol (full-duplex).
- Bypass bus mode: use a custom UART protocol.

4.5.4 Maxim DRS select

This parameter is only supported by MAX9276 and MAX9275. There are two modes of operation:

- DRS = low rate
- DRS = high rate (default)

4.5.5 APIX mode

This parameter is only supported by INAP375T and INAP375R. The APIX functionality provides a high-speed Gigabit video link in combination with full-duplex communication over two wire pairs. There are two modes of operation:

- APIX1 mode
- APIX2 mode (default)

4.5.6 TI low frequency mode

This parameter is only supported by DS90UB947 and DS90UB948.

- Normal mode (default)
- Low frequency mode enabled

4.5.7 TI LVDS link data mapping

This parameter is only supported by DS90UB947 and DS90UB948. The device can be configured to accept 24-bit color with two different mapping schemes:

- Mapping mode 0 (default): SPWG mapping.
- Mapping mode 1: OpenLDI mapping.

4.5.8 Transceiver power down

This parameter is only supported by MAX9260, MAX9272, MAX9276 and DS90UB914. The devices have a power-down mode which reduces power consumption:

- Transceiver is powered up (default).
- Transceiver is powered down.

4.5.9 Sideband pass through mode

This parameter is supported only depending on the serializer used. The devices pass a sideband command through to further sideband subscribers.:

- disabled (default)
- I^{2}C pass through
- UART pass through (only for MAX9276 v1.1)

4.6 Sideband Settings

Different sideband communication modes can be set for the various supported serializers. The **Video Dragon** supports SPI, I^{2}C and UART. Depending on the currently selected LVDS interface, **Dragon Suite** will provide only the appropriate data mode options.

Detailed information about sideband can be found in the chapter Sideband Communication.

Data Modes are only available with activated sideband features.

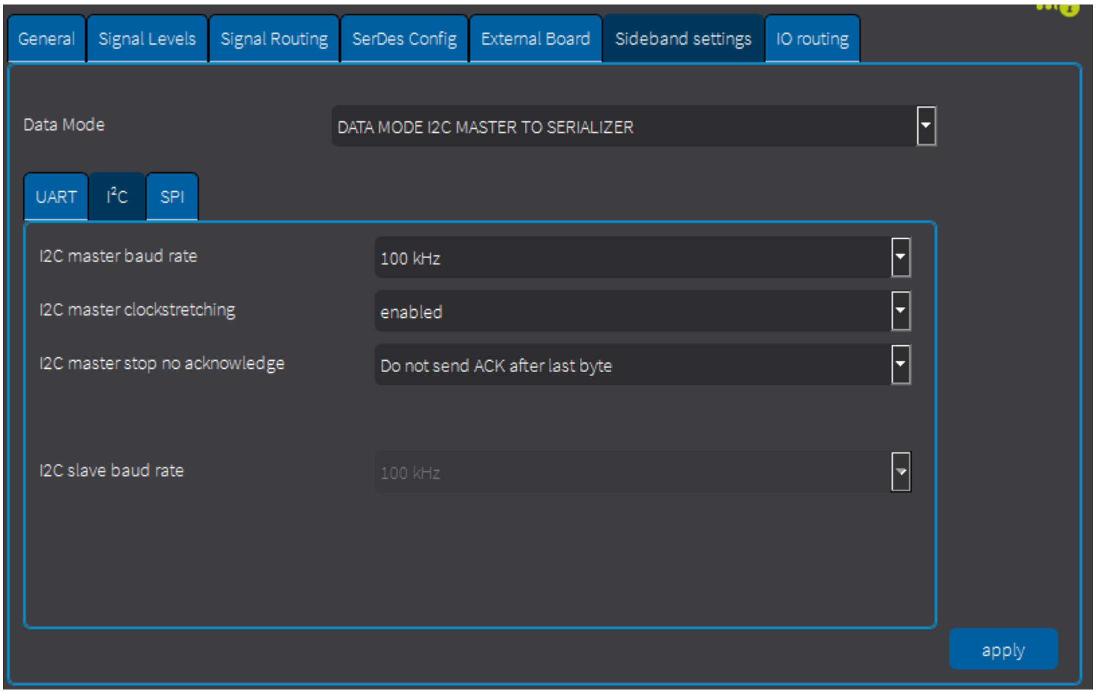


Figure 4.12 **Frame Generator** settings window - Sideband Settings

TIP The sideband settings can be set independently of all other parameters. Use the **Apply** button in the lower right corner of the Settings window.

The following Data Modes are possible:

- I²C master to deserializer: **Video Dragon** is I²C master at deserializer chip.
- I²C master to serializer: **Video Dragon** is I²C master at serializer chip.
- SPI master to deserializer: **Video Dragon** is SPI master at deserializer chip.
- SPI master to serializer: **Video Dragon** is SPI master at serializer chip.
- SPI passthrough dual: **Video Dragon** is connected to transmitter and receiver in "dual mode".
- SPI receiver dual: **Video Dragon** is connected to receiver in "dual mode".
- UART to deserializer: **Video Dragon** is connected to UART interface of deserializer chip.
- UART to serializer: **Video Dragon** is connected to UART interface of serializer chip.
- SPI transmitter dual: **Video Dragon** is connected to transmitter in "dual mode".
- SPI slave to deserializer: **Video Dragon** is SPI slave at deserializer chip.
- SPI slave to serializer: **Video Dragon** is SPI slave at serializer chip.
- I²C slave to deserializer: **Video Dragon** is I²C slave at deserializer chip.
- I²C slave to serializer: **Video Dragon** is I²C slave at serializer chip.

Depending on the selected Data Mode, the sub-tab for the corresponding sideband is automatically opened.

4.6.1 UART

For UART mode the following parameters can be adapted:

- Baud rate (default value: 115200 baud): Sets the earliest possible value to this parameter.
- Parity (default value: no parity): Sets, if a parity will be build and transferred.

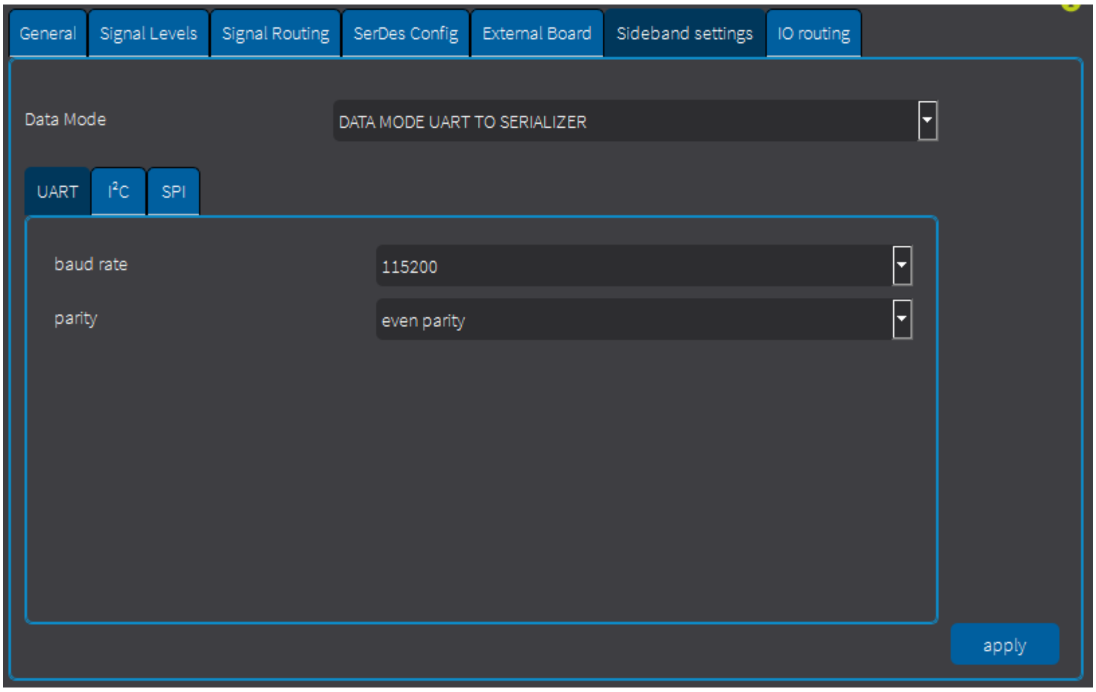


Figure 4.13 Sideband Settings - UART

4.6.2 I²C

For I²C mode the following parameters can be adapted:

- I²C master baud rate (default value: 400kHz): Baud rate of I²C master can be 100kHz or 400kHz.
- I²C master clockstretching (default value: enable): Enables or disables the master response on the deferment of the clock, organized by the slave.
- I²C master stop no acknowledge (default value: Do not send ACK after last byte): I²C master sends an acknowledge or not after the last received byte from the slave.
- I²C slave baud rate (default value: 100kHz): Baud rate of I²C slave can be 100kHz or 400kHz.

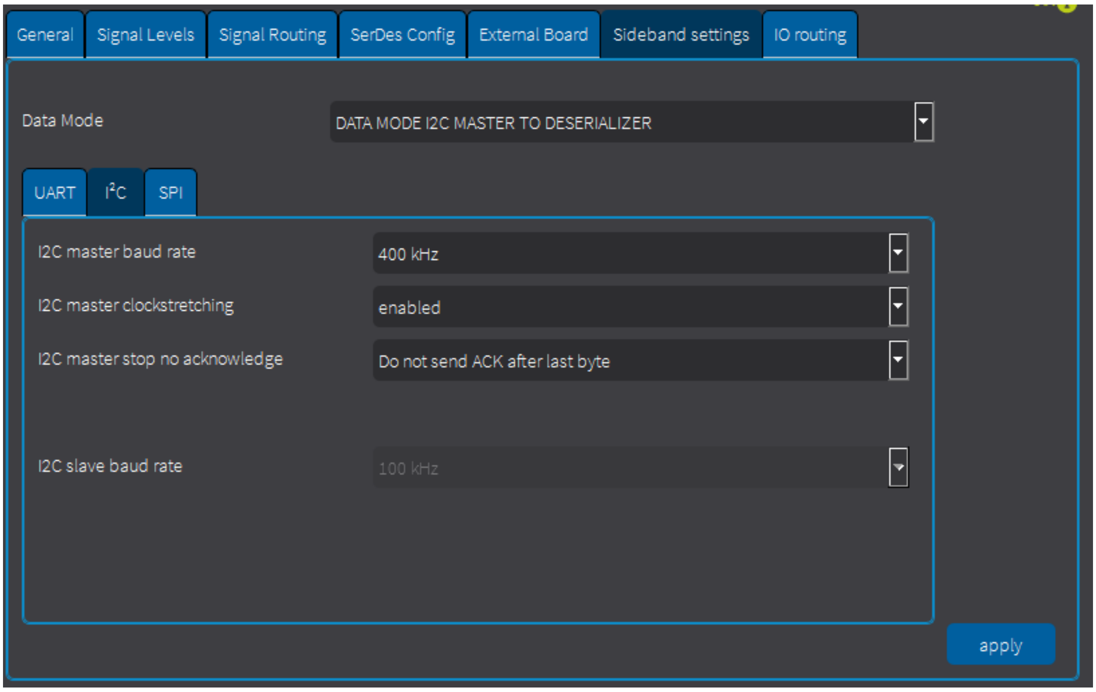


Figure 4.14 Sideband Settings - I²C

4.6.3 SPI

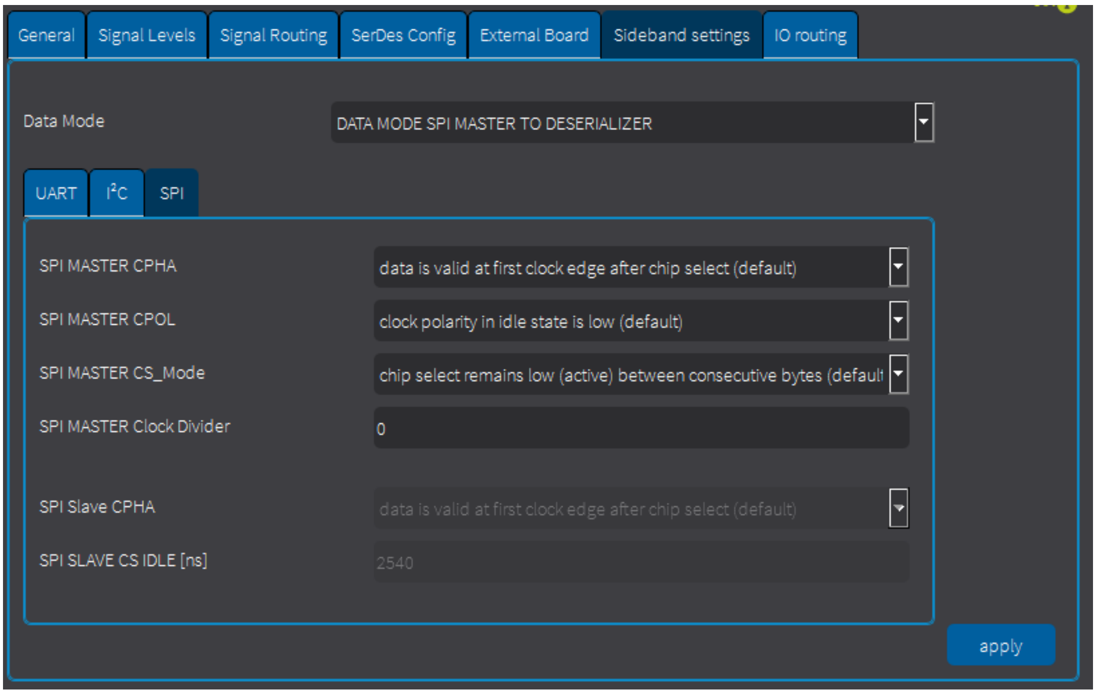


Figure 4.15 Sideband Settings - SPI

For SPI mode the following parameters can be adapted:

- SPI master CPHA (default value: data is valid at first clock edge after chip select): SPI clock phase - Valid data at first respectively second clock edge after chip select.
- SPI master CPOL (default value: clock parity in idle state is low): SPI clock polarity in idle state.
- SPI master CS_mode (default value: chip select remains low (active) between consecutive bytes): SPI chip select behavior between consecutive bytes of the same transfer.

- SPI master clock divider (default value: 0): determines SPI clock frequency based on freq. 25MHz
  - 0: divider is 1; clock frequency is 25MHz
  - 1: divider is 2; clock frequency is 12.5MHz
  - 2: divider is 3; clock frequency is 8.33MHz
  - ...
  - 255: divider is 256; clock frequency is 0.1953125MHz
- SPI slave CPHA (default value: data is valid at first clock edge after chip select): SPI clock phase - Valid data at first respectively second clock edge after chip select.
- SPI slave CS idle [ns] (default value: 20): SPI chip select idle minimum in 20ns steps
  - 0: 127 * 20ns = 2.54µs chip select idle after transfer
  - 1: 1 * 20ns = 20ns
  - 2: 2 * 20ns = 40ns
  - ...
  - 255: 255 * 20ns = 5.1µs

4.7 IO Routing

The **GOEPEL electronics Video Dragon** has an IO trigger function with which certain events can be routed to a trigger event. The routing can be defined in the IO Trigger Dialog window. The defined connections can also be found in the IO Routing tab of the Settings window for an easier overview.

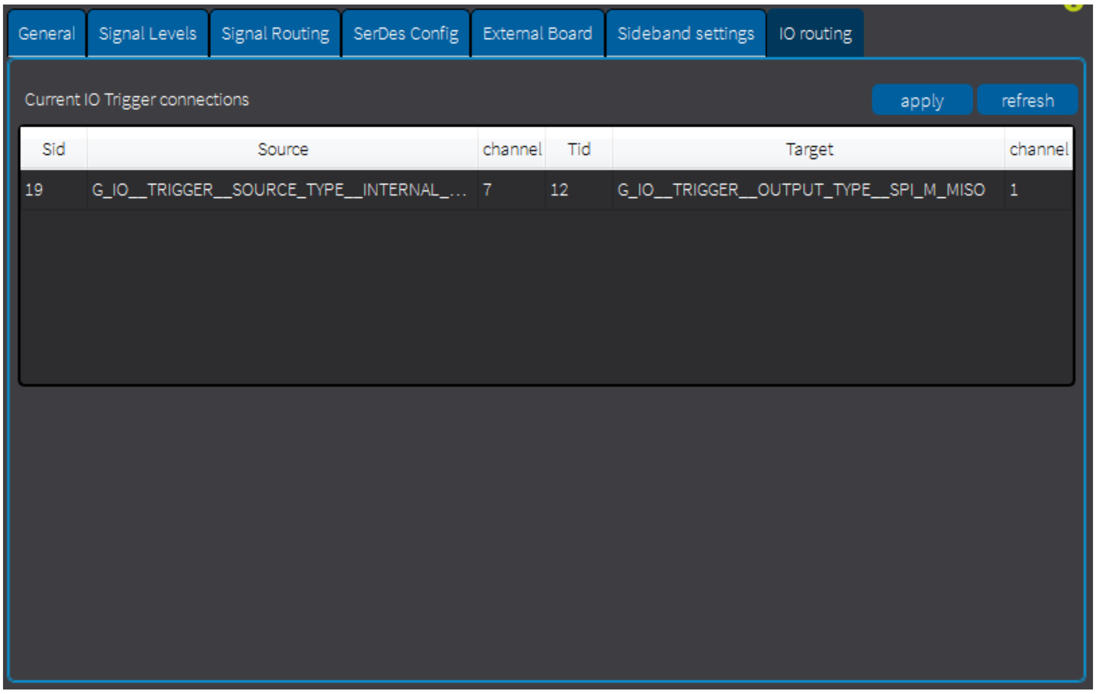


Figure 4.16 **Frame Generator** settings window - IO Routing

The channel entries of the connection table can be changed manually by clicking on the desired field and editing the value. Use the **Apply** button to set the values. Use the **Refresh** button to load the actual settings.

4.8 Ethernet

The Ethernet tab supports the **MII Multiplexer** functionality (Media Independent Interface Multiplexer). With the MII Multiplexer you can establish connections between different communication sources and destinations directly on the device.

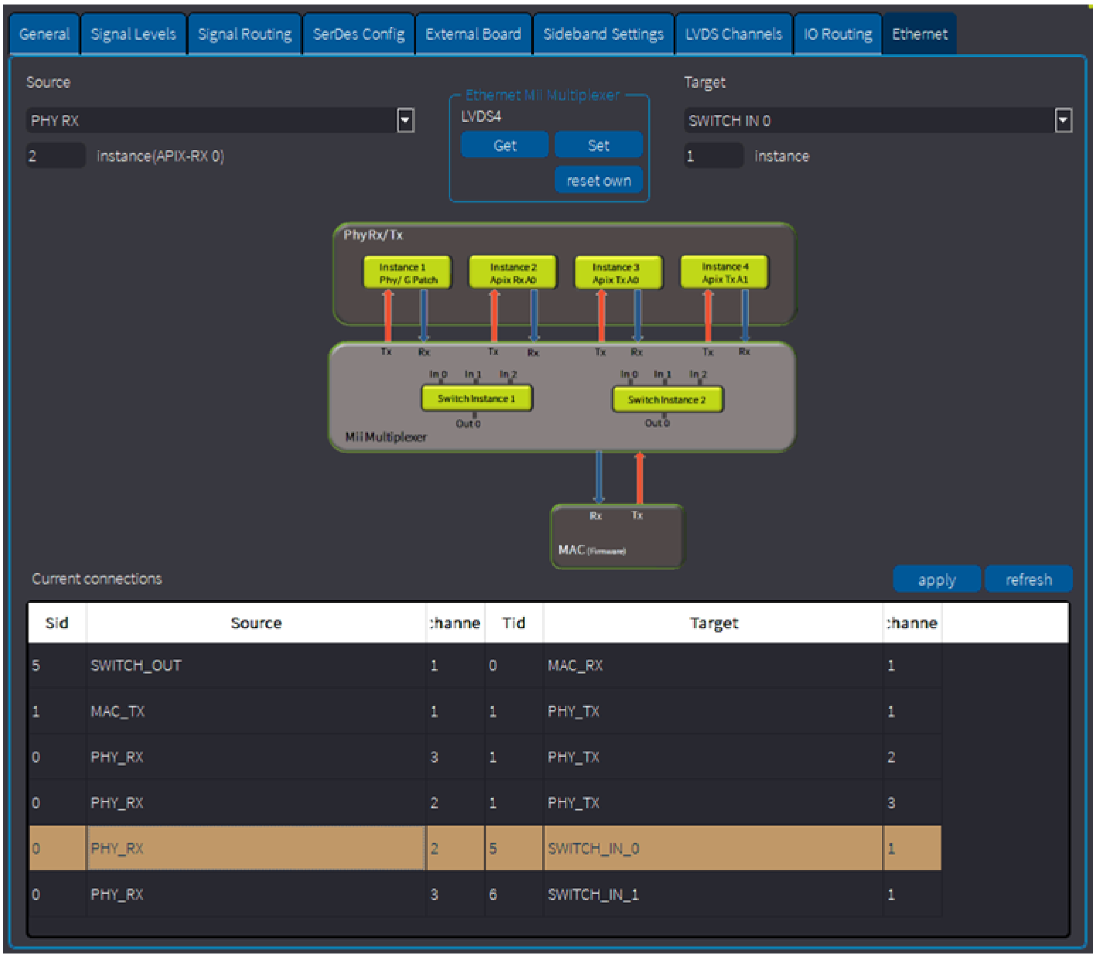


Figure 4.17 **Frame Generator** settings window - Ethernet

Depending on the selected interface the corresponding sources and targets are available. Which ones are available can be seen in the graphic, which is automatically displayed in the tab.

Regarding LVDS devices, this feature is only supported for **Media Interface** boards with Apix MII function (Currently these are the boards INAP562T and INAP562R).

Using the switches of the MII multiplexer, data can be routed from Apix Rx to Apix Tx (Phy) and/or to the MAC of the firmware for instance. The Apix chip has two **A-Shell** channels. In the INAP562T, both **A-Shell** channels are routed out as MII (Tx). With the INAP562R only one (Rx) is routed out. All MIIs are bidirectional.

All possible PHY Rx and MAC Tx instances can be routet to an input of the switch and all PHY Tx and MAC Rx instances can be routet to the output. The possible sources and targets are listed in the drop-down lists. With the help of the graphic the correct instance can be specified in the input field below the dropdown list. All incoming packets of the switch are routed to the switch output. The priority is indicated by the number of the input. So Switch Input0 has the highest priority and Switch Input2 the lowest.

This priority distribution is only valid for the Apix interface. This is not the case for other **GOEPEL electronics** devices with MII Multiplexer.

In the middle of the dialog window there are three buttons:

Button	Description
Get	Load the current MII Multiplexer setting values of the selected interface.
Set	Overwrite all current MII Multiplexer settings on the device with the settings displayed on the tabs of the window.
reset own	Reset all MII Multiplexer settings of the selected interface.

Table 4.2 Ethernet tab buttons

The defined connections can be found in the connection table below the graphic. The channel entries of the connection table can be changed manually by clicking on the desired field and editing the value. Use the **Apply** button to set the values. Use the **Refresh** button to load the actual settings.

4.8.1 Example

This is a short example of pass-through of data from Apix Rx-A0 to Apix Tx-A0. Instance 2 of Phy_Rx needs to be routet to Instance 3 of Phy_Tx. The other way around Instance 3 of Phy_Rx needs to be routet to Instance 2 of Phy_Tx.

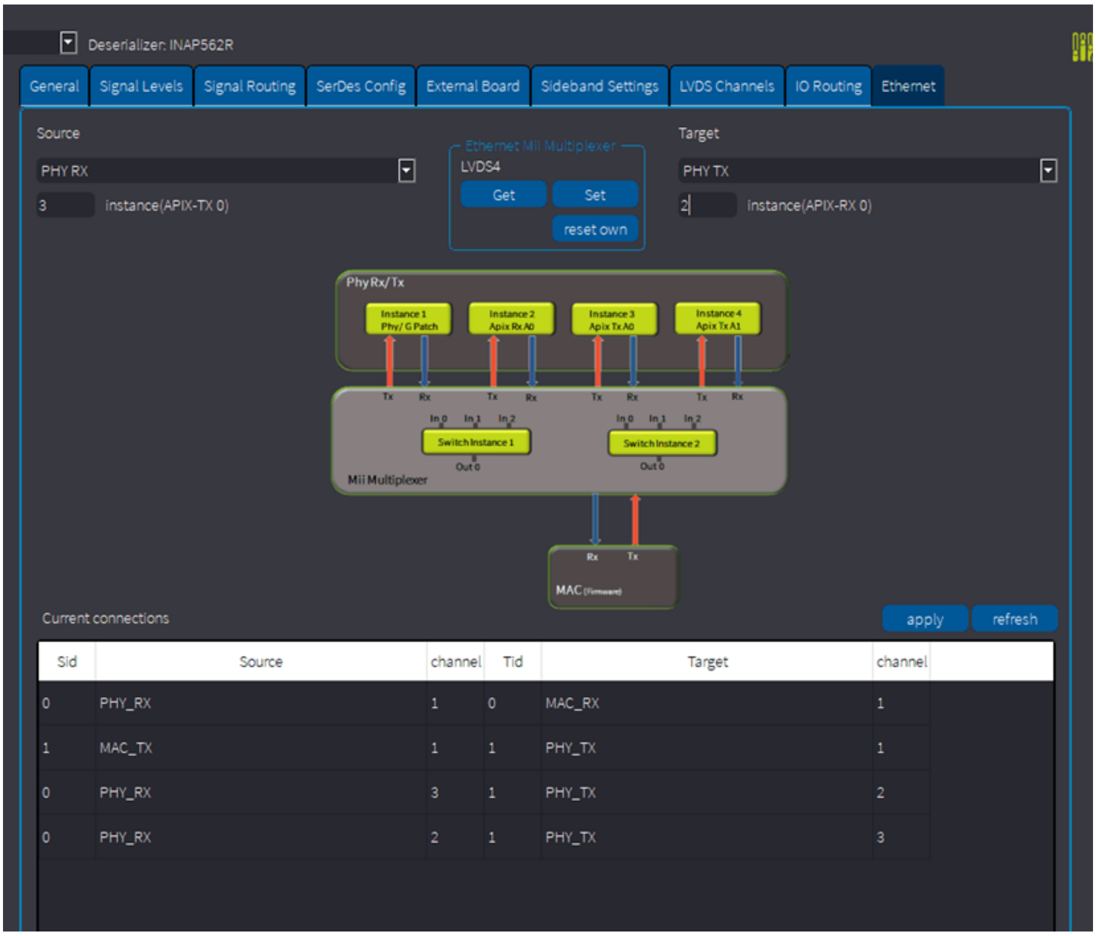


Figure 4.18 Example of MII Multiplexer configuration

4.9 LVDS Info

The **Frame Generator** LVDS Info window for the selected interface can be opened with the icon ()(ALT + 6).

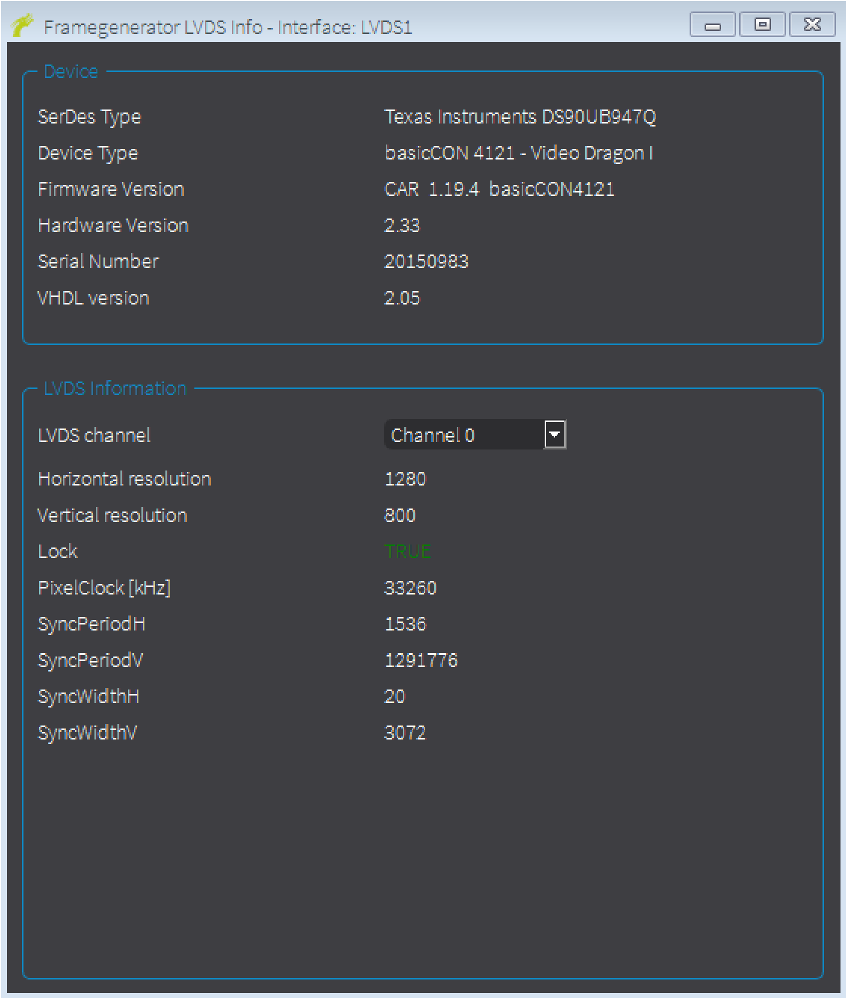


Figure 4.19 **Frame Generator** settings window - LVDS Info

The device information contains:

- SerDes Type: The type of the serializer mounted on the currently installed extension board of the LVDS device.
- Device Type: The type of the LVDS device.
- Firmware Version: The firmware version running on the device.
- Hardware Version: The hardware revision number.
- Serial Number: The serial number of the LVDS device.
- VHDL Version: The version number of the VHDL design of the FPGA device.

The LVDS information contains information about the current video stream:

- LVDS Channel: Select the LVDS Channel. (only supported for **Video Dragon 6222**)
- Horizontal Resolution: Number of horizontal pixel clock cycles where the Data Enable signal is active. The value corresponds to the horizontal resolution of the image source.

- Vertical Resolution: Number of vertical pixel clock cycles where the Data Enable signal is active. The value corresponds to the vertical resolution of the image source.
- Lock: Indicates whether the deserializer is synchronous to the LVDS bit stream. (only for supporting Media Interface Boards)
- PixelClock: Frequency of pixel clock in kHz.
- SyncPeriodH: Absolute number of pixel clock cycles between two horizontal synchronization edges.
- SyncPeriodV: Absolute number of pixel clock cycles between two vertical synchronization edges.
- SyncWidthH: Number of pixel clock cycles during the horizontal synchronization is active. The value corresponds to the Horizontal Sync Width of the image source.
- SyncWidthV: Number of pixel clock cycles during the vertical synchronization is active. The value corresponds to the product of the total horizontal active area hTotal and the Vertical Sync Width of the image source.

5 Frame Generator Dialog Window

Open the **Frame Generator** dialog window with one of the alternatives illustrated in chapter Using the GUI. The dialog window has four segments that allow you to manage the files on the device and view the frames.

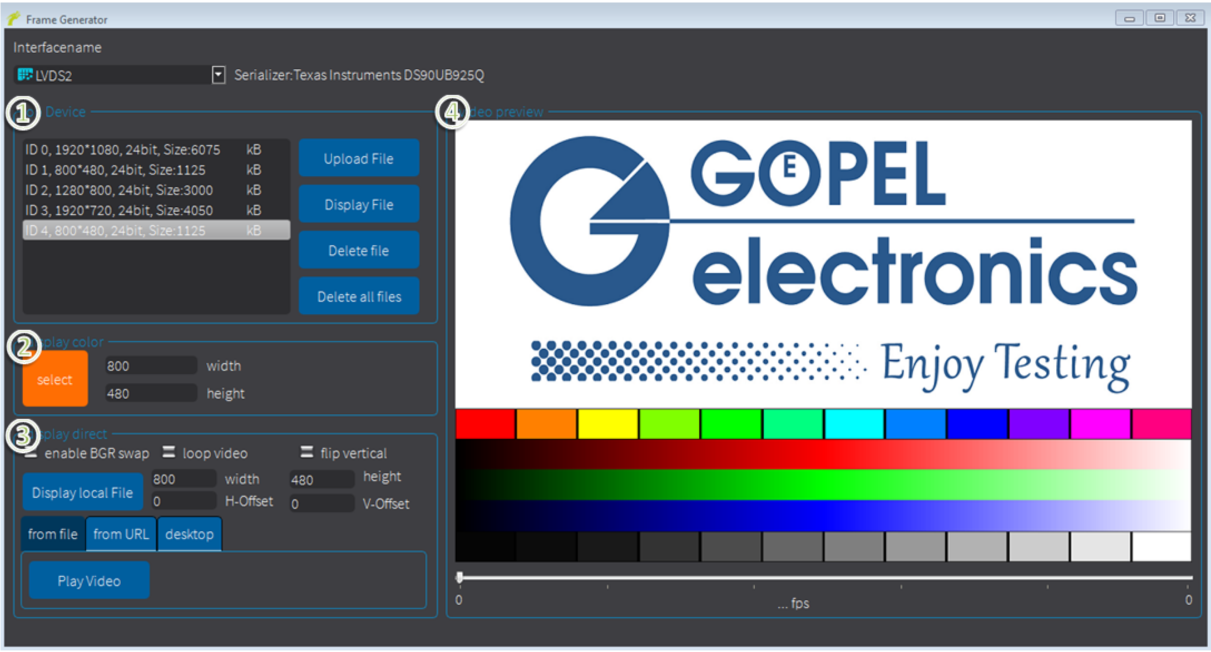


Figure 5.1 **Frame Generator** dialog window

The Dialog Window consists of the following areas:

1. The Files on Device overview
2. The Display Color box
3. The Display Direct box
4. The Video Preview

All available **Frame Generator** interfaces are listed in the drop down menu.

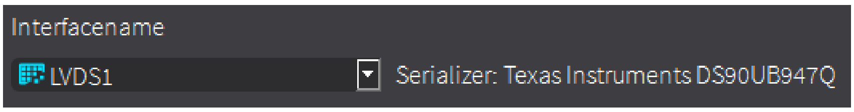


Figure 5.2 Choose interface name

The currently selected interface is shown in the text field of the drop down menu.

When working with **Video Dragon 6222** use the button to switch between Files on Device and Pattern Generator.

5.1 Files on the Device

This dialog segment displays the files stored on the device and has several buttons to manage them.

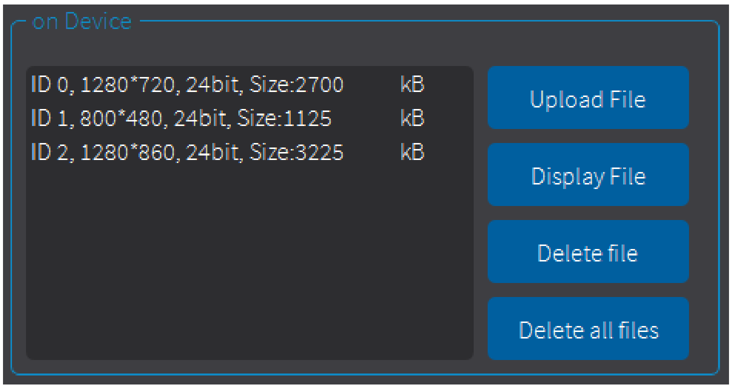


Figure 5.3 On Device segment

The table lists all files stored in the internal **Frame Generator** memory. Each file has a file number (ID) that starts with 0. When you upload a new file, it gets the lowest vacant ID. Next to the ID, the table displays the width and height of the stored images in pixels, the bit depth in bits and the file size in kilobytes. To select a file, left-click in the list. The file is highlighted in the list and the image is also displayed in the preview segment.

On the right side of the segment are four buttons with the following functions:

Button	Description
Upload File	Open a window to select a path to the desired file. The file must be in **24-bit bitmap** format. The selected file is saved in the internal memory of the **Frame Generator** and displayed in the file list on the left.
Display File	Display the file selected in the file list. The selected file width or height can not be larger than configured in the **Frame Generator** settings.
Delete file	Delete the selected file from the internal memory of the **Frame Generator** immediately.
Delete all files	Delete all files from the internal memory of the **Frame Generator** immediately and reset the internal file system to a clean state. Depending on the number of files stored in the memory, this action may take several minutes.

Table 5.1 File management buttons

5.2 Display color

The display color segment provides the ability to select a color to be displayed with the **Frame Generator** as a frame.

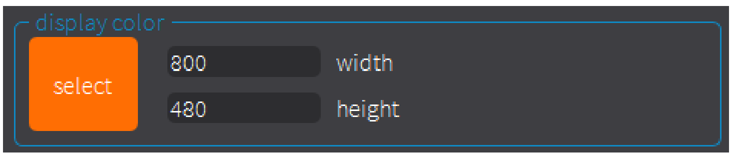


Figure 5.4 Display color segment

The width and height of the frame are arbitrary. Any value from 1 to the maximum width or height configured in the **Frame Generator** settings is valid. Clicking in the colored **select** field opens an additional window for configuring the displayed color.

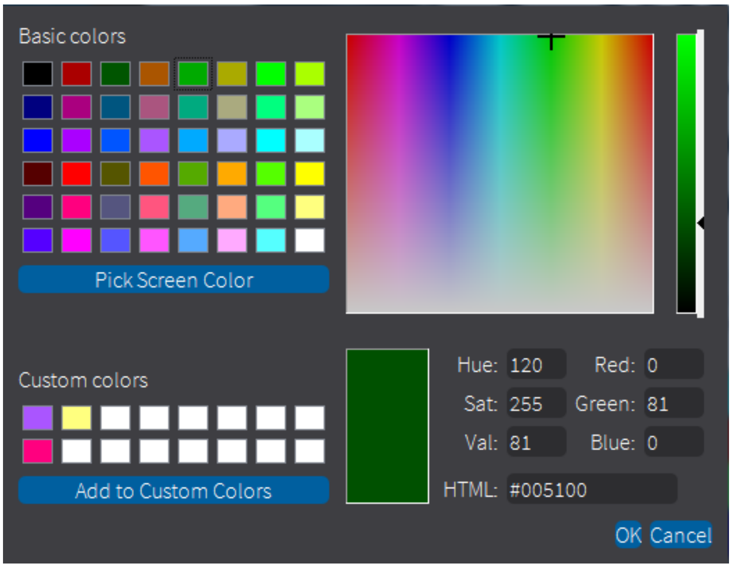


Figure 5.5 Pick a color

After selecting a color and confirming with **OK,** the color is displayed with the **Frame Generator** .

5.3 Display Direct

This segment allows a file or video to be displayed from the PC with the **Frame Generator** without having to upload the file to the device.

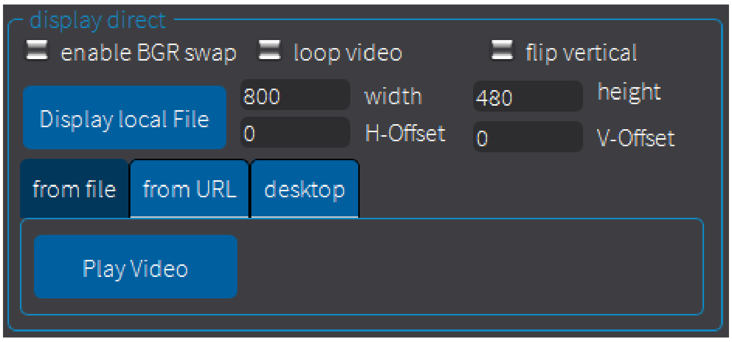


Figure 5.6 Display Direct segment

If the file size is larger than the defined active area, the file is trimmed into the vertical and horizontal active area, beginning at the top left pixel of the file. The size can also be set in this segment using the width and height edit fields. Furthermore, an offset can be defined so as not to start the frame at the upper left pixel but at the desired one.

The specified offset plus file size must not be larger than defined in **Frame Generator** settings.

The **Display local File** button selects an image file to be displayed with the **Frame Generator** . The file type must be a prevalent image file type, such as JPEG or bitmap.

To display a video, go to the **from file** tab and use the **Play Video** button. Then a window opens for selecting a file. After selecting a video file, the **Frame Generator** displays the video prompt.

Use the **from URL** tab to display a stream from a URL, such as a webcam.

The **desktop** tab is used to display the current desktop screen. When using several screens, the desired screen can be selected using the drop down menu. The **Frame Generator** can display the entire screen by downscaling the overall frame to the size of the active area. Therefor use the flag **scale** . To display only a cutout of the overall frame, use the flag **crop** .

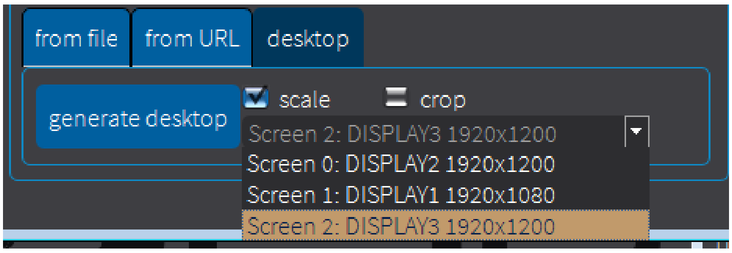


Figure 5.7 Display desktop screen

For direct video presentation, there are some functions that are additionally selectable with the appropriate flag. You can enable the BGR swap, loop the video in a loop by restarting it at the end or flip the image vertically.

5.4 Video Preview

This segment displays the preview image from a file selected in the device file list. Additionally, selected videos played in segment Display Direct are displayed. The scale at the bottom of the segment shows the video timeline and its frame rate in frames per second. While displaying, you can use this timeline to rewind and rewind the video.

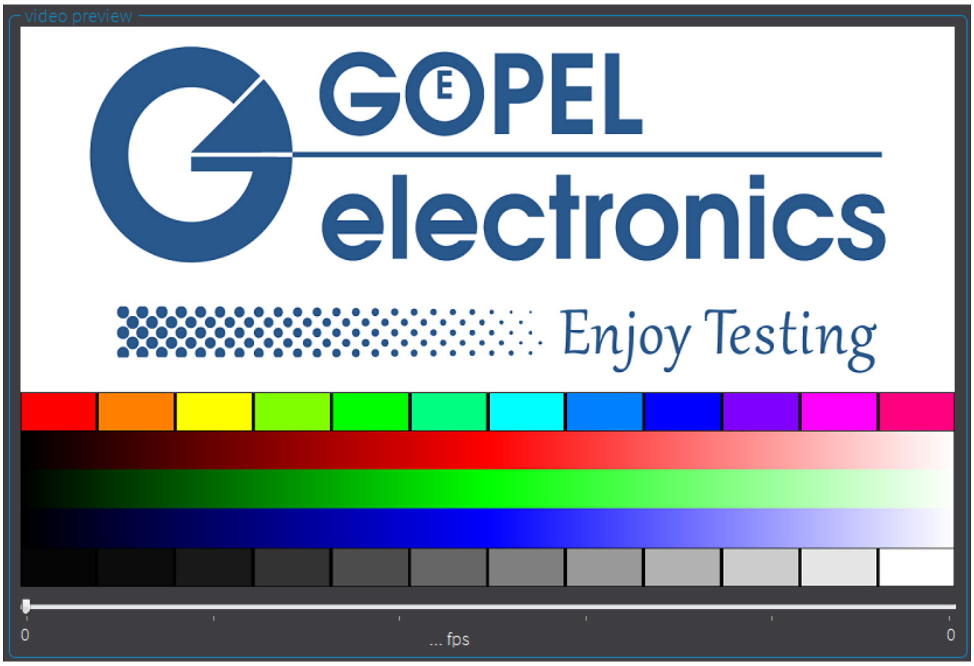


Figure 5.8 Video Preview segment

5.5 Pattern Generator

This feature is only supported by **Video Dragon 6222** .

The **Video Dragon 6222** provides the ability to generate LVDS image data internally. Therefore, an external source is not necessary.

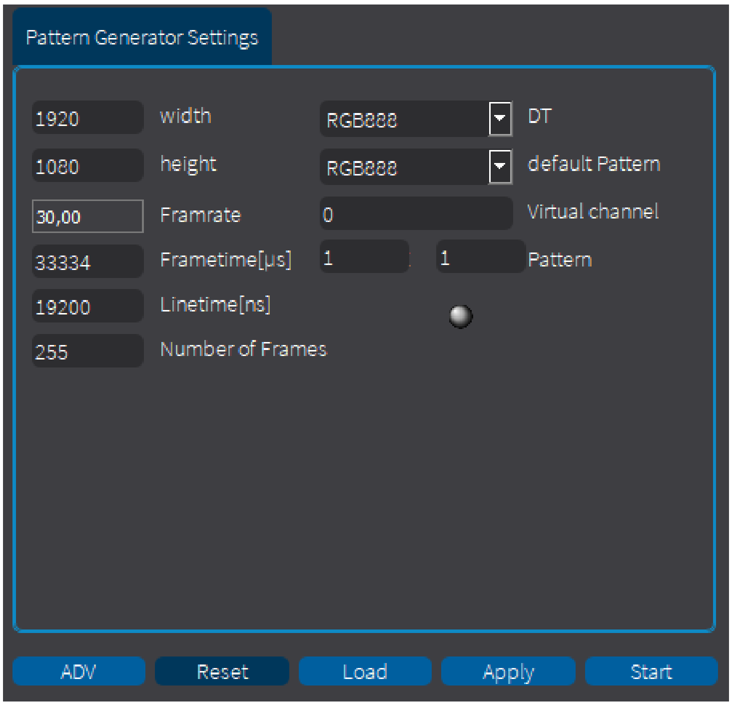


Figure 5.9 Pattern Generator

In this segment the buttons have the following functions:

Button	Description
Init	Start initialisation and unlock the settings options.
Apply	Apply the current settings.
Start	Start the Pattern Generator.
Stop	Stop the Pattern Generator.
Reset	Reset all settings to default.
ADV	Open Advanced Settings Window for detailed color setting.

Table 5.2 Pattern Generator buttons

When the **Init** button is clickeded, the Pattern Generator is initialized to the default values. Now the Settings parameters can be changed as desired and set with **Apply** . The following parameters are available:

Parameter	Description
Width/ Height	Width and Height of the image.
Framerate	Framerate of the image.
Frametime	Frametime of the image in microseconds (read only).
Linetime	Linetime of the image in nanoseconds (read only).
Number of Frames	Number of frames that are displayed from 1 to 254. 255 means infinite displaying.
DT	Picture format of the pattern (RGB888 and YUV 4:2:2 8 Bit mode are possible).
Virtual Channel	Virtual channel ID for MIPI CSI-2.
Pattern	Number of individual pattern segments in horizontal and vertical dimension from 1 to 8.

Table 5.3 Pattern Generator settings

The Pattern Generator can only be started after initialization. As long as an image is being generated, the indicator light in the Pattern Generator Settings window lights up green.

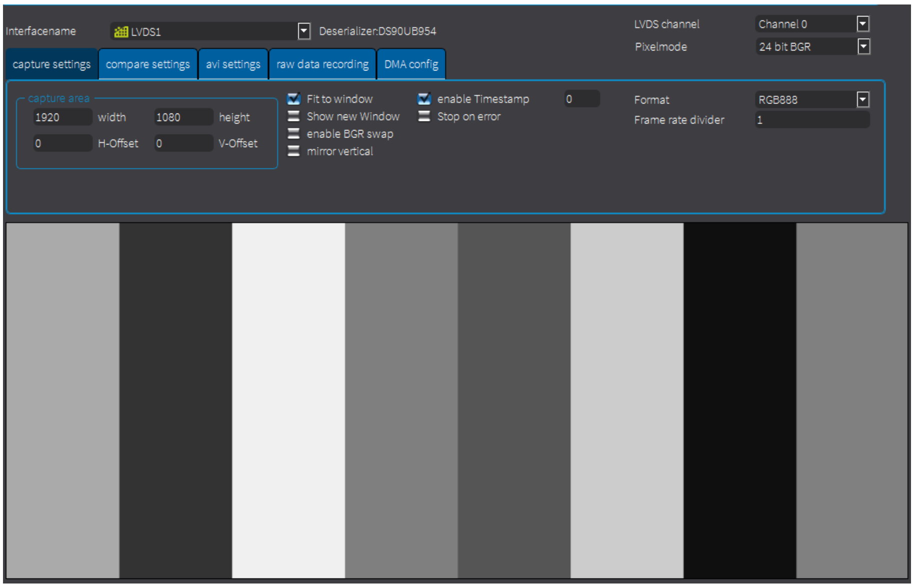


Figure 5.10 Pattern Generator - Capture 8 x 1 pattern

5.5.1 Advanced Pattern Generator

This feature is only available when Dragon Suite Advanced is activated.

In the advanced Pattern Generator window, the 8 refernce pixels of the individual patterns can also be set. Define the number of vertical and horizontal patterns (8 at maximum) and the table below is enlarged or reduced accordingly. Click into the table cells and change the values. The values result from 2 rows of 4 pixels each as RGB hex values.

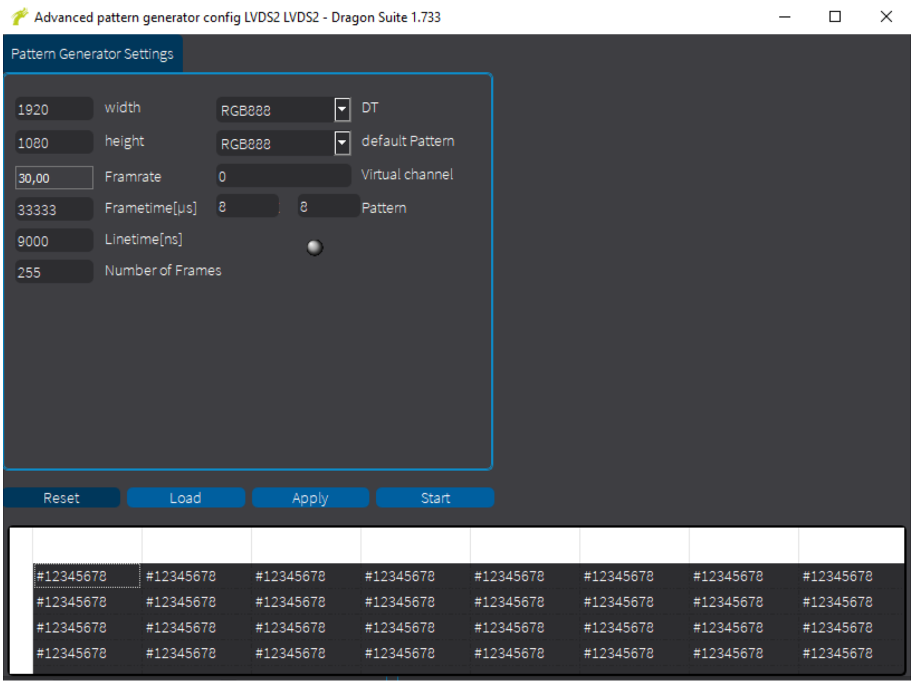


Figure 5.11 Advanced Pattern Generator

6 Setting up the Frame Grabber

Open the **Frame Grabber** settings window by using one of the alternatives illustrated in chapter Using the GUI. The settings window shows all supported device settings. Most of the functionalities of the **Dragon Suite** depend on valid settings of the interface. So setting up the device should be the first step after starting the software.

Generally parallel usage of several interfaces with the **G-API** is possible. But the **Dragon Suite** supports the usage of only one **Frame Grabber** interface at a time.

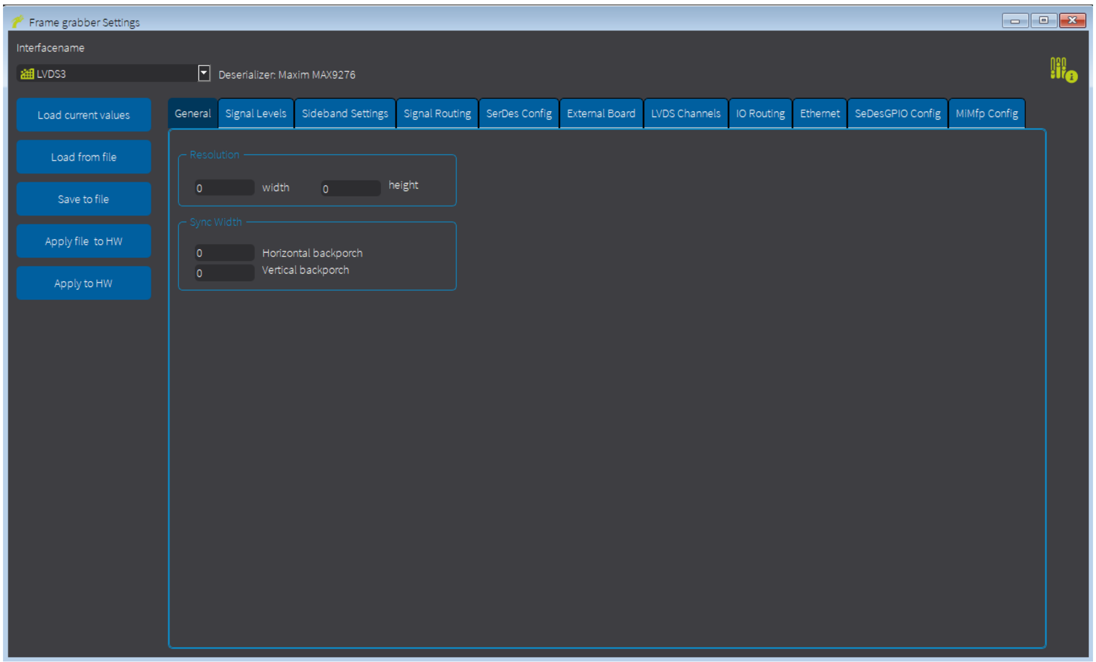


Figure 6.1 **Frame Grabber** settings window

All available **Frame Grabber** interfaces are listed in the drop down menu.

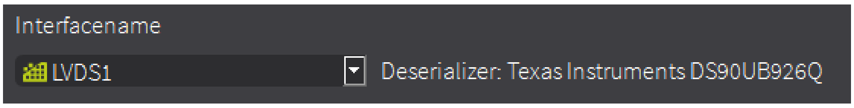


Figure 6.2 Choose interface name

The currently selected interface is shown in the text field of the drop down menu.

On the left of the settings window are four buttons:

Button	Description
Load current values	Load the current setting values of the selected interface.
Load from file	Load the values for the selected interface by importing an external XML file. Importing a settings file does not overwrites the current settings on the device. To overwrite the device settings, use the **Apply to HW** Button.
Save to File	Save the current settings of the selected interface by exporting them to an external XML file.
Apply to HW	Overwrite all current settings on the device with the settings displayed on the tabs of the window.

Table 6.1 **Frame Grabber** setting buttons

A right click on the **Load from file** button opens a selection of the last opened files. This facilitates the search for frequently used configurations. The history can be cleared with **clear history.**

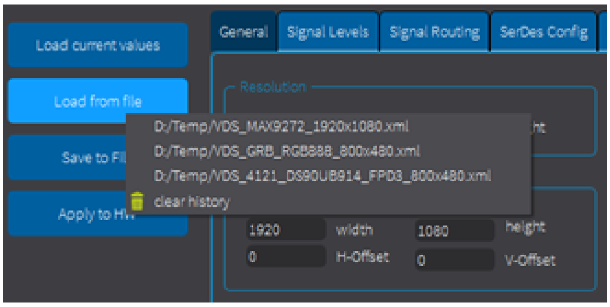


Figure 6.3 **Frame Grabber** Load from file button

6.1 General

The general deserializer settings are resolution and sync width parameters for **basicCON 4121** and resolution for **Video Dragon 6222** .

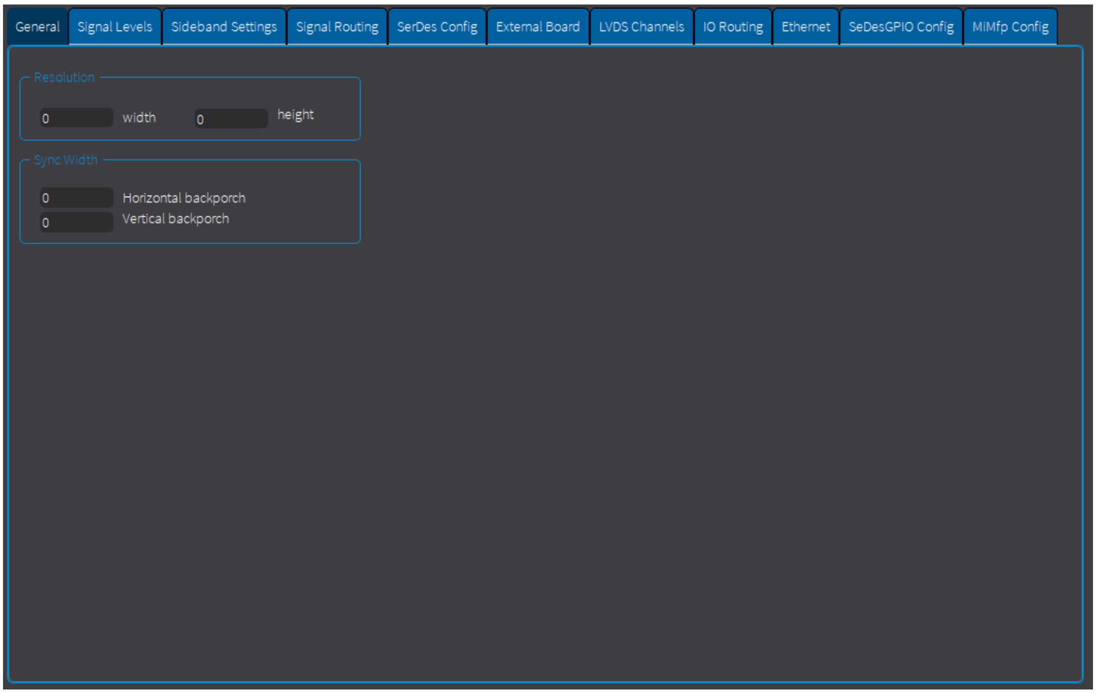


Figure 6.4 **Frame Grabber** settings window - General (for **basicCON 4121**)

The **Resolution** is specified by the number of columns (width) and the number of rows (height) of a bitmap graphic. The maximum resolution of the frame that can be captured is displayed in width and height. These parameters are purely informative for the user and do not need to be modified.

In exceptional cases, no bit should be assigned to the data enable signal in the signal routing settings (see section Signal Routing). If no data enable signal is being transmitted within the LVDS stream, the vertical and horizontal back porches must be set to allow the LVDS device to determine the beginning of the valid pixel data within the stream. These back porch settings in the General Settings tab are called **Sync Width.** The **Vertical back porch** must be set to the number of lines transmitted after a vertical sync signal until a line of valid pixel data begins. The **Horizontal back porch** must be set to the number of columns (pixels) that are transmitted within one line after a horizontal sync signal until valid pixel data begins. When a data enable signal is transmitted and correctly assigned in signal routing, the values of the porches (sync widths) are not important and will not be evaluated. Likewise, the value of the signal level setting of the data enable signal is not evaluated if no data enable signal is transmitted or assigned in the signal routing.

6.2 Signal Levels

The valid signal levels and edges are defined in the Signal Levels tab.

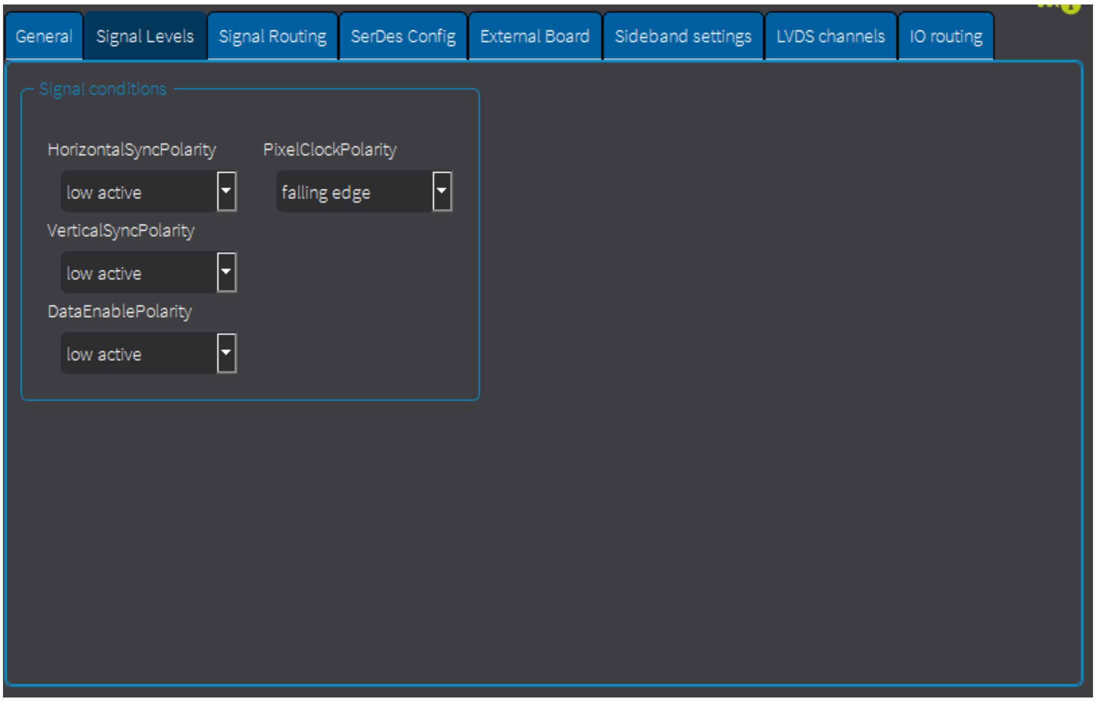


Figure 6.5 **Frame Grabber** settings window - Signal Level

For the synchronization signals, the **Polarity** (low or high active) defines which signal level indicates vertical or horizontal synchronization.

The level setting of the **Data Enable** signal specifies at which level pixel data is being transmitted.

Pixel Clock Polarity describes at which edge of the pixel clock signal (rising or falling) the values of the signal are to be sampled.

6.3 Signal Routing

Since there is no common standard for mapping the video signals to the 32 serialized bits in the LVDS video stream, this mapping must be defined for each device. This could be done on the Signal Routing tab.

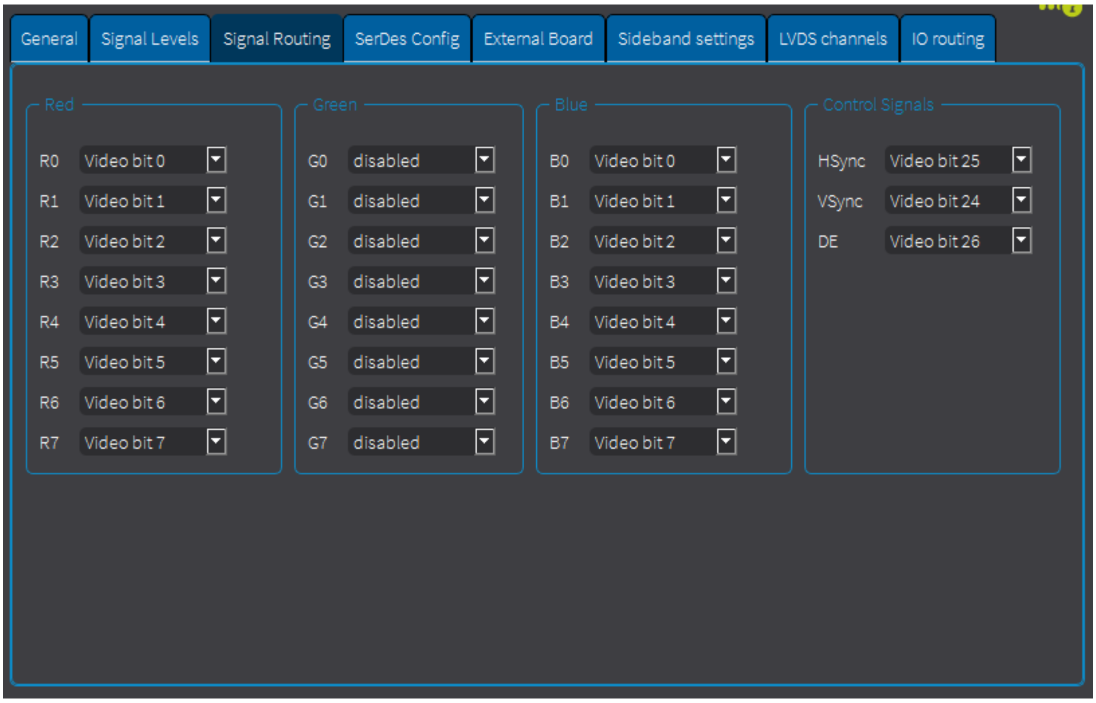


Figure 6.6 **Frame Grabber** settings window - Signal Routing

To deserialize the data within the LVDS data stream, the deserializer needs to know which bit of the data stream should be deserialized to which data. Within the signal routing tab it is possible to define the assignment of each serialized bit of the LVDS data stream to its meaning in the non-serialized video data. Therefore, the combo boxes of the tab provide the ability to set the color bits. Since the resulting RGB frame has a color depth of 24 bit, each color (red, green and blue) has a maximum depth of 8 bits. The selection of the corresponding bit in the video stream is made by selecting the correct from the possibilities of the combo box. All 32 bits of the stream and the value **disabled** can be selected. **Disabled** means that the signal has no correspondent in the video stream. This should be done, for example, if a video stream contains frames with a color depth of less than 24 bits. The significance of the color bits is ascending. For example, this means for an 8-bit color value of red, **R0** is the least significant bit, and **R7** is the most significant bit.

The control signal boxes include the vertical sync signal **VSync,** which indicates the end of the transmission of a complete frame, while the horizontal sync signal **HSync** indicates the end of the transmission of a line. The third control signal is the **Data Enable** signal. In a continuous stream, the video signal may contain porches in each single line and between the end and the beginning of a new frame. The data enable signal indicates whether pixel data is currently being transmitted.

The figure **Frame Grabber** settings window - Signal Routing shows a possible example of video routing. The video stream contains 24-bit RGB frames, with the first 8 bits in the stream representing the 8-bit blue value in ascending order (bit 0 is the least significant, bit 7 is the most significant bit). The following 16 bits represent the green and red values, but without any order. The horizontal sync signal is bit 24, followed by the vertical sync signal and the data enabled signal. Bits 27 to 31 have no assigned values.

There is no common standard for serializing the video data into a 32-bit LVDS stream. The assignment of the several bits to their respective meaning within the deserialization process therefore requires exact knowledge of the settings of the serializer used. The use of wrong settings will most likely lead to incorrect frame data (i.e. wrong colors) or no valid frame data at all. The LVDS device can not determine if the user has entered valid settings and can report capture errors due to these settings, even though the transmitted frame data is correct.

6.4 SerDes Config

Dragon Suite gives the opportunity to configure the **Video Dragon** deserializer by manipulating the bus registers.

Manual manipulating of the configuration data list needs an extreme good knowledge of the meaning of each register value. This can only be obtained from the data sheets of the deserializer. Even a single wrong setting of only one register may render the complete configuration invalid and the deserializer inoperative.

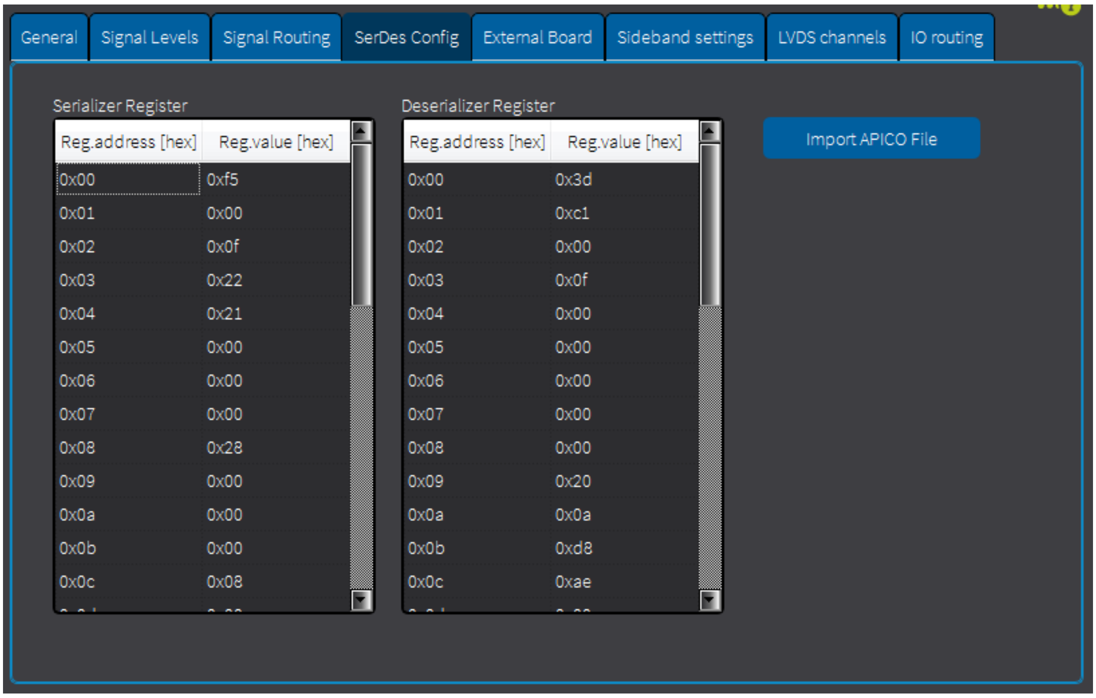


Figure 6.7 **Frame Grabber** settings window - SerDes Config

To set up the deserializer configuration, you must understand the structure of the configuration data: Each deserializer stores its configuration in a set of one-byte register values that can be read from or written to the deserializer. Some deserializer types group subsets of their registers to different internal devices, others have only one internal device. The value can be presented as a decimal or hexadecimal value, which is selected by right-clicking in the list.

In case the **Frame Grabber** is implemented with a serializer too, the serializer register settings also can be configured on this tab.

To add a new row to the register list, right-click in the list and select **add row** from the sub menu. A blank row appears at the bottom of the list. To delete a single row, right-click in the relevant row and select **delete row** . It is also possible to delete all rows.

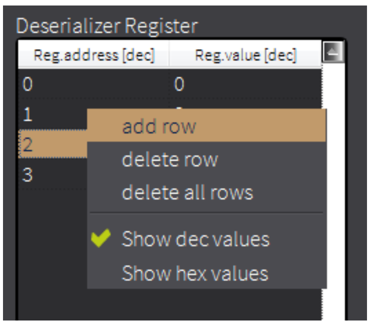


Figure 6.8 Add row to Deserializer Register

To read a register, double-click an existing register address item and change its value to the requested register address. Loading the current values updates the register data from the specified address.

Figure 6.9 Change register address of Deserializer Register

The register data can be changed by double-clicking in the corresponding entry and changing the value. The Apply to HW button immediately overwrites the current value on the deserializer.

Devices with an Apix Media Interface Board require a special configuration via an Apico file. This file can be loaded via the **Import Apico File** button. The button can only be used with the appropriate hardware.

6.5 External Board

Different modes can be set for the various supported deserializers. Depending on the currently selected LVDS interface, only the corresponding setting options are available.

If you set **undefined** in one of the settings, the device configuration is not taken into account.

Further information on the individual setting options can always be found in the data sheet of the respective deserializer.

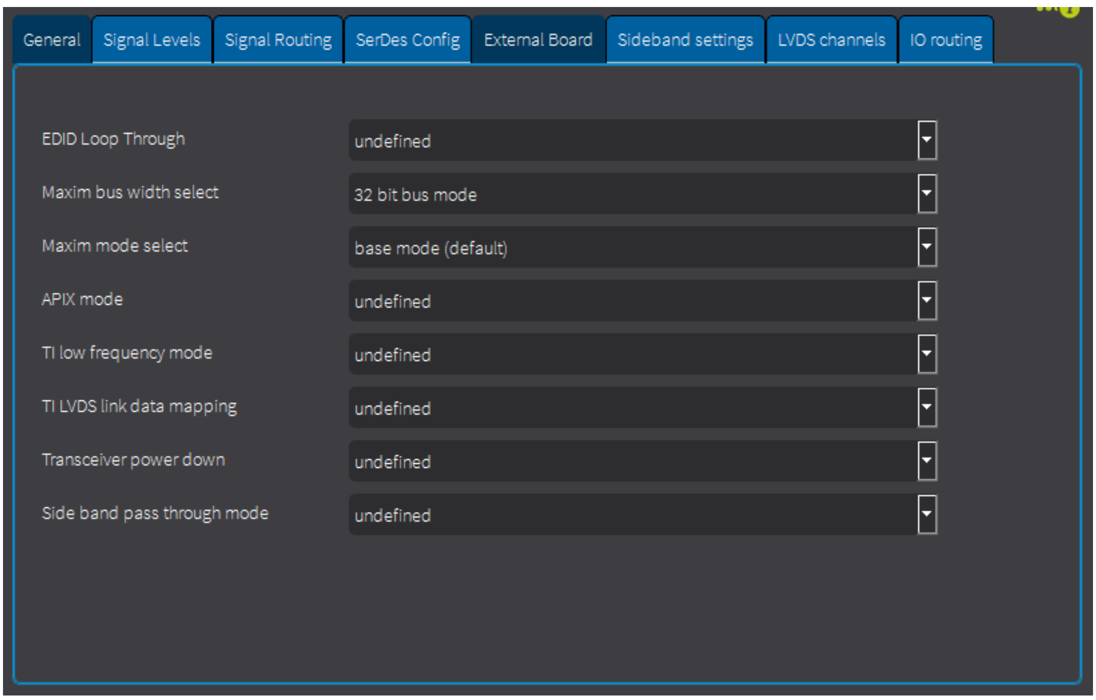


Figure 6.10 **Frame Grabber** settings window - External Board

6.5.1 EDID Loop Through

Extended Display Identification Data (EDID) is a 128-byte data format for displays that describes their capabilities, such as manufacturer, date of manufacturing or display size. This parameter is only supported by ADV7611 (HDMI).

- No loop through (default): the EDID data of serdes extension board are used.
- Loop through enabled: the EDID data of output device is looped to input.

6.5.2 Maxim bus width select

This parameter is only supported by MAX9260, MAX9272, MAX9276, MAX9259, MAX9271 and MAX9275. It describes the data format of the device.

- 24 bit bus mode (default): the first 21 bits contain video data.
- 32 bit bus mode: the first 29 bits contain video data.
- High bandwidth mode (27bit; only supported by MAX9276 and MAX9275): the first 24 bits contain video data or special control signal packets.

The last 3 bits always are the embedded audio channel bit, the forward control channel bit and the parity bit of the serial word.

6.5.3 Maxim mode select

This parameter is only supported by MAX9260, MAX9272, MAX9276, MAX9259, MAX9271 and MAX9275. Two modes of control-channel operation are available for this device:

- Base mode (default): use either I^2C (half-duplex) or GMSL UART protocol (full-duplex).
- Bypass bus mode: use a custom UART protocol.

6.5.4 Maxim DRS select

This parameter is only supported by MAX9276 and MAX9275. There are two modes of operation:

- DRS = low rate
- DRS = high rate (default)

6.5.5 APIX mode

This parameter is only supported by INAP375T and INAP375R. The APIX functionality provides a high-speed Gigabit video link in combination with full-duplex communication over two wire pairs. There are two modes of operation:

- APIX1 mode
- APIX2 mode (default)

6.5.6 TI low frequency mode

This parameter is only supported by DS90UB947 and DS90UB948.

- Normal mode (default)
- Low frequency mode enabled

6.5.7 TI LVDS link data mapping

This parameter is only supported by DS90UB947 and DS90UB948. The device can be configured to accept 24-bit color with two different mapping schemes:

- Mapping mode 0 (default): SPWG mapping.
- Mapping mode 1: OpenLDI mapping.

6.5.8 Transceiver power down

This parameter is only supported by MAX9260, MAX9272, MAX9276 and DS90UB914. The devices have a power-down mode which reduces power consumption:

- Transceiver is powered up (default).
- Transceiver is powered down.

6.5.9 Sideband pass through mode

This parameter is supported only depending on the serializer used. The devices pass a sideband command through to further sideband subscribers.:

- disabled (default)
- I^2C pass through
- UART pass through (only for MAX9276 v1.1)

6.6 Sideband Settings

Different sideband communication modes can be set for the various supported deserializers. The **Video Dragon** supports SPI, I^2C and UART. Depending on the currently selected LVDS interface, **Dragon Suite** will provide only the appropriate data mode options.

Detailed information about sideband can be found in the chapter Sideband Communication.

Data Modes are only available with activated sideband features.

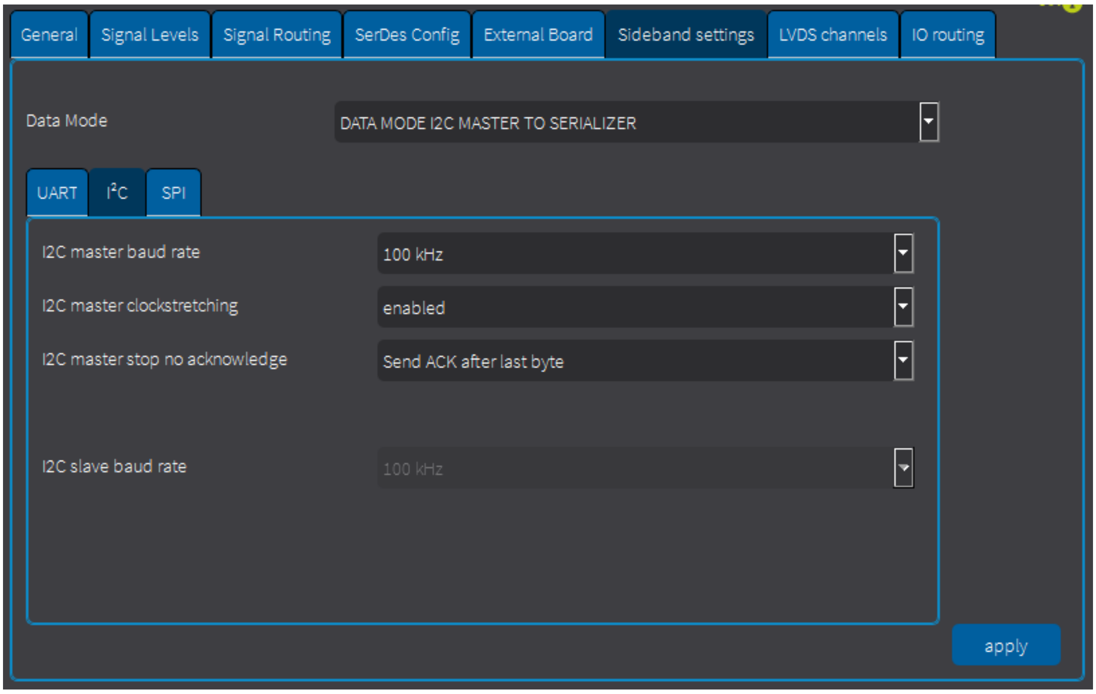


Figure 6.11 **Frame Grabber** settings window - Sideband Settings

TIP

The sideband settings can be set independently of all other parameters. Use the **Apply** button in the lower right corner of the Settings window.

The following Data Modes are possible:

- I^2C master to deserializer: **Video Dragon** is I^2C master at deserializer chip.
- I^2C master to serializer: **Video Dragon** is I^2C master at serializer chip.
- SPI master to deserializer: **Video Dragon** is SPI master at deserializer chip.
- SPI master to serializer: **Video Dragon** is SPI master at serializer chip.
- SPI passthrough dual: **Video Dragon** is connected to transmitter and receiver in "dual mode".
- SPI receiver dual: **Video Dragon** is connected to receiver in "dual mode".
- UART to deserializer: **Video Dragon** is connected to UART interface of deserializer chip.
- UART to serializer: **Video Dragon** is connected to UART interface of serializer chip.
- SPI transmitter dual: **Video Dragon** is connected to transmitter in "dual mode".
- SPI slave to deserializer: **Video Dragon** is SPI slave at deserializer chip.
- SPI slave to serializer: **Video Dragon** is SPI slave at serializer chip.
- I^2C slave to deserializer: **Video Dragon** is I^2C slave at deserializer chip.
- I^2C slave to serializer: **Video Dragon** is I^2C slave at serializer chip.

Depending on the selected Data Mode, the sub-tab for the corresponding sideband is automatically opened.

6.6.1 UART

For UART mode the following parameters can be adapted:

- Baud rate (default value: 115200 baud): Sets the earliest possible value to this parameter.
- Parity (default value: no parity): Sets, if a parity will be build and transferred.

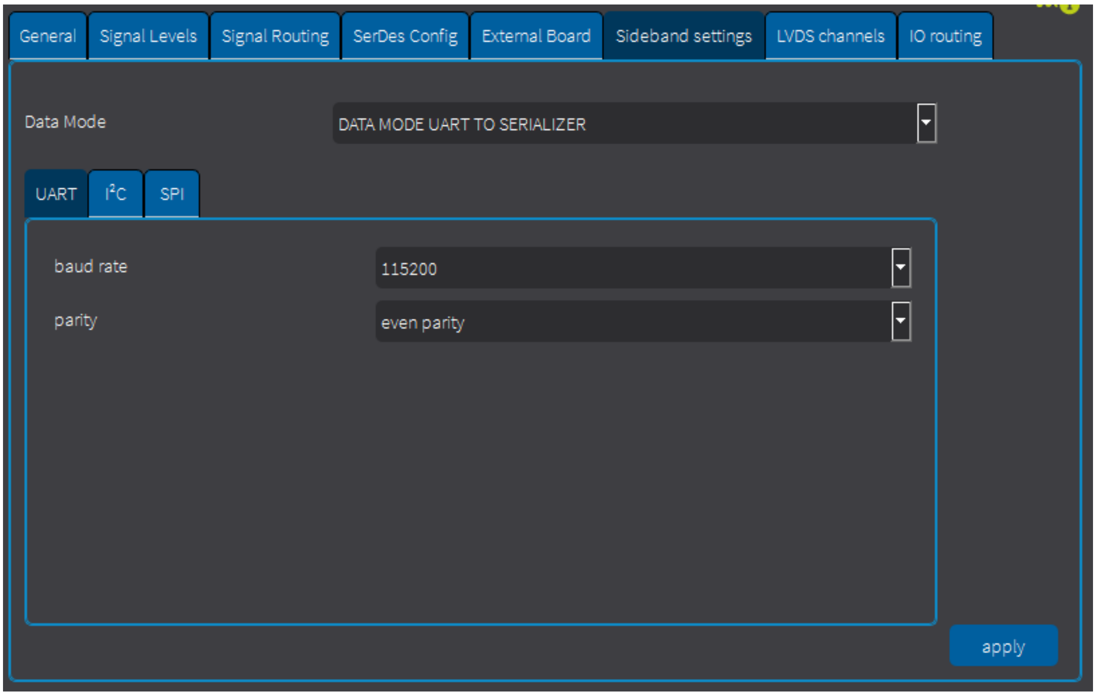


Figure 6.12 Sideband Settings - UART

6.6.2 I²C

For I²C mode the following parameters can be adapted:

- I²C master baud rate (default value: 400kHz): Baud rate of I²C master can be 100kHz or 400kHz.
- I²C master clockstretching (default value: enable): Enables or disables the master response on the deferment of the clock, organized by the slave.
- I²C master stop no acknowledge (default value: Do not send ACK after last byte): I²C master sends an acknowledge or not after the last received byte from the slave.
- I²C slave baud rate (default value: 100kHz): Baud rate of I²C slave can be 100kHz or 400kHz.

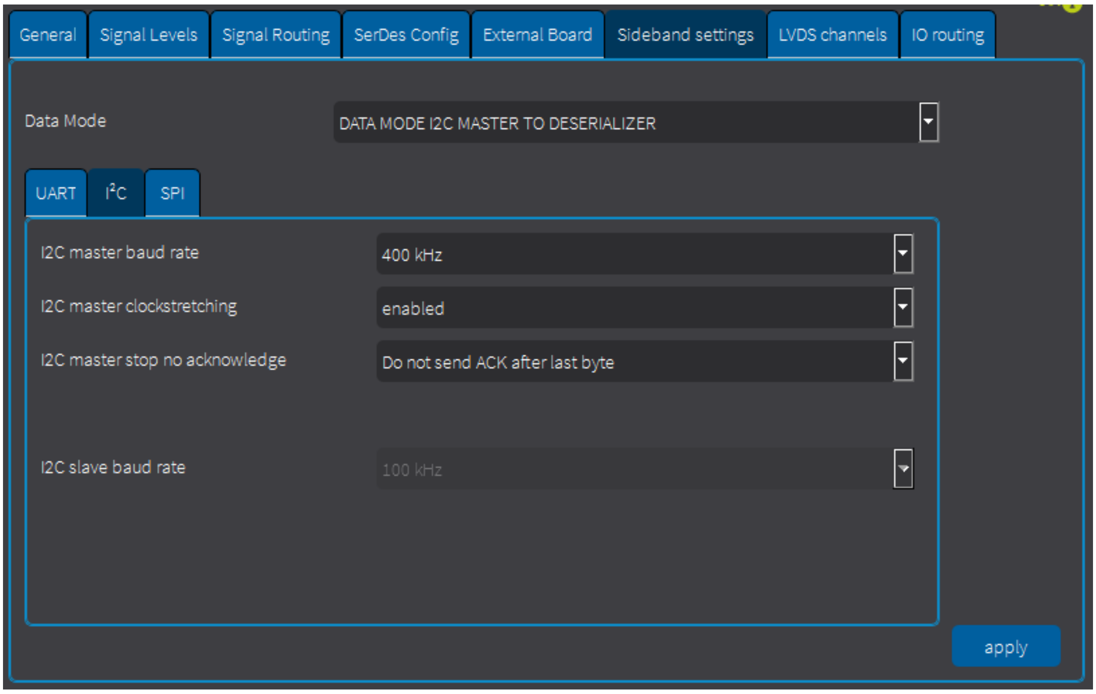


Figure 6.13 Sideband Settings - I²C

6.6.3 SPI

For SPI mode the following parameters can be adapted:

- SPI master CPHA (default value: data is valid at first clock edge after chip select): SPI clock phase - Valid data at first respectively second clock edge after chip select.
- SPI master CPOL (default value: clock parity in idle state is low): SPI clock polarity in idle state.
- SPI master CS_mode (default value: chip select remains low (active) between consecutive bytes): SPI chip select behavior between consecutive bytes of the same transfer.
- SPI master clock divider (default value: 0): determines SPI clock frequency based on freq. 25MHz
  - 0: divider is 1; clock frequency is 25MHz
  - 1: divider is 2; clock frequency is 12.5MHz
  - 2: divider is 3; clock frequency is 8.33MHz
  - ...
  - 255: divider is 256; clock frequency is 0.1953125MHz
- SPI slave CPHA (default value: data is valid at first clock edge after chip select): SPI clock phase - Valid data at first respectively second clock edge after chip select.
- SPI slave CS idle [ns] (default value: 20): SPI chip select idle minimum in 20ns steps
  - 0: 127 * 20ns = 2.54µs chip select idle after transfer
  - 1: 1 * 20ns = 20ns
  - 2: 2 * 20ns = 40ns
  - ...

- 255: 255 * 20ns = 5.1µs

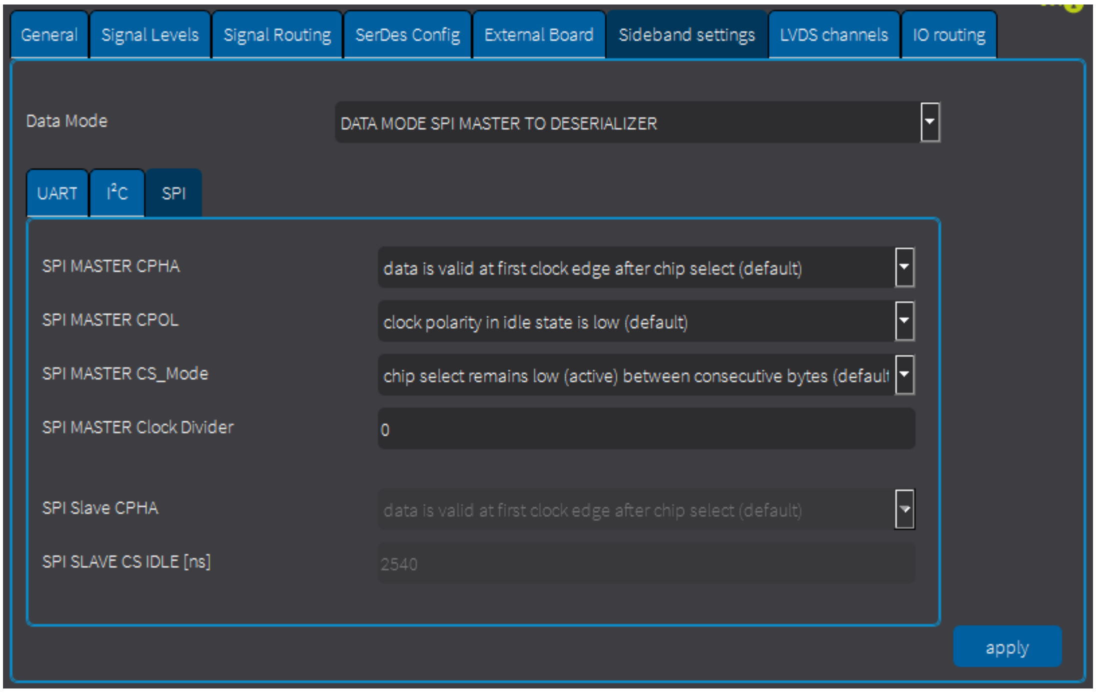


Figure 6.14 Sideband Settings - SPI

6.7 LVDS Channels

Use this tab to configure the capture parameters for the individual channels of the **Video Dragon** . The Capture Area parameter is available for **basicCON 4121** and **Video Dragon 6222** .

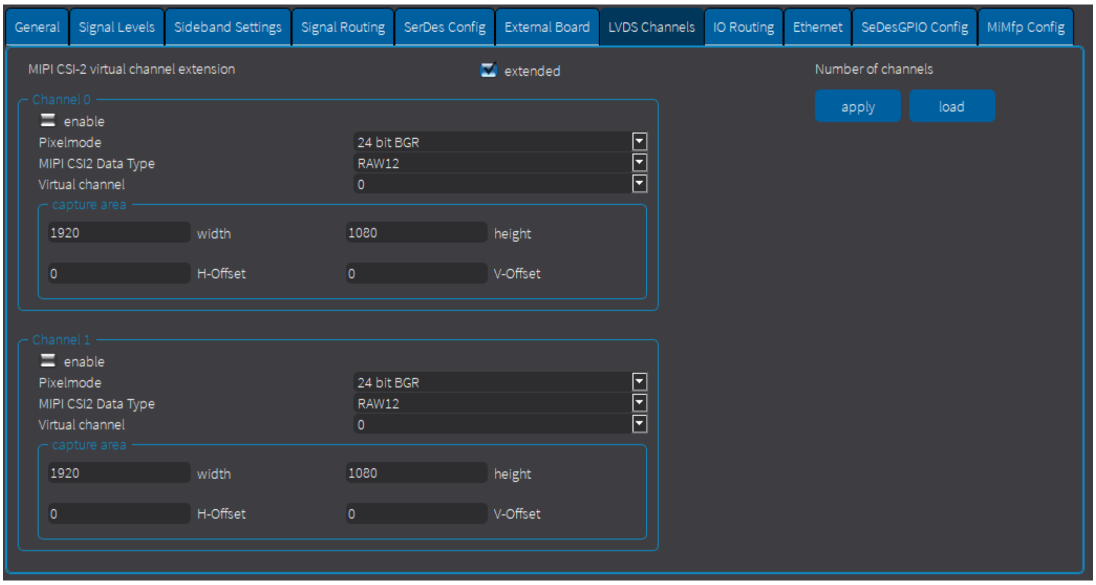


Figure 6.15 LVDS Channels

Capture Area defines the resolution of the frame to be captured, starting with the upper left pixel of the transmitted frame. Any value from 1 to the real maximum width or height of the transmitted video frames is valid . If the values are below the transmitted frame, the captured frame is part of the transmitted frame, starting with the upper left pixel. Larger values than the transmitted frame will result in an error message as soon as a frame has to be captured. To capture a frame with a defined offset, use H-Offset to ignore columns on the left side of the frame, or V-Offset to disregard rows at the top of the frame.

The following properties are only supported for **Video Dragon 6222** with MIPI CSI-2 virtual channel extension:

From **Video Dragon 6222** , several physical channels (Channel 0, Channel 1, etc.) lead to the PC. The number of virtual channels depends on the **Media Interface** module used. Depending on the configuration, the virtual channels can go through one of the physical channels.

Enable or disable the physical channel by checking the **enable** box. You can select the **Pixel Mode** and the **Data Type** for each physical channel.

For the **Pixel Mode** the following modes are possible:

Mode	Description
24 bit BGR	24 bit BGR format with 3 bytes per pixel. Per byte one color channel is transmitted. Byte 0: Blue (bit 7..0) Byte 1: Green (bit 7..0) Byte 2: Red (bit 7..0)
12 bit	RAW12 format. Here two pixels split into three bytes (little endian). Byte 0: Pixel 1 (bit 7..0) Byte 1: Pixel 2 (bit 3..0) und Pixel 1 (bit 11..8) Byte 2: Pixel 2 (bit 11..4)
RAW 8 (only Video Dragon 2)	RAW8 format. Per byte one pixel is transmitted. Byte 0: Pixel 1 (bit 7..0)

Mode	Description
	Byte 1: Pixel 2 (bit 7..0) Byte 2: Pixel 3 (bit 7..0) Byte 3: Pixel 4 (bit 7..0)
RAW 10 MIPI CSI2 (only Video Dragon 2)	RAW10 format. Here four pixels split into five bytes. Recommended memory storage format. Byte 0: Pixel 1 (bit 9..2) Byte 1: Pixel 2 (bit 9..2) Byte 2: Pixel 3 (bit 9..2) Byte 3: Pixel 4 (bit 9..2) Byte 4: Pixel 4 (bit 1..0) and Pixel 3 (bit 1..0) and Pixel 2 (bit 1..0) and Pixel 1 (bit 1..0)
RAW 12 MIPI CSI2 (only Video Dragon 2)	RAW12 Format. Here two pixels split into three bytes (according to MIPI-CSI-2 specification). Byte 0: Pixel 1 (bit 11..4) Byte 1: Pixel 2 (bit 11..4) Byte 2: Pixel 2 (bit 3..0) and Pixel 1 (bit 3..0)
RAW 16 MIPI CSI2 (only Video Dragon 2)	RAW16 format. One pixel is set to two bytes (according to MIPI-CSI-2 specification). Byte 0: Pixel 1 (bit 15..8) Byte 1: Pixel 1 (bit 7..0)
RAW 12 16 LE (only Video Dragon 2)	RAW12 format. One pixel is set to two bytes, with byte 0 shifted 4 bits to the left (little endian). Byte 0: Pixel 1 (bit 3..0) 0b0000 Byte 1: Pixel 1 (bit 11..4)
RAW 16 LE (only Video Dragon 2)	RAW16 format. One pixel is set to two bytes (little endian). Byte 0: Pixel 1 (bit 7..0) Byte 1: Pixel 1 (bit 15..8)
YUV 422 10 MIPI CSI2 (only Video Dragon 2)	YUV422 10 bit MIPI CSI-2. Here YUV data were split into five bytes. Recommended memory storage format. Byte 0: U1 (bit 9..2) Byte 1: Y1 (bit 9..2) Byte 2: V1 (bit 9..2) Byte 3: Y2 (bit 9..2) Byte 4: Y2 (bit 1..0) and V1 (bit 1..0) and Y1 (bit 1..0) and U1 (bit 1..0)

Table 6.2 Pixel Modes for Frame Grabber

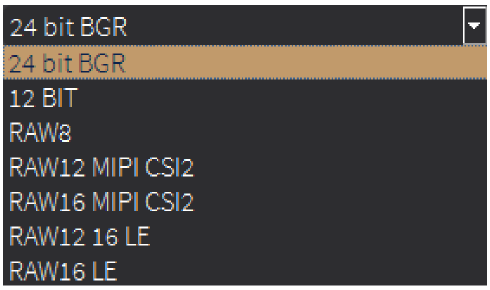


Figure 6.16 Pixel Mode

For the **MIPI CSI-2 Data Type** the following data formats are possible:

- RGB888
- RAW8
- RAW10

- RAW12
- RAW16
- YUV422 8 bit
- YUV422 10 bit
- embedded data (raw data packet, not necessarily just video data)

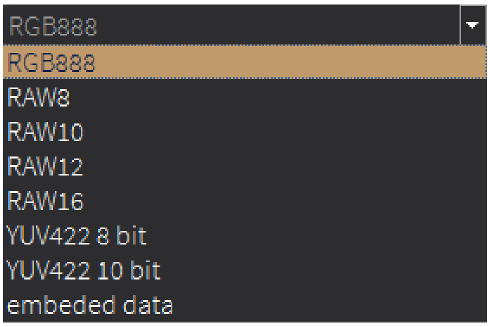


Figure 6.17 MIPI CSI-2 Data Type

From the **Virtual Channel** drop-down menu, you can select the virtual channel to be routed to the appropriate physical channel. Depending on whether the extended box is checked or not, you can choose between 4 or 16 (extended) virtual channels. The parameter applies globally to all physical channels.

Write the displayed settings in the window by clicking the **Apply to HW** button. When you use this button, the settings of the other tabs are not written to the device.

6.8 IO Routing

The **GOEPEL electronics Video Dragon** has an IO trigger function with which certain events can be routed to a trigger event. The routing can be defined in the IO Trigger Dialog window. The defined connections can also be found in the IO Routing tab of the Settings window for an easier overview.

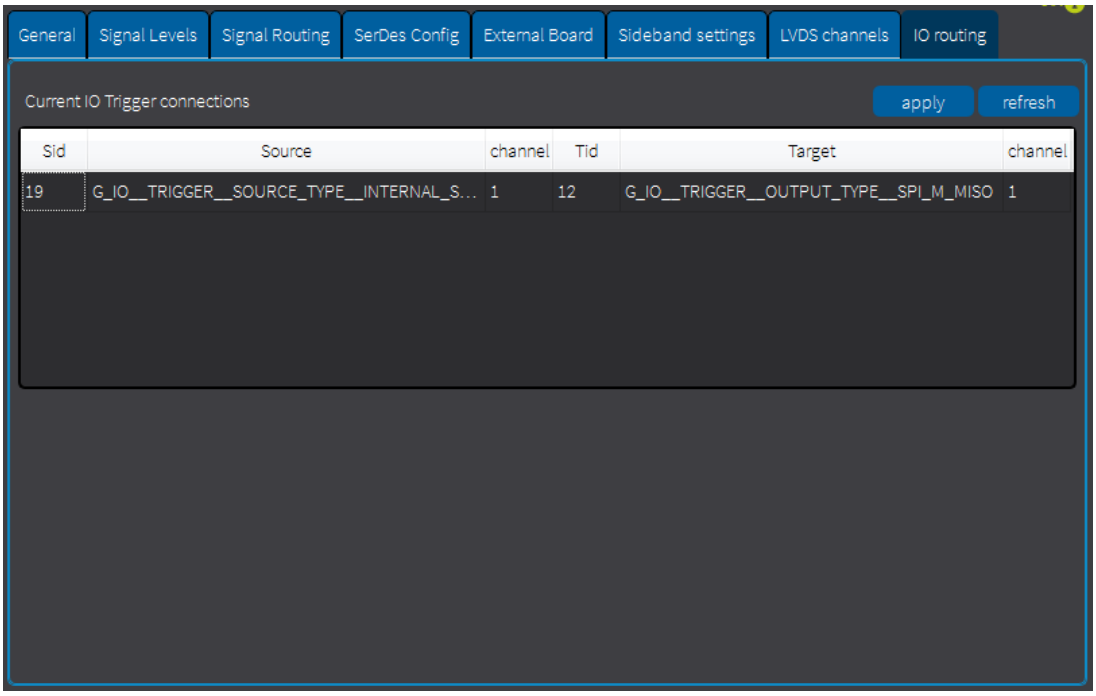


Figure 6.18 **Frame Grabber** settings window - IO Routing

The channel entries of the connection table can be changed manually by clicking on the desired field and editing the value. Use the **Apply** button to set the values. Use the **Refresh** button to load the actual settings.

6.9 Ethernet

The Ethernet tab supports the **MII Multiplexer** functionality (Media Independent Interface Multiplexer). With the MII Multiplexer you can establish connections between different communication sources and destinations directly on the device.

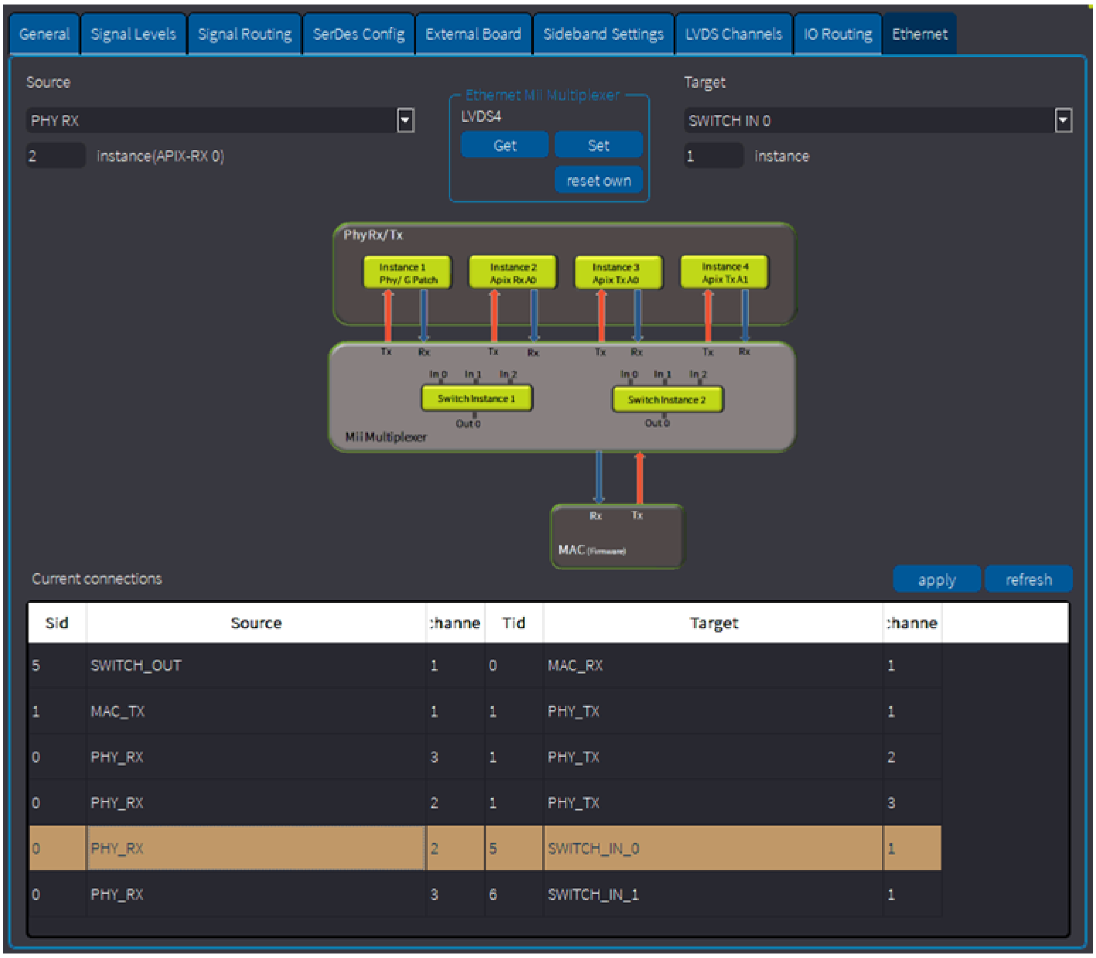


Figure 6.19 **Frame Grabber** settings window - Ethernet

Depending on the selected interface the corresponding sources and targets are available. Which ones are available can be seen in the graphic, which is automatically displayed in the tab.

Regarding LVDS devices, this feature is only supported for **Media Interface** boards with Apix MII function (Currently these are the boards INAP562T and INAP562R).

Using the switches of the MII multiplexer, data can be routed from Apix Rx to Apix Tx (Phy) and/or to the MAC of the firmware for instance. The Apix chip has two **A-Shell** channels. In the INAP562T, both **A-Shell** channels are routed out as MII (Tx). With the INAP562R only one (Rx) is routed out. All MIIs are bidirectional.

All possible PHY Rx and MAC Tx instances can be routet to an input of the switch and all PHY Tx and MAC Rx instances can be routet to the output. The possible sources and targets are listed in the drop-down lists. With the help of the graphic the correct instance can be specified in the input field below the dropdown list. All incoming packets of the switch are routed to the switch output. The priority is indicated by the number of the input. So Switch Input0 has the highest priority and Switch Input2 the lowest.

This priority distribution is only valid for the Apix interface. This is not the case for other

GOEPEL electronics devices with MII Multiplexer.

In the middle of the dialog window there are three buttons:

Button	Description
Get	Load the current MII Multiplexer setting values of the selected interface.
Set	Overwrite all current MII Multiplexer settings on the device with the settings displayed on the tabs of the window.
reset own	Reset all MII Multiplexer settings of the selected interface.

Table 6.3 Ethernet tab buttons

The defined connections can be found in the connection table below the graphic. The channel entries of the connection table can be changed manually by clicking on the desired field and editing the value. Use the **Apply** button to set the values. Use the **Refresh** button to load the actual settings.

6.9.1 Example

This is a short example of pass-through of data from Apix Rx-A0 to Apix Tx-A0. Instance 2 of Phy_Rx needs to be routet to Instance 3 of Phy_Tx. The other way around Instance 3 of Phy_Rx needs to be routet to Instance 2 of Phy_Tx.

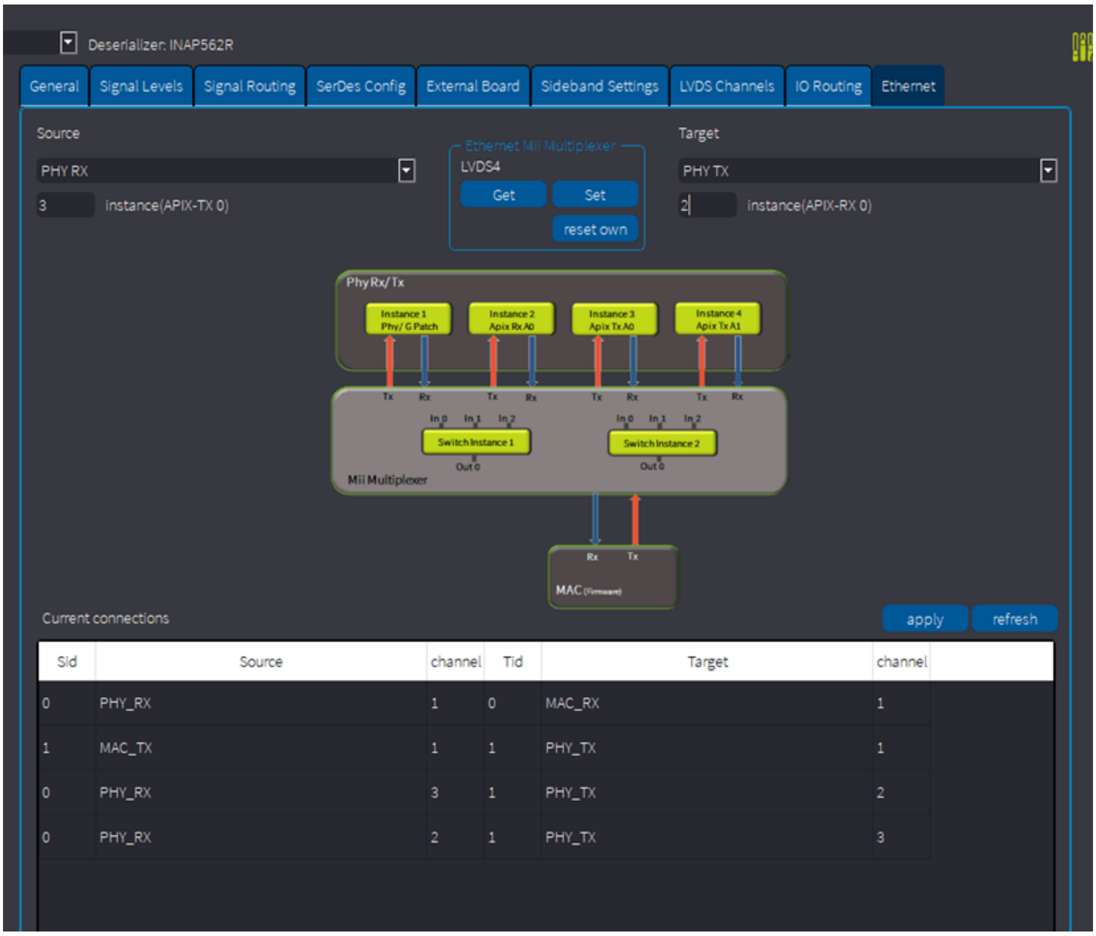


Figure 6.20 Example of MII Multiplexer configuration

6.10 LVDS Info

The **Frame Grabber** LVDS Info window for the selected interface can be opened with the icon ( )(ALT + 6). The window consists of two parts: the Device Information and the LVDS Information. For **basicCON 4121** and **Video Dragon 6222** the device information is the same. However, the LVDS information differs between the two devices.

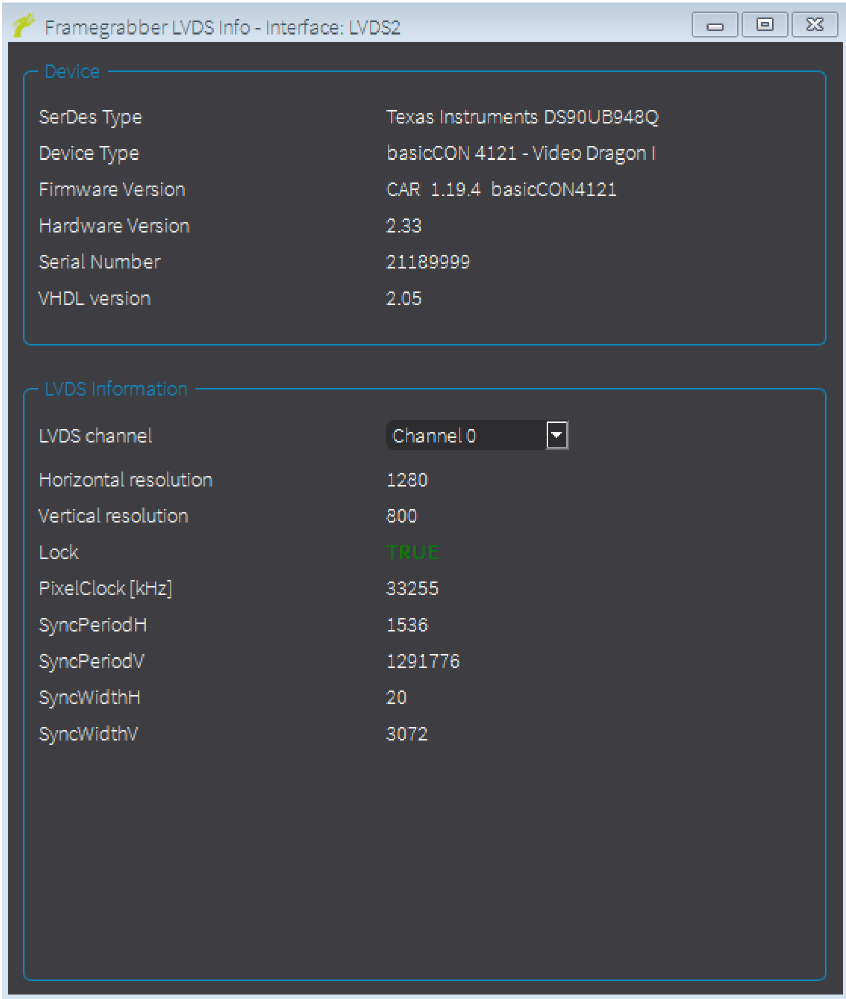


Figure 6.21 **Frame Grabber** settings window - LVDS Info of **basicCON 4121**

The device information contains:

- SerDes Type: The type of the deserializer mounted on the currently installed extension board of the LVDS device.
- Device Type: The type of the LVDS device.
- Firmware Version: The firmware version running on the device.
- Hardware Version: The hardware revision number.
- Serial Number: The serial number of the LVDS device.
- VHDL Version: The version number of the VHDL design of the FPGA device.

6.10.1 LVDS Information of basicCON 4121

The LVDS information contains information about the current **basicCON 4121** video stream:

- LVDS Channel: Select the LVDS Channel. (for **basicCON 4121** only one channel is supported)
- Horizontal Resolution: Number of horizontal pixel clock cycles where the Data Enable signal is active. The value corresponds to the horizontal resolution of the image source.
- Vertical Resolution: Number of vertical pixel clock cycles where the Data Enable signal is active. The value corresponds to the vertical resolution of the image source.
- Lock: Indicates whether the deserializer is synchronous to the LVDS bit stream.
- PixelClock: Frequency of pixel clock in kHz.
- SyncPeriodH: Absolute number of pixel clock cycles between two horizontal synchronization edges.
- SyncPeriodV: Absolute number of pixel clock cycles between two vertical synchronization edges.
- SyncWidthH: Number of pixel clock cycles during the horizontal synchronization is active. The value corresponds to the Horizontal Sync Width of the image source.
- SyncWidthV: Number of pixel clock cycles during the vertical synchronization is active. The value corresponds to the product of the total horizontal active area hTotal and the Vertical Sync Width of the image source.

6.10.2 LVDS Information of Video Dragon 6222

The LVDS information contains information about the current **Video Dragon 6222** video stream:

- LVDS Channel: Select the LVDS Channel.
- Horizontal Resolution: Number of horizontal pixel clock cycles where the Data Enable signal is active. The value corresponds to the horizontal resolution of the image source.
- Vertical Resolution: Number of vertical pixel clock cycles where the Data Enable signal is active. The value corresponds to the vertical resolution of the image source.
- Lock: Indicates whether the deserializer is synchronous to the LVDS bit stream.
- Frame rate: The speed at which the images are sent in FPS (Frames per second).
- CSI2 data type: MIPI CSI-2 type of the data.
- Pixel mode: Pixel Mode of the data.
- Frame counter: Number of captured frames.
- DMA channel: Used DMA channel.
- Transfer mode: Mode, how the data is transferred (Possible modes: Command-Response mode, DMA mode).
- Device framebuffer count: Number of frame buffers within the device.
- Frame rate divisor: Specifies whether all frames or every x-th frame should be captured.
- Device framebuffer overwrite: Specifies whether full frame buffers are overwritten when new frames receive.
- Frame count: Number of frames to capture. When number of frames to capture is reached, frame capturing is automatically disabled.
- Capture enable state: Specifies whether Capture is enabled or disabled.

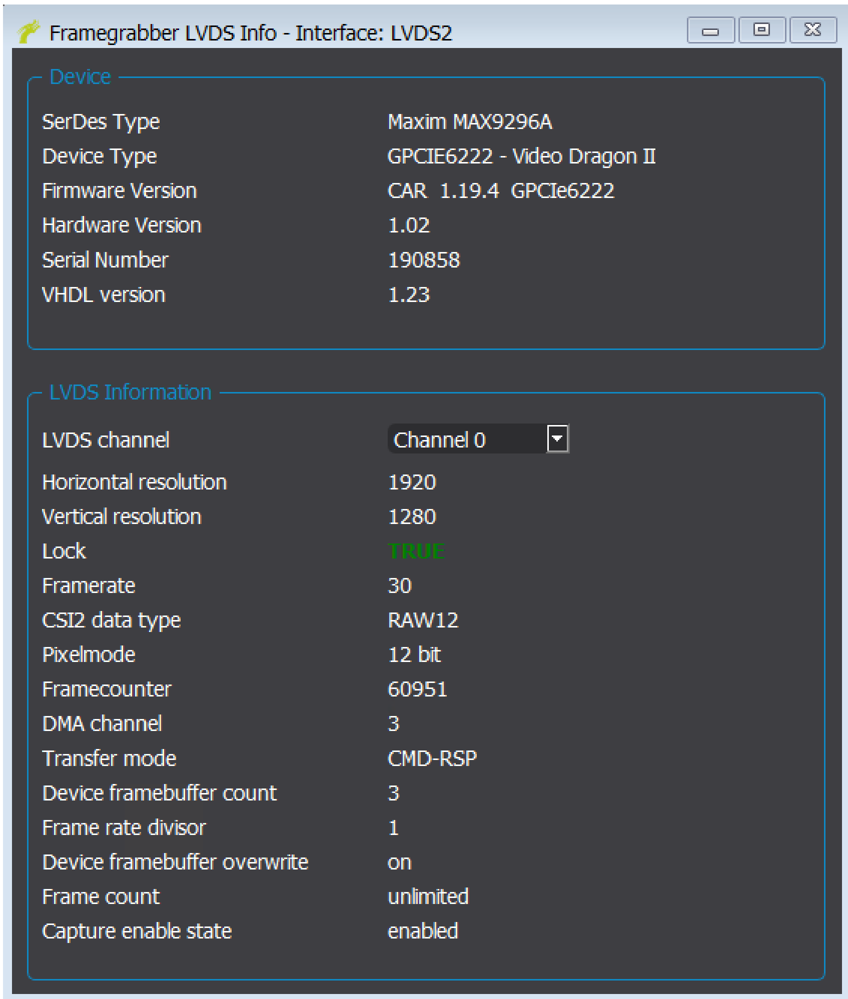


Figure 6.22 **Frame Grabber** settings window - LVDS Info of **Video Dragon 6222**

7 Frame Grabber Dialog Window

Open the **Frame Grabber** dialog window with one of the alternatives illustrated in chapter Using the GUI. The dialog window has several segments to manage the capturing process.

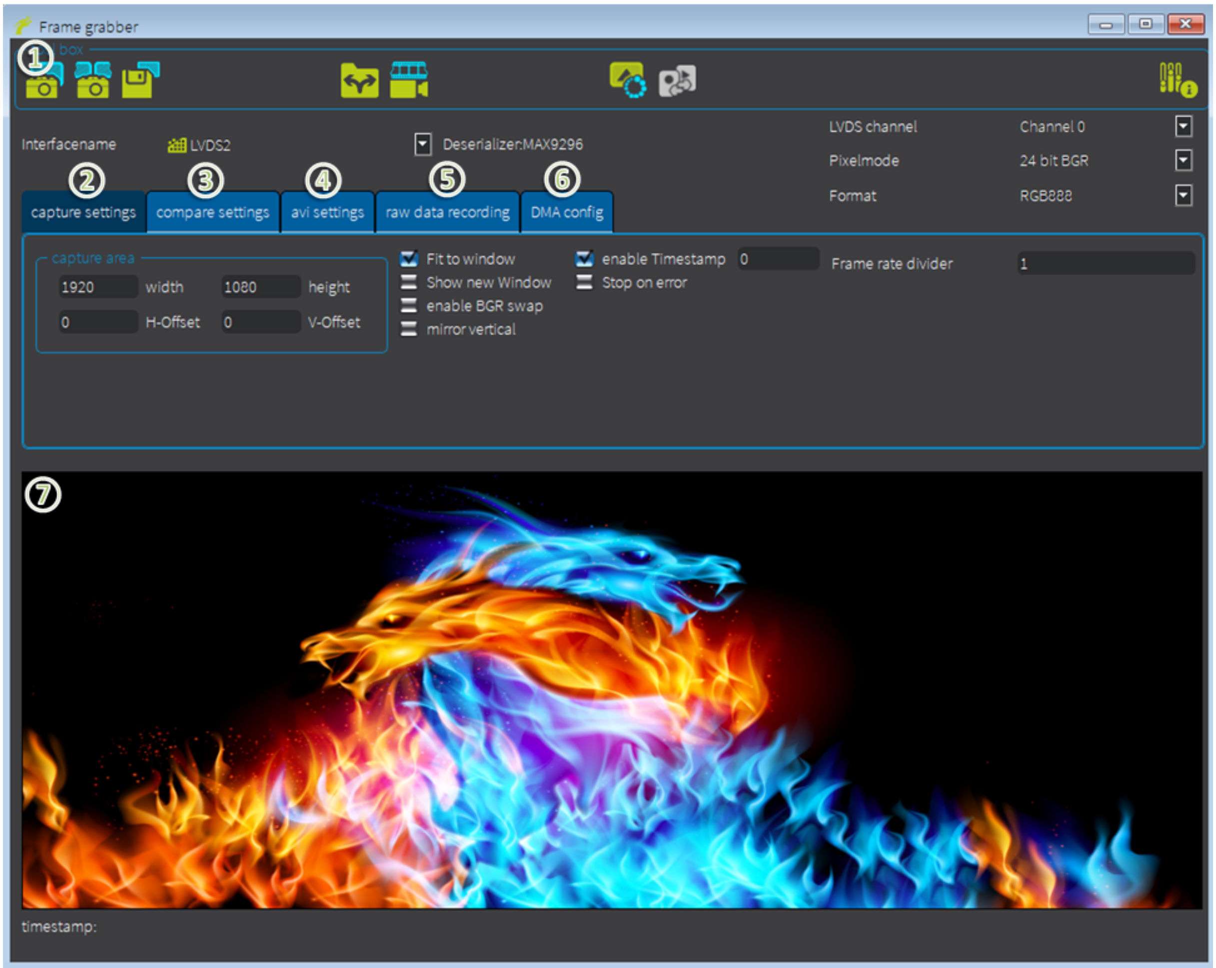


Figure 7.1 **Frame Grabber** dialog window

The Dialog Window consists of the following areas:

1. The Tool Box
2. The Capture Settings tab
3. The Compare Settings tab
4. The Avi Settings tab
5. The Raw Data Recording tab
6. The DMA Config tab
7. The Frame Area

All available **Frame Grabber** interfaces are listed in the drop down menu.

Figure 7.2 Choose interface name

The currently selected interface is shown in the text field of the drop down menu.

The **LVDS Channel** setting in the dropdown menu on the right can be used for **Video Dragon 6222** only. Here you can choose between LVDS channels 0 and 1.

Figure 7.3 LVDS Channel

The Color Data and Data Format of the captured frame can be changed in the dropdown menus on the right.

For the **Color Data Format** the following data formats are possible:

- RGB888
- RGB444
- YUV422 8bit
- YUV422 10bit
- YUV422 12bit
- YUV422 DBL
- Bayer BG
- Bayer GB
- Bayer RG
- Bayer GR
- GREY 8bit
- GREY 12bit

Figure 7.4 Data format

For the **Pixel Mode** the following modes are possible:

- 24 bit BGR
- 12 Bit
- RAW 8 (Video Dragon 2)
- RAW 10 MIPI CSI2 (Video Dragon 2)
- RAW 12 MIPI CSI2 (Video Dragon 2)
- RAW 16 MIPI CSI2 (Video Dragon 2)

- RAW 12 16 LE (Video Dragon 2)
- RAW 16 LE (Video Dragon 2)
- YUV 422 10 MIPI CSI2 (Video Dragon 2)

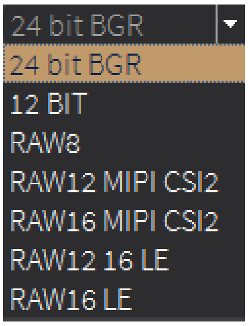


Figure 7.5 Pixel Mode

For a more complete way to customize the LVDS channels for Video Dragon 2, see the LVDS Channels tab in Frame Grabber Settings section.

7.1 Tool Box

The Tool Box segment provides several icons with the following functions:

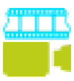
Icon	Description
	Capture a single frame from the selected device and display it in the frame area (ALT + 1).
	Capture images continuously and display them in the frame area. The capturing starts immediately after clicking and stops after clicking again. The icon will remain red while capturing is in progress (ALT + 2).
	Save the last captured frame to a file. The last captured frame can always be seen in the frame area (ALT + 3).
	Choose a path to the directory where video files created in **Record mode** will be stored (ALT + 4).
	Record continuously captured frames to an AVI video file. If no path is selected yet, a window opens in which you can select the directory. An existing file will be overwritten. The recording starts immediately after clicking and stops after clicking again. The icon will remain red while capturing is in progress (ALT + 5).
	Load a reference frame for compare function by selecting an image file (ALT + r).
	Compare a captured frame to the loaded reference frame (ALT + c).
	Open LVDS info window (ALT + 6).

Table 7.1 Toolbar icons

TIP

When you save the captured frame (), you can also save the image as **.dat** -file. This will save the raw data directly from the buffer without changing it. However, this only works after recording a single frame!

7.2 Capture Settings

As in chapter Framegrabber General Settings, the Capture Area defines the resolution of the frame to be captured, starting at the specified offset.

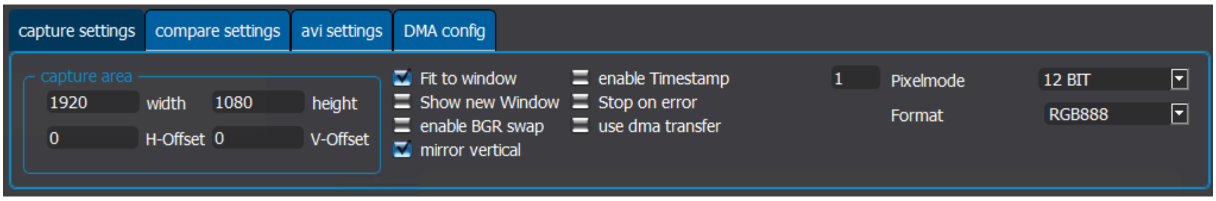


Figure 7.6 Capture area segment

Additionally in this tab are several flags to configure the frame area:

Flag	Description
Fit to window	The captured frame is adapted to the entire frame area size. Changing the size of the dialog window also changes the size of the captured frame.
Show new window	The captured frame will be displayed in a new full-screen window. At the bottom of the new window, the window coordinates and RGB data of the pixel where the mouse is located are displayed. You can also use the mouse scroll wheel to zoom into the image until the single RGB pixel values are displayed.
Enable BGR swap	Change from RGB to BGR image format.
Mirror vertical	The captured frame is displayed upside down.
Enable timestamp	The timestamp of the captured frame is displayed at the bottom of the frame area.
Stop on error	The capturing stops when an error is recognized.
Use dma transfer	Using DMA (Direct Memory Access), the frames can be stored directly in the main memory in order to achieve a faster data transfer and to relieve the processor. (Only supported for **G PCIe 6222** when using PCIe connection)

Table 7.2 Frame area flags

7.3 Compare Settings

The **Dragon Suite** provides the ability to compare a captured frame with a reference frame. This function compares the frames pixel by pixel to see if they match.

The **Compare Area** parameters define the resolution of the frame to be captured, starting at the specified offset. The allowable tolerance range for the RGB values can be set in section **Color Tolerance.** For each color - red, green and blue - a separate tolerance value between 0 and 255 can be set. This value specifies by how many pixels the corresponding color of the captured frame may deviate from the color value of the reference frame. For example, if the color value Red in the reference frame is 132 pixels and the tolerance value is 10 pixels, the pixel value of the captured frame may be between 122 and 142. Otherwise, this pixel is considered to be faulty.

Figure 7.7 Compare settings segment

To use the comparison function, a reference frame must be loaded via the Load Reference button. To compare the frames, use the Compare button. The loaded reference frame is compared with the last captured frame, which is also displayed in the Frame Area. If no frame has been captured before, the function returns an error.

The comparison result is displayed in the Message Box below. In case the frames do not match, the number of different pixels is displayed.

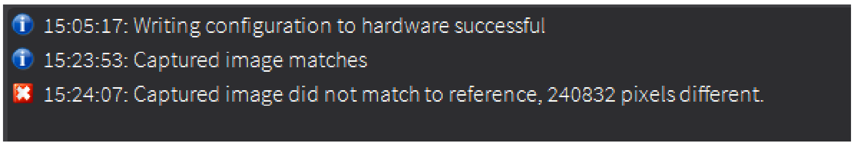


Figure 7.8 Capture and Compare info

7.4 Avi Settings

In this segment, the settings for recording videos can be defined.

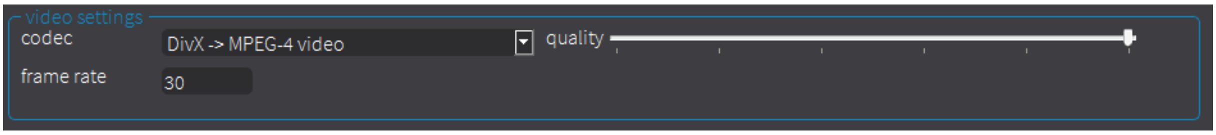


Figure 7.9 Video settings segment

The selectable **Codec** defines the compression type for the recording AVI video files.

Using no compression results in large AVI video files. Each frame requires the product of frame width * frame height * 3 bytes for 24-bit frames.

The **Frame rate** sets the maximum rate for capturing frames in continuous mode. This affects both the capture mode and the recording of video files.

The **Image Quality** slider sets the compression parameter for the selected video compression codec. Depending on the selected codec, the value has different impacts. The lower the number, the higher the compression and the lower the image quality. For more information about the relationship between value and compression of the video, see the documentation for the corresponding codec.

7.5 Raw Data Recording

Recording raw data is only possible in **Dragon Suite Advanced.** This segment is described in chapter Dragon Suite Advanced.

7.6 DMA Configuration

Using DMA (Direct Memory Access), the frames can be stored directly in the main memory in order to achieve a faster data transfer and to relieve the processor.

This feature is only supported for **G PCIe 6222** .

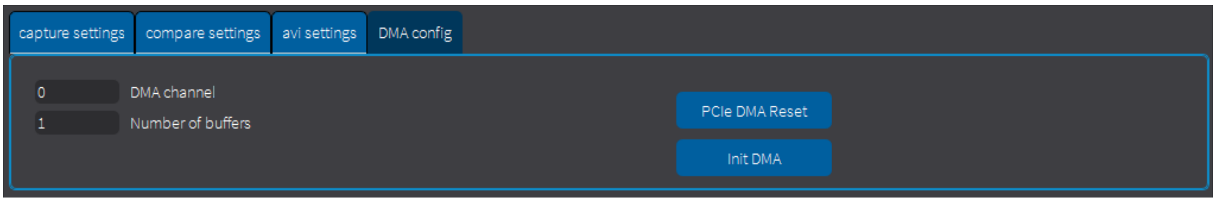


Figure 7.10 DMA Configuration segment

The **Video Dragon** currently has 4 **DMA channels** . Of these, channels 3 and 4 can be used for user-specific configuration. The desired DMA channel can be entered in the "DMA Channel" field, starting with 0. So for using channels 3 and 4, DMA channel 2 or 3 must be entered here. If a DMA channel can not be used, an error message appears. The **Number of Buffers** indicates how many frames are to be cached. The DMA configuration can be set by clicking the **Init DMA** button and reset with the **DMA Reset** button.

7.7 Frame Area

The frame area always displays the last captured frame.

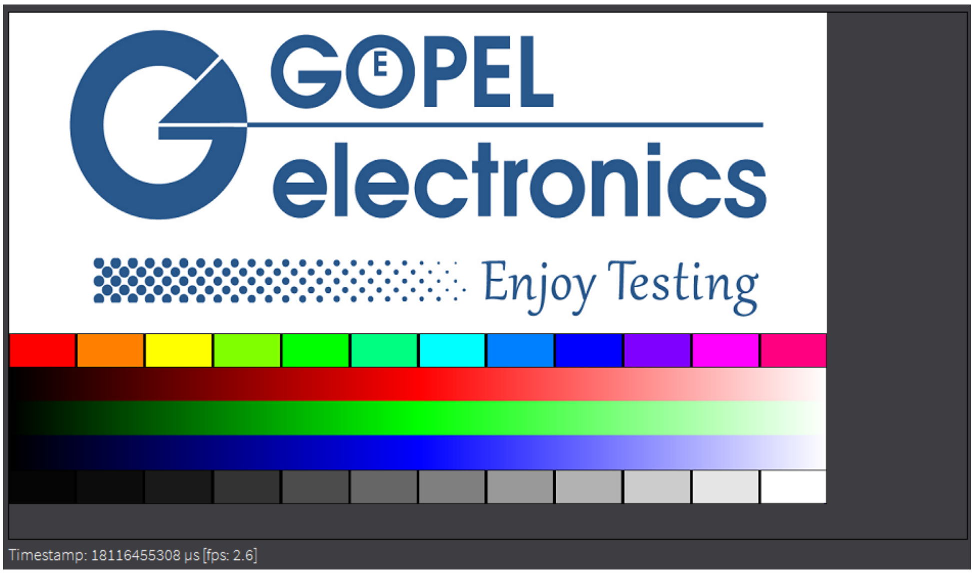


Figure 7.11 Frame area segment

Below is the timestamp, if desired. It also displays the frame rate at which the image is captured.

8 Sideband Communication

Parallel to the image transfer, the control data can be read in or out via sideband. Depending on the **Media Interface** module the **Video Dragon** suppports I^2C, SPI and UART. With appropriate commands, the registers of the chips of both the **Video Dragon** as well as the remote station can be read or written. For some hardware, it is imperative that e.g. a sideband display must receive a "wake-up" command to start communication or receive and process image data.

The sideband communication is only available with enabled sideband activation. The activation can be obtained through **GOEPEL electronics** sales department.

Manual manipulating of the configuration data list needs an extreme good knowledge of the meaning of each register value. This can only be obtained from the data sheets of the serializer and deserializer. Even a single wrong setting of only one register may render the complete configuration invalid and the device inoperative.

An incorrectly configured device can be reset to its default values by a restart.

The following figure shows the sideband communication between the serializer and the deserializer in a schematic way:

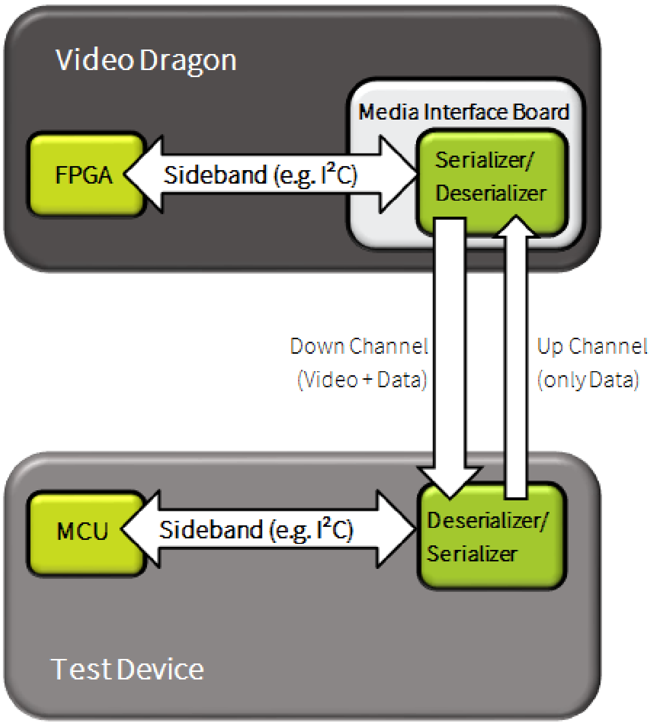


Figure 8.1 Sideband - Schematic Illustration

The data communication signal (I^2C, UART or SPI) from the FPGA is routed to the chip and serialized there. The transmission then takes place via the LVDS data stream in parallel with the video signal. In the receiving chip, the data is deserialized and returned to the outside. The communication is usually bidirectional. In general, the up channel has a lower data rate than the down channel.

8.1 Configuration of Sideband Communication via G-API

To configure and use the sideband communication of the **Video Dragon** the `G_Lvds_Common_Data_*` commands of the **G-API** are used. The configuration is divided into setting the data mode and adjusting the parameters.

Please compare the **G-API** Documentation for more information.

8.1.1 Setting the Data Mode

The data mode is set within the command `G_Lvds_Common_Data_Property_SetById` by the property `G_LVDS__COMMON__DATA__PROPERTY_ID__DATA_MODE`. For these, the following options are currently available:

Data mode	Description
`NONE`	No Data Mode is set
`I2C_MASTER_TO_DESERIALIZER`	Video Dragon is I²C Master on deserializer chip
`I2C_MASTER_TO_SERIALIZER`	Video Dragon is I²C Master on serializer chip
`I2C_SLAVE_TO_DESERIALIZER`	Video Dragon is I²C Slave on deserializer chip
`I2C_SLAVE_TO_SERIALIZER`	Video Dragon is I²C Slave on serializer chip
`SPI_MASTER_TO_DESERIALIZER`	Video Dragon is SPI Master on deserializer chip
`SPI_MASTER_TO_SERIALIZER`	Video Dragon is SPI Master on serializer chip
`SPI_PASSTHROUGH_DUAL`	Video Dragon combines data from INAP375R and INAP375T in "dual-mode"
`SPI_RECEIVER_DUAL`	Video Dragon is connected to INAP375R in "Dual Mode"
`SPI_TRANSMITTER_DUAL`	Video Dragon is connected to INAP375T in "Dual Mode"
`SPI_SLAVE_TO_DESERIALIZER`	Video Dragon is SPI Slave on deserializer chip
`SPI_SLAVE_TO_SERIALIZER`	Video Dragon is SPI Slave on serializer chip
`UART_TO_DESERIALIZER`	Video Dragon is connected to the UART interface of the deserializer chip
`UART_TO_SERIALIZER`	Video Dragon is connected to the UART interface of the serializer chip

Table 8.1 Available sideband modes

The data mode currently set on the chip can be queried by the command `G_Lvds_Common_Data_Property_GetById`.

8.1.2 Setting the Parameters

Depending on the Data Mode set, certain parameters for the sideband communication can be set. This also happens in the command `G_Lvds_Common_Data_Property_SetById`. The prefix of the property name `G_LVDS__COMMON__DATA__PROPERTY_ID__*DATAMODE*_` indicates for which data mode a property is valid (`*DATAMODE*` can be `I2C_MASTER`, `I2C_SLAVE`, `SPI_MASTER`, `SPI_SLAVE` or `UART`).

The following table explains the available parameters, with the default values highlighted in bold:

Parameter	Description
`I2C_MASTER_BAUDRATE`	Baud rate of the I²C master: **100kHz** (0) or 400kHz (1)
`I2C_MASTER_CLOCK_STRETCHING`	Clock stretching is the ability of the I²C slave to delay the master-driven clock. For this, the master must read back the clock to respond accordingly. This parameter **enables** (1) or disables (0) the reaction to the delay of the master.
`I2C_MASTER_STOP_NACK`	The I²C master sends an ACK (0) or **no ACK** (1) from the slave after the last received byte.
`I2C_SLAVE_BAUDRATE`	Baud rate of the I²C slave: **100kHz** (0) or 400kHz (1).

Parameter	Description
`SPI_MASTER_CPHA`	Sets the clock phase of the SPI master data: The data is valid on the **first edge** (0) after the ChipSelect or on the second edge (1).
`SPI_MASTER_CPOL`	Sets the clock polarity of the SPI signal: In idle state, the clock is **Low** (0) or High(1).
`SPI_MASTER_CS_MODE`	Sets the behavior of the ChipSelect signal during a transfer over several bytes: ChipSelect remains **active** between the individual bytes (0) or changes briefly to inactive (1).
`SPI_MASTER_CLOCK_DIVIDER`	Determines the SPI clock frequency based on a basic clock of 25MHz according to the following equation: Clock = 25MHz/(Divider + 1) For example **Divider = 0** results in a clock of 25MHz; Divider = 1 results in a clock of 12,5MHz etc.
`SPI_SLAVE_CPHA`	Sets the clock phase of the SPI slave data: The data is valid on the **first edge** (0) after the ChipSelect or on the second edge (1).
`SPI_SLAVE_CS_IDLE`	Sets the minimum time that the ChipSelect must be inactive for the slave to detect the completion of a transfer as a multiple of 20ns. The specified value sets the time to a multiple of 20ns, e.g.: **0 = 127*20ns = 2,54µs** 1 = 1*20ns = 20ns 2 = 2*20ns = 40ns usw.
`UART_BAUDRATE`	The baud rate that is closest to the specified value is set (the real value can be read out). Default: **115200 baud**
`UART_PARITY`	Defines whether a parity bit is formed and transmitted. **0 = no parity bit** , 1 = even parity, 2 = odd parity

Table 8.2 Available sideband parameters

8.2 I²C Master Mode

In the I²C master mode, the **Video Dragon** is the I²C master for the (de-)serializer chip used on the **Media Interface** module. Depending on this, the mode can be set to `I2C_MASTER_TO_DESERIALIZER` or `I2C_MASTER_TO_SERIALIZER`.

8.2.1 Communication

Once the correct data mode and parameters have been set, communication can be made using the firmware command `G_Lvds_Common_Data_I2cTransfer`. The start command initiates a communication and determines the (successful) completion by polling the status. With this command, any communication on the bus can be initiated independently of any protocol.

I²C Transfer on the Bus

The transfer on the I²C bus is bytewise. First, the master sends a start signal followed by the bytes it wants to send to the slave. Then he drives the clock for as many bytes as he wants to read bytes from the slave, which in turn now drives the data line with the answer. Thereafter, a stop signal is sent from the master and communication is completed.

The bytes sent by the master are individually acknowledged by the slave (ACK) or not (NACK). Accordingly, the master acknowledges the bytes from the slave (ACK) or not (NACK). Additionally, the master can provide another byte (in addition to the first one) that it writes with a start signal. This "Repeated Start" signal does not require a previous stop signal.

I^2C Transfer by Command

The command to start the transfer has the following parameters:

Parameter	Description
`NumberOfTxBytes`	The number of bytes to be written by the master. The master drives the clock for exactly as many bytes and places the data on the data line.
`NumberOfRxBytes`	The number of bytes to be read by the master. The master drives the clock after writing for exactly as many bytes and reads the data from the data line.
`SendStartMask`	Bytemask indicating the position of an additional (Repeated) start signal (Example: 0x02 = 2nd byte is provided with start signal, 0x04 = 3rd byte, ...).
`TxData`	The data bytes to be written.

Table 8.3 Parameters of the I^2C transfer command

The transfer command has the following return values:

Return Value	Description
`AckMask`	Byte mask indicating the position of the acknowledged (ACK) bytes (Example: 0x01 = 1st byte acknowledged, 0x02 = 2nd byte, 0x03 = 1st and 2nd byte, ...).
`NumberOfRxBytes`	The number of bytes read by the master (or written by the slave).
`RxData`	The read data bytes.

Table 8.4 Return values of the I^2C transfer command

I^2C Protocol

When transmitting with I^2C, the same protocol is used in most cases. This is for writing or reading registers of a slave device. When writing a register, the following data is transferred in this order:

1. Device address (7bit address) and bit (0) for write access
2. Register address (1 byte or more, depending on the device)
3. Register value (1 byte or more, depending on the device)

All bytes must be confirmed by the slave. At the beginning and end of the transfer, there is a start or stop signal.

Reading a register is done in two steps. The device and register addresses are written to the slave first, then the device address is written again to initiate reading of the contents:

1. Device address (7bit address) and bit (0) for write access
2. Register address (1 byte or more, depending on the device)
3. Repeated Start signal is sent
4. Device address (7bit adress) and bit (1) for read access
5. Register value is sent by the slave (1 byte or more, depending on the device)

All master bytes must be confirmed by the slave. The master confirms all slave bytes (register value) except the last. At the beginning and end of the transfer, there is a start or stop signal.
A read access to a 1-byte register is shown as an example in the following graphic:

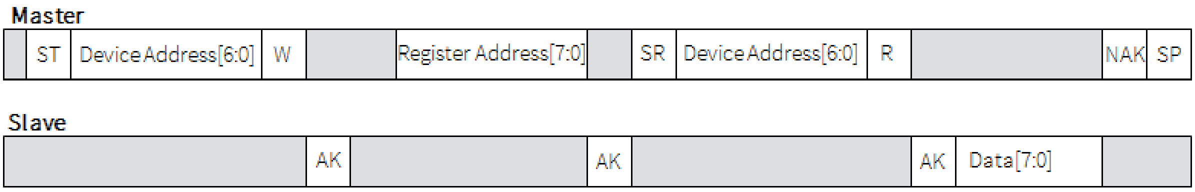


Figure 8.2 I²C protocol

8.3 I²C Slave Mode

In the I²C slave mode, the **Video Dragon** is the I²C slave for the (de-)serializer chip used on the **Media Interface** module. Depending on this, the mode can be set to `I2C_SLAVE_TO_DESERIALIZER` or `I2C_SLAVE_TO_SERIALIZER`.

8.3.1 Communication

In I²C slave mode, no independent communication can be triggered because only the master drives the clock of the I²C bus and also determines the type of access. In slave mode, the **Video Dragon** therefore simulates one or more slave devices in the manner of the protocol, as described in section I²C Protocol of the chapter I²C Master Mode. The slave then behaves accordingly and responds to requests initiated by the master.

The commands `G_Lvds_Common_Data_I2cSlaveDevice_Define` and `G_Lvds_Common_Data_I2cSlaveRegister_Define` are used to define the simulated devices and their associated registers. Here, a device must always be defined first before an associated register can be created. Equivalently, with the commands `G_Lvds_Common_Data_I2cSlaveDevice_Delete` and `G_Lvds_Common_Data_I2cSlaveRegister_Delete` devices or registers are removed from the simulation. Deleting a device automatically removes all associated registers.

I²C Slave Definition by command

The command for defining a slave device to be simulated has the following parameters:

Parameter	Description
`Flags`	`NONE` .. no flag set `REG_ADD_WIDTH_16BIT` .. the address width of the device registers is 2 bytes (otherwise only 1 byte). `DATA_WIDTH_16BIT` .. the data width of the device registers is 2 bytes (otherwise only 1 byte). `ACCESS_WO_ADDRESS` .. the master sends only the device address but no register address for access. `OVERWRITE_DEVICE` .. an already defined device with this address will be overwritten (otherwise an error message appears, that the device does not exist). `DEVICE_ADDRESS_7BIT` .. the specified device address is 7bit (otherwise 8).
`DeviceAddress`	The I²C address to which the device should respond.
`NAckByte`	Specifies from how many bytes the slave should respond with a NACK. 0 … every byte is confirmed, 1 … no confirmation from the 1st byte, …

Table 8.5 Parameter of the I²C slave device command

The command for defining a slave register to be simulated has the following parameters:

Parameter	Description
`Flags`	`NONE` .. no flag set `OVERWRITE_REGISTER` .. an already defined register with this address will be overwritten (otherwise an error message appears, that the device does not exist). `DEVICE_ADDRESS_7BIT` .. the specified device address is 7bit (otherwise 8).
`DeviceAddress`	The I^2C address to which the device should respond.
`RegisterAddress`	The address of the register that is defined.
`RegisterValue`	The value of the register that is defined.

Table 8.6 Parameters of the I^2C slave register command

8.4 SPI Mode

An SPI bus system typically consists of a master and many slaves. The clock line is the same for all participants and is driven by the master. The data from the master goes via the MOSI line (MasterOutSlaveIn) to the slaves. The slaves provide their data via MISO (MasterInSlaveOut). The selection of the slave with which the master communicates is made via the corresponding SlaveSelect (SS). Further communication parameters are the clock polarity and the clock phase. The clock polarity indicates whether the master clock is high or low active. The clock phase defines on which edge of the clock the data is taken over.

8.5 SPI Master Mode

In the SPI master mode, the **Video Dragon** is the SPI master for the (de-)serializer chip used on the **Media Interface** module. Depending on this, the mode can be set to `SPI_MASTER_TO_DESERIALIZER` or `SPI_MASTER_TO_SERIALIZER`.

8.5.1 Communication

Once the correct data mode and parameters have been set, communication can be made using the firmware command `G_Lvds_Common_Data_SpiTransfer`.

SPI Transfer on the Bus

The transfer on the SPI bus is bytewise. There are no special control signals. In addition to the clock driven by the master, there is a dedicated data line for the data from the master to the slave and one for the data from the slave to the master. A bidirectional communication can thus take place simultaneously. For addressing a slave device, a special line (ChipSelect) is used. This is pulled by the master to a special level and leads individually to only one slave. The data lines are called **MOSI** – Master Out Slave In and **MISO** – Master In Slave Out.
Bidirectional communication is exemplified in the following figure:

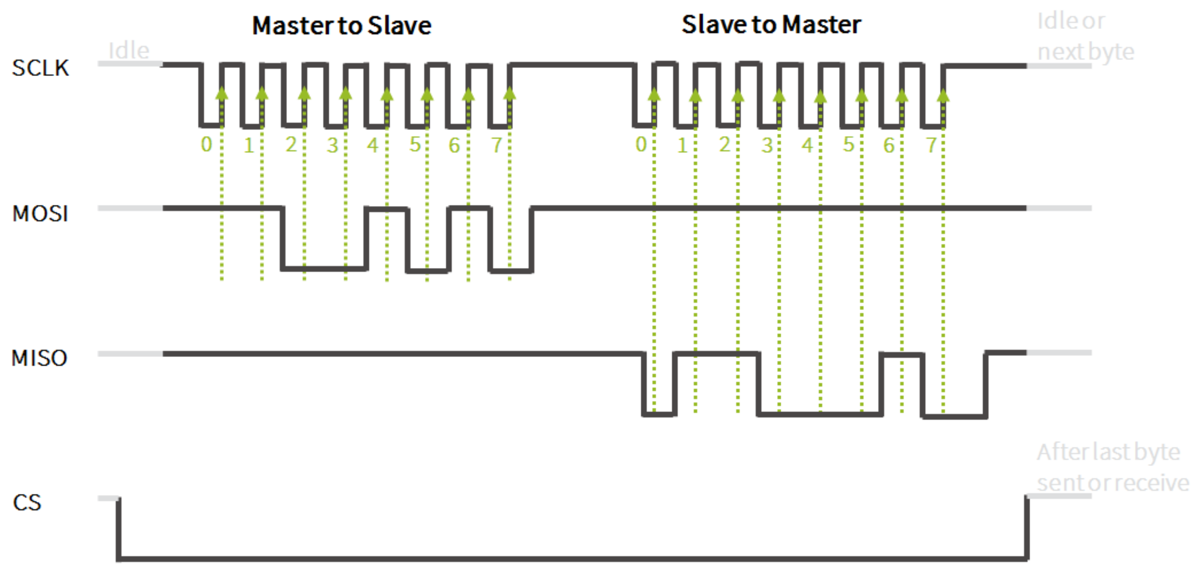


Figure 8.3 Bidirectional communication of the SPI bus transfer

There is no special protocol for communication via SPI. The validity of the data (at the first or second edge of the clock) and the idle state of the clock (high or low level) are also not normalized and therefore parameterizable under the parameters **CPHA** and **CPOL** (see chapter Setting the Parameters). For example, in the figure above, CPHA is set to the second edge and CPOL is set to high.
The other parameters define the data rate and the behavior of the ChipSelect signal between individual bytes.

SPI Transfer by Command

The command to start the transfer has the following parameters:

Parameter	Description
`NumberOfTxBytes`	The number of bytes to be written by the master. The master drives the clock and puts the data on the data line.
`NumberOfRxBytes`	The number of bytes to be read by the master. The master drives the clock and reads the data from the data line.
`TxData`	The data bytes to be written.

Table 8.7 Parameters of the SPI transfer command

Reading and writing are always done simultaneously on the bus. Therefore, the total number of bytes for which the master drives the clock is the maximum of the values of the bytes to be written and read. The count of the bytes starts from the beginning of the transfer. This must be taken into account in the number of bytes to be read. For example, if you want to read a response byte after 3 bytes written by the master, you must set the number of bytes to be written to 3 and those of the bytes to be read to 4 (3 + 1). At first 3 bytes are read during writing. These should each have the value 0 due to the prevailing resting level on the MISO line. Subsequently, the payload byte is read with the answer in 4th place.

The transfer command has the following return values:

Return Value	Description
`NumberOfRxBytes`	The number of bytes read by the master (written by the slave).
`RxData`	The value of the read data bytes (valid from 0 to `NumberOf-ReadBytes` – 1).

Table 8.8 Return values of the SPI Transfer command

8.6 SPI Slave Mode

In the SPI slave mode, the **Video Dragon** is the SPI slave for the (de-)serializer chip used on the **Media Interface** module. Depending on this, the mode can be set to `SPI_SLAVE_TO_DESERIALIZER` or `SPI_SLAVE_TO_SERIALIZER`.

8.6.1 Communication

In SPI slave mode, no independent communication can be triggered because only the master drives the clock of the SPI bus and also determines the type of access. Nevertheless, data can be read and written. The used command `G_Lvds_Common_Data_SpiTransfer` is the same as in the SPI Master Mode. The **Video Dragon** stores data received via the MOSI line in an internal FIFO so that bytes that have already been read can be retrieved. The start command thus does not initiate a transfer, but provides information about whether a successful transfer initiated by the master has taken place and reads out received data. The bytes to be written transferred by parameter are sent to the master from the beginning of the subsequent transfer. As soon as the master starts a new communication and restarts the clock, the data is put on the bus. Again, it should be noted that for a reasonable answer, it may be necessary to prepend some bytes with zero values to the actual user data (see note in chapter SPI Transfer by Command.

8.7 SPI Dual Mode

SPI Dual Mode is only used with INAP375 boards that communicate via the Apix standard. The special feature of these chips is that a direct bi-directional communication is possible, in which both chips involved can trigger a transfer. This is possible because two parallel SPI interfaces are used, and only the MOSI data line is used at a time. An answer to sent data thus does not take place in the same transfer and is limited by the cycle length and time of the master. But by the independent spontaneous sending of data on the self-driven parallel bus.

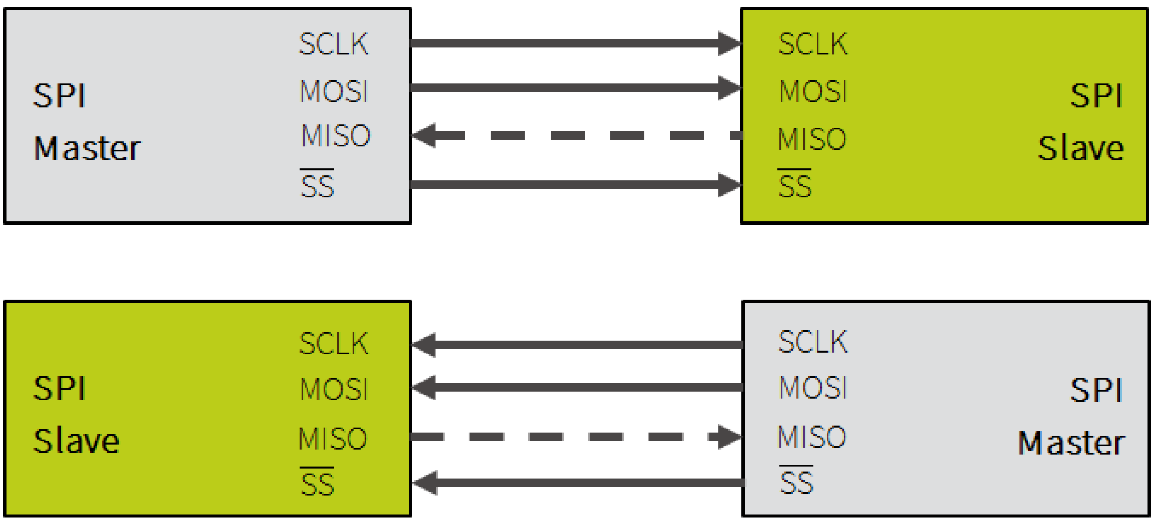


Figure 8.4 Direct bidirectional communication in SPI Dual Mode

The transmitter mode is used for the INAP375T and the receiver mode for the INAP375R.

8.7.1 Communication

SPI Transfer by Command

The commands used are the same as in SPI Master Mode. They differ only in their execution on the bus. Thus, for read bytes, the clock is not driven by the master, but (as in slave mode) the bytes of a previous transfer are fetched from the internal FIFO. If there were no received bytes, the return value of the number of bytes read equals 0. The writing of bytes is thus triggered by the command and takes place accordingly in time thereafter. The reading refers to a transfer that took place before the command was executed.

The sideband always transmits a fixed number of 8 bytes. If fewer bytes are transferred to the command for transmission, they are padded with zero bytes for the transfer to the required number. When reading it should be noted that 8 bytes should be read, even if the number of user data is lower. Any following zeros can be ignored when evaluating the data.

8.8 UART Mode

For UART mode the data mode of the **Video Dragon** has to be set to `UART_TO_DESERIALIZER` or `UART_TO_SERIALIZER`.

8.8.1 Communication

There are no master or slave roles in UART communication. Both parties can send data via their TX line, which serves as the RX line for reading. The writing of data therefore triggers a transfer on the TX line. The reading refers to already transmitted data, which are stored in a FIFO when received from the **Video Dragon** . Once the correct data mode and parameters have been set, communication can be made using the firmware command `G_Lvds_Common_Data_UartTransfer`.

UART Transfer by Command

The command to start the transfer has the following parameters:

Parameter	Description
`Flags`	`NONE` .. no flag set. `READ_ALL` .. regardless of `NumberOfReadBytes`, all data is read. The FIFO is completely emptied. `WITH_TIMESTAMP` .. the receive data is read out with a 64-bit time stamp (time of receipt).
`NumberOfTxBytes`	The number of bytes to write.
`NumberOfRxBytes`	The number of bytes to read.
`TxData`	The data bytes to be written.

Table 8.9 Parameters of the UART transfer command

The transfer command has the following return values:

Return Value	Description
`NumberOfRxBytes`	The number of bytes to read.
`RxData`	The value of the read data bytes (valid from 0 to `NumberOfReadBytes` – 1), or first all timestamps followed by all data bytes if timestamps are enabled.

Table 8.10 Return values of the UART transfer command

8.9 Sideband Communication Tool

The Sideband Communication tool is for reading or writing register data to a device via UART, I²C or SPI.

The sideband communication is only available with enabled sideband activation. The activation can be obtained through **GOEPEL electronics** sales department.

Manual manipulating of the configuration data list needs an extreme good knowledge of the meaning of each register value. This can only be obtained from the data sheets of the serializer and deserializer. Even a single wrong setting of only one register may render the complete configuration invalid and the device inoperative.

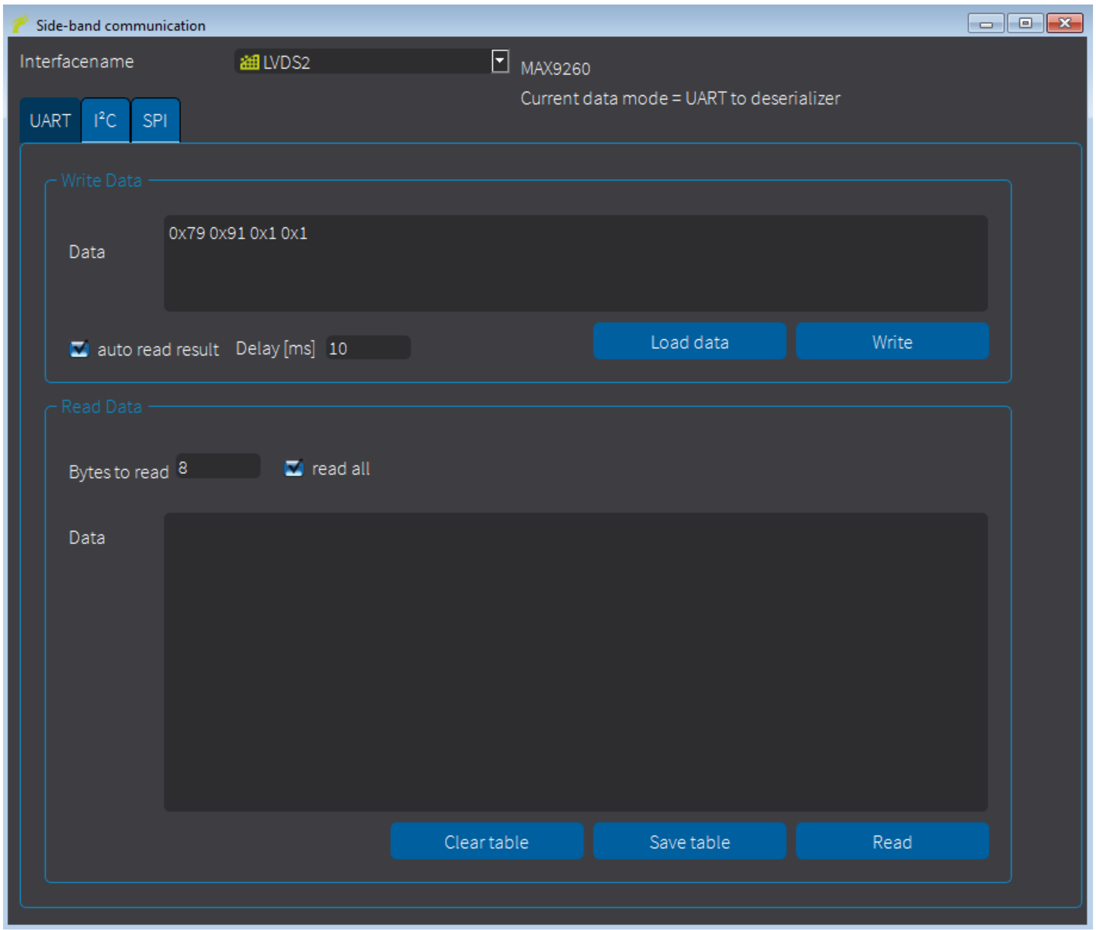


Figure 8.5 Sideband Communication

First select a data mode and set it on the sideband settings tab of the device to work correctly with this tool. The current data mode is displayed below the extension board information.

All available device interfaces are listed in the drop down menu.

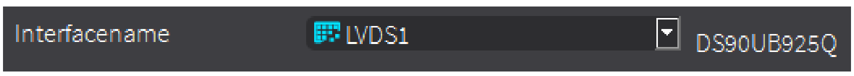


Figure 8.6 Choose interface name

The currently selected interface is displayed in the text field of the drop-down menu. Selecting the interface will automatically open the sideband tab of the defined data mode.

8.9.1 UART

For UART communication, there is no master or slave roles. Both parties can send data over their Tx line, which is used as Rx line on the other. The writing of data triggers a transfer on the Tx line. The reading of data refers to already transmitted data that is stored in the FIFO when the **Video Dragon** receives it.

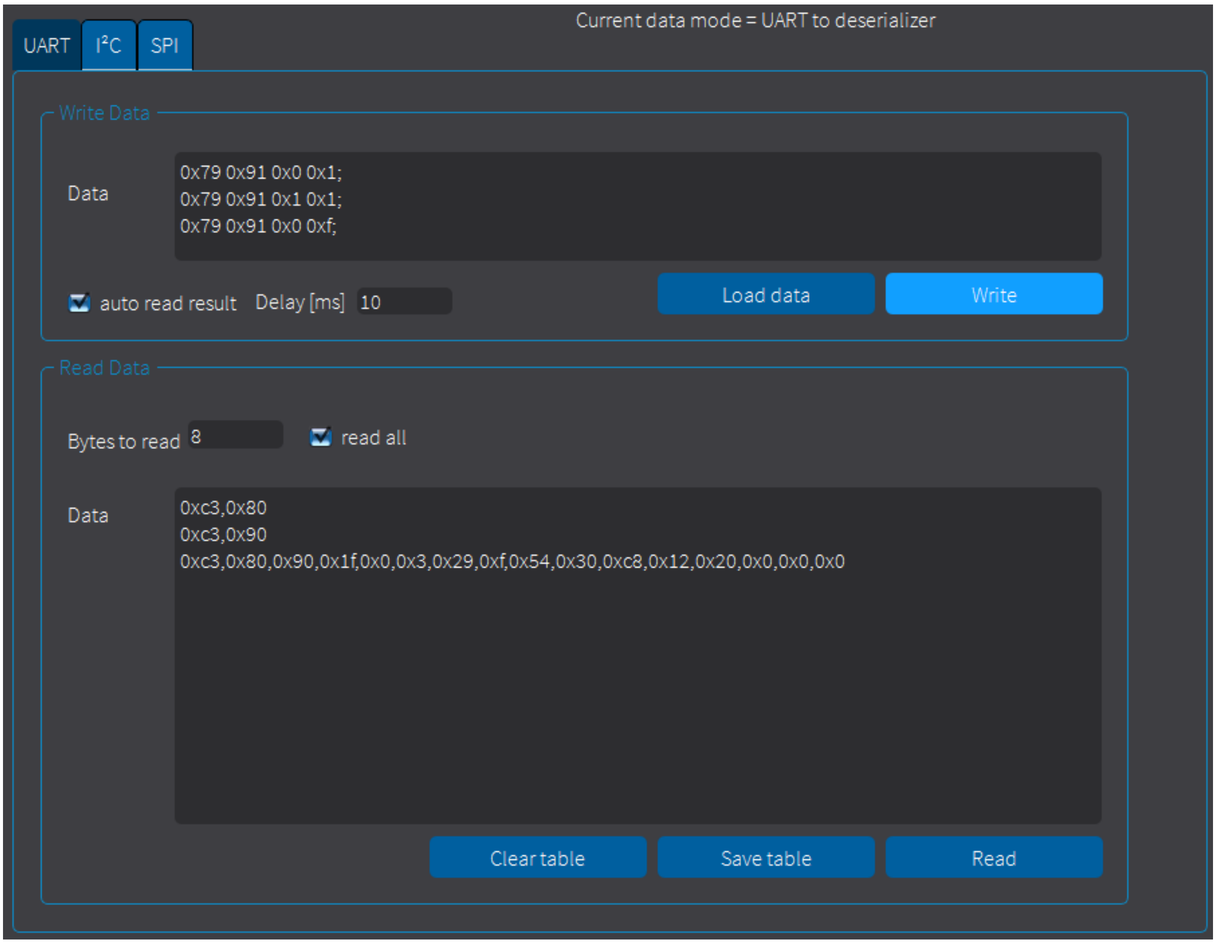


Figure 8.7 UART tab

To start the transfer, use the **Write Data** table. Enter the data in the table, following the UART Protocol specified in the serializer / deserializer data sheet. The package values can be entered in decimal or hexadecimal format. Separate the single package values with "," (decimal point) or " " (blank space). To write several package frames at once, use ";" (semicolon) to separate them. Select the **auto read result** checkbox to automatically read the received results after writing and display them in the **Read Data** table. Add a **Delay** in [ms] to define the time delay between two frames (useful only if several frames are written simultaneously). Press the **Write** button to write the data displayed in the table to the device.

The data can also be loaded from a text file via the **Load data** button. The data of each sequences must be defined in the order shown in the example below.

```
# Video Dragon Suite UART sequence import file
# syncByte, device address, register address, number of bytes to read/write, data to write, ...
0x79,0x91,0x0,0x1;
0x79,0x91,0x0,0x2;
0x79,0x91,0x0,0x3;
0x79,0x91,0x0,0x4;
0x79,0x91,0x0,0x5;
```

Figure 8.8 UART sequence file example

If the auto-read function is not checked, the result data can be read using the **Read** button. It is possible to **read all** data using the check box or to define a specific number of **Bytes to read**. Use the **Clear table** button to clear the table, or the **Save table** button to save the table entries in a text file.

8.9.2 I²C

Transmission on the I²C bus is performed byte by byte. First, the master sends a start signal followed by the bytes it wants to send to the slave. Then the master continues the clock for as many bytes as it would like to read from the slave. The slave drives the data line with the response. Thereafter, the master sends a stop signal and the communication is terminated.

The bytes sent by the master are individually acknowledged by the slave with a confirmation signal (ACK) or not (NACK). Similarly, the master acknowledges the bytes from the slave (ACK) or not (NACK).

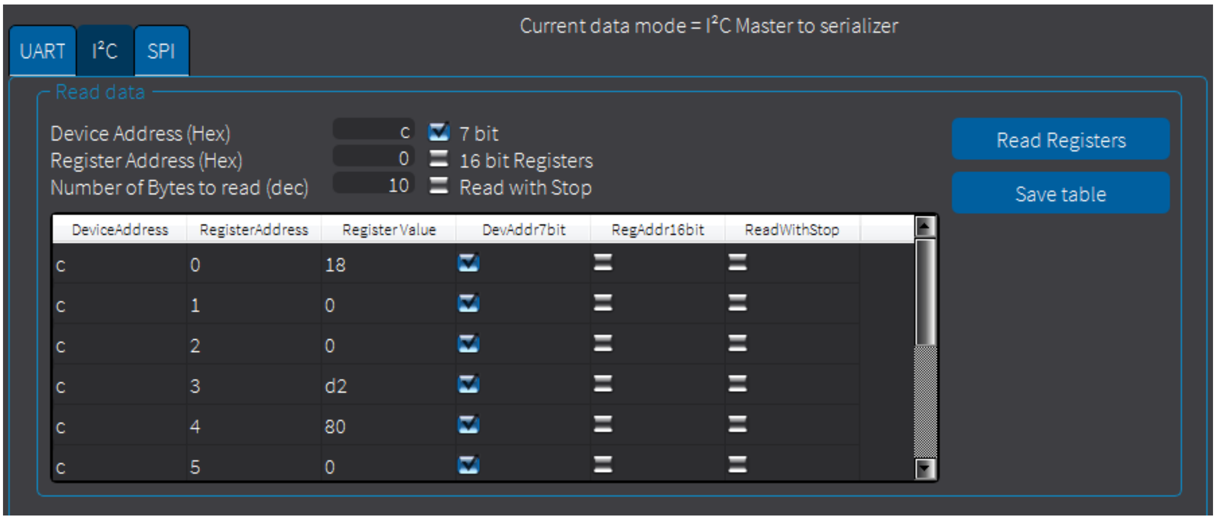


Figure 8.9 I²C Read data

To read register data via I²C, first define the device address (hex value) and the first register address (hex value) from which the registers are to be read. The device addressing can be in 7-bit or 8-bit format, chosen in the **7-bit** check box. If you do not check **7 bit**, the 8-bit format is set. The registers can be read in 8-bit or 16-bit format, chosen in the **16 bit Registers** check box. If you do not check **16-bit**, the 8-bit format is set. This settings can be found in the specifications of the device. In addition, specify the number of bytes you want to read starting at the defined register address (decimal value). The single registers are read and written simultaneously. Select the **Read with Stop** check box to first write all registers before reading them.

Click the **Read Registers** button to read the data and display them in the data list. Save this data with the button **Save table** in a file.

```
# DeviceAddress,RegisterAddress,RegisterValue,7-bit Device Address ? , 16-Bit RegisterAddress? , Read with Stop ?, Waittime between lines
0x71,0x4,0x0,1,0,0,100
0x71,0x4,0x1,1,0,0,10
0x71,0x4,0x3,1,0,0,10
0x71,0x2,0xffe,1,0,0,20
0x71,0x6,0xff,1,0,0,20
```

Figure 8.10 I²C sequence file example

There are two ways to write data to the device. The first option is to import a list of data from a text file using the **Import Data** button. Select a path to the file and the list will appear in the data table.

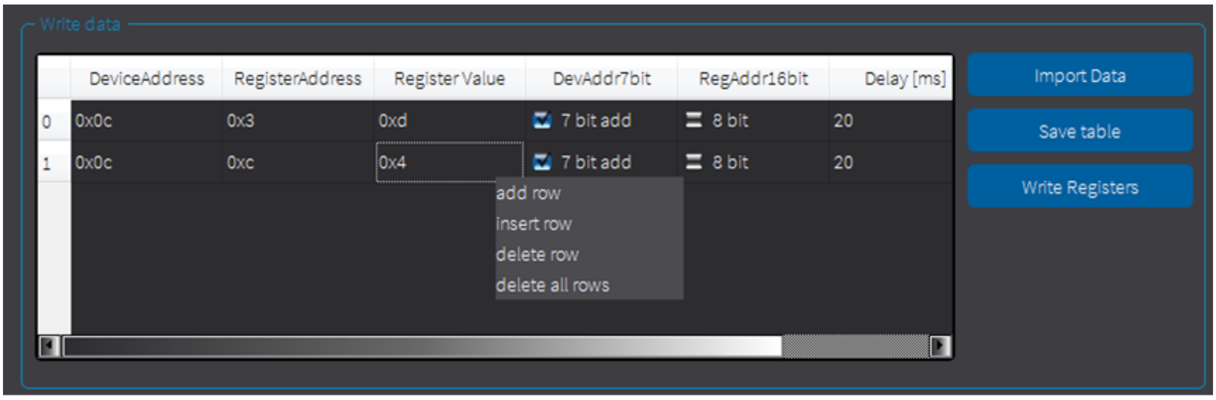


Figure 8.11 I^{2}C Write data

The other way is to add data one by one by hand. Right-click in the table and select **add row** to add a register row at the end of the table. To create a new row at a specific location in the table, select **insert row** . The data can be costumed by clicking in the list entry and changing the value. The last value in the row defines the write delay after each register(default is 20).

This data table can be saved to a file by using the **Save table** button.

Press the **Write Registers** button to write the table data to the selected device.

8.9.3 SPI

As with the I^{2}C, the **Video Dragon** also takes over the master role of the chip on the extension board in SPI master mode. The transmission also takes place byte by byte. In addition to the clock controlled by the master, there is a dedicated line for the data from the master to the slave (MOSI - Master Out Slave In) and one for the data from the slave to the master (MISO - Master In Slave Out). Therefore, bidirectional communication can take place simultaneously. For addressing a slave device, the Chip Select line is used.

There is no specific protocol for SPI communication. The data validity (on the first or second edge of the clock) and the idle state of the clock (high or low level) are also not standardized. Thus they can be parametrized under the parameters CPHA and CPOL.

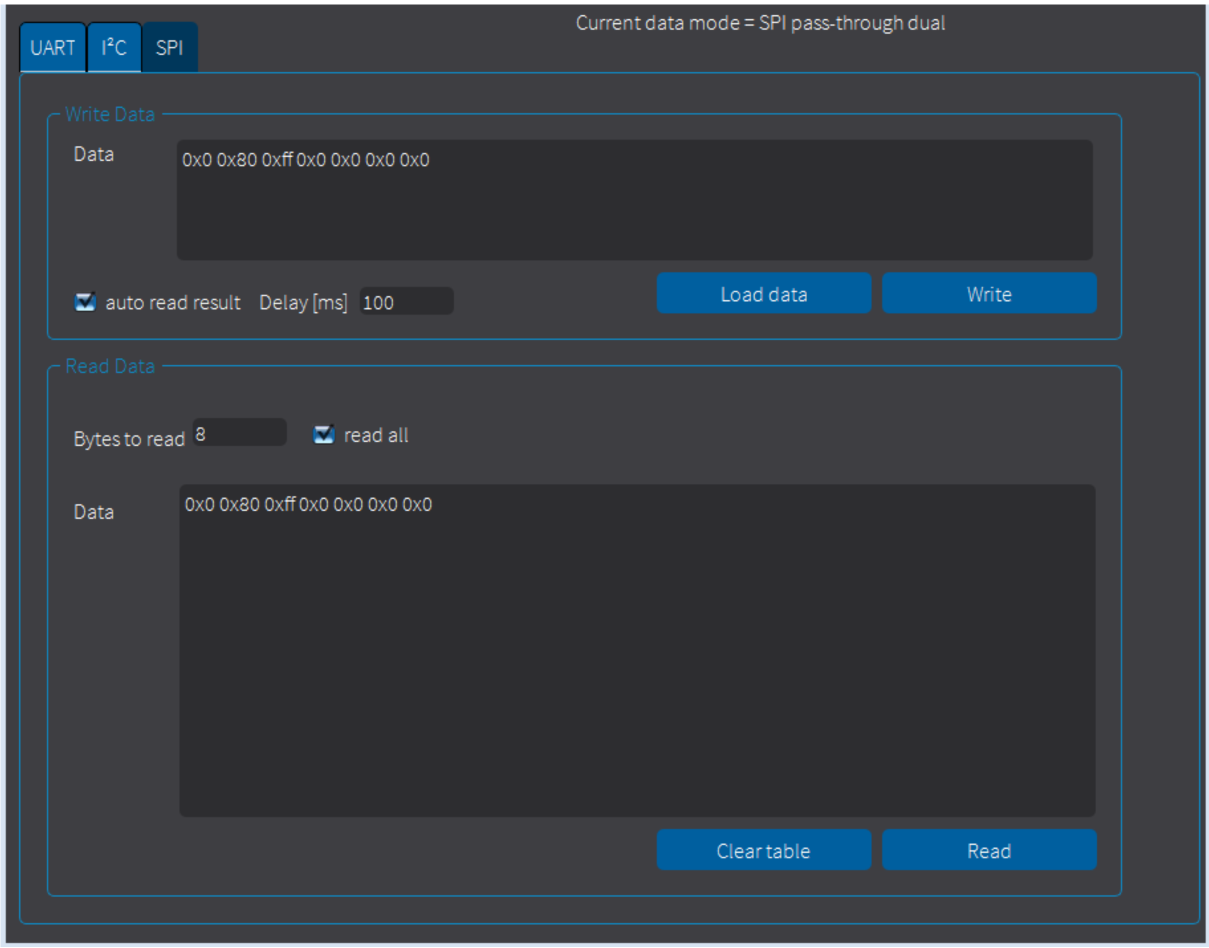


Figure 8.12 SPI tab

To start the transfer, use the **Write Data** table. Enter the data in the table, following the SPI Protocol specified in the serializer / deserializer datasheet. The package values can be entered in decimal or hexadecimal format. Separate the single package values with "," (decimal point) or " " (blank space). The data can also be loaded from a text file via the **Load data** button. To write several package frames at once, use ";" (semicolon) to separate them. Select the **auto read result** check box to automatically read the received results after writing and display them in the **Read Data** table. Add a **Delay** in [ms] to define the time delay between two frames (useful only if several frames are written simultaneously). Press the **Write** button to write the data displayed in the table to the device.

If the auto-read function is not checked, the result data can be read using the **Read** button. It is possible to **read all** data using the check box or to define a specific number of **Bytes to read** . Use the **Clear table** button to clear the table, or the **Save table** button to save the table entries in a text file.

8.9.4 Indigo

Currently, this feature is only supported by **basicCON 4121** .

The Indigo tab is used for sending automotive shell (AShell) messages via Apix. Thanks to AShell, the data exchange of the bidirectional APIX connection is error-free and secure.

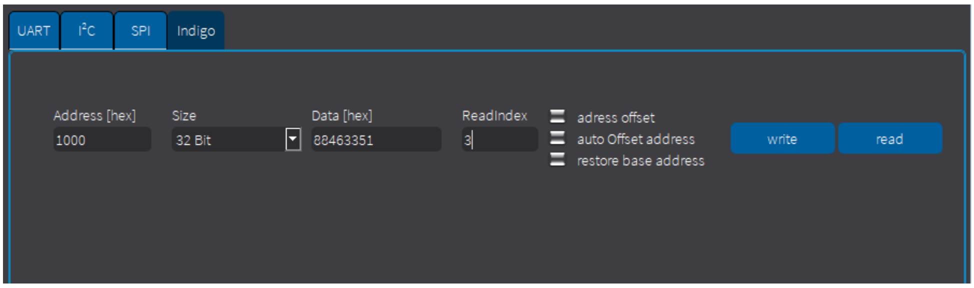


Figure 8.13 Indigo tab

Insert the address, the data size (8/ 16/ 32 bit) and the data to be written. Click the **Write** button to send a command or the **read** button to read an AShell message.

9 File Manager Tool

Amongst other things, the **FS** (File System) software interface allows you to create, copy, delete, execute and search files on the hardware. Thus, a uniform access to the OnBoard File System is possible. The **Dragon Suite** file manager helps organize the File Systems of **GOEPEL electronics** devices.

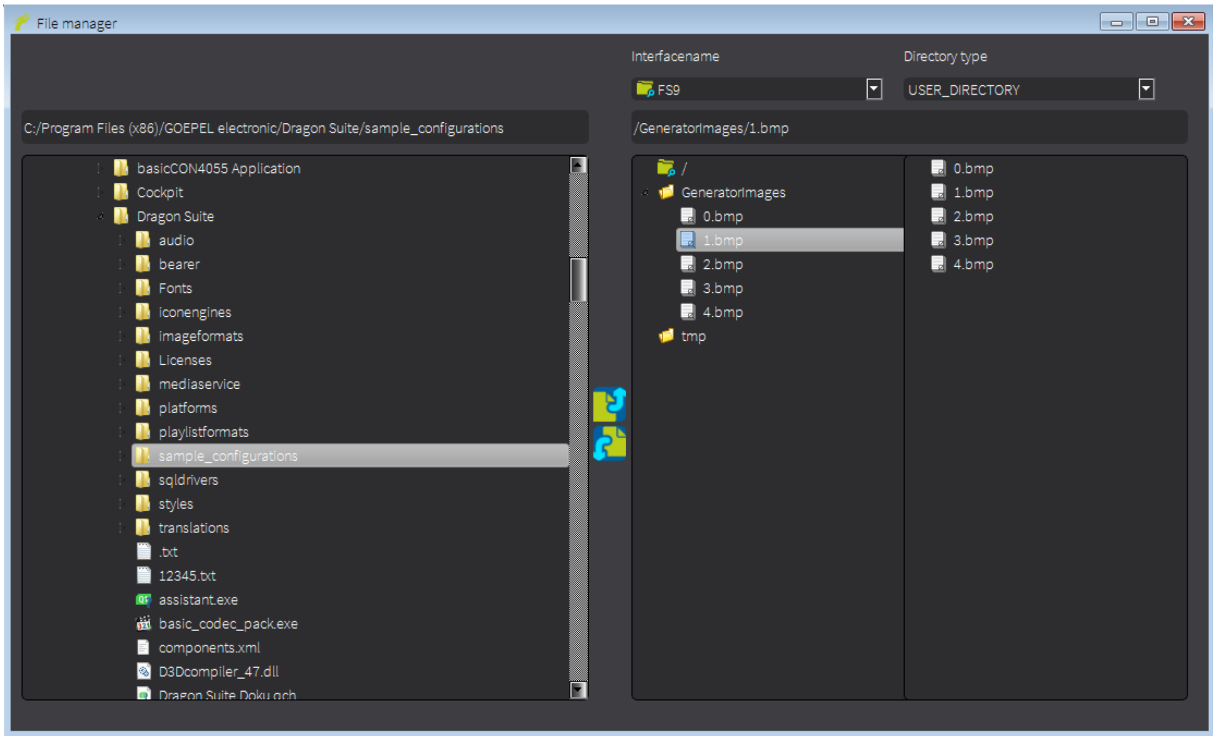


Figure 9.1 File Manager

On the left side of the file manager, the data files of the PC are listed. The right side shows the files of the **GOEPEL electronics** device. On the upper right side there is a drop down menu with all available **GOEPEL electronics** interfaces. The currently selected interface is displayed in the text field of the drop down menu.

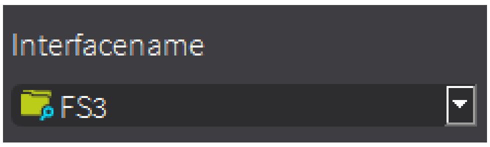


Figure 9.2 Interface name

To copy a file from the PC to the device, select the appropriate file on the left and the correct folder on the right.

Press  to add this file to the device. Use  to download this file from the device tp the PC.

Right-clicking on a device file (right side) opens a small menu with options for deleting or renaming the selected file or to creating a new folder.

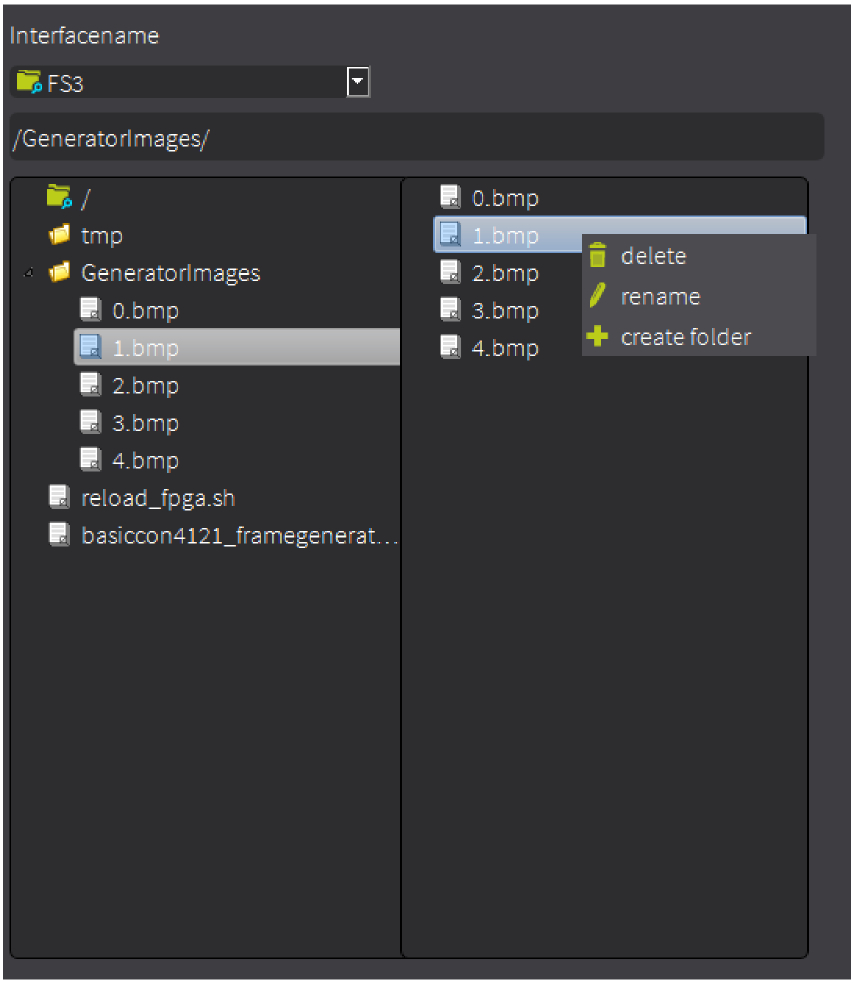


Figure 9.3 FS Functions

10 IO Tool

Open the IO dialog by using one of the alternatives illustrated in chapter Using the GUI.

All available IO interfaces are listed in the drop down menu.

Figure 10.1 Choose interface name

10.1 Digital IO

This dialog window shows the digital inputs and outputs of the connected **GOEPEL electronics** hardware. Depending on the connected hardware and its number of digital IOs the tab can display, for example, 6 digital outputs (**basicCON 4121**), 16 (**GOEPEL electronics** relay card) or even a different quantity.

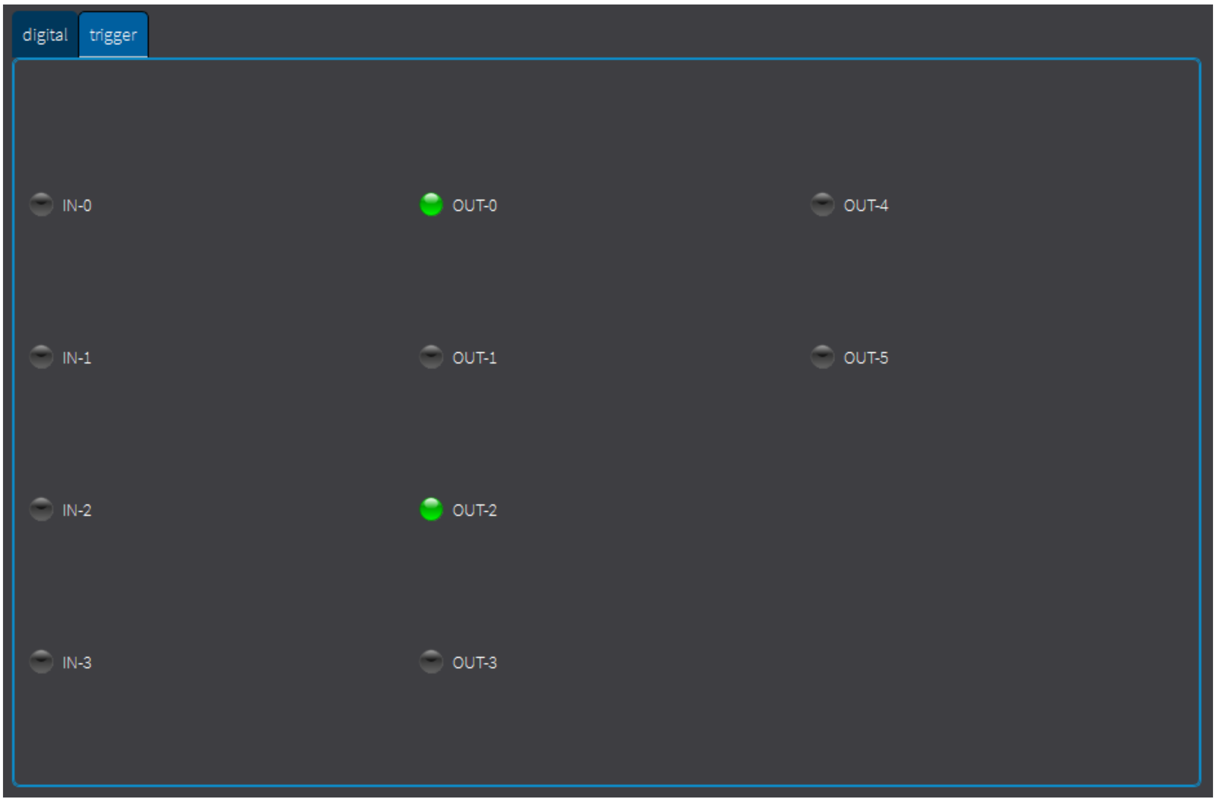


Figure 10.2 Digital IO

If a digital input or output is set to 1, the control element of this input or output lights up green in the dialog. Is it set to 0, it does not light up. The digital outputs can be set manually in the dialog by setting the control element to 1 (lit green) or 0 (not lit) by clicking on it. This does not apply to the digital inputs, here the control element is only used for status indication.

10.2 Trigger

With the help of the IO tool, the trigger functionality can be controlled by configuring the trigger matrix. It is used to assign a trigger source to a trigger output.

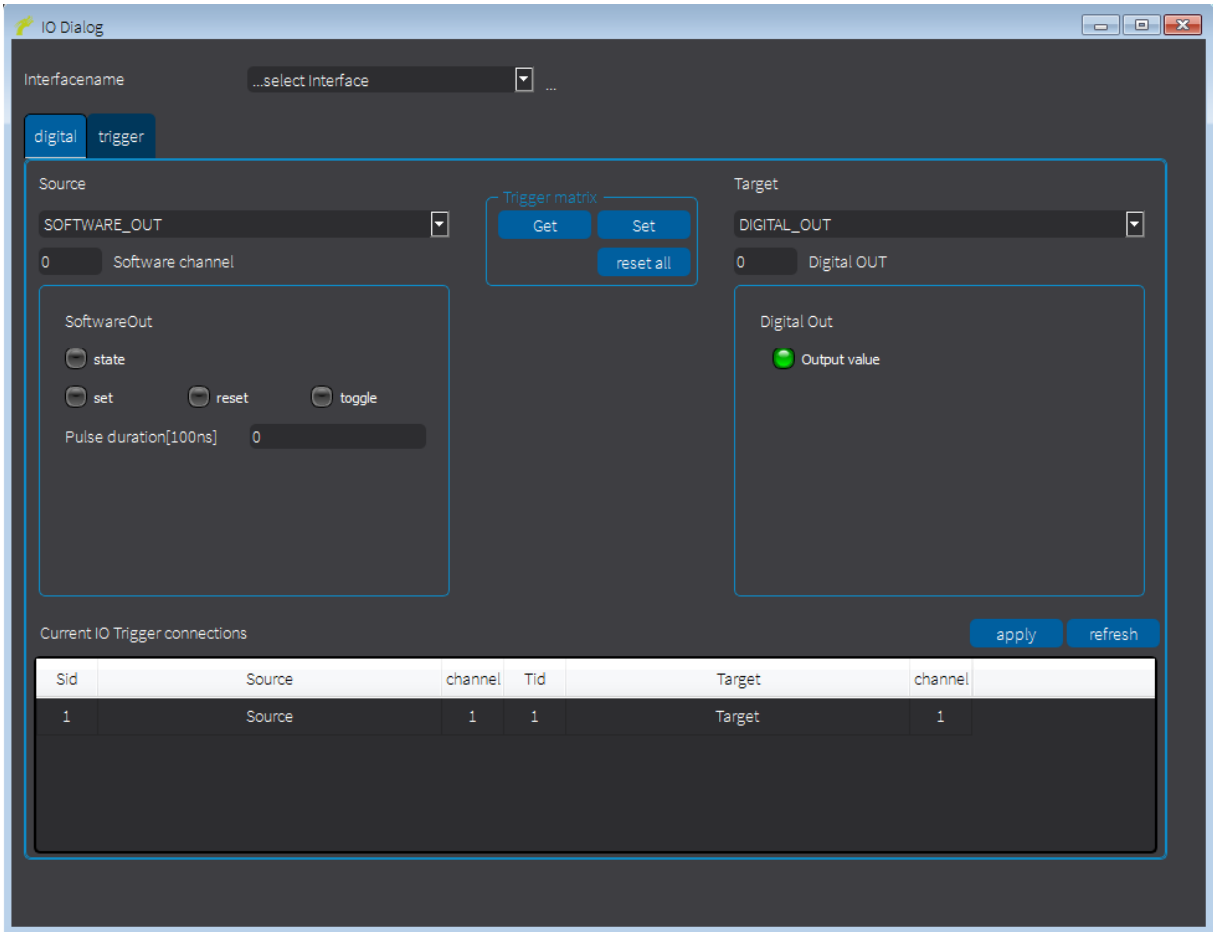


Figure 10.3 IO Tool

Seen across all **GOEPEL electronics** devices, there are a variety of possible trigger applications. This document only discusses those trigger signals that are supported by the video devices **basicCON 4121** and **Video Dragon 6222** .

The trigger source can be generated internally or taken from an external input. The available sources are listed in the drop down menu.

Figure 10.4 Choose IO source

The following source values are possible:

Source	Description
NO_SOURCE	No source is set.
DIGITAL_IN	The trigger signal is taken from a digital input.
SOFTWARE_OUT	If the trigger source is assigned to a software output, the trigger signal can be generated internally.
TRIGGER_BUS_IN	If the trigger source is assigned to the trigger bus in, trigger signals of the trigger bus line of a PXI, PCI or USB Rack are used for triggering.
LVDS_VIDEO_LOCK	The LVDS video lock signal is used as trigger source.
LVDS_VIDEO_ACTIVE	The LVDS video active signal is used as trigger source.
LVDS_GRABBER_READY	The LVDS Frame Grabber ready signal is used as trigger source.
LVDS_GRABBER_COMPLETE	The LVDS Frame Grabber complete signal is used as trigger source.
LVDS_GRABBER_ERROR	The LVDS Frame Grabber error signal is used as trigger source.
LVDS_0_SER_DES_GPIO	A GPIO signal of the serializer or deserializer board of the first LVDS interface is used as trigger source.
LVDS_1_SER_DES_GPIO	A GPIO signal of the serializer or deserializer board of the second LVDS interface is used as trigger source.
LVDS_2_SER_DES_GPIO	A GPIO signal of the serializer or deserializer board of the third LVDS interface is used as trigger source.
LVDS_3_SER_DES_GPIO	A GPIO signal of the serializer or deserializer board of the fourth LVDS interface is used as trigger source.
LVDS_0_TRIGGER_OUT	A trigger output signal of the first LVDS interface is used as trigger source.
LVDS_1_TRIGGER_OUT	A trigger output signal of the second LVDS interface is used as trigger source.
LVDS_2_TRIGGER_OUT	A trigger output signal of the third LVDS interface is used as trigger source.
LVDS_3_TRIGGER_OUT	A trigger output signal of the fourth LVDS interface is used as trigger source.
UART_TX	The UART output is used as trigger source.
INTERNAL_SOURCE	An internal source is used as trigger source.
SPI_M_MOSI	The SPI Master Master Out Slave In is used as trigger source.
SPI_M_SCLK	The SPI Master Serial Clock is used as trigger source.
SPI_M_SS0	The SPI Master Slave Select 0 is used as trigger source.
SPI_M_SS1	The SPI Master Slave Select 1 is used as trigger source.
SPI_M_SS2	The SPI Master Slave Select 2 is used as trigger source.
SPI_S_MISO	The SPI Slave Master In Slave Out is used as trigger source.
LVDS_MI_0_MFP	The LVDS media interface 0 multi function pin is used as a trigger source.

Table 10.1 Available Trigger Sources

The possible trigger sources depend on the used Media Interface.

The trigger signals can be used internally or routed to an output. The available outputs are listed in the drop down menu.

Figure 10.5 Choose IO output

The following output values are possible:

Output	Description
TRIGGER_BUS_OUT	If the trigger output is assigned to the trigger bus out, all trigger signals will be routed to the dedicated trigger bus line that feeds all devices of one PXI, PCI or USB bus.
DIGITAL_OUT	The trigger signals are routed to a digital output port. Further configuration options for this parameter are explained below.
SOFTWARE_IN	If the trigger output is assigned to a software input, the trigger signal can be used by the communication interfaces of the device.
LVDS_GRABBER__START	The LVDS Grabber device will start the capture operation when the trigger source is active.
LVDS_GRABBER__STOP	The LVDS Grabber device will stop the capture operation when the trigger source is active.
LVDS_0_SER_DES_GPIO	The trigger signals are routed to a serializer/ deserializer GPIO of the first LVDS interface.
LVDS_1_SER_DES_GPIO	The trigger signals are routed to a serializer/ deserializer GPIO of the second LVDS interface.
LVDS_2_SER_DES_GPIO	The trigger signals are routed to a serializer/ deserializer GPIO of the third LVDS interface.
LVDS_3_SER_DES_GPIO	The trigger signals are routed to a serializer/ deserializer GPIO of the fourth LVDS interface.
LVDS_0_TRIGGER_IN	The trigger signals are routed to the trigger input of the first LVDS interface.
LVDS_1_TRIGGER_IN	The trigger signals are routed to the trigger input of the second LVDS interface.
LVDS_2_TRIGGER_IN	The trigger signals are routed to the trigger input of the third LVDS interface.
LVDS_3_TRIGGER_IN	The trigger signals are routed to the trigger input of the fourth LVDS interface.
UART_RX	The trigger signals are routed to the UART input.
SPI_M_MISO	The trigger signals are routed to SPI Master In Slave Out.
SPI_S_MOSI	The trigger signals are routed to SPI Master Out Slave In.
SPI_S_SCLK	The trigger signals are routed to SPI Slave Serial Clock.
SPI_A_MISO	The trigger signals are routed to SPI Analyzer Master In Slave Out.
SPI_A_MOSI	The trigger signals are routed to SPI Analyzer Master Out Slave In.
SPI_A_SCLK	The trigger signals are routed to SPI Analyzer Serial Clock.
SPI_A_SS0	The trigger signals are routed to SPI Slave Select 0.
SPI_A_SS1	The trigger signals are routed to SPI Slave Select 1.
SPI_A_SS2	The trigger signals are routed to SPI Slave Select 2.
LVDS_MI_0_MFP	The trigger signals are routed to the LVDS media interface 0 multi function pin.

Table 10.2 Available Trigger Outputs

The possible trigger outputs depend on the hardware used.

Depending on the source or output value, another selection field automatically appears under the dropdown lists. In most cases, this is just the input field for a channel. Enter the necessary **source** or **target channel** here. The count starts at 0. The number of channels depends on your device.

Figure 10.6 Enter source channel

If, for example, **SER_DES_GPIO** is selected as source or output, a larger configuration field opens.

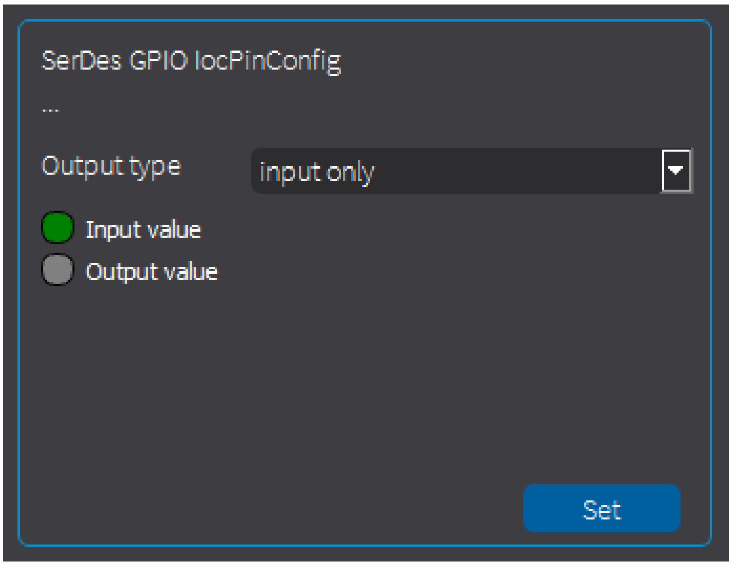


Figure 10.7 SerDes GPIO configuration

The GPIOs can be used to trigger specific actions. In the configuration field you can set the input/ output controller pin configuration of the serializer/ deserializer board. For the **Output Type** of the SerDes GPIO the following values are possible:

- input only
- push/ pull output mode
- open drain output mode

In addition, the configuration field contains two digital display segments: **Input Value** and **Output Value.** **Input Value** shows only the on/ off value and can not be changed manually. If the value is 1, the item turns green. The **Output Value** can only be changed if the output type is "push/ pull output mode" or "open drain output mode". The digital value can be changed manually to on (green) or off.

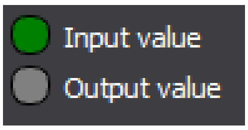


Figure 10.8 Input/ Output value

With the button "Set" the SerDes GPIO configuration is set. If the output value has been set to 1, the input value now also goes to 1 (and turns green), since output and input are internally connected.

If, for example, **DIGITAL_OUT** is selected as the output, a configuration field opens as well. Here the digital output value can be set to 0 or 1 (green).

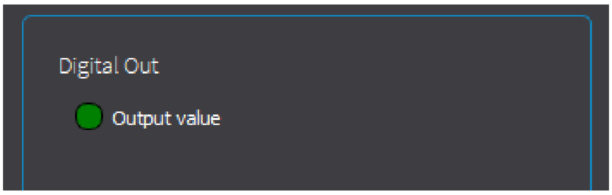


Figure 10.9 Digital Out Value

In the middle of the dialog window there are three buttons:

Button	Description
Get	Load the current IO setting values of the selected interface.
Set	Overwrite all current IO settings on the device with the settings displayed on the tabs of the window.
reset all	Reset all IO settings of the selected interface.

Table 10.3 IO dialog buttons

The trigger functionality of the devices offers a variety of configuration options. The software's IO tool offers only a subset of this options. What options are also possible, can be read in the manual, in order to realize this in own applications (if necessary).

The lower part of the dialog window contains a table with the currently set **IO Trigger Connections.** The channel entries can be changed manually by clicking on the desired field and editing the value. Use the **Apply** button to set the values. Click the **Refresh** button to load the actual settings.

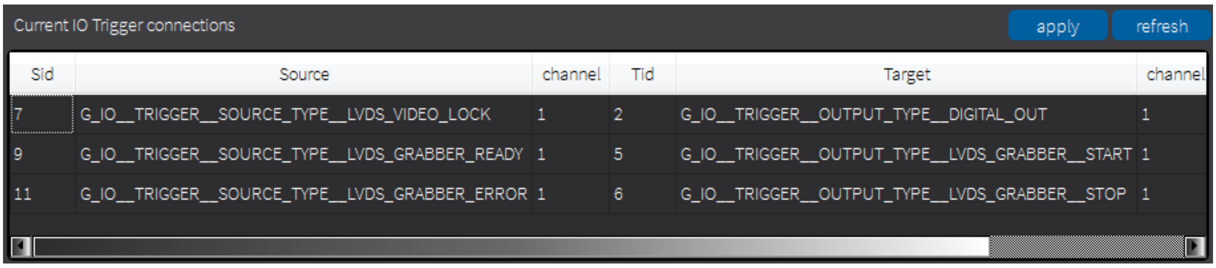


Figure 10.10 Current IO Trigger connections

By using the right click, one or all entries can be deleted from the list. **This action also deletes the routed trigger!** In addition, the table entry can be copied to the clipboard as a script command line by right clicking.

The IO Trigger connection table can also be found and edited in the Settings Window of the respective hardware.

10.2.1 Examples

Example 1: Routing Video Lock to Digital Output

The hardware used is an LVDS frame grabber. The trigger source is `LVDS_VIDEO_LOCK` on channel 0. As output the digital output 2 is defined. With "Set" this configuration is set. As soon as a lock signal occurs, the digital output 2 is activated.

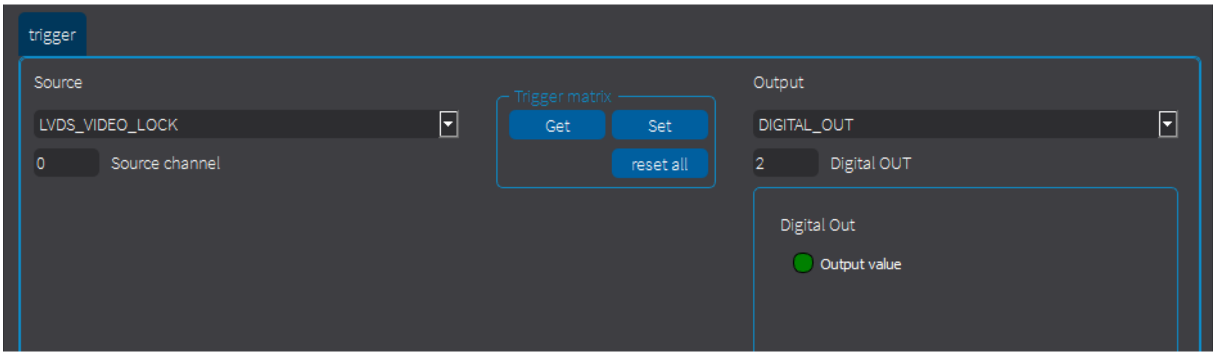


Figure 10.11 Routing Video Lock to Digital Output

Example 2: Routing SPI Analyzer to SPI Master for SPI Monitoring

To monitor SPI communication of the video device, the SPI Analyzer must be routed to the SPI Master signals. Therefore route the targets `SPI_A_MOSI`, `SPI_A_SCLK` and `SPI_A_SS0` to the sources `SPI_M_MOSI`, `SPI_M_SCLK` and `SPI_M_SS0`. Now the SPI communication can be monitored in the Monitor Dialog.

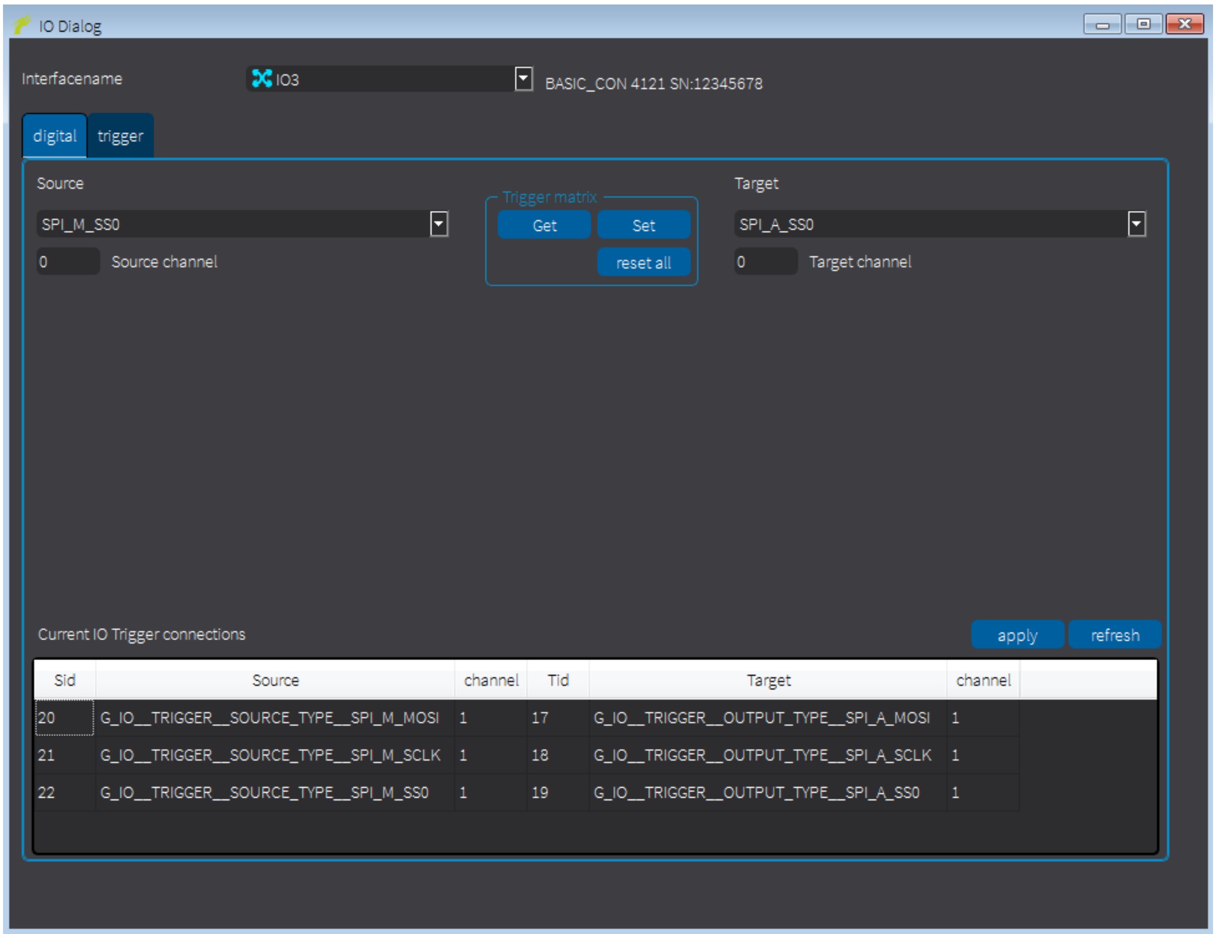


Figure 10.12 Routing SPI Analyzer to SPI Master

The SPI monitor is only available in Dragon Suite Advanced

11 CAN Tool

Open the CAN dialog by using one of the alternatives illustrated in chapter Using the GUI. In the upper tab the CAN node can be initialized. The lower tab is for defining a CAN - UART Gateway.

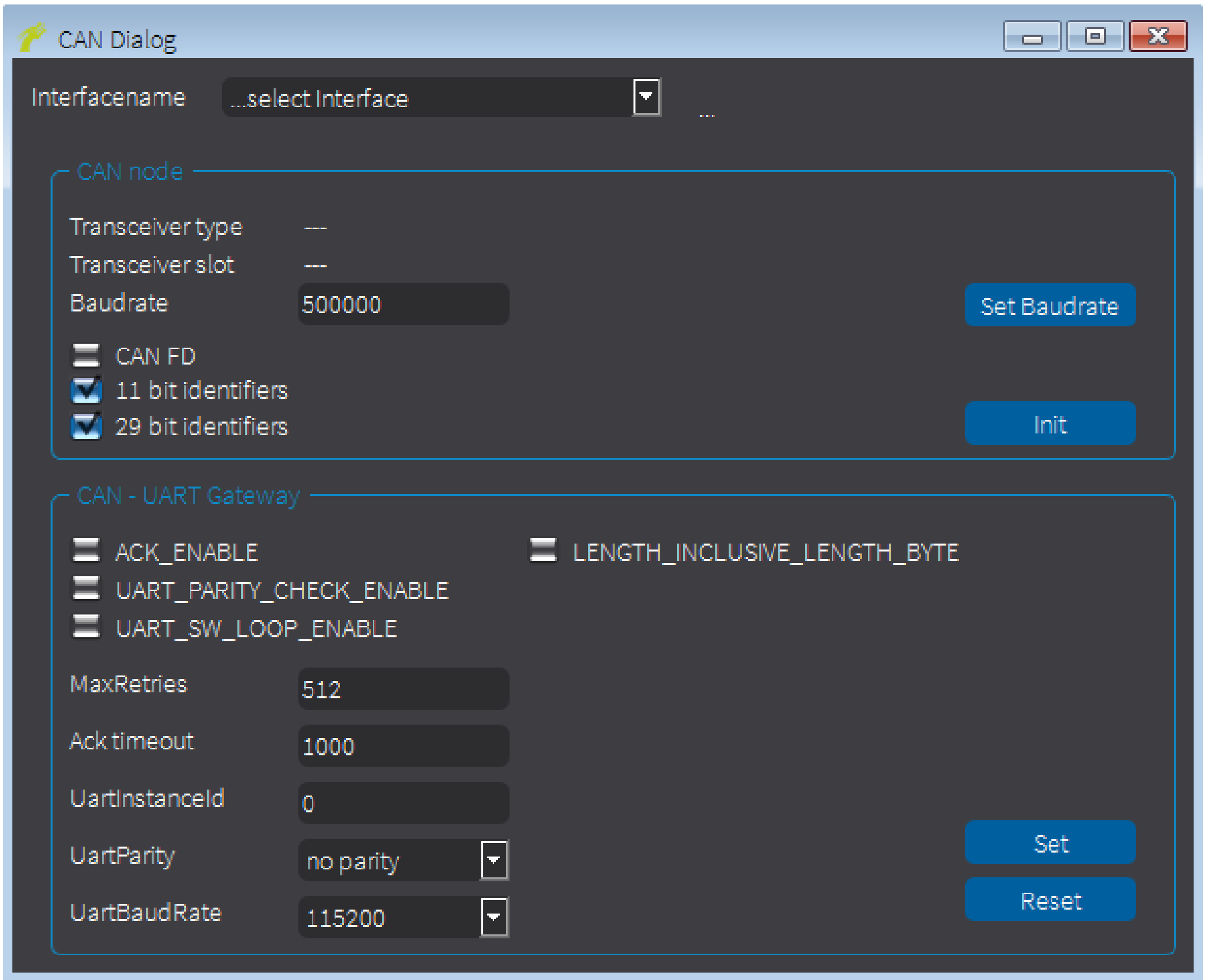


Figure 11.1 CAN Tool

All available CAN interfaces are listed in the drop down menu.

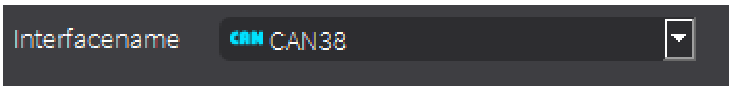


Figure 11.2 Choose interface name

11.1 CAN Node

After selecting the interface, the current values are displayed in the tab. **Transceiver type** and **Transceiver slot** are immutable values.

Enter a baud rate value (in baud, e.g. 500000 for 500kBaud) and click the "SetBaudrate"-Button to set the baud rate of the CAN node. CAN FD and Extended Identifiers can be enabled or disabled by setting the associated flags and clicking the "Init"-Button. When using CAN FD, the CAN FD baud rate can also be set.

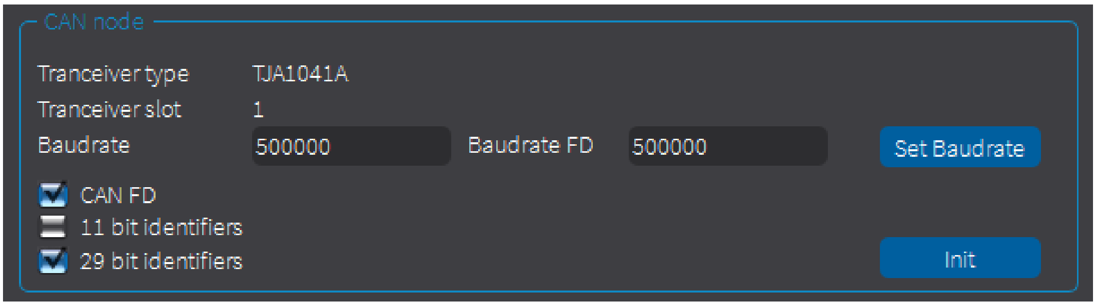


Figure 11.3 CAN Node

11.2 CAN - UART Gateway

The CAN - UART Gateway can be set to configure a gateway for gating CAN messages to UART and vice versa. Some test devices can not communicate directly via CAN, e.g. to receive a wake-up message. The Video Drag-on offers the possibility to receive CAN messages via the CAN interface and to route them to a UART signal. The UART signal can then be received and processed by the test device via the LVDS stream.

A configuration of the IO interface is necessary to use the CAN - UART Gateway (see Example).

For using the CAN - UART Gateway the Data Mode of the device must be set to "UART to serializer" or "UART to deserializer".

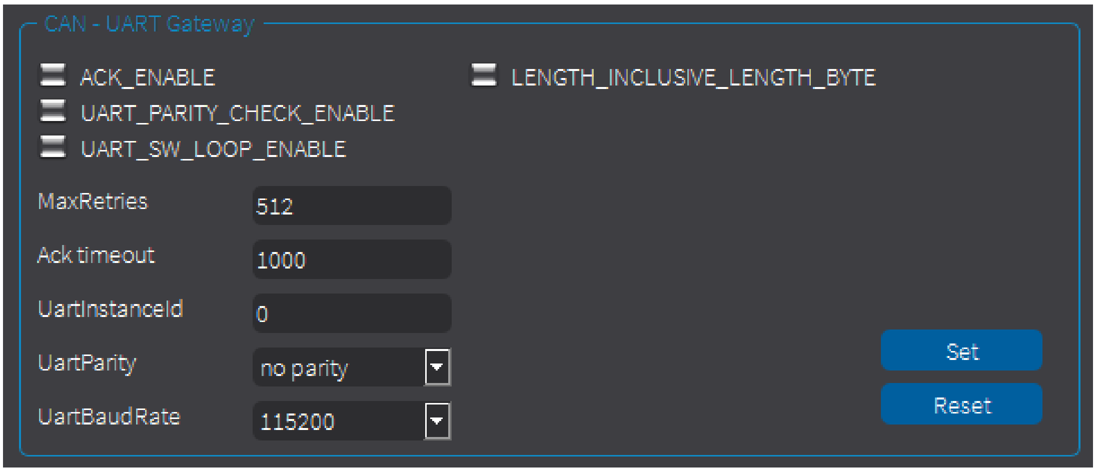


Figure 11.4 CAN - UART Gateway

The Gateway tab contains several flags for configuration:

Flag	Description
ACK_ENABLE	Enables the acknowledge handling. (Not supported yet)
UART_PARITY_CHECK_ENABLE	Enables parity checking in received UART frames. Frames with a false parity bit are disregarded.
UART_SW_LOOP_ENABLE	If the flag is set, the received CAN messages are ignored and the received UART messages are not gated to CAN. Received UART messages are looped back to UART.
LENGTH_INCLUSIVE_LENGTH_BYTE	If the flag is set, this length byte also contains the length byte itself in addition to the number of bytes after the length byte. If the flag is not set, it only contains the number of bytes after the length byte.

Table 11.1 CAN - UART Gateway flags

The parameter **MaxRetries** specifies the maximum number of TX repeats if no acknowledge is received. If the specified maximum is reached, the frame is discarded. **AckTimeout** defines the time (in µs) to wait for an acknowledge before repeating the message. In **UartInstanceId** a UART instance has to be defined, starting with 0. The **UART parity** can be set to even, odd or none. Additionally the **UART baud rate** needs to be specified. By clicking the "Set"-Button the parameters are written to the device. The "Reset"-Button resets the UART Gateway

11.2.1 Example

This chapter shows a short example of the CAN - UART Gateway configuration. For this purpose, in addition to the CAN configuration, the IO trigger interface must also be set. For this example, a **G PCIe 6222** is used. The CAN configuration is shown in the following picture:

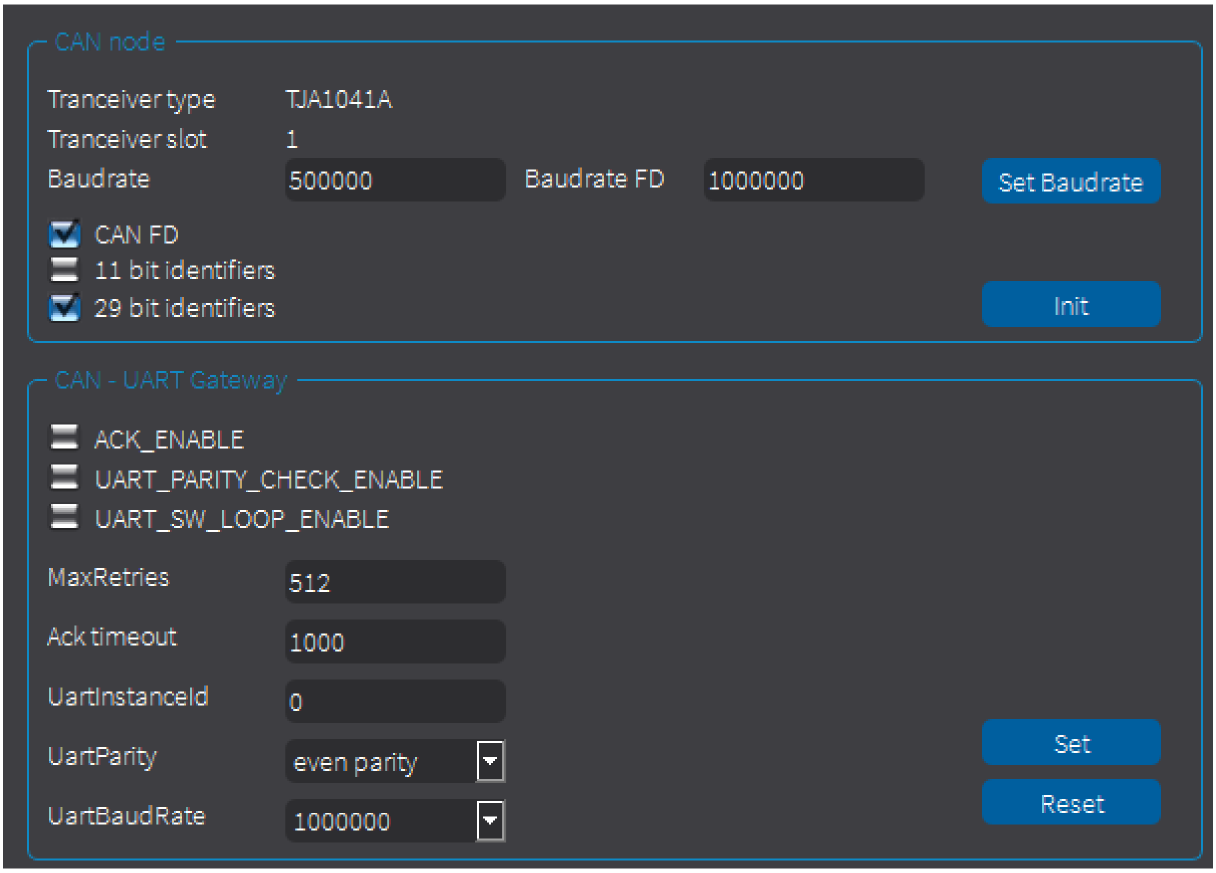


Figure 11.5 CAN configuration

In addition to the CAN configuration, it must of course be specified how the CAN signal is routed to the UART output. In our example the GPIO 0 is used for the CAN signal (as source). The output is the UART_Rx signal.

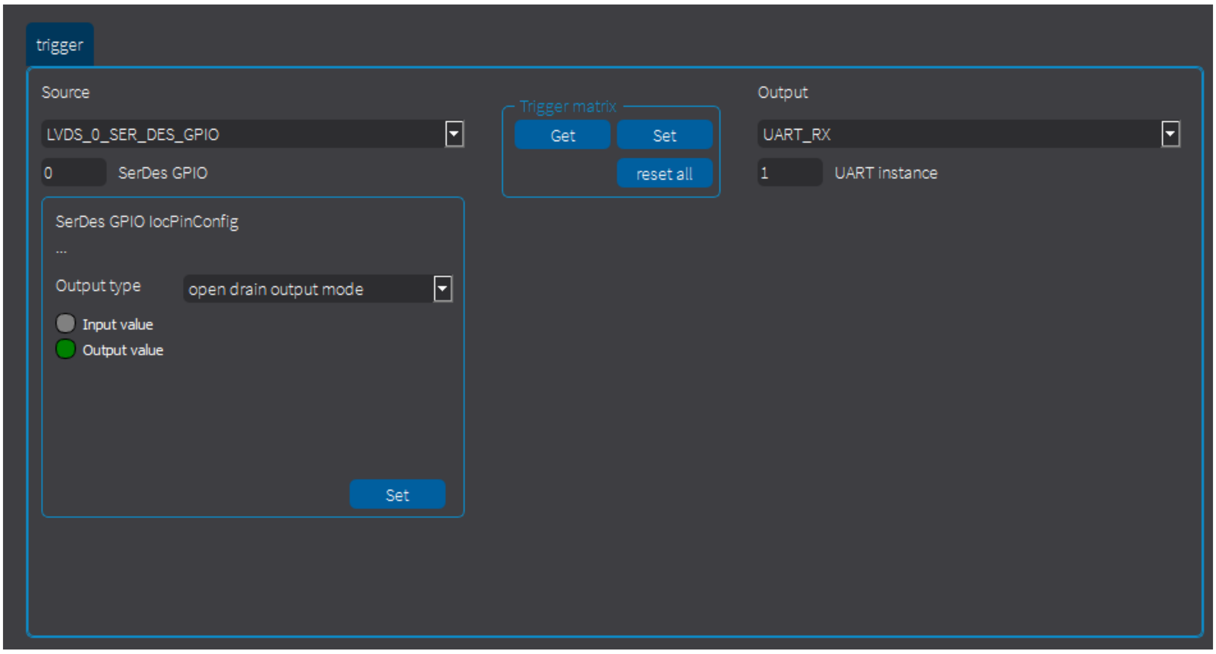


Figure 11.6 IO configuration Rx

Equivalently, the UART_Tx signal is set as source and routed to GPIO 3. Thus, a communication can take place in both directions.

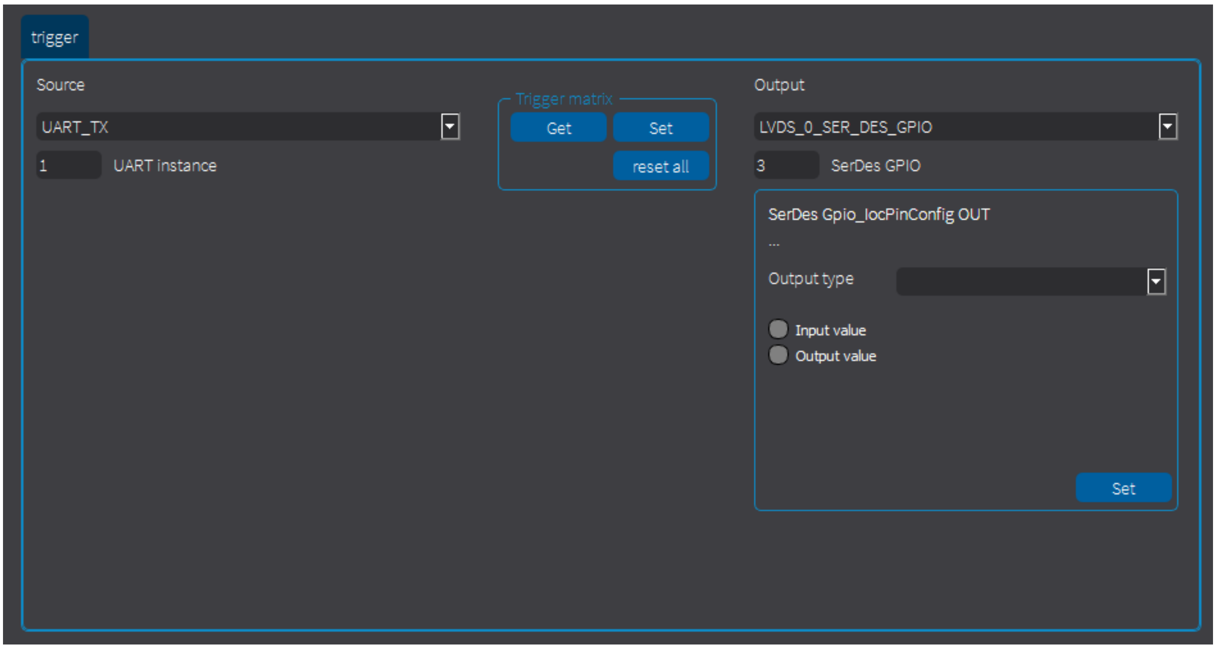


Figure 11.7 IO configuration Tx

12 Command Line Interface

The **Dragon Suite** provides the ability to work with the command line interface. Open the GUI-less **Dragon Suite** as shown in the example below.

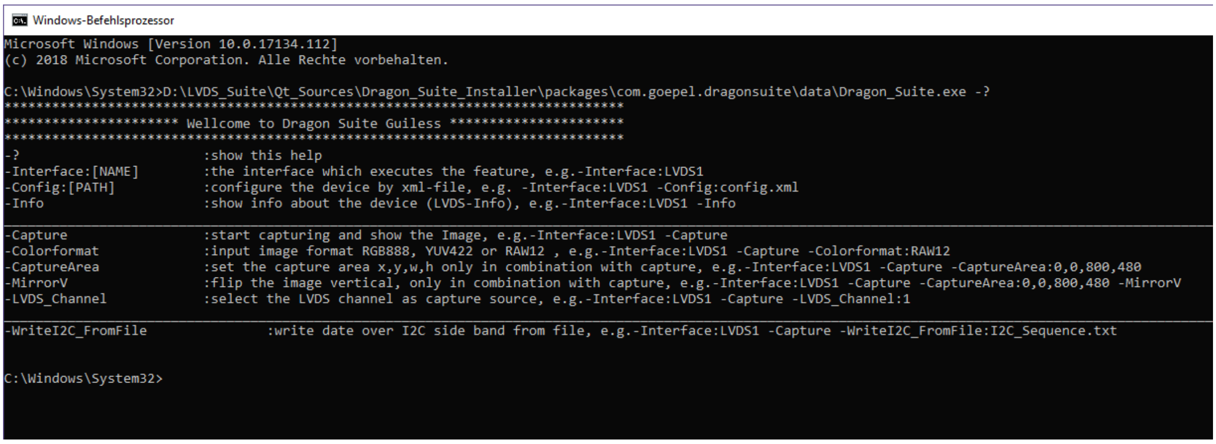


Figure 12.1 GUI-less **Dragon Suite**

13 Dragon Suite Advanced

Some **Dragon Suite** features are only available for **Dragon Suite Advanced** . The features explained in this chapter can only be used in advanced mode.

Category	Feature	BASIC	ADVANCED	Comment
Generator Config	Basic configuration	X	X	
	SerDes + MiMfP GPIO config		X	
Grabber Config	Basic configuration	X	X	
	SerDes + MiMfP GPIO config		X	
	SerDes + MiMfP GPIO config		X	
Generator Dialog	RGB generating	X	X	
	YUV, RAW generating		X	coming soon
	Generating on multiple channels simultaneously		X	coming soon
	Advanced pattern generator		X	
	Video output from file to PC or desktop (mirror)	X	X	
	Video output of recorded RAW data		X	coming soon
Grabber Dialog	Basic capturing	X	X	
	Advanced color format conversion (Bayer, Grey 8+12 bit, YUYV-UYVY 8+10 bit)		X	
	YUV, with limited FR		X	
	RAW data recording		X	
	RAW data to AVI converter		X	
	Grab on multiple channels simultaneously (raw)		X	
Sideband Dialog	UART, I2C, SPI	X	X	
	Send/Receive Ashell Messages		X	
	MII / Ethernet		X	coming soon
Monitoring	CAN Monitor		X	
	Sideband SPI Monitor		X	
	Sideband I^2C Monitor		X	coming soon
	Sideband MII Monitor		X	coming soon
	Sideband UART Monitor		X	coming soon
FS Interface	File System interface	X	X	
IO Interface	r/w digital IOs	X	X	
	Trigger settings	X	X	
Sequence Interface	Sequence Interface		X	coming soon
Script Interface	Script Interface		X	
CAN Interface	Basic message functions		X	only via Script Interface

Category	Feature	BASIC	ADVANCED	Comment
Ethernet	Send/ Receive UDP frames per Fifo		X	only via Script Interface

Table 13.1 Dragon Suite Advanced feature overview

The Advanced version of the **Dragon Suite** is only available through paid licenses, which you can purchase from our sales team. The activation is done via the hardware activation of your **Video Dragon** . If a device with this activation is detected when the software starts, it automatically opens in Advanced Mode. The activation can be done at any time, even after the purchase of the hardware. A free trial license can also be requested from the sales department.

13.1 Script Interface

The Script Interface is used to run Java (ECMA) scripts to automatically control the functions of the **Video Dragon** Hardware. Among other things, this includes configuring the **Video Dragon** , generating or capturing frames to file, switching IOs, and communicating via sideband. This means that all work steps can be completed with just a few clicks. With manual single steps the effort is considerably higher. In addition, all the manual steps must be carried out again and again with each restart or hardware change. The Script Interface makes work easier not only in development, but also in the manufacturing process.

For example, the following sequence can be performed with only one script:

- Configuring the Frame Grabber LVDS interface
- Configuring the Frame Grabber register settings
- Configuring the Frame Grabber CAN - UART Gateway and SerDes GPIOs
- Sending I²C commands from the Frame Grabber to a connected camera
- Output I²C response strings
- Capturing frames to a file

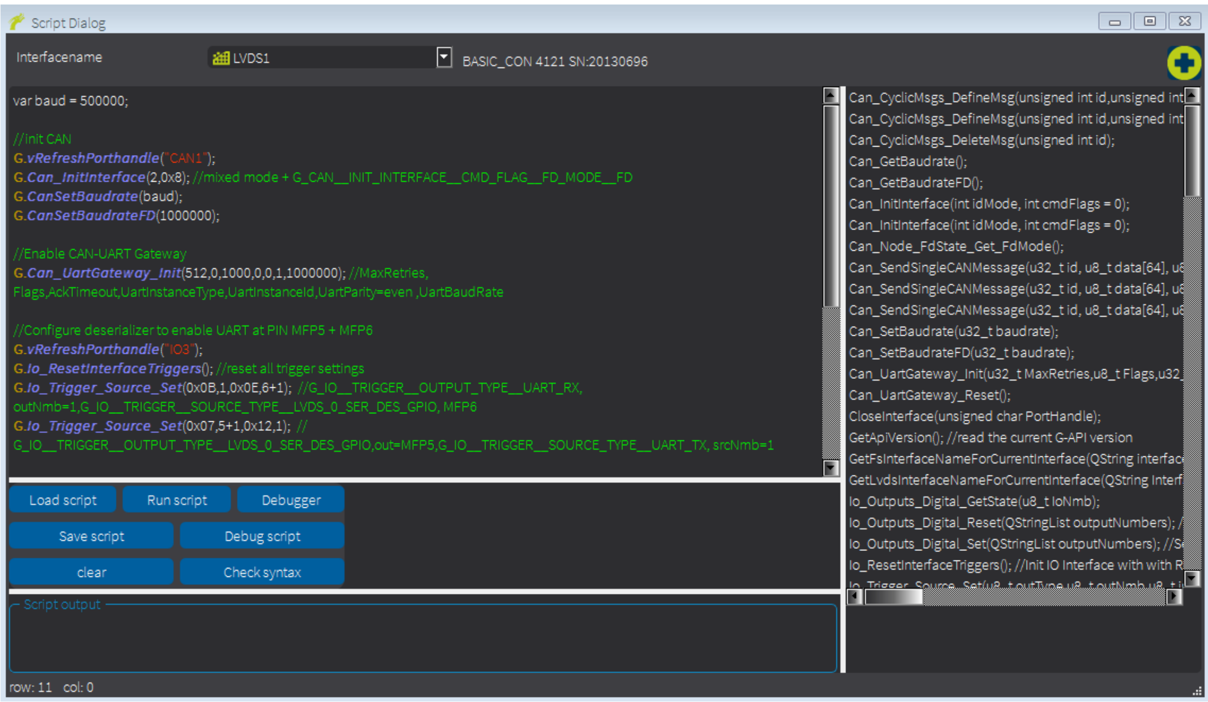


Figure 13.1 Script Interface

All available LVDS interfaces are listed in the drop down menu.

Figure 13.2 Choose interface name

TIP

The interface to be addressed can be changed in the script with `RefreshPorthandle();`.

Thanks to syntax highlighting and autocompletion, creating the script is as easy as possible. Autocompletion is intended for words longer than 2 characters. In addition, you can trigger the autocompletion with the shortcut Ctrl + E.

```
var i = 0;
while(1)
{
		for(i=0; i<255;i++)
		{
		  G.LVDS_DisplayColor(1280,640,i,0,255-i);
		  G.delay(100);
		}
		for(i=0; i<255;i++)
		{
		  G.LVDS_DisplayColor(1280,640,255,i,255-i);
		  G.delay(100);
		}
 }
```

Figure 13.3 Main Window of Script Interface

TIP

Enter **"G."** followed by Ctrl + E to access **Dragon Suite** and **G-API** methods. Use right-click to undo, copy and replace, or to select the entire typing.

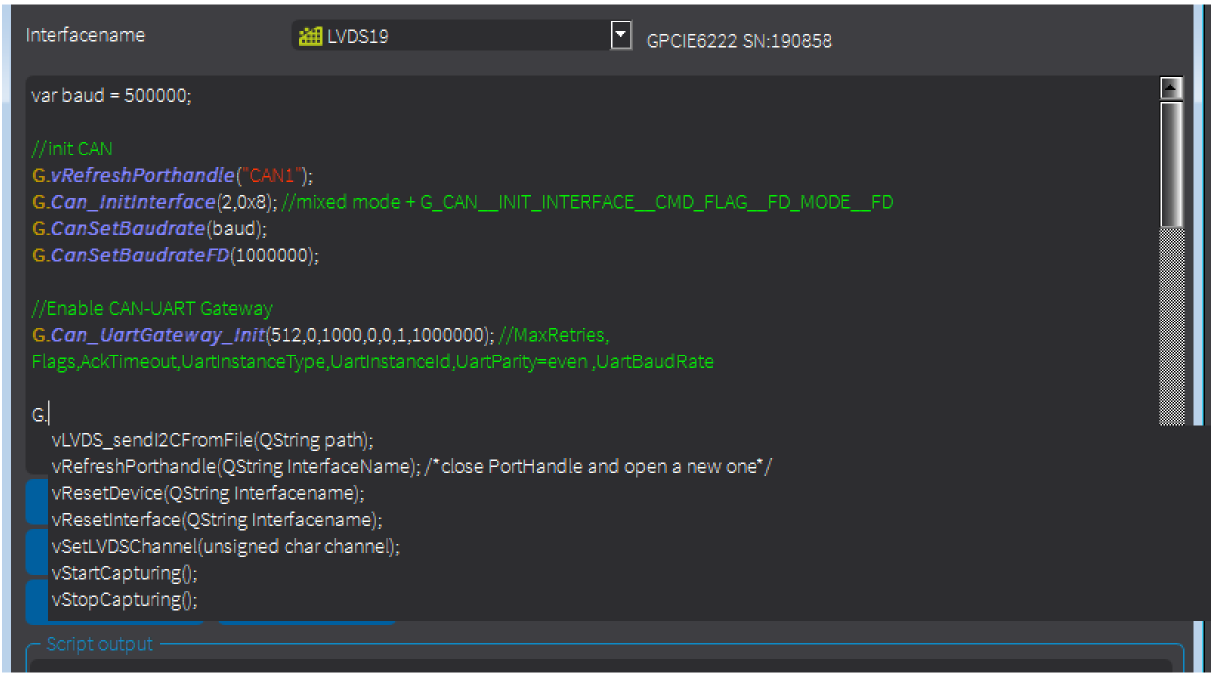


Figure 13.4 Autocompletion in Script Interface

There are several buttons in the Script Dialog window:

Button	Description
Load script	Load the script for the selected interface by importing an external Java (ECMA) script file (*.js).
Save script	Save the current script of the selected interface by exporting them to an external Java (EC-MA) script file (*.js).
clear	Deletes the entries in the main window.
Run script	Starts the loaded script on the selected interface
Debug script	Open the Qt Script Debugger Tool and start debugging of the script
Check syntax	Check the syntax of the script automatically and display found errors.
Debugger	Open the Qt Script Debugger Tool

Table 13.2 Script Interface buttons

The **Script Output** gives feedback when the script is executed or displays any errors.

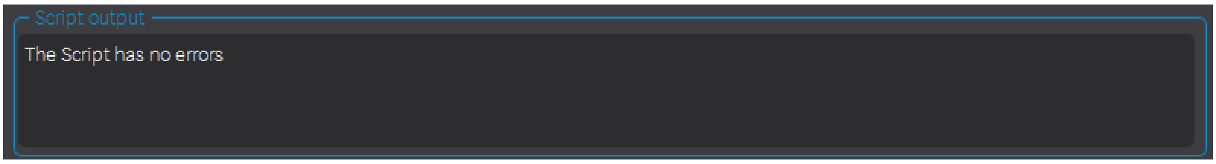


Figure 13.5 Script Output

TIP

Below the script output you can find the information in which row and column of the main window the cursor is located. This makes it easier to search for longer scripts.

The functions that can be used can be found in the **Help** window on the right. Open or close it by using the icon. The search for functions can be simplified by using the lower input field.

TIP

The functions can be dragged and dropped into the main window.

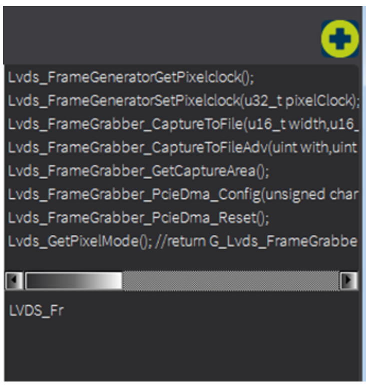


Figure 13.6 Searching for functions in Script Interface

TIP

Some sample scripts can be found in the installation folder of the Dragon Suite.

13.2 Raw Data Recording

Raw Data Recording is a Frame Grabber feature. This is used to store and play back received raw data.

This feature is only supported for **Video Dragon 6222** .

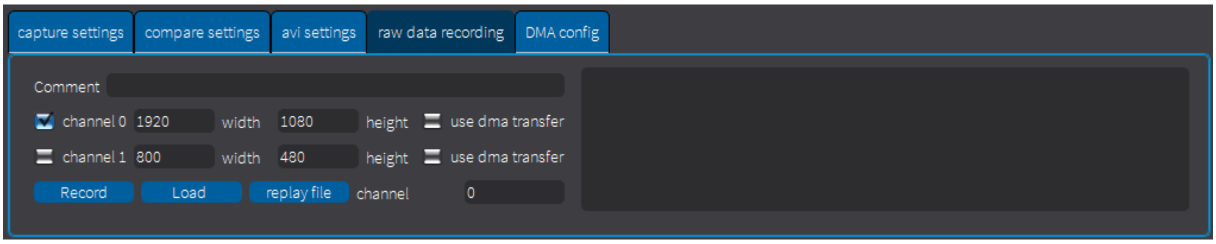


Figure 13.7 Raw data recording

Use the icon to select path to the directory where the *.raw files will be stored. Use **Comment** to insert a short note at the beginning of the RAW file. Select channel 1 or 2 and specify the resolution. Also select whether to use DMA transfer.

Below are three buttons:

Button	Description
Record	Start recording the raw data and save it in the path specified before.
Load	Load safed *.raw file
replay file	Replay loaded raw data

Table 13.3 Raw data recording buttons

Each recorded file is saved with a common file header. The structure is as follows:

```
typedef struct{
     u32_t version; //use current suite version for header version
     u32_t headerSize;
     u8_t numberOfChannels;  //number of LVDS channels to capture and save
     u8_t LVDSChannel[0xFF]; //the real selected LVDS channel
     u8_t pixelmode[0xFF];
     u32_t width[0xFF]; //frame width
     u32_t height[0xFF]; //frameheight
     u32_t dataSize[0xFF] ; //per channel
     u16_t reserved;
     u8_t CSI2_dataType[0xFF];
     u8_t virtualChannel[0xFF];
     u32_t frameHeaderSize; //for image/frame data
     u32_t framecounter[0xFF];
     u32_t deviceFramecounter[0xFF]; //device frame counter differnce while recording
     qulonglong duration; // total duration im ms
     char comment[0xFF]; //description for the file

     u16_t reserved0;
```

Figure 13.8 Raw data file header

The single frames are stored in the following structure:

```
typedef struct{
      u16_t version;
      u16_t rows;
      u16_t cols;
      u8_t pixelmode;
      u8_t CSI2_dataType;
      u8_t virtualChannel;
      u8_t dataChannel; //current channel in file
      u32_t headerSize;
      u32_t dataSize; //size of the video frame in Bytes
      qulonglong timestamp;
      u32_t frameNumber;
      u16_t reserved0;

}RawDataStreamFrameHeader_t;
```

Figure 13.9 Raw data frame header

When loading the file you will find the header information in the right window.

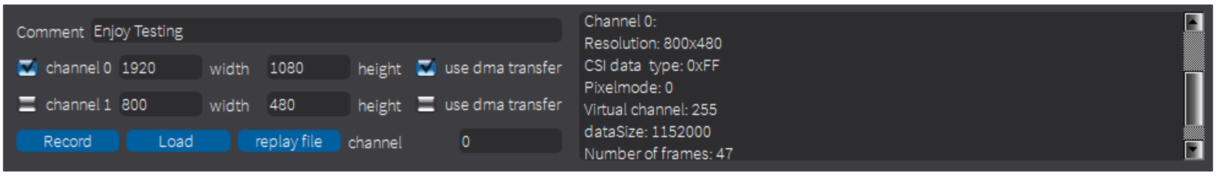


Figure 13.10 Loaded raw data

To replay the file it is not necessary that the Frame Grabber is configured. However, it may be necessary to make adjustments in the Capture Settings (Pixel Mode, Color Data Format). Select the desired channel before replaying (To the right of the **Replay** button). Pressing the **Replay** button will play the loaded file once completely. During replay the process can be stopped with the same button.

13.3 Monitor Dialog

With the Monitor feature CAN and SPI signals can be monitored.

Not every media interface supports all sideband functions mentioned here. Furthermore, sideband and/or CAN must be activated for the device.

All available LVDS interfaces are listed in the drop down menu.

Figure 13.11 Choose interface name

The currently selected interface is displayed in the text field of the drop-down menu. Selecting the interface will automatically open the corresponding tab.

13.3.1 CAN Monitor

Set the monitor settings before starting the monitor. Determine whether **Rx** signals, **Tx** signals or **error frames** should be monitored. A combination of the signals is of course also possible. The monitored frames can be displayed consecutively below each other or **listed.** Listed means that for each signal there is one monitor entry whose time stamp is overwritten when this signal is repeated. **Data block size** indicates after how many bytes a line break occurs (the value must be at least 8).

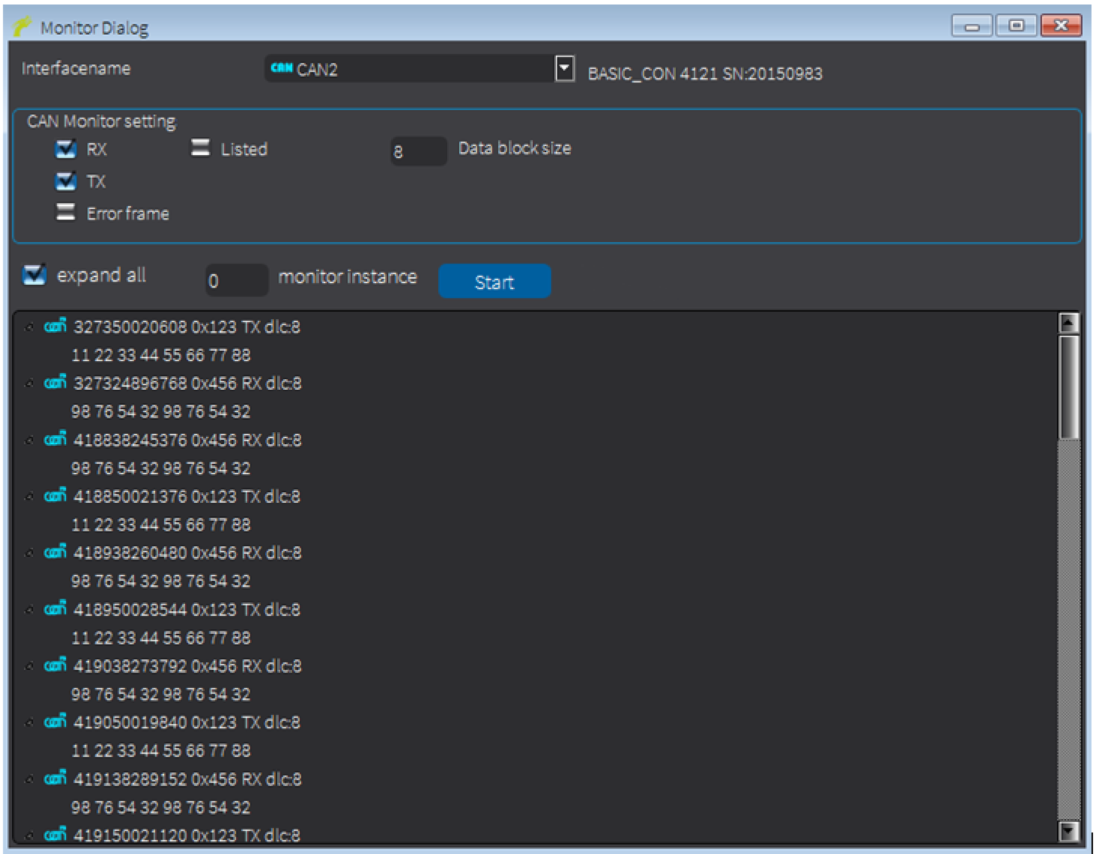


Figure 13.12 CAN Monitor - not listed & expand mode

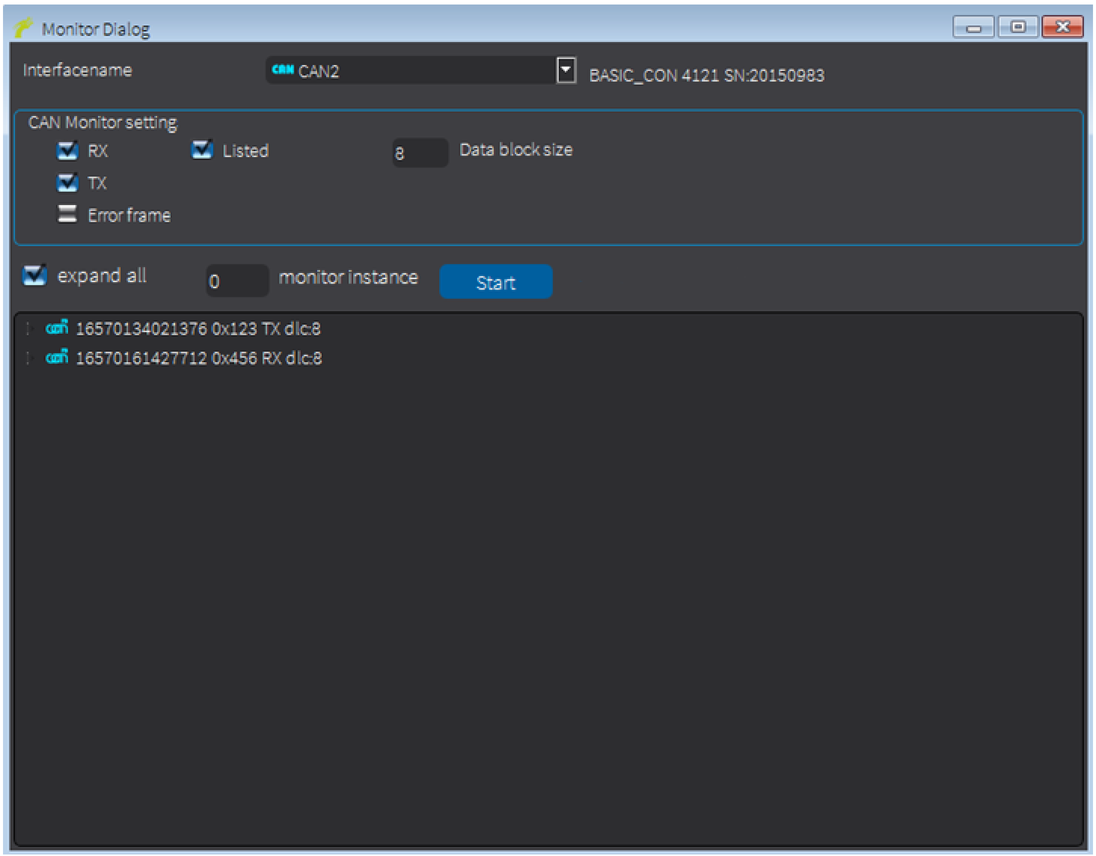


Figure 13.13 CAN Monitor - listed & no expand mode

The signals can be displayed in **expand** mode or not. In expand mode, the data belonging to the signal ID is displayed in a second line. Without this mode the data is hidden, but can be opened individually in the monitor field. To do so, click on the small triangle to the left of the signal.

Figure 13.14 Hide CAN signal data

The displayed signal parameters are described in the following figure:

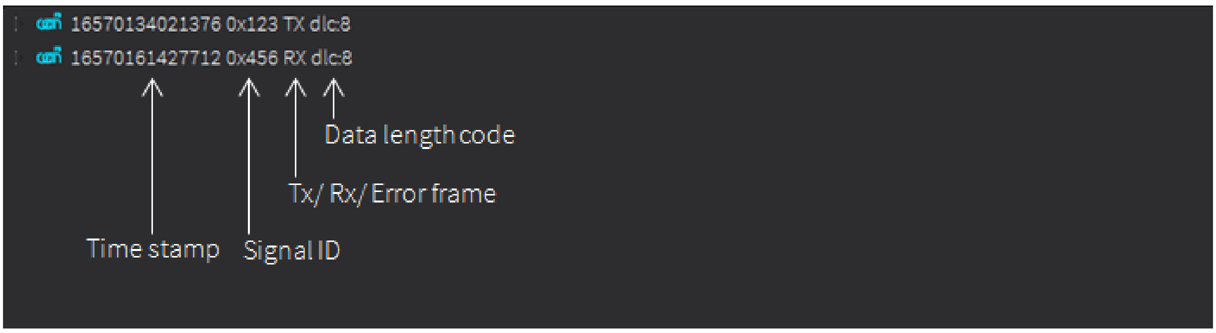


Figure 13.15 Monitored CAN signal parameters

13.3.2 SPI Monitor/ SPI Analyzer

The API of the SPI Analyzer consists of two sections. The first section, **SPI_A** , configures the parameters, which are the same for all monitors connected to this analyzer node and depend on the monitored SPI bus. These parameters include:

- CPha (ClockPhase): Data is active on first or second edge
- DataWidth: Data width from 8 to 32 bits
- SS_Usage: Which SlaveSelect should be monitored (Bit0..SS0,Bit1..SS1,..)
- SS_Polarity: Which polarity do these SlaveSelects have (0..Low,1..High)
- SS_IdleTime: How long is the SlaveSelect at least inactive for the last SPI transfer to be considered completed (in ns)

The clock polarity is determined automatically.

In the second API section, **SPIA_Monitor** , several monitors can be configured for one SPI Analyzer node. This can be useful to allow each monitor to listen to its own SlaveSelect or a combination of SlaveSelects. The parameters are:

- BufferSize: internal buffer for monitor data
- SS_Enable: SlaveSelects observed by this monitor. They must be activated in the first API section (SS_Usage).

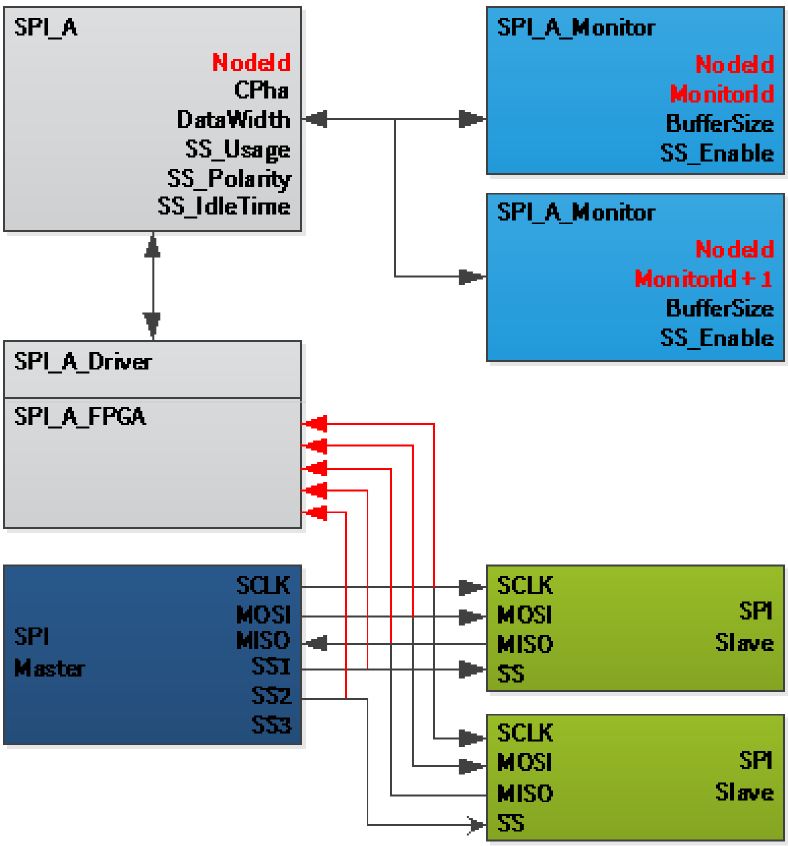


Figure 13.16 SPI Analyzer

For the actual monitoring function it is necessary to route the inputs of the SPI Analyzer. For this we need the Clock, MISO, MOSI and SlaveSelect signals. These are supplied with the appropriate signal sources via TriggerSource_Set as corresponding targets. The routing can for example be like this:

- TargetType: SPI_A_SCLK (SPI Analyzer Clock Signal)
- TargetChannel: 0 (node ID of the SPI Analyzer)
- SourceType: LVDS_MI_0_MFP (Multifunction pins of the **Media Interface** LVDS)
- SourceChannel: 10

Since the SPI Monitor is configured via the IO interface, the IO interface must be opened here. Additionally the triggers must be configured correctly. Further information and **examples** can be found in the chapter IO Tool and at the end of this chapter.

Set the monitor settings before starting the monitor. Determine whether **MOSI** signals, **MISO** signals or which **SS** signals should be monitored. A combination of the signals is of course also possible. Set the **Analyzer Channel**, starting with 0. Only one channel is possible for **basicCON 4121**, the **Video Dragon 6222** supports up to 4 channels. **Data block size** indicates after how many bytes a line break occurs (the value must be at least 8).

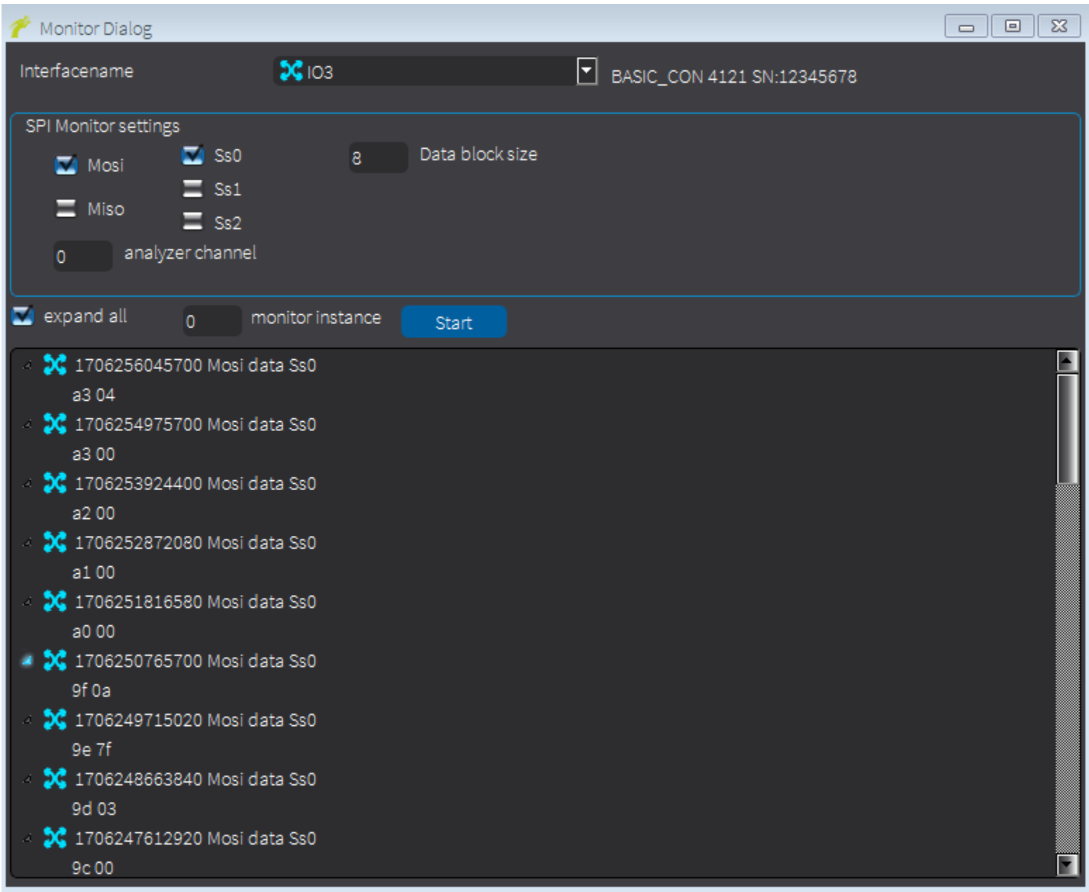


Figure 13.17 SPI Monitor

The data is written to the monitor from bottom to top and from right to left.

The signals can be displayed in **expand** mode or not. In expand mode, the data belonging to the signal is displayed in a second line. Without this mode the data is hidden, but can be opened individually in the monitor field. To do so, click on the small triangle to the left of the signal.

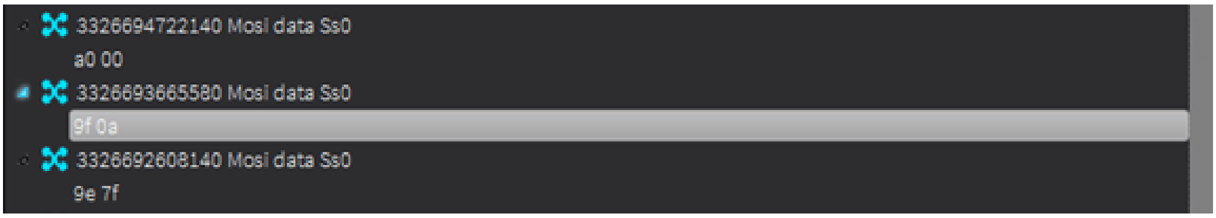


Figure 13.18 Hide SPI signal data

It is possible to monitor over several instances. For example, one **monitoring instance** can be used for MOSI and one for MISO. To display the monitoring instances simultaneously, the dialog box can be opened multiple times.

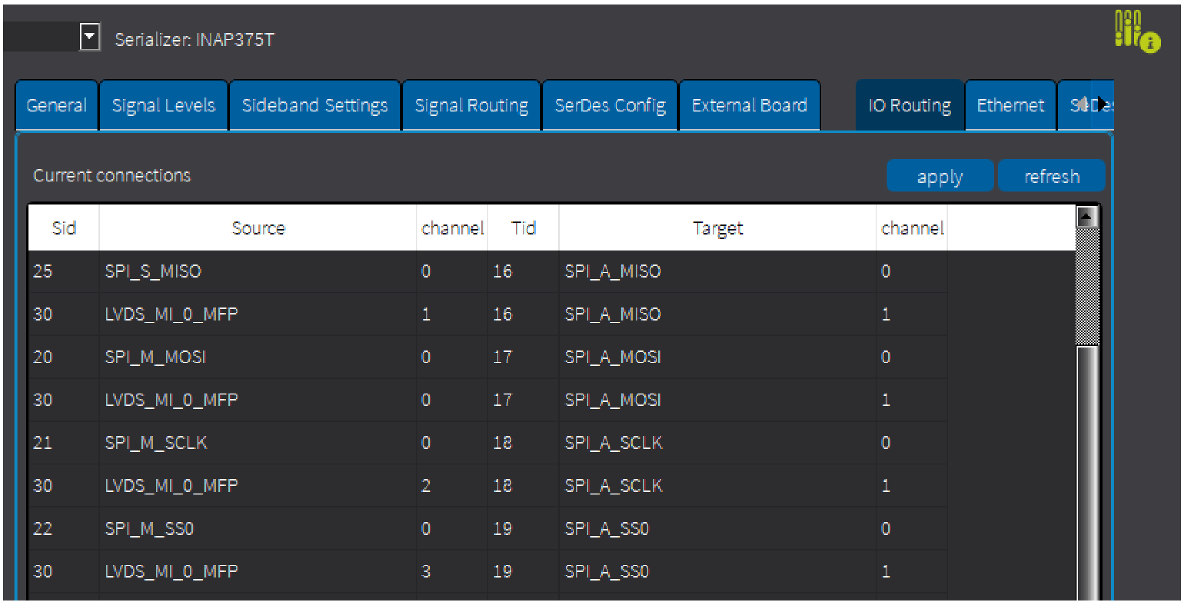
Use the **Save** Button to save the monitor data to a *.txt file.

Example for SPI Monitor

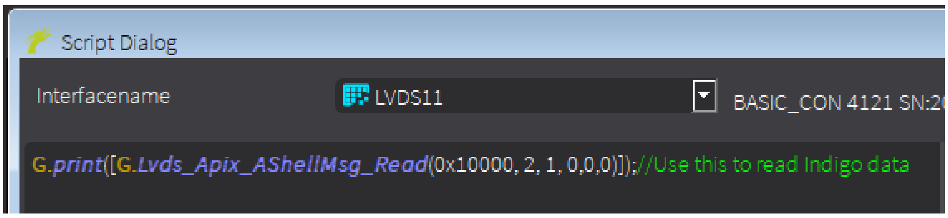
For understanding, here is an example of configuring the SPI analyzer via the IO interface and sending a read command to read the serial number of an Apix chip. For this purpose, a **basicCON 4121** with INAP375T **Media Interface** board is used, which communicates with an Apix deserializer.

The following steps were done:

1. Set the IO triggers to configure the SPI analyzer. We want to read the sent and received messages with the monitor. For this two monitor instances must be configured. The first instance is for reading the sent messages (instance 0), the second instance is for reading the received messages (instance 1). In the picture below you can see how the IO triggers are connected to each other. The target channels represent the monitor instances. For instance 0 we use SPI triggers, for instance 1 the multifunction pins are used as source.

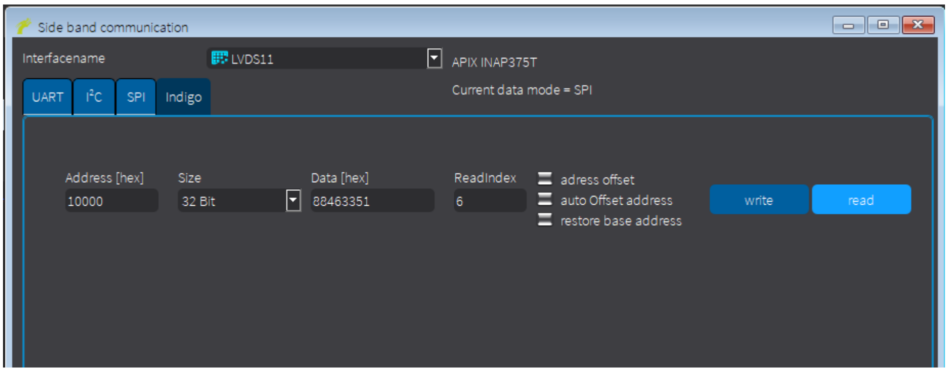


2. There are two ways to read the data. The first is to use the Script Interface. The G-API function to read the AShell message is included in the Script Interface. Adapted to our DUT the script command is `G.Lvds_Apix_AShellMsg_Read(0x10000, 2, 1, 0,0,0)`.

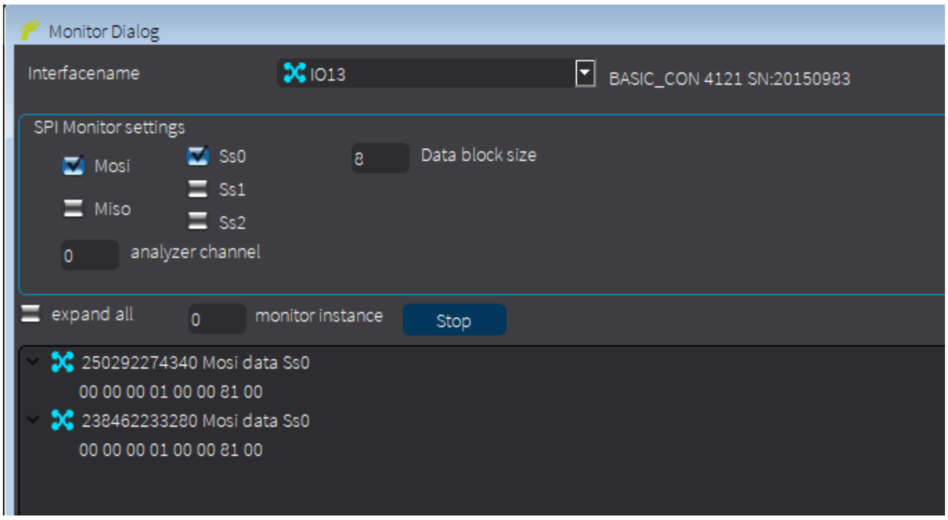


About the parameter values:The address is 0x10000 hex. Since the data size is 32 bit, the second value is "2". The Read Index is defined as "1". Since we do not set a flag, the remaining three values are specified as "0". If the command is packed in a `G.print` command, the answer is printed in the script output, in our case the serial number of the chip.

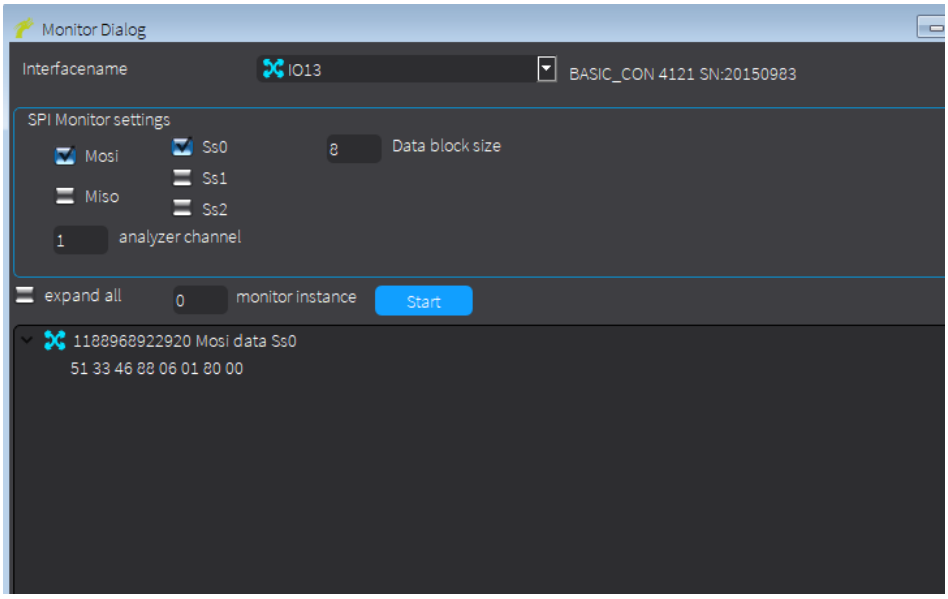
The second option is to use the Indigo tab in the sideband window. Here also the address and the data size are specified. The response (serial number) then appears in the data field.



3. The SPI messages can be read in the SPI Monitor. Before sending the AShell Read command the monitor must be started (Start button). If we look at monitor instance 0 (analyzer channel), we see the sent data.



At monitor instance 1 (analyzer channel) we read the received data (from right to left). The serial number is 88463351.



14 First Steps

Prerequisite for these steps is a successful installation of the **G-API** and the **Dragon Suite** .

The supported **basicCON 4121** and **Video Dragon 6222** devices can be used with different Media Interface Modules and must be configured differently. For this chapter, it is assumed that a DS90UB954 deserializer module is installed in a **G PCIe 6222** Board, and a compatible video source is already set up and ready for use.

14.1 System Structure

1. Install the appropriate module on the main board (in this example it is DS90UB954). (Typically, this step is not required because a module is already installed at delivery.)
2. Install the **GOEPEL electronics** device in your (switched off) test system or your PC.
3. Connect your video source to the input connector of the **GOEPEL electronics** device using the supplied video cable.
4. Switch on the test system and thus the **GOEPEL electronics** device. As soon as the device is ready, LED2 starts flashing.

14.2 Registration

Before you can use the **GOEPEL electronics** device for the first time, it must have been registered in the **G-API** . The **G-API** is responsible for all future communication between the control PC or laptop and the **GOEPEL electronics** device. This registration is simply done by starting the **HardwareExplorer** . The following figure shows a **G PCIe 6222** board with four interfaces:

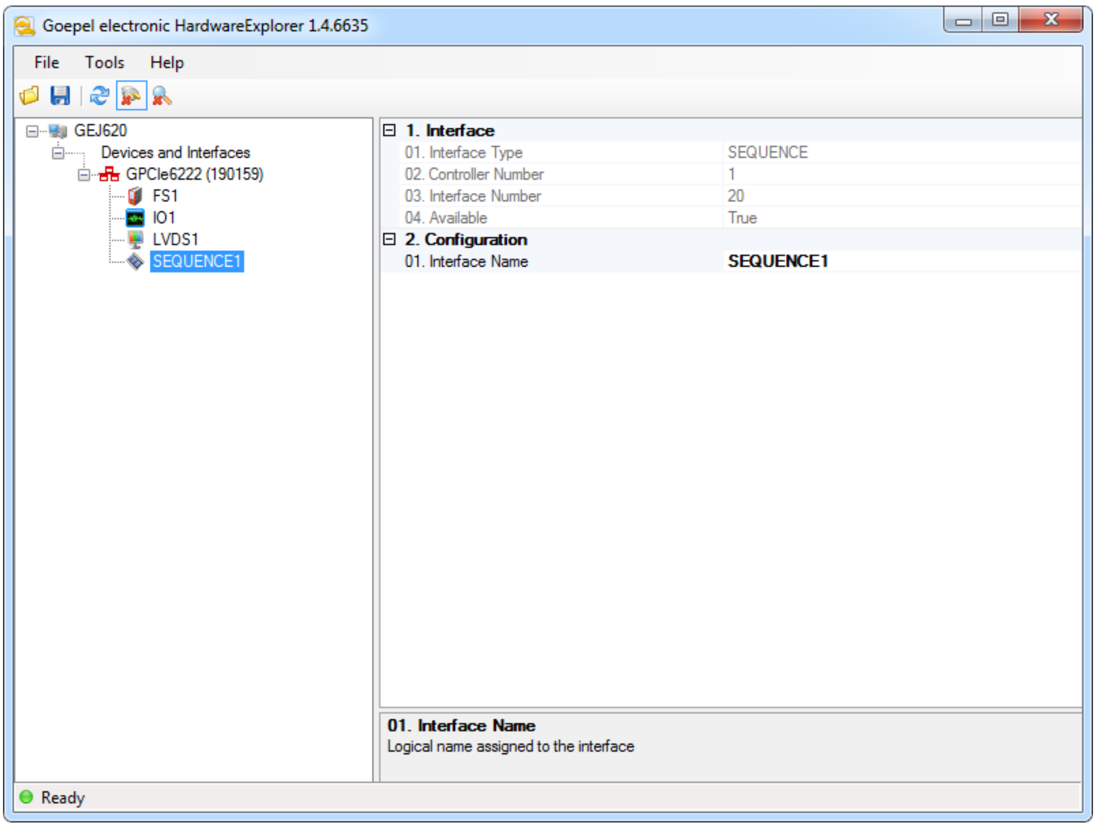


Figure 14.1 HardwareExplorer with **G PCIe 6222** board

The **GOEPEL electronics** device can be seen in the left column with all available software interfaces. If several devices are connected, the corresponding device can be identified by its serial number, which is shown in parentheses. The name of the LVDS interface (e.g., "LVDS1") is important to the following steps.

For more information about the **G-API** , its installation, and the **Hardware Explorer** , see **G-API Quick-start Guide** .

14.3 Configuration

Before the capturing of frames is possible, the device must be configured according to the currently transmitted video signal.

1. Start the **GOEPEL electronics Dragon Suite** software. On the left side in the Interface Tree the **G PCIe 6222** Board and its interfaces appear, similar to the **Hardware Explorer** .

2. The icon opens the Settings Window for the Frame Grabber. Select the corresponding LVDS interface as Interface Name from the preselection of the drop-down list (here "LVDS1").

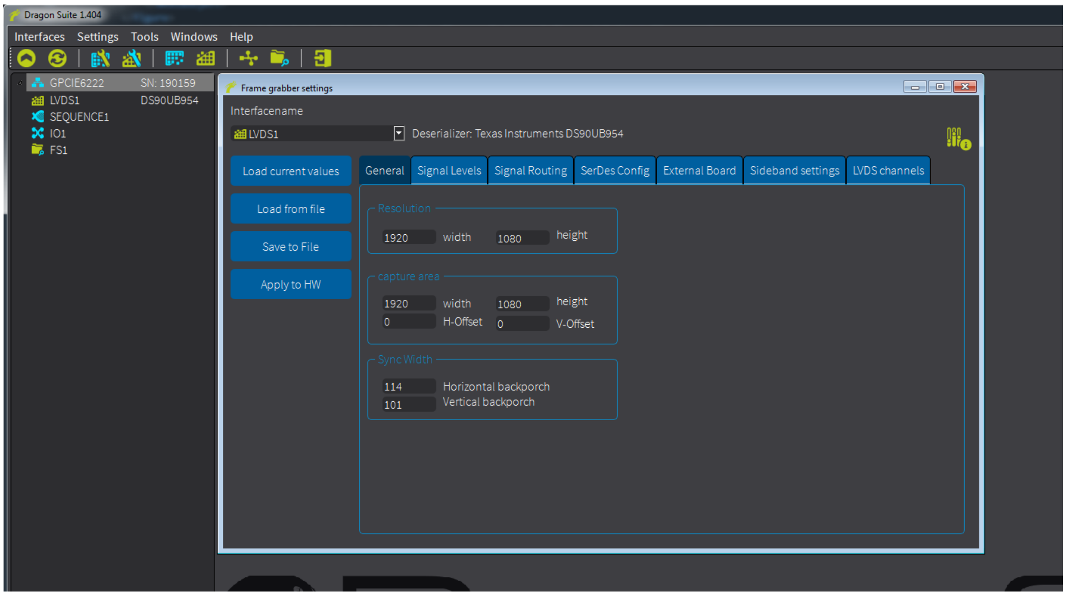


Figure 14.2 Frame Grabber Configuration

All settings for the deserializer can be made in this dialog.

3. After selecting the LVDS interface, the values in the configuration window are overwritten with the current values of the **GOEPEL electronics** device automatically. The current configuration parameters can also be loaded into the settings window via the button "Load current values".

4. Enter the desired resolution in "Capture Area". This must not be higher than the resolution of the incoming frame. Confirm this entry with the button "Apply to HW".

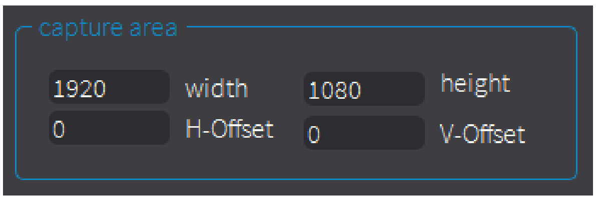


Figure 14.3 Frame Grabber Capture Area

5. Switch to the LVDS Channels tab and adjust the parameters of the physical channels according to your test requirements. Confirm this entry with "Apply" in the lower right corner of the dialog box.

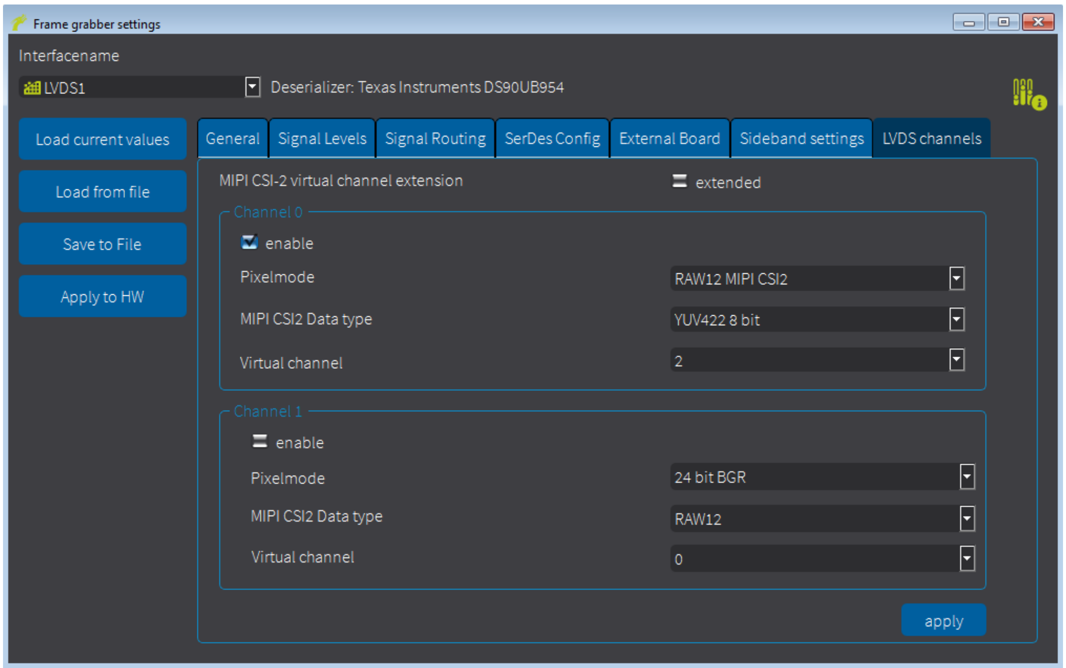


Figure 14.4 Frame Grabber LVDS Channels

6. In most cases, the configuration registers of the deserializer must be adapted to the test environment. In this example, the registers are written using I²C communication. Switch to the Sideband Settings tab and change the Data Mode to "I²C Master to Deserializer". Adjust the other parameters (baud rate etc.) according to your test requirements. Confirm this entry with the button "Apply to HW".

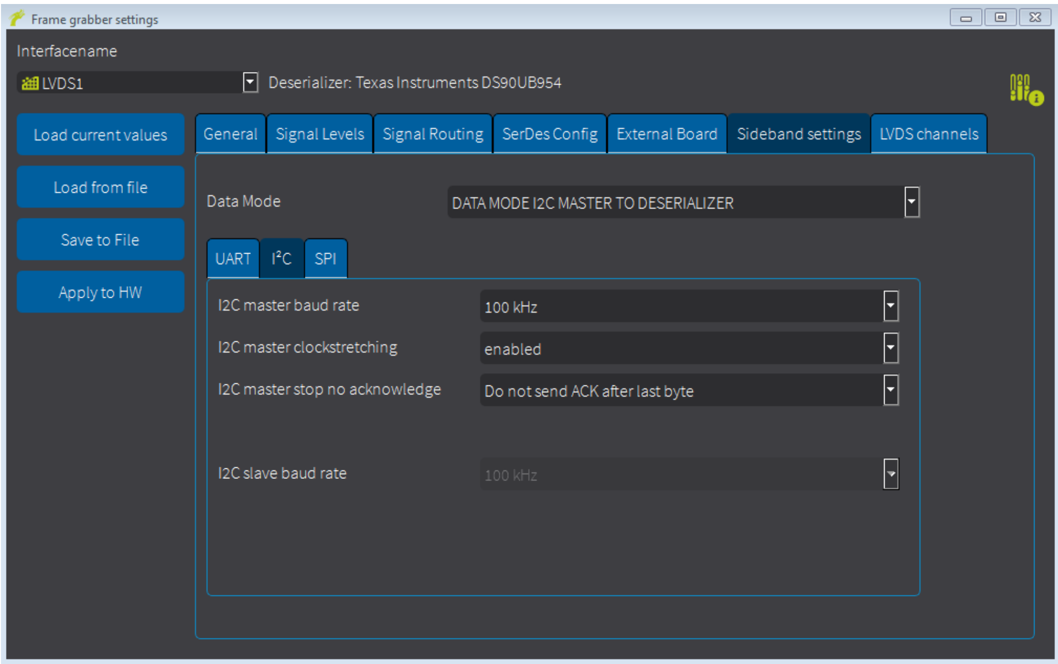


Figure 14.5 Frame Grabber Sideband Settings

7. Open the Sideband Communication Window with the icon . Also select the LVDS interface here ("LVDS1"). The tab for I²C opens automatically if the Data Mode was successfully overwritten before.

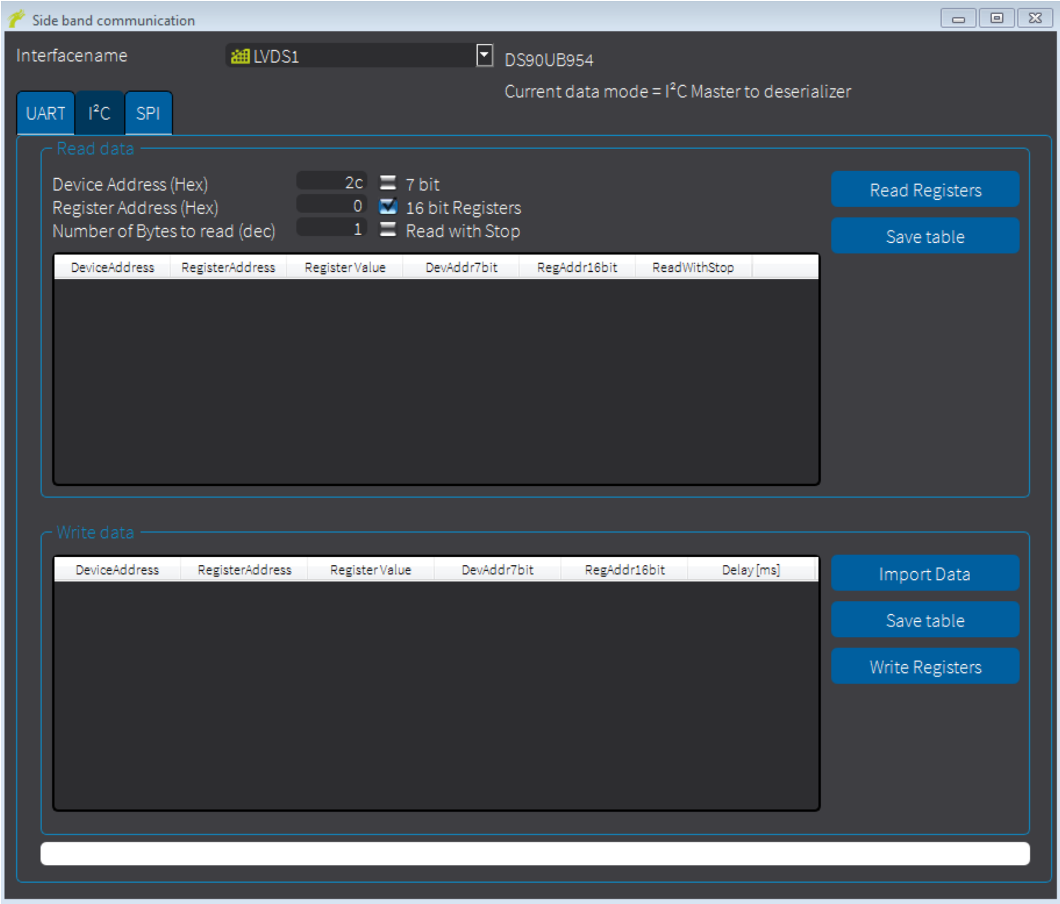


Figure 14.6 Frame Grabber Sideband Communication

Here, the configuration registers of the deserializer can be read out and overwritten. Reading is done by entering the device address and register address and the "Read Registers" button. Save the table with "Save Table". Now you can change the desired tabs in the saved text file. With "Import Data" the changed list is imported again and by clicking on "Write Registers" the registers of the deserializer are changed.

The device is now configured and ready to capture frames.

14.4 Capturing

The icon opens the Frame Grabber Dialog Window. It is possible to capture both single frames and a sequence of frames.

The icon captures a single frame from the video stream and displays it in the dialog window. The capturing of a sequence of frames is started by pressing the icon . After each captured frame of the sequence, the dialog window is updated to show the frame being captured.

Your **Video Dragon** and also the **Dragon Suite** offers a multitude of additional functions. In this chapter, only general instructions for working with the device could be given by way of example.

15 Common Error Messages

The following table shows common **G-API** error messages and how to fix them:

Error Code	Error Message	Description
0x000000E0	FW - function not available	The selected feature (e.g., Sideband Communication or CAN) is not activated for this device.
0x00090002	LVDS - capturing not in progress	The deserializer does not receive image data from the serializer. Check the lock between serializer and deserializer. The cause may also be due to a bad signal, so that a frame grabber has difficulty capturing. Also check the connection between the PC and the device.
0x00090006	LVDS - no video lock	There is no lock between serializer and deserializer. Make sure the configurations of the devices match. In most cases, the Signal Levels (Polarities) or the Control Signals (HSync, VSync, DataEnable) do not match.
0x00090007	LVDS - initializing after lock lost	The link between serializer and deserializer was lost. This must now be reinitialized. The cause may be due to a bad signal, so that a frame grabber has difficulty capturing. Also check the connection between the PC and the device.
0x0009010A	LVDS - config - capture area bigger than frame	The defined Capture Area of the Frame Grabber is larger than the transmitted video frame.
0x00090201	LVDS-data-extensionboard function not available	One of the selected modes is not supported for the serializer/ deserializer.
0x0009020D	LVDS - data no acknowledge	The device receives no response from the receiver during sideband communication (e.g., I^2C). There may be no connection or the recipient address is incorrect.
0x0206000D	LVDS-sync error-videolock not active	There is no lock between serializer and deserializer. Make sure the configurations of the devices match. In most cases, the Signal Levels (Polarities) or the Control Signals (HSync, VSync, DataEnable) do not match.
0x2000107	API - event timeout	The command triggered a timeout in the **G-API** . There was no frame at the grabber input. The cause may be due to a bad signal, so that a frame grabber has difficulty capturing. Also check the connection between the PC and the device.
0x2000205	API - no response from device	The **G-API** has lost connection to the device. The cause maybe due to a bad signal, so that a frame grabber has difficulty capturing. Also check the connection between the PC and the device.

Table 15.1 Common Error Messages

Some error messages can occur at the same time and have the same cause. This is because some applications call multiple functions of the **G-API** , and then each returns an error.

TIP

Please make sure to keep the **G-API** and the **Dragon Suite** up to date. This can prevent some errors.

TIP

Always check the connection between the PC and the **GOEPEL electronics** device. We recommend the use of Ethernet. A USB2.0 connection, for example, is too slow to achieve error-free capture. It may also be necessary to restart the device.

16 Service and Support

16.1 Spare Parts and Accessories

If necessary, please contact our sales department:

GOEPEL electronics GmbH
ATS-Vertrieb
Göschwitzer Str. 58 / 60
D-07745 Jena
Tel.: +49-3641-6896-508
E-Mail: ats.sales@goepel.com
http://www.goepel.com

16.2 Warranty and Repair

16.2.1 Conditions

We guarantee the accuracy of the test system for a period of 24 months from the date of sale. The warranty does not apply to errors that are based on improper interventions or changes or improper use.

16.2.2 Identification

Furthermore, we ask you to announce possible warranty cases as such. Repair orders without reference to an existing warranty claim will in any case initially be paid. If the warranty has expired, we will of course also repair your test system in accordance with our general installation and service conditions.

If necessary, please contact our support service:

GOEPEL electronics GmbH
ATS-Support
Göschwitzer Str. 58 / 60
D-07745 Jena
Tel.: +49-3641-6896-597
E-Mail: ats.support@goepel.com
http://www.goepel.com

17 Disposal

17.1 Disposal of used Electrical / Electronic Equipment

The device implements the following EU directives:

- 2012/19/EU (WEEE) Waste Electrical and Electronic Equipment and
- 2011/65/EU on the restriction of the use of certain hazardous substances in electronic equipment (RoHS directive)

At the end of the life of the device, this product must not be disposed of with other household waste. The improper disposal of this type of waste can have a negative impact on the environment and health due to the potential hazardous substances in electrical and electronic equipment. Dispose of the product at a suitable collection point.

When disposing of the device in countries outside the EU, local laws and regulations must be observed.

17.2 Disposal of used Disposable / Rechargeable Batteries

At the end of the service life of disposable/ rechargeable batteries, these must not be disposed of with the normal household waste. Dispose of the disposable/ rechargeable batteries at a recycling center for disposable batteries and rechargeable batteries.

Please dispose of only discharged disposable/ rechargeable batteries.

Index