Produktbeschreibung

USB 4120
basicCON 4120

LVDS Framegrabber
Nutzerhandbuch Version 1.4

Get the total Coverage!

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

© 2013 GÖPEL electronic GmbH. Alle Rechte vorbehalten.

Die in diesem Handbuch beschriebene Software sowie das Handbuch selbst dürfen nur in Übereinstimmung mit den Lizenzbedingungen verwendet oder kopiert werden.
Zu Sicherungszwecken darf der Käufer eine Kopie der Software anfertigen.

Der Inhalt des Handbuchs dient ausschließlich der Information, ist nicht als Verpflichtung der GÖPEL electronic GmbH anzusehen und kann ohne Vorankündigung verändert werden.

Hard- und Software unterliegen ebenso möglichen Veränderungen im Sinne des technischen Fortschritts.

Die GÖPEL electronic GmbH übernimmt keinerlei Gewähr oder Garantie für Genauigkeit und Richtigkeit der Angaben in diesem Handbuch.

Ohne vorherige schriftliche Genehmigung der GÖPEL electronic GmbH darf kein Teil dieser Dokumentation in irgendeiner Art und Weise übertragen, vervielfältigt, in Datenbanken gespeichert oder in andere Sprachen übersetzt werden (es sei denn, dies ist durch die Lizenzbedingungen ausdrücklich erlaubt).

Die GÖPEL electronic GmbH haftet weder für unmittelbare Schäden noch für Folgeschäden aus der Anwendung ihrer Produkte.

Gedruckt: 23.04.2013

Alle in diesem Handbuch verwendeten Produkt- und Firmennamen sind Markennamen oder eingetragene Markennamen ihrer jeweiligen Eigentümer.

Stand: April 2013

1 INSTALLATION 1-1

1.1 USB 4120/ BASICCON 4120 HARDWAREINSTALLATION 1-1
1.2 TREIBERINSTALLATION 1-2

2 HARDWARE 2-1

2.1 BESTIMMUNG 2-1
2.2 TECHNISCHE DATEN 2-4
2.2.1 Abmessungen 2-4
2.2.2 Elektrische Kennwerte 2-4
2.3 AUFBAU 2-5
2.3.1 Allgemeines 2-5
2.3.2 Stromversorgung 2-6
2.3.3 Adressierung 2-7
2.3.4 LVDS-Schnittstelle 2-7
2.3.5 Digitale I/O Schnittstelle 2-8
2.3.6 LED Anzeige 2-9
2.3.7 Empfänger-Aufsätze 2-10
2.4 LIEFERHINWEISE 2-17

3 ANSTEUERSOFTWARE 3-1

3.1 PROGRAMMIEREN ÜBER G-API 3-1
3.2 PROGRAMMIEREN ÜBER DLL-FUNKTIONEN 3-1
3.2.1 Driver_Info 3-3
3.2.2 DLL_Info 3-4
3.2.3 Xilinx_ Version 3-5
3.2.4 Write_ COMMAND 3-6
3.2.5 Read_ COMMAND 3-7
3.2.6 IdentInfo 3-8
3.2.7 Reset 3-10
3.2.8 Get HardwareInfo 3-11
3.2.9 SyncInfo 3-13
3.2.10 InitInfo 3-15
3.2.11 Init 3-18
3.2.12 Set ClockSource 3-22
3.2.13 Set BitMask 3-24
3.2.14 External TriggerMode_Start 3-26
3.2.15 External TriggerMode_Stop 3-28
3.2.16 External TriggerMode_ GetState 3-29
3.2.17 Capture 3-31
3.2.18 Compare 3-33
3.2.19 Capture ToBuffer 3-35
3.2.20 Capture ToFile 3-37
3.2.21 Read CapturedFrame ToBuffer 3-39
3.2.22 Read CapturedFrame ToFile 3-41
3.2.23 Load Reference FromBuffer 3-43
3.2.24 Load Reference FromFile 3-45
3.2.25 Read Reference ToBuffer 3-47
3.2.26 Read Reference ToFile 3-49
3.2.27 Deserializer Configuration Vector 3-51
3.3 PROGRAMMIEREN MIT LABVIEW 3-54
3.4 NUTZUNG WEITERER GÖPEL SOFTWARE 3-54
3.5 STEUERBEFEHLE USB-CONTROLLER 3-55
3.5.1 USB-Befehlsaufbau 3-55
3.5.2 USB- Antwortaufbau 3-55
3.5.3 USB-Befehle 3-55

1 Installation

1.1 USB 4120 / basicCON 4120 Hardwareinstallation

Das LVDS Framegrabber-Board USB 4120 wird in einem GÖPEL electronic USB-Rack installiert (USB 1004/ USB 1008/ USB 1016).

Warnung

Stellen Sie bitte unbedingt sicher, dass alle Hardware Installationsarbeiten im ausgeschalteten Zustand Ihres Systems erfolgen!

Zur Installation eines USB 4120-Boards wird das USB-Gerätesystem entsprechend der Beschreibung im zugehörigen Manual geöffnet. Wählen Sie einen freien Steckplatz in Ihrem System aus und führen das Board vorsichtig in den vorbereiteten Steckplatz ein.
Mit dem an der Frontplatte befindlichen Hebel wird es das letzte Stück eingeschoben.

Warnung

Fassen Sie das Board bei der Montage nur an den Rändern an. Berühren Sie niemals die Oberfläche, da sonst akute Zerstörungsgefahr durch elektrostatische Aufladung besteht.

Bei der Hardwareinstallation des basicCON 4120 sind nur die Kabel für USB, LVDS und ggf. die Stromversorgung anzuschließen (siehe Hardware).

Zur Positionierung mehrerer USB 4120-Boards im USB-Rack bzw. basicCON 4120-Baugruppen am PC/ Laptop beachten Sie bitte die Hinweise im Abschnitt Adressierung.

1.2 Treiberinstallation

Windows Treiber

Um die GÖPEL electronic USB-Treiber auf Ihrem System einzurichten, muss das GUSB Treiber Setup ausgeführt werden.
Starten Sie dazu dass auf der mitgelieferten CD enthaltene Setup Programm *GUSB-Setup-*.exe* (der Stern steht für die Versionsnummer) und folgen Sie den Anweisungen.

Der zur Verfügung stehende Devicetreiber unterstützt sowohl Windows® 2000-/ XP- als auch Windows® 7 (32Bit und 64Bit) Systeme.

Wenn Sie eigene Software für USB 4120 bzw. basicCON 4120-Baugruppen erstellen wollen, benötigen Sie die Dateien für die anwenderspezifische Programmierung (*.VI, *.LLB). Diese werden nicht automatisch übernommen und müssen deshalb manuell von der mitgelieferten CD in Ihr Entwicklungsverzeichnis kopiert werden.

Die USB-Schnittstelle nutzt, falls möglich, die high-speed Datenrate entsprechend USB2.0 Spezifikation, ansonsten USB1.1 (full-speed).

Nach der Treiberinstallation können Sie überprüfen, ob die Baugruppen einwandfrei vom System eingebunden wurden.

Die folgende Abbildung zeigt die erfolgreiche Einbindung von vier USB 4120 Boards bzw. basicCON 4120-Baugruppen (USB 4120):

Abbildung 1-1: Anzeige Geräte-Manager

Beachten Sie bitte, dass der Geräte-Manager ALLE USB-Controller anzeigt, die vom GUSB-Treiber unterstützt werden.

2 Hardware

2.1 Bestimmung

Das LVDS Framegrabber-Board USB 4120 ist ein Signalverarbeitungsboard der GÖPEL electronic GmbH mit USB 2.0-Interface.

Dieses Board wird speziell in der Automobiltechnik zur Aufzeichnung und Auswertung von Bilddaten verwendet, die über LVDS-Systeme übertragen werden.

Folgende Eigenschaften kennzeichnen den LVDS Framegrabber:

- LVDS Datenraten von bis zu 1.200Mbit/s
- Pixeltakt bis 50MHz bei einer Farbtiefe von 24 Bit
- 1 Megapixel maximale Auflösung
- USB 2.0 Schnittstelle
- Digitale I/O-Schnittstelle bis 24V zur Echtzeittriggerung
- 8 MByte SRAM Onboard
- Referenzbildspeicher für Onboard-Bildvergleich
- Auswechselbarer LVDS-Empfängeraufsatz
  Verfügbare Typen: DS90UR124, DS90CF364, DS90UR906, DS90UB926, INAP125R24APIX, MAX9248, MAX9260 und CXB1458R
- Volle Flexibilität in Auflösung, Farbtiefe und Synchronisation zur genauen Abstimmung des Framegrabbers mit der Quelle
- Visualisierung der Controllerzustände mittels vier LEDs
  (siehe LED Anzeige) auf der Frontplatte

Die Empfängeraufsätze DS90UR906 und DS90UB926 verfügen über einen zweiten LVDS-Anschluss. Dieser gibt das LVDS-Eingangssignal wieder aus, nachdem es von einem Repeater aufbereitet wurde. Dieses Signal kann z.B. für ein Display weiter verwendet werden.

USB 4120 ist ein Signalverarbeitungsboard für den Einsatz in einem USB-Rack der GÖPEL electronic GmbH.
Die Stromversorgung erfolgt über das Netzteil des USB-Racks.

basicCON 4120 ist ein GÖPEL electronic GmbH Stand-alone-Gerät auf der Grundlage eines USB 4120-Signalverarbeitungsboards zum Anschluss an einen PC oder Laptop, das für den eigenständigen Einsatz außerhalb komplexer Testsysteme entwickelt wurde.

Das basicCON 4120 kann über die USB-Schnittstelle oder die Buchsen für ext. Power Supply (7-25VDC) versorgt werden
(siehe Stromversorgung).

Die externe Spannungszufuhr und die digitale I/O Schnittstelle erlauben nach entsprechender Konfiguration die Nutzung des basicCON 4120 zum Bildvergleich auch ohne USB-Anschluss
(siehe Digitale I/O Schnittstelle).

Für den Betrieb von USB 4120-Boards ist ein GÖPEL electronic USB-Rack erforderlich (USB 1004, USB 1008 oder USB 1016), das bis zu 16 GÖPEL electronic USB-Boards aufnehmen kann.

Abbildung 2-1: USB 4120

Abbildung 2-2: basicCON 4120

Im basicCON 4120 nach Abbildung 2-2 ist ein anderer Empfänger-Aufsatz (Deserializer) eingebaut als im USB 4120-Board von Abbildung 2-1.
Deshalb ist die Anzahl der LVDS-Buchsen unterschiedlich (siehe auch LVDS-Schnittstelle und Empfänger-Aufsätze).

An der Geräterückseite des basicCON 4120 befinden sich die folgenden Anschlüsse:

Abbildung 2-3: basicCON 4120 - Rückseite

- USB-B-Buchse für das USB 2.0 Interface mit USB-Standardbelegung
- DC-Buchse für das Steckernetzteil
- Bananenbuchsen zur Stromversorgung

Warnung

Bitte nutzen Sie für die externe Stromversorgung entweder die beiden Bananenbuchsen ODER die DC-Buchse.

Zur Stromversorgung vergleichen Sie bitte die Hinweise im entsprechenden Kapitel.

2.2 Technische Daten

2.2.1 Abmessungen

(Breite x Höhe x Tiefe):

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

Die Angaben für USB 4120 beziehen sich auf ein Board in einem GÖPEL electronic USB-Rack.

2.2.2 Elektrische Kennwerte

Symbol	Kennwert	min.	typ.	max.	Einheit	Bemerkung
V_{SS}	Versorgungsspannung		5		V	Versorgung über USB (basicCON 4120)
I_{SS}	Versorgungsstrom		300	400	mA	
V_{SExt}	Externe Versorgungsspannung	8	12	25	V	I_{Max} = 200mA bei 12V Optional für basicCON 4120
V_{SDIO}	Stromversorgung der digitalen I/O Schnittstelle	5		27	V	Optional (siehe Digital I/O Schnittstelle)
V_{LVDS}	Differenzielle LVDS Eingangsspannung	±50	±500		mV	Abhängig vom Aufsatzboard
I_{LVDS}	Differenzieller LVDS Eingangsstrom		±40		µA	Abhängig vom Aufsatzboard
Z_{LVDS}	LVDS Eingangsimpedanz		100		Ω	
f_{LVDS}	LVDS Eingangsfrequenz			1200	MHz	Abhängig vom Aufsatzboard
V_{DIn}	Eingangsspannung digitale I/O	5		27	V	5V ohne Versorgung über DIO-Schnittstelle
I_{DIn}	Eingangsstrom digitale I/O			12	mA	Bei 27V
V_{DOut}	Ausgangsspannung digitale I/O	5		27	V	
I_{DOut}	Ausgangsstrom digitale I/O			250	mA	Bei 27V

Warnung

Bitte verwenden Sie zum Anschluss der Baugruppe an die USB-Schnittstelle des PCs/ Laptops nur das im Lieferumfang enthaltene USB-Kabel.

Andere Kabel sind u. U. nicht geeignet!

2.3 Aufbau

2.3.1 Allgemeines

Die Baugruppen werden bereits mit einem LVDS-Empfänger-Aufsatzboard ausgeliefert.

Folgende Empfänger sind momentan verfügbar:

LVDS-Deserializer	Eigenschaften	Unterstützte LVDS-Serializer
CXB1458R	24 Bit Farbtiefe 4 Bit Control 7,6-42MHz Pixeltakt	CXB1457R
DS90CF364	18 Bit Farbtiefe 3 Bit Control 20-50MHz Pixeltakt	DS90C363
DS90UB926	24 Bit Farbtiefe 3 Bit Control 5-50MHz Pixeltakt	DS90UB925 DS90UH925 (ohne HDCP-Verschlüsselung) DS90UR905 DS90UR907
DS90UR124	21 Bit Farbtiefe 3 Bit Control 18-43MHz Pixeltakt	DS90UR241 (DS90C241)
DS90UR906	24 Bit Farbtiefe 3 Bit Control 5-50MHz Pixeltakt	DS90UR905 (DS90UR124 DS90C241)
INAP125R24 APIX	24 Bit Farbtiefe 3 Bit Control 6-32MHz Pixeltakt	INAP125T24
MAX9248	18 Bit Farbtiefe 9 Bit Control 18-42MHz Pixeltakt	MAX9247 (MAX9217)
MAX9260	24 Bit Farbtiefe 8 Bit Control 12,5-50MHz Pixeltakt	MAX9259

Abbildung 2-4 zeigt den schematischen Aufbau der Baugruppen in einem Blockschaltbild.

Bei basicCON 4120-Baugruppen bzw. USB 4120-Boards dient ein USB2.0-Controller als Interface zum USB-Bus. Dieser beinhaltet alle notwendigen Funktionsblöcke, die für eine Kommunikation mit dem Bus notwendig sind.

Das Board USB 4120, dass für den Einsatz in einem GÖPEL electronic USB-Rack konstruiert wurde (USB 1004, USB 1008, USB 1016), besitzt im Gegensatz zu basicCON 4120 zusätzliche Signalleitungen (GPIO) über den USB-Connector, welche zur Kommunikation mehrerer Karten untereinander verbunden werden können.
Für die Grundfunktionen werden diese Leitungen nicht benötigt.

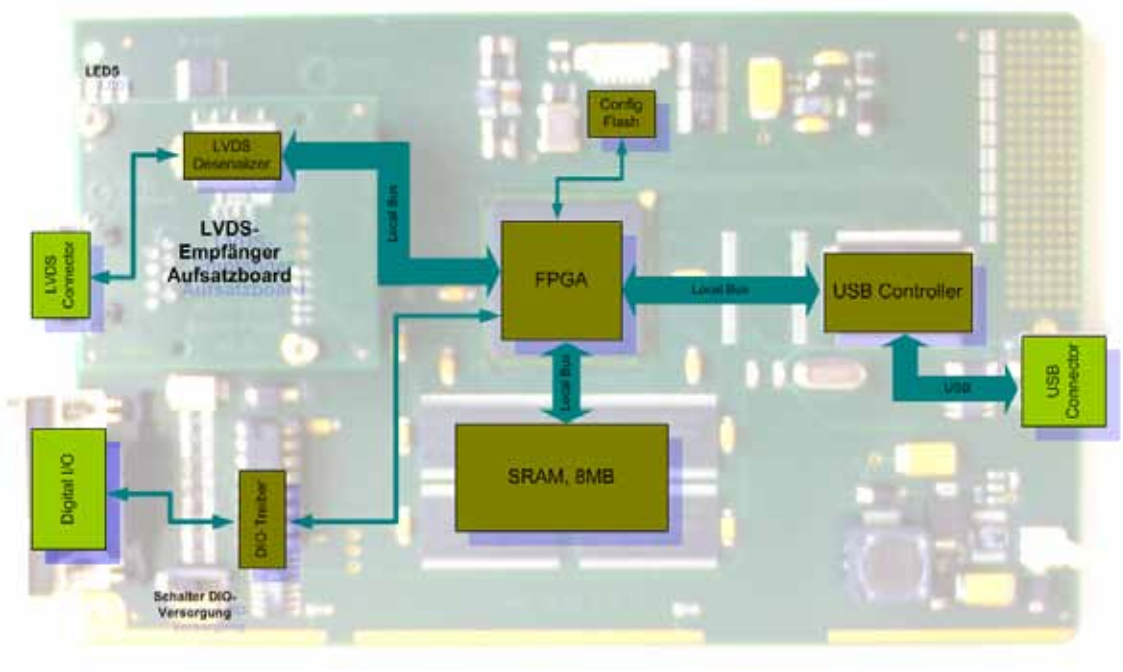


Abbildung 2-4: Blockdiagramm eines Signalverarbeitungsboards USB 4120

Alle Boards verfügen über einen HF-Steckverbinder zum Anschluss der LVDS-Eingangssignale, einen 9-poligen Dsub-Stecker für die digitale Input-/ Output-Schnittstelle (DIO) sowie vier Zustands-LEDs.

Das USB 4120-Board besitzt einen 132-poligen Backplane-Steckverbinder zum Anschluss in einem GÖPEL electronic USB-Rack.

Das basicCON 4120 verfügt über einen USB-B Anschluss sowie zur externen Stromversorgung über zwei 4mm-Bananenbuchsen und eine Buchse zum Anschluss eines Steckernetzteils.

2.3.2 Stromversorgung

Das Board USB 4120 wird über das USB-Rack versorgt, in dem es installiert wurde.

Die Stromversorgung des basicCON 4120 kann über die USB-Schnittstelle erfolgen, sofern diese für den benötigten Strom von ca. 500mA konfiguriert ist.
Im Betrieb an passiven USB-Hubs oder Laptops mit geringerer USB-Stromversorgung muss das basicCON 4120 extern über die beiden Buchsen für ext. Power Supply (rot = plus/ blau = minus) mit 7-25VDC (und ca. 200mA bei 12V) oder über die Buchse für das beigelegte Steckernetzteil mit dem Hohlstecker (2,1 x 5,5mm/ Polarität + innen) versorgt werden, siehe auch Abbildung 2-3.

2.3.3 Adressierung

Die Adressierung von USB 4120-Boards im GÖPEL electronic USB-Rack oder von basicCON 4120-Baugruppen erfolgt ausschließlich über deren Seriennummern (siehe auch Ansteuersoftware):
Die Baugruppe mit der kleinsten Seriennummer ist immer das Gerät Nummer 1.

Tip

Zur Erhöhung der Übersichtlichkeit empfehlen wir, mehrere USB 4120-Boards in aufsteigender Reihenfolge der Seriennummern im USB-Rack anzuordnen (bzw. mehrere basicCON 4120-Baugruppen in der gleichen Reihenfolge am PC/ Laptop anzuschließen).

2.3.4 LVDS-Schnittstelle

LVDS ist ein serieller Breitband-Übertragungsstandard, der im Automotive-Bereich zur Übertragung von Videodaten ständig an Bedeutung gewinnt.

Die Daten werden auf zwei Signalleitungen differenziell mit sehr geringen Spannungen und Strömen mit Frequenzen von über 1 GHz übertragen. Deshalb sind geeignete Steckverbinder und Kabel zur störungsfreien Übertragung besonders wichtig.

Für die Empfängeraufsätze DS90UR124, DS90UR906, DS90UB926, INAP125R24 APIX, MAX9248, MAX9260 und CXB1458R wird folgender Steckverbinder verwendet:

D4S20A-40ML5-Z von Rosenberger

Pinbelegung LVDS In:

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

Die Empfängeraufsätze DS90UR906 und DS90UB926 verfügen über einen zweiten LVDS-Anschluss (LVDS Out). Dieser gibt das LVDS-Eingangssignal wieder aus, nachdem es von einem Repeater aufbereitet wurde.
Dieses Signal kann z.B. für ein Display weiter verwendet werden.

Pinbelegung LVDS Out (nur für DS90UR906 und DS90UB926):

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

Der Empfängeraufsatz DS90CF364 arbeitet mit 4 LVDS-Kanälen (3x Daten, 1x Takt) und besitzt deshalb folgenden 8poligen Steckverbinder:

RJ45 – Buchse Kat6

Pinbelegung LVDS In:

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

Zum Anschluss sollten nur STP-Kabel mit 100Ω Impedanz und entsprechendem Steckverbinder gewählt werden.
Diese Kabel können auch über die GÖPEL electronic GmbH bezogen werden.

2.3.5 Digitale I/O Schnittstelle

Über den Fronsteckverbinder Digital I/O (DIO) steht eine digitale Schnittstelle zur Verfügung. Mit Hilfe des auf dem entsprechenden Board befindlichen Schalters bestimmen Sie, ob die Versorgung des Ausgangstreibers über die DIO (5 bis 27V) oder intern über 5V erfolgen soll.

Bei Auslieferung ist der Schalter in der Stellung für die Versorgung über die DIO (siehe folgende Abbildung).

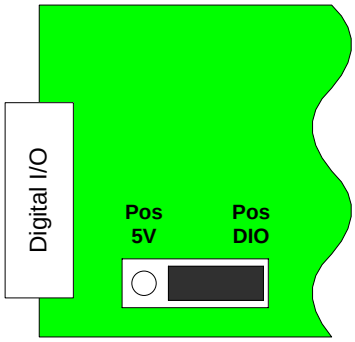


Abbildung 2-5: Schalter DIO-Versorgung

Mit Hilfe der DIO kann das Board auch ohne PC oder Laptop betrieben werden, nachdem die Konfiguration über USB erfolgt ist. In diesem Fall ist dafür zu sorgen, dass die Stromversorgung weiterhin gewährleistet ist (siehe Stromversorgung).

Mit der DIO können Bildabgriffe ausgelöst und automatisch onboard mit dem Referenzbild verglichen werden. Diese Schnittstelle bietet jedoch keinerlei Möglichkeiten der Konfiguration oder der spezifischen Bildauswertung.

Ein Vorteil der DIO ist die Möglichkeit der Echtzeittriggerung des Framegrabbers. Nur durch Verwendung der DIO kann gewährleistet werden, dass im Videostream ein ganz bestimmtes Bild abgegriffen wird.

Die Tabelle zeigt die Pinbelegung der Digitalen I/O Schnittstelle:

Pin Nr.	Signalname	Erklärung
1	FG_Ready_Out	Ausgang Framegrabber bereit für DIO Steuerung
2	Comp_Fail_Out	Ausgang Bildvergleicher hat mindestens einen Fehler erkannt
3	Ext_Trigger_In	Eingang externer Trigger startet den Bildabgriff und anschließenden Vergleich mit dem Referenzbild
4	Ext_Res1_In	Eingang Reserve 1
5	DIO_VCC	DIO Versorgung (+5..+27V) bei Schalterstellung DIO (Abbildung 2-5)
6	FG_Pass_Out	Ausgang Framegabber Pass zeigt an, dass die Triggerung abgeschlossen wurde und das Bild fehlerfrei verglichen wurde
7	FG_Lock_Out	Ausgang Framegabber Lock zeigt an, dass das LVDS Signal gelockt ist
8	Ext_Res0_In	Eingang Reserve 0
9	DIO_GND	DIO Ground Pin

2.3.6 LED Anzeige

Die auf der Frontplatte angeordneten Leuchtdioden geben Auskunft über den momentanen Betriebszustand des Framegrabbers.

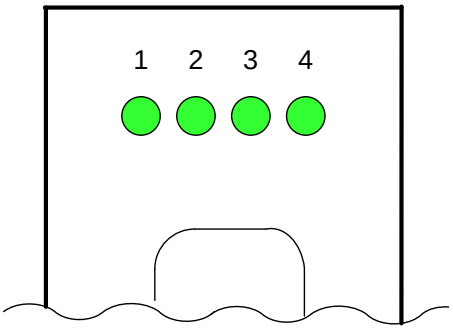


Abbildung 2-6::
LED Anzeige Frontansicht

Die Anzeigezustände dieser LEDs werden in folgender Tabelle erläutert:

LED	Bemerkung
LED 1 leuchtet	Pixelfehler beim Onboard Bildvergleich
LED 2 leuchtet	Fehler beim Bildscan (Sync- oder Auflösungsfehler)
LED 3 leuchtet	Externer Trigger Modus an
LED 4 leuchtet	LVDS Lock aktiv

2.3.7 Empfänger-Aufsätze

Durch verschiedene Aufsatzboards kann der LVDS Framegrabber auf die Signalquelle abgestimmt werden.

Entscheidend ist auch die beliebige Zuweisung der LVDS-Bits zu den jeweiligen Farb- oder Kontroll-Bits des Bildes.
Dazu muss der Framegrabber passend zur Belegung der LVDS-Quelle konfiguriert werden.

Die folgenden Abbildungen zeigen die Belegung des jeweiligen LVDS-Empfängers (Deserializers), welche bei der Konfiguration bzw. Initialisierung des Framegrabbers beachtet werden muss.

DS90UR124

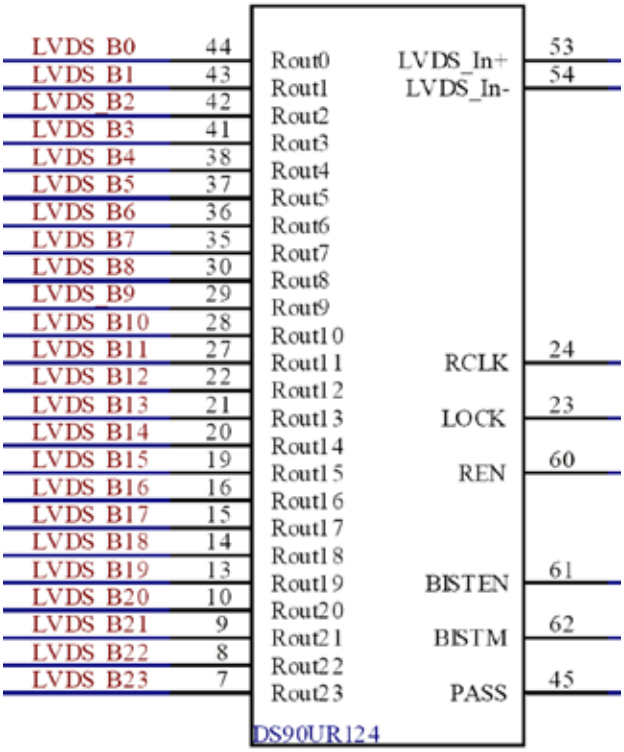


Abbildung 2-7:: Pinbelegung DS90UR124

Dieses Aufsatzboard verfügt über einen Schalter/ Jumper, mit dem das Board entweder kompatibel zu Serializer DS90UR241 oder DS90C241 gemacht werden kann.

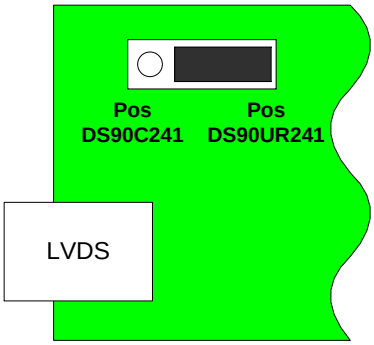


Abbildung 2-8:: Serializer-Umschaltung

DS90CF364

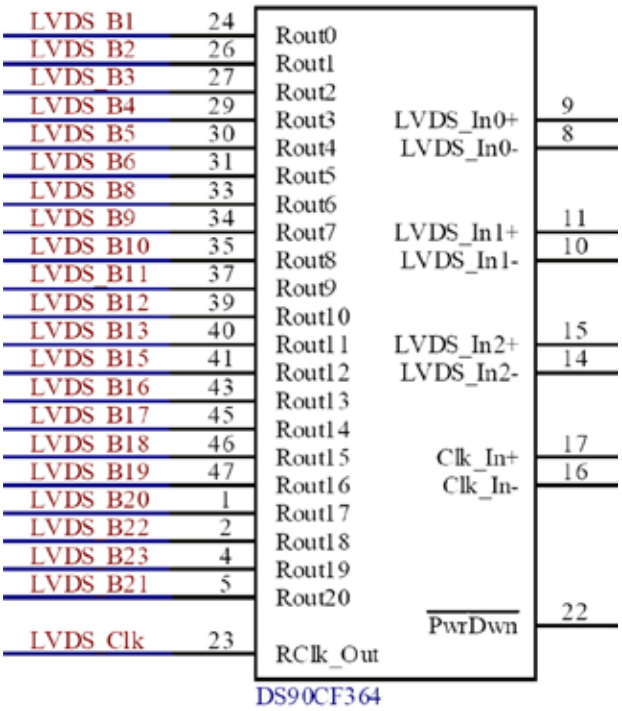


Abbildung 2-9::
Pinbelegung DS90CF364

DS90UR906

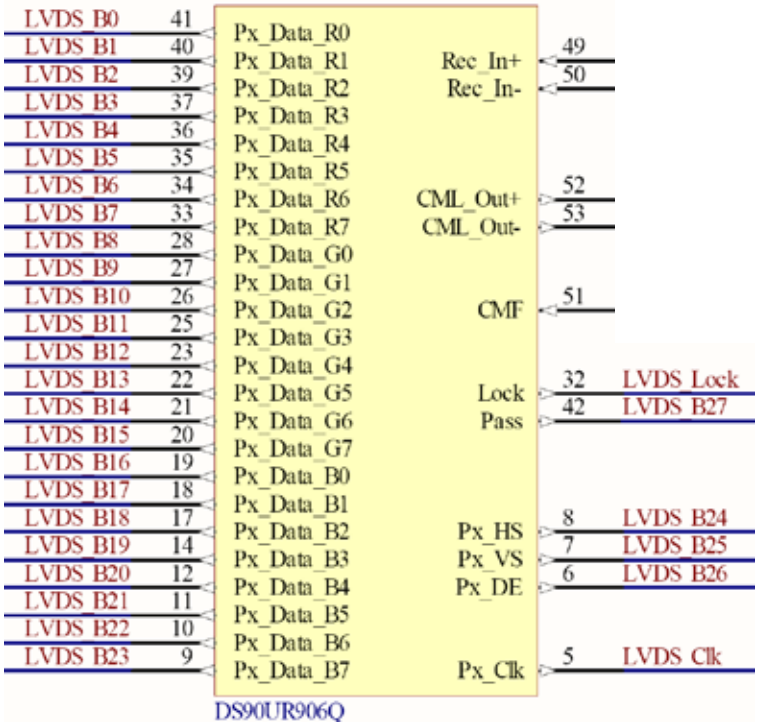


Abbildung 2-10: Pinbelegung DS90UR906

Bei diesem Aufsatzboard muss der Deserializer über ein I2C-Interface konfiguriert werden.
Die Konfiguration übernimmt der Framegrabber. Sie kann jederzeit vom User geändert werden.

Dieses Aufsatzboard kann über die Konfiguration sowohl zu den Serializern DS90UR905, DS90UR241 als auch DS90C241 kompatibel gemacht werden.

Für die Kompatibilität zu den Serializern DS90UR241 und DS90C241 muss allerdings die Belegung der Serializer dem Datenblatt (DS90UR906) entsprechen.

Außerdem verfügt dieses Aufsatzboard über einen Repeater, der das LVDS-Eingangssignal aufbereitet und auf einem zweiten LVDS-Steckverbinder wieder ausgibt.

DS90UB926

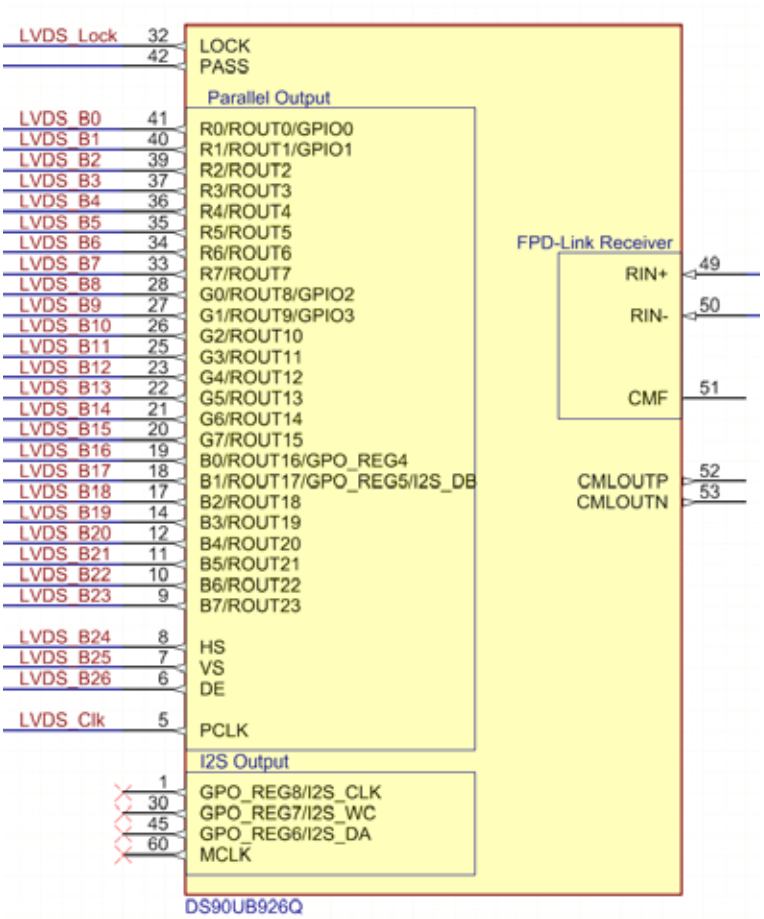


Abbildung 2-11: Pinbelegung DS90UB926

Bei diesem Aufsatzboard muss der Deserializer über ein I2C-Interface konfiguriert werden.
Die Konfiguration übernimmt der Framegrabber. Sie kann jederzeit vom User geändert werden.

Dieses Aufsatzboard ist durch verschiedene Konfigurationen sowohl zu den Serializern DS90UB925 und DS90UH926 als auch DS90UR905 und DS90UR907 kompatibel.
Bei der Kompatibilität zum DS90UH926 ist allerdings zu beachten, dass der DS90UB926 keine HDCP Entschlüsselung unterstützt.

Außerdem verfügt dieses Aufsatzboard über einen Repeater, der das LVDS-Eingangssignal aufbereitet und auf einem zweiten LVDS-Steckverbinder wieder ausgibt.

INAP125R24 APIX

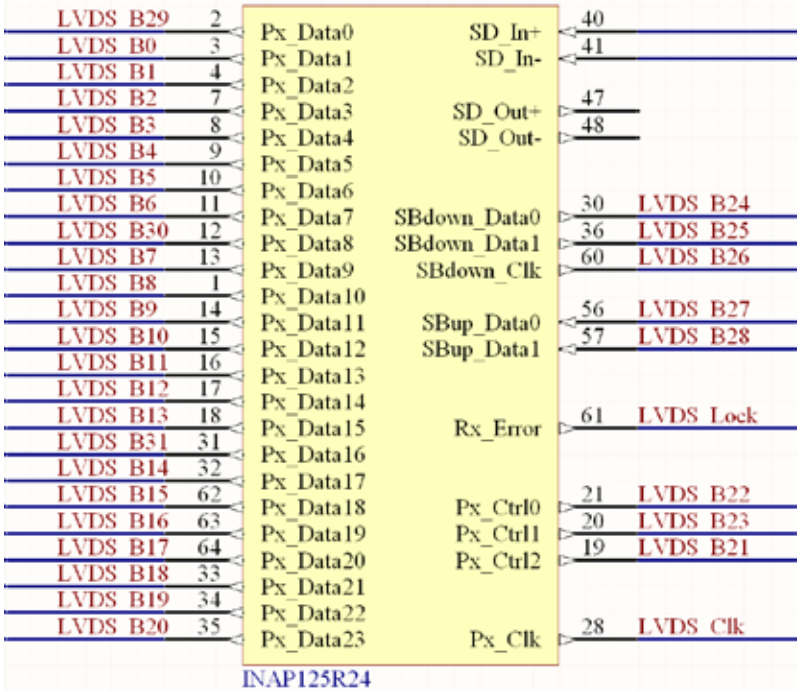


Abbildung 2-12: Pinbelegung INAP125R24 APIX

Bei diesem Aufsatzboard muss der Deserializer über ein serielles Interface konfiguriert werden.

Die Konfiguration übernimmt der Framegrabber. Sie kann jederzeit vom User geändert werden.

Der Upstream und die Sidebandkanäle werden nicht verwendet.

MAX9248

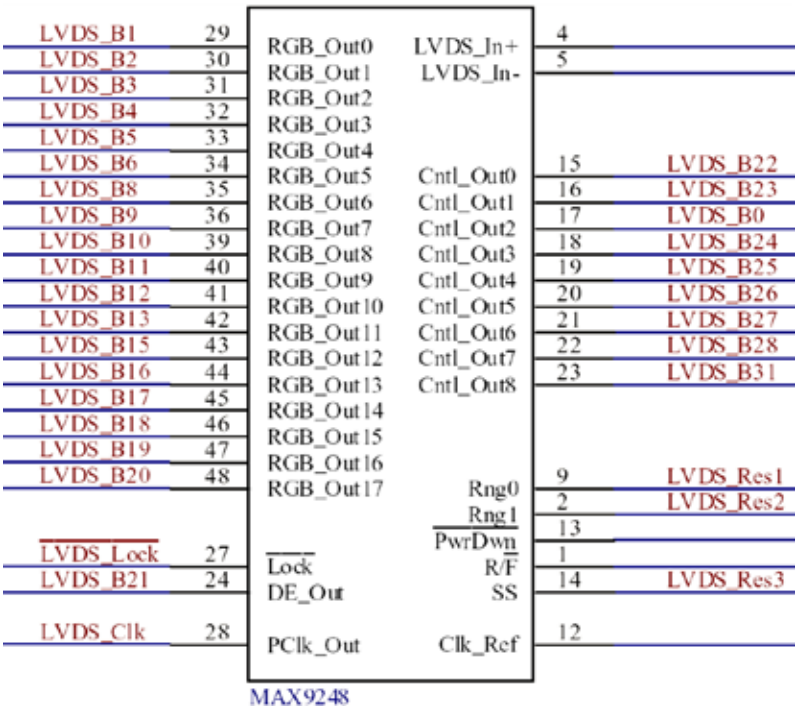


Abbildung 2-13: Pinbelegung MAX9248

Für dieses Aufsatzboard muss die Quarzfrequenz auf den Sender abgestimmt werden. Bitte geben Sie daher bei der Bestellung die Pixelfrequenz des Senders an.

MAX9260

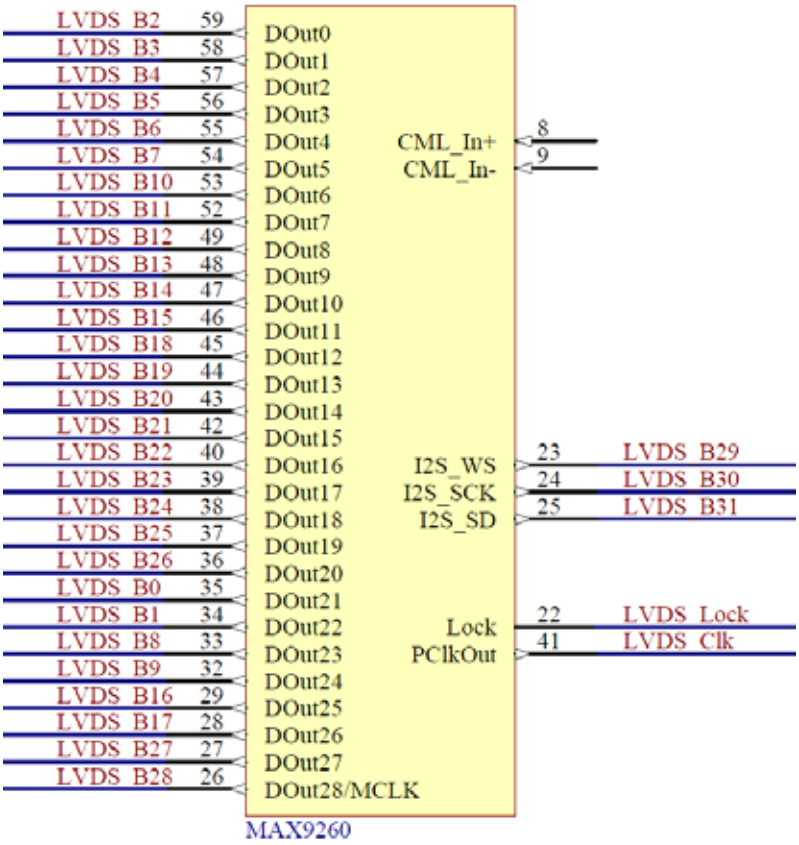


Abbildung 2-14: Pinbelegung MAX9260

CXB1458R

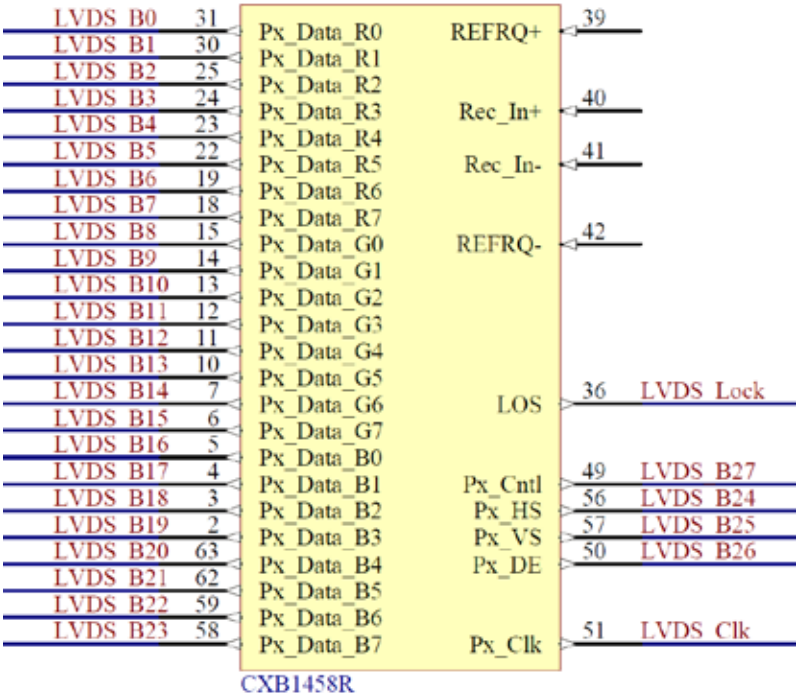


Abbildung 2-15: Pinbelegung CXB1458R

2.4 Lieferhinweise

USB 4120 Boards/ basicCON 4120 Baugruppen sind in folgenden Varianten und mit folgendem Zubehör erhältlich:

- USB 4120.10 mit Aufsatzboard DS90UR124
- USB 4120.20 mit Aufsatzboard DS90CF364
- USB 4120.30 mit Aufsatzboard DS90UR906
- USB 4120.40 mit Aufsatzboard INAP125R24 APIX
- USB 4120.50 mit Aufsatzboard MAX9248 (bitte bei Bestellung die Pixelfrequenz angeben)
- USB 4120.60 mit Aufsatzboard MAX9260
- USB 4120.70 mit Aufsatzboard CXB1458R
- USB 4120.80 mit Aufsatzboard DS90UB926

- basicCON 4120.10 mit Aufsatzboard DS90UR124
- basicCON 4120.20 mit Aufsatzboard DS90CF364
- basicCON 4120.30 mit Aufsatzboard DS90UR906
- basicCON 4120.40 mit Aufsatzboard INAP125R24 APIX
- basicCON 4120.50 mit Aufsatzboard MAX9248 (bitte bei Bestellung die Pixelfrequenz angeben)
- basicCON 4120.60 mit Aufsatzboard MAX9260
- basicCON 4120.70 mit Aufsatzboard CXB1458R
- basicCON 4120.80 mit Aufsatzboard DS90UB926

- ST 4120.10 Aufsatzboard DS90UR124
- ST 4120.20 Aufsatzboard DS90CF364
- ST 4120.30 Aufsatzboard DS90UR906
- ST 4120.40 Aufsatzboard INAP125R24 APIX
- ST 4120.50 Aufsatzboard MAX9248 (bitte bei Bestellung die Pixelfrequenz angeben)
- ST 4120.60 Aufsatzboard MAX9260
- ST 4120.70 Aufsatzboard CXB1458R
- ST 4120.80 Aufsatzboard DS90UB926

3 Ansteuersoftware

Zur Einbindung der USB 4120/ basicCON 4120-Hardware in eigene Applikationen existieren zwei Möglichkeiten:

- Programmieren über G-API
- Programmieren über DLL-Funktionen

3.1 Programmieren über G-API

Das bevorzugte User Interface für diese GÖPEL Hardware ist die G-API (GÖPEL-API).

Sie finden alle benötigten Informationen im Ordner *G-API* der mitgelieferten CD.

3.2 Programmieren über DLL-Funktionen

Die Programmierung über DLL-Funktionen ist nur für Projekte erforderlich, bei denen nicht mit der GÖPEL G-API gearbeitet wird.

Mit den nachfolgend beschriebenen Funktionsaufrufen können USB 4120-Boards oder basicCON 4120-Geräte direkt aus diversen Hochsprachen angesprochen werden (VisualC++, CVI).

Der in der folgenden Funktionsbeschreibung verwendete Begriff GUSB_Platform ist der Name eines USB-Treibers der GÖPEL electronic GmbH.
Im Folgenden gelten alle Aussagen zu USB 4120-Boards auch für basicCON 4120-Geräte.

Informationen zu den Strukturen, Funktionen und Error-Codes enthält das C Header-File *GUSB_Platform.h* der mitgelieferten CD.

Windows Device Treiber

Die für die Programmierung unter Verwendung des Windows Device Treibers nutzbaren DLL-Funktionen sind in den folgenden Abschnitten beschrieben:

- Driver_Info
- DLL_Info
- Xilinx_Version
- Write_COMMAND
- Read_COMMAND
- IdentInfo
- Reset
- Get HardwareInfo
- SyncInfo
- InitInfo
- Init
- Set ClockSource
- Set BitMask
- External TriggerMode_Start
- External TriggerMode_Stop
- External TriggerMode_GetState
- Capture
- Compare
- Capture ToBuffer
- Capture ToFile
- Read CapturedFrame ToBuffer
- Read CapturedFrame ToFile
- Load Reference FromBuffer
- Load Reference FromFile
- Read Reference ToBuffer
- Read Reference ToFile
- Deserializer Configuration Vector

3.2.1 Driver_Info

Die Funktion GUSB_Platform_Driver_Info dient zur Status-Abfrage des Hardware-Treibers und zur internen Initialisierung der erforderlichen Handles.

Diese Funktion MUSS einmalig vor dem Aufruf aller anderen Funktionen des GUSB_Platform Treibers ausgeführt werden.

Format:

```
int GUSB_Platform_Driver_Info(GUSB_Platform_DriverInfo *pVersion
                              unsigned int             LengthInByte)
```

Parameter:

pVersion

Zeiger auf eine Datenstruktur
Zur Struktur siehe das File *GUSB_Platform.h* auf der mitgelieferten CD

LengthInByte

Größe des Speicherbereichs, auf den pVersion zeigt, in Bytes

Beschreibung:

Die Funktion GUSB_Platform_Driver_Info gibt Informationen über den Status des Hardware-Treibers zurück.

Dazu muss der Funktion die Adresse des Zeigers pVersion übergeben werden. Mit Hilfe des Parameters LengthInByte prüft die Funktion intern den korrekt initialisierten Anwenderspeicher.

Die Funktion füllt die Struktur, auf die pVersion zeigt, mit Angaben zur Treiberversion, der Anzahl aller sich im System befindenden USB Controller (die von diesem Treiber unterstützt werden), und Informationen darüber, wie z.B. die Seriennummer(n).

Die Bereitstellung der Hardwareinformationen und die Initialisierung der zugehörigen Handles sind für die weitere Nutzung der USB-Hardware zwingend erforderlich.

3.2.2 DLL_Info

Die Funktion GUSB_Platform_DLL_Info dient zur Abfrage von Informationen über die DLL.

Format:

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

Parameter

DLLinformation

Zeiger auf eine Datenstruktur
Zur Struktur siehe das File *GUSB_Platform.h* auf der mitgelieferten CD

Beschreibung:

Die Funktion GUSB_Platform_DLL_Info gibt die Struktur DLLinfo zurück. Der erste Integerwert enthält die Versionsnummer der *GUSB_Platform.dll*.

Beispiel:

Die Versionsnummer **`1.23`** wird als Wert **`123`** zurück gegeben, Version **`1.60`** als Wert **`160`**.

3.2.3 Xilinx_ Version

Die Funktion GUSB_Platform_Xilinx_Version ermöglicht das Auslesen der geladenen XILINX-Firmwareversion.

Format:

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

Parameter:

DeviceName

Typ des adressierten Gerätes (Nummer, die in *GUSB_Platform_def.h* deklariert ist, für USB 4120 = 19)

DeviceNumber

Nummer des adressierten Gerätes. Wenn mehrere Geräte gleichen Typs angeschlossen sind, erfolgt die Nummerierung in aufsteigender Reihenfolge der Seriennummern (das Gerät mit der NIEDRIGSTEN Seriennummer hat immer DeviceNumber 1).

Version

XILINX Softwareversion

Beschreibung:

Mit der Funktion GUSB_Platform_Xilinx_Version kann die Versionsnummer der im FPGA geladenen Software ausgelesen werden.

Beispiel:

Die Versionsnummer **2.34** wird als Wert **234** zurück gegeben, Version **2.60** als Wert **260**.

3.2.4 Write_COMMAND

Die Funktion GUSB_Platform_Write_COMMAND dient zum Senden eines Configurations-Befehls zum USB Controller.

Format:

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

Parameter:

DeviceName

Typ des adressierten Gerätes (Nummer, die in *GUSB_Platform_def.h* deklariert ist, für USB 4120 = 19)

DeviceNumber

Nummer des adressierten Gerätes. Wenn mehrere Geräte gleichen Typs angeschlossen sind, erfolgt die Nummerierung in aufsteigender Reihenfolge der Seriennummern (das Gerät mit der NIEDRIGSTEN Seriennummer hat immer DeviceNumber 1).

pWrite

Zeiger auf den Bereich für Schreibdaten (z. Zt. max. 512 Byte pro Befehl, siehe auch Steuerbefehle USB Controller)

DataLength

Anzugebende Byteanzahl der in den Speicherbereich, auf den pWrite zeigt, zu schreibenden Daten

Beschreibung:

Die Funktion GUSB_Platform_Write_COMMAND sendet einen Befehl zum USB Controller.
Die allgemeine Struktur ist im Abschnitt Steuerbefehle USB Controller beschrieben.

3.2.5 Read_COMMAND

Die Funktion GUSB_Platform_Read_COMMAND dient zum Lesen einer Antwort vom USB Controller.

Format:

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

Parameter:

DeviceName

Typ des adressierten Gerätes (Nummer, die in *GUSB_Platform_def.h* deklariert ist, für USB 4120 = 19)

DeviceNumber

Nummer des adressierten Gerätes. Wenn mehrere Geräte gleichen Typs angeschlossen sind, erfolgt die Nummerierung in aufsteigender Reihenfolge der Seriennummern (das Gerät mit der NIEDRIGSTEN Seriennummer hat immer DeviceNumber 1).

pRead

Zeiger auf den Lesepuffer
(Nach erfolgreicher Funktionsausführung befinden sich im Lesepuffer die Daten, bestehend aus Antwortkopf und Antwortbytes, z. Zt. max. 512 Byte pro Antwort, siehe auch Steuerbefehle USB Controller.)

DataLength

Vor Funktionsaufruf: Anzugebende Größe des Lesepuffers in Bytes

Nach Funktionsausführung: Anzahl der tatsächlich gelesenen Bytes

Beschreibung:

Die Funktion GUSB_Platform_Read_COMMAND liest die älteste vom USB Controller geschriebene Antwort zurück.

Werden mehrere Antworten vom USB Controller bereit gestellt, werden maximal zwei dieser Antworten in den Puffer des USB Controllers geschrieben.
Weitere ggf. bereit gestellte Antworten gehen verloren!

3.2.6 IdentInfo

Die Funktion GUSB_Platform_4120_IdentInfo liefert allgemeine Informationen für das mit DeviceNumber indizierte USB 4120-Board.

Format:

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

Parameter:

DeviceNumber

Nummer des adressierten Gerätes. Wenn mehrere Geräte gleichen Typs angeschlossen sind, erfolgt die Nummerierung in aufsteigender Reihenfolge der Seriennummern (das Gerät mit der NIEDRIGSTEN Seriennummer hat immer DeviceNumber 1).

IdentData

Zeiger auf eine Datenstruktur

Zur Struktur siehe das File *GUSB_Platform.h* auf der mitgelieferten CD

LengthInByte

Größe des Speicherbereichs, auf den IdentData zeigt, in Bytes

Nach Funktionsausführung: Anzahl der tatsächlich gelesenen Bytes

State

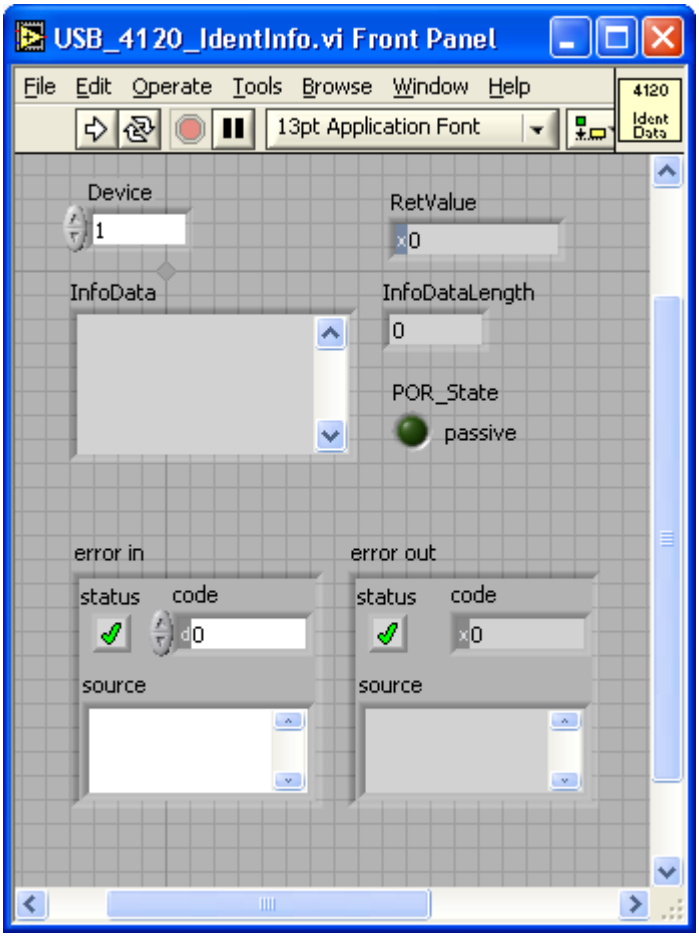
Bitselektive Statusinformationen des Boards

Bit 0:	PowerOnReset 0 -> passiv 1 -> aktiv
Bits 1..31:	not used

Beschreibung:

Die Funktion liefert allgemeine Informationen und Konfigurationszustände des Gerätes.

Zugehöriges LabVIEW – VI:



3.2.7 Reset

Die Funktion GUSB_Platform_4120_Reset setzt alle FPGA-Register des mit DeviceNumber indizierten USB 4120-Boards auf ihre Defaultwerte zurück.

Format:

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

Parameter:

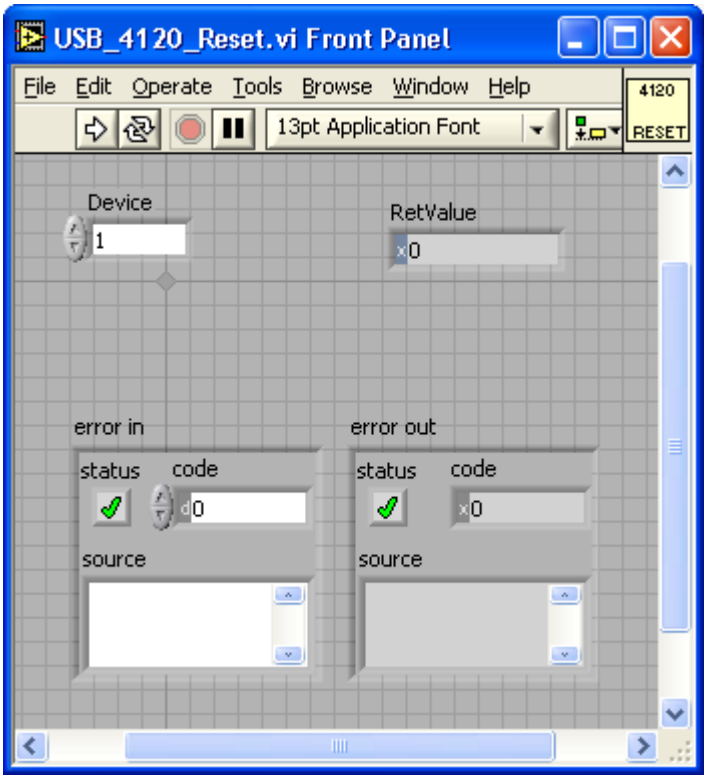
DeviceNumber

Nummer des adressierten Gerätes. Wenn mehrere Geräte gleichen Typs angeschlossen sind, erfolgt die Nummerierung in aufsteigender Reihenfolge der Seriennummern (das Gerät mit der NIEDRIGSTEN Seriennummer hat immer DeviceNumber 1).

Beschreibung:

Alle Registerwerte des FPGAs werden zurück gesetzt.

Zugehöriges LabVIEW – VI:



3.2.8 Get HardwareInfo

Die Funktion GUSB_Platform_4120_GetHardwareInfo liefert allgemeine Informationen und Zustände für das mit DeviceNumber indizierte USB 4120-Board.

Format:

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

Parameter:

DeviceNumber

Nummer des adressierten Gerätes. Wenn mehrere Geräte gleichen Typs angeschlossen sind, erfolgt die Nummerierung in aufsteigender Reihenfolge der Seriennummern (das Gerät mit der NIEDRIGSTEN Seriennummer hat immer DeviceNumber 1).

VHDL_Version

Zeigt die VHDL-Design Version des FPGAs an
(Bsp.: Wert 120 -> Version 1.20)

HW_Version

Zeigt die Version des Platinenlayouts an
(Bsp.: Wert 11 -> Version 1.1)

Deserializer

Zeigt den Typ des Empfänger-ICs auf dem Aufsatzboard an
(siehe Datei *GUSB_Platform.h*)

LockState

Zeigt an, ob ein LVDS-Signal anliegt und der Deserializer auf dieses Signal gelockt ist
(siehe Datei *GUSB_Platform.h*)

ConfigError

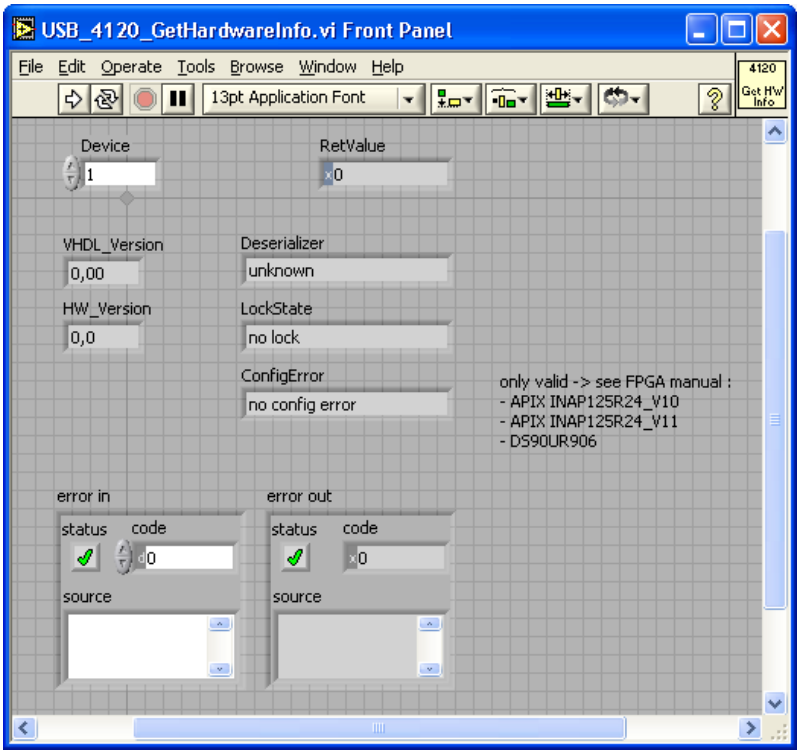
Zeigt an, ob bei der Konfiguration des Deserializers (GUSB_Platform_4120_DeserializerConfigurationVector()) ein Fehler aufgetreten ist
(nur gültig bei den Deserializer-Typen APIX INAP125R24_V10, APIX INAP125R24_V11 und DS90UR906)
(siehe Datei *GUSB_Platform.h*)

Beschreibung:

Momentan sind folgende Deserializertypen verfügbar:

- DS90UR124
- DS90CF364
- INAP125R24_V10
- INAP125R24_V11
- MAX9248
- DS90UR906
- DS90UB926
- CXB1458R
- MAX9260

Zugehöriges LabVIEW – VI:



3.2.9 SyncInfo

Die Funktion GUSB_Platform_4120_SyncInfo liefert Informationen über die Synchronisierungs-Eigenschaften des LVDS-Datenstromes des mit DeviceNumber indizierten USB 4120-Boards.

Format:

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

Parameter:

DeviceNumber

Nummer des adressierten Gerätes. Wenn mehrere Geräte gleichen Typs angeschlossen sind, erfolgt die Nummerierung in aufsteigender Reihenfolge der Seriennummern (das Gerät mit der NIEDRIGSTEN Seriennummer hat immer DeviceNumber 1).

PixelClock

Zeigt die Frequenz des Taktes in kHz an, mit dem die Pixel übertragen werden (Genauigkeit mindestens 50kHz)

DataEnableNumberOfColumn

Zeigt die Anzahl der Spalten an, bei denen das DataEnable-Signal aktiv ist (entspricht horizontaler Auflösung)

Die Definition des aktiven DataEnable-Signals wird in GUSB_Platform_4120_Init() (siehe Init) mit dem Parameter SignalLevel.DataControls gesetzt (low-aktiv oder high-aktiv).

(nur gültig, wenn in GUSB_Platform_4120_Init() der Parameter RoutingControls.DataEnable ungleich K_4120_LVDS_BIT_DISABLE gesetzt wurde)

DataEnableNumberOfLines

Zeigt die Anzahl der Zeilen an, bei denen das DataEnable Signal aktiv ist (entspricht vertikaler Auflösung).

Die Definition des aktiven DataEnable-Signals wird in GUSB_Platform_4120_Init() (siehe Init) mit dem Parameter SignalLevel.DataControls gesetzt (low-aktiv oder high-aktiv).

(nur gültig, wenn in GUSB_Platform_4120_Init() der Parameter RoutingControls.DataEnable ungleich K_4120_LVDS_BIT_DISABLE gesetzt wurde.)

HSyncWidth

Zeigt die Anzahl der Takte an, bei denen das horizontale Synchronsignal aktiv ist

Die Definition des aktiven Hsync-Signals wird in GUSB_Platform_4120_Init() (siehe Init) mit dem Parameter SignalLevel.HSync gesetzt (low-aktiv oder high-aktiv).

VSyncWidth

Zeigt die Anzahl der Takte an, bei denen das vertikale Synchronsignal aktiv ist
Die Definition des aktiven Vsync-Signals wird in GUSB_Platform_4120_Init() (siehe Init) mit dem Parameter SignalLevel.VSync gesetzt (low-aktiv oder high-aktiv).

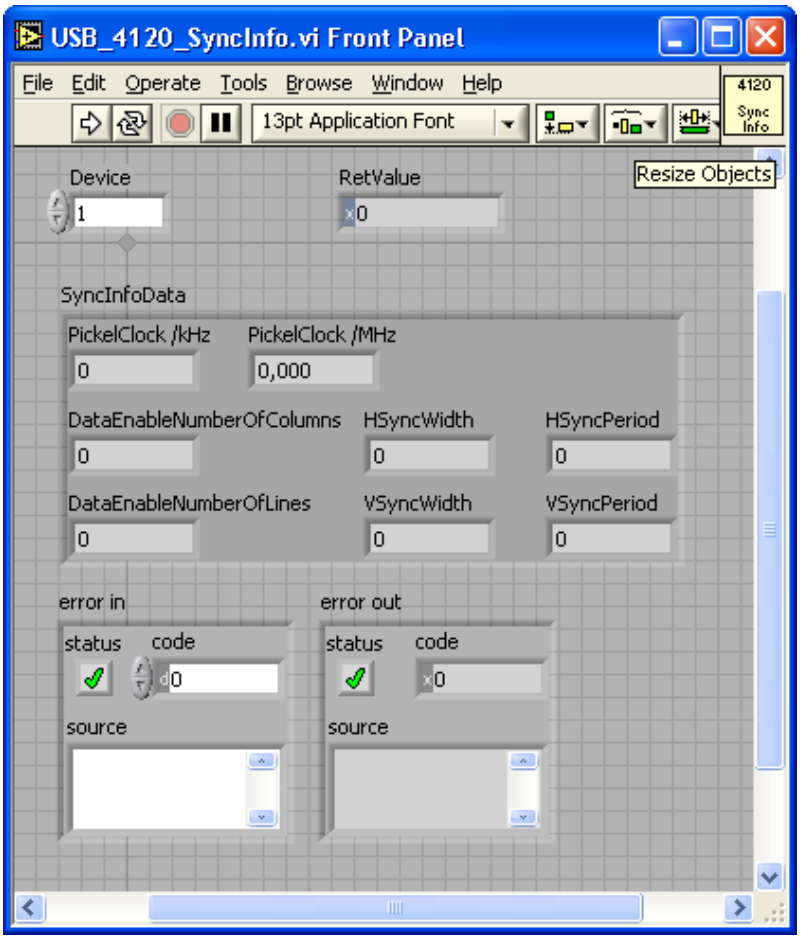
HSyncPeriod

Zeigt die Anzahl der Takte zwischen zwei gleichen horizontalen Synchronflanken an

VSyncPeriod

Zeigt die Anzahl der Takte zwischen zwei gleichen vertikalen Synchronflanken an

Zugehöriges LabVIEW – VI:



3.2.10 InitInfo

Die Funktion GUSB_Platform_4120_InitInfo liefert Informationen über den Initialisierungszustand des mit DeviceNumber indizierten USB 4120-Boards.

Format:

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

Parameter:

DeviceNumber

Nummer des adressierten Gerätes. Wenn mehrere Geräte gleichen Typs angeschlossen sind, erfolgt die Nummerierung in aufsteigender Reihenfolge der Seriennummern (das Gerät mit der NIEDRIGSTEN Seriennummer hat immer DeviceNumber 1).

InitResolution.NumberOfColumns

Zeigt die eingestellte Anzahl der Spalten des Bildes an (horizontale Auflösung)

InitResolution.NumberOfLines

Zeigt die eingestellte Anzahl der Zeilen des Bildes an (vertikale Auflösung)

SyncWidth.NumberOfColumnsSync

Zeigt die eingestellte Anzahl der Leerspalten in der horizontalen Austastlücke nach dem aktiven horizontalen Synchronsignal an

(nur gültig, wenn in GUSB_Platform_4120_Init() (siehe Init) der Parameter RoutingControls.DataEnable gleich K_4120_LVDS_BIT_DISABLE gesetzt wurde)

SyncWidth.NumberOfLinesSync

Zeigt die eingestellte Anzahl der Leerzeilen in der vertikalen Austastlücke nach dem aktiven vertikalen Synchronsignal an

(nur gültig, wenn in GUSB_Platform_4120_Init() (siehe Init) der Parameter RoutingControls.DataEnable gleich K_4120_LVDS_BIT_DISABLE gesetzt wurde)

SignalLevel.HSync

Zeigt die eingestellte Polarität des horizontalen Synchronsignals an

0 -> low-aktiv

1 -> high-aktiv

SignalLevel.VSync

Zeigt die eingestellte Polarität des vertikalen Synchronsignals an

0 -> low-aktiv
1 -> high-aktiv

SignalLevel.DataEnable

Zeigt die eingestellte Polarität des DataEnable-Signals an

0 -> low-aktiv
1 -> high-aktiv

EdgesSelection.Clock

Zeigt die eingestellte Flanke des Taktes an,
bei der die Daten eingelesen werden

0 -> falling edge
1 -> rising edge

RoutingColors.B0...RoutingColors.B7

Zeigt die eingestellte Quelle (LVDS Bit-Nummer)
der Farbbits Blau 0..7 an

RoutingColors.G0...RoutingColors.G7

Zeigt die eingestellte Quelle (LVDS Bit-Nummer)
der Farbbits Grün 0..7 an

RoutingColors.R0...RoutingColors.R7

Zeigt die eingestellte Quelle (LVDS Bit-Nummer)
der Farbbits Rot 0..7 an

RoutingControls.VSync

Zeigt die eingestellte Quelle (LVDS Bit-Nummer)
des vertikalen Synchron-Signals an

RoutingControls.HSync

Zeigt die eingestellte Quelle (LVDS Bit-Nummer)
des horizontalen Synchron-Signals an

RoutingControls.DataEnable

Zeigt die eingestellte Quelle (LVDS Bit-Nummer)
des DataEnable-Signals an

RoutingControls.C3...RoutingControls.C7

Zeigt die eingestellte Quelle (LVDS Bit-Nummer)
der Controlbits 3..7 (optionale Steuerbits) an

Beschreibung:

Die Definitionen der verwendeten Strukturen befinden sich in der Datei *GUSB_Platform.h* auf der mitgelieferten CD.

Die Elemente der Strukturen RoutingColors und RoutingControl zeigen die aktuelle Zuordnung der LVDS-Bits an:

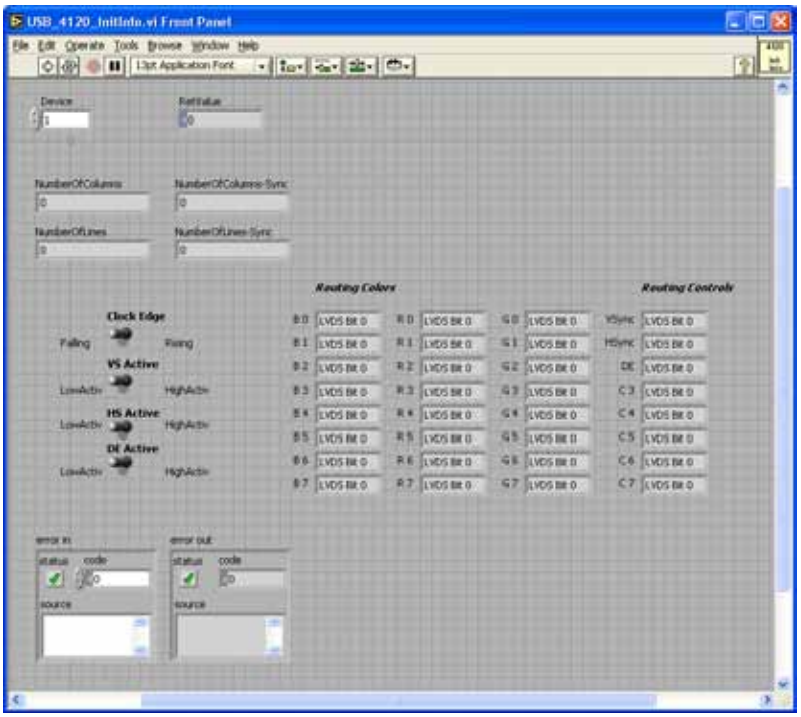
0 -> LVDS-Bit 0
1 -> LVDS-Bit 1

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

254 -> not used
255 -> disable

Die Erklärung zur Zuordnung der LVDS-Bits finden Sie in der Hardwarebeschreibung, Abschnitt Empfänger-Aufsätze.

Zugehöriges LabVIEW – VI:



3.2.11 Init

Die Funktion GUSB_Platform_4120_Init löst ein RESET aus und konfiguriert das mit DeviceNumber indizierte USB 4120-Board zur Abstimmung mit der Bildquelle.

Format:

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

Parameter:

DeviceNumber

Nummer des adressierten Gerätes. Wenn mehrere Geräte gleichen Typs angeschlossen sind, erfolgt die Nummerierung in aufsteigender Reihenfolge der Seriennummern (das Gerät mit der NIEDRIGSTEN Seriennummer hat immer DeviceNumber 1).

InitResolution.NumberOfColumns

Setzt die Anzahl der Spalten des Bildes (horizontale Auflösung)

InitResolution.NumberOfLines

Setzt die Anzahl der Zeilen des Bildes (vertikale Auflösung)

SyncWidth.NumberOfColumnsSync

Setzt die Anzahl der Leerspalten in der horizontalen Austastlücke nach dem aktiven horizontalen Synchronsignal
(nur gültig, wenn der Parameter RoutingControls.DataEnable gleich K_4120_LVDS_BIT_DISABLE gesetzt wurde)

SyncWidth.NumberOfLinesSync

Setzt die Anzahl der Leerzeilen in der vertikalen Austastlücke nach dem aktiven vertikalen Synchronsignal
(nur gültig, wenn der Parameter RoutingControls.DataEnable gleich K_4120_LVDS_BIT_DISABLE gesetzt wurde)

SignalLevel.HSync

Setzt die Polarität des horizontalen Synchronsignals
0 -> low-aktiv
1 -> high-aktiv

SignalLevel.VSync

Setzt die Polarität des vertikalen Synchronsignals
0 -> low-aktiv
1 -> high-aktiv

SignalLevel.DataEnable

Setzt die Polarität des DataEnable-Signals
0 -> low-aktiv
1 -> high-aktiv

EdgeSelection.Clock

Setzt die Flanke des Taktes, bei der die Daten eingelesen werden
0 -> falling edge
1 -> rising edge

RoutingColors.B0...RoutingColors.B7

Setzt die Quelle (LVDS Bit-Nummer) der Farbbits Blau 0...7

RoutingColors.G0...RoutingColors.G7

Setzt die Quelle (LVDS Bit-Nummer) der Farbbits Grün 0...7

RoutingColors.R0...RoutingColors.R7

Setzt die Quelle (LVDS Bit-Nummer) der Farbbits Rot 0...7

RoutingControls.VSync

Setzt die Quelle (LVDS Bit-Nummer) des vertikalen Synchronsignals

RoutingControls.HSync

Setzt die Quelle (LVDS Bit-Nummer) des horizontalen Synchronsignals

RoutingControls.DataEnable

Setzt die Quelle (LVDS Bit-Nummer) des DataEnable-Signals

RoutingControls.C3...RoutingControls.C7

Setzt die Quelle (LVDS Bit-Nummer) der Controlbits 3...7
(optionale Steuerbits)

Beschreibung:

Die Definitionen der verwendeten Strukturen befinden sich in der Datei *GUSB_Platform.h* auf der mitgelieferten CD.

Die Elemente der Strukturen RoutingColors und RoutingControl zeigen die aktuelle Zuordnung der LVDS-Bits an.

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

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

254 -> not used
255 -> disable

Die Erklärung zur Zuordnung der LVDS-Bits finden Sie in der Hardwarebeschreibung, Abschnitt Empfänger-Aufsätze.

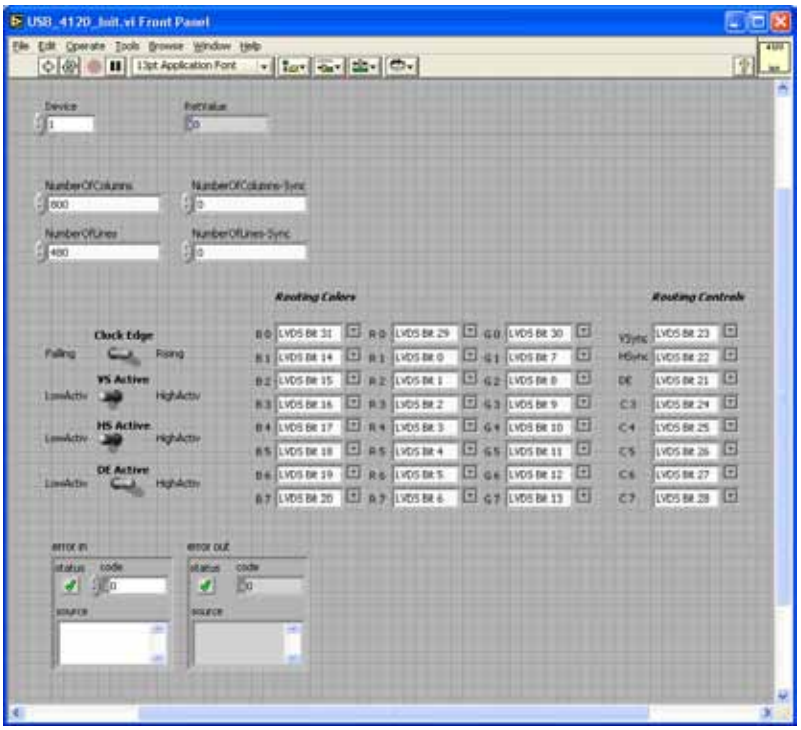
Wird eine Bitzuordnung bei den RoutingColors (RGB) mit disable initialisiert, setzt der Framegrabber dieses Bit bei den abgetasteten Bilddaten fest auf 1.

Wird eine Bitzuordnung bei den RoutingControls mit disable initialisiert, setzt der Framegrabber dieses Bit bei den abgetasteten Bilddaten fest auf 0.

Aufbau der 32 Datenbits eines Pixels:

Bits	31..27	26	25	24	23..16	15..8	7..0
Inhalt	C7..C3	DE	HS	VS	R7..R0	G7..G0	B7..B0
	Steuersignale				Farbinformationen		

Zugehöriges LabVIEW – VI:



3.2.12 Set ClockSource

Die Funktion GUSB_Platform_4120_SetClockSource setzt die Quelle des LVDS-Taktes des mit DeviceNumber indizierten USB 4120-Boards.

Format:

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

Parameter:

DeviceNumber

Nummer des adressierten Gerätes. Wenn mehrere Geräte gleichen Typs angeschlossen sind, erfolgt die Nummerierung in aufsteigender Reihenfolge der Seriennummern (das Gerät mit der NIEDRIGSTEN Seriennummer hat immer DeviceNumber 1).

Mode

Quelle des vom Framegrabber benutzten LVDS-Taktes

0 -> Standard Pixel Clock

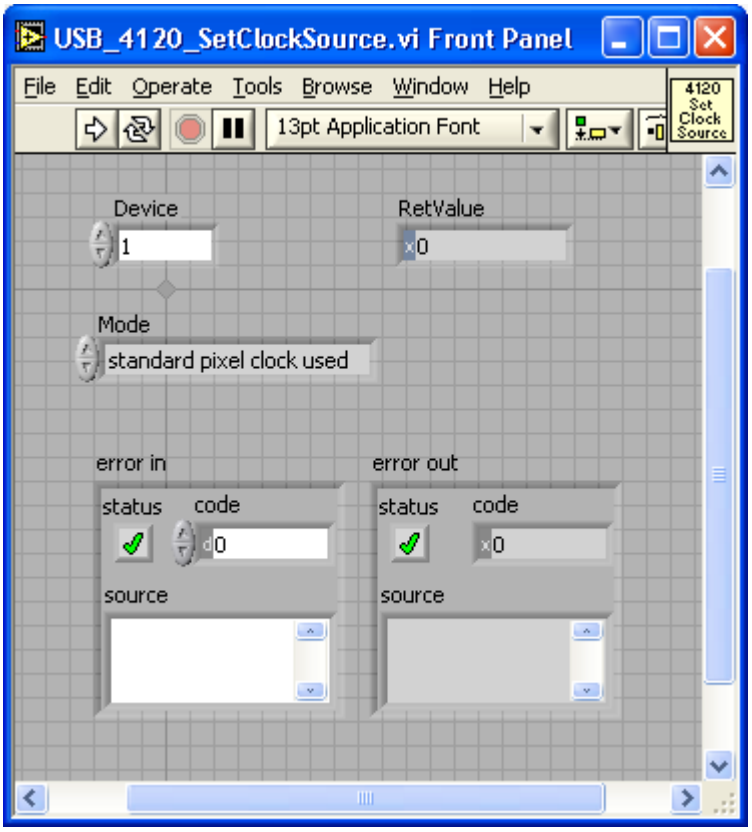
1 -> LVDS-Bit 19

Beschreibung:

Das Umschalten der Quelle des LVDS-Taktes stellt eine Sonderfunktion dar.

Die Grundeinstellung der Karte ist die Verwendung des Standard-Pixel-Taktes.

Zugehöriges LabVIEW – VI:



3.2.13 Set BitMask

Die Funktion GUSB_Platform_4120_SetBitMask setzt die Maske, die für einen internen Bildvergleich des mit DeviceNumber indizierten USB 4120-Boards verwendet wird.

Format:

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

Parameter:

DeviceNumber

Nummer des adressierten Gerätes. Wenn mehrere Geräte gleichen Typs angeschlossen sind, erfolgt die Nummerierung in aufsteigender Reihenfolge der Seriennummern (das Gerät mit der NIEDRIGSTEN Seriennummer hat immer DeviceNumber 1).

BitMask

Bits, die in der Maske auf „0“ gesetzt sind, werden beim Pixelvergleich nicht betrachtet.

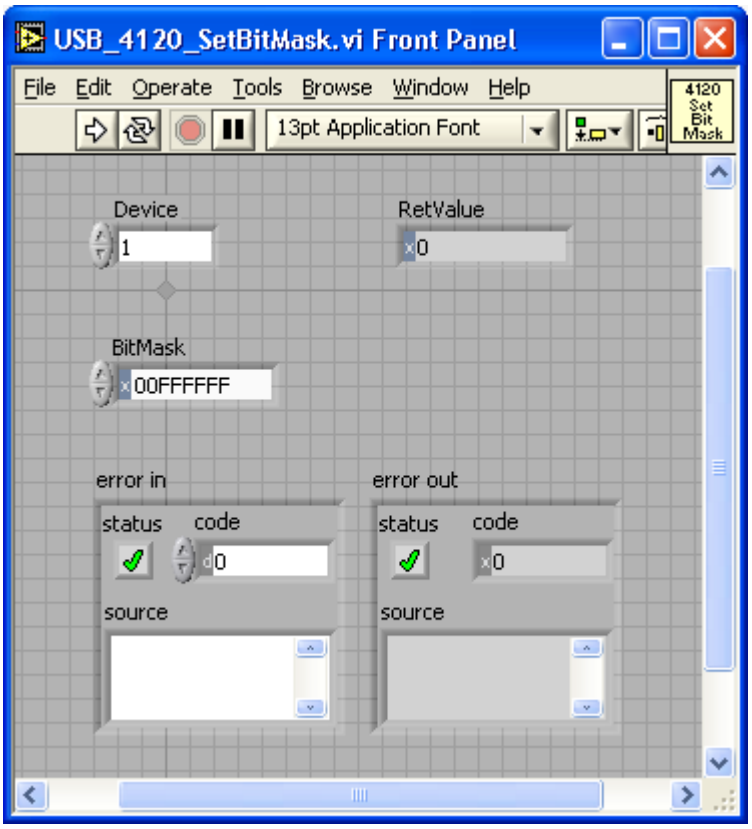
Beschreibung:

Die Funktion maskiert Bits für den internen Bildvergleich.

Bedeutung der einzelnen Bits:

Bits	31..27	26	25	24	23..16	15..8	7..0
Inhalt	C7..C3	DE	HS	VS	R7..R0	G7..G0	B7..B0
	Steuersignale				Farbinformationen		

Zugehöriges LabVIEW – VI:



3.2.14 External TriggerMode_Start

Die Funktion GUSB_Platform_4120_ExternalTriggerMode_Start aktiviert den externen Triggermode des mit DeviceNumber indizierten USB 4120-Boards.

Format:

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

Parameter:

DeviceNumber

Nummer des adressierten Gerätes. Wenn mehrere Geräte gleichen Typs angeschlossen sind, erfolgt die Nummerierung in aufsteigender Reihenfolge der Seriennummern (das Gerät mit der NIEDRIGSTEN Seriennummer hat immer DeviceNumber 1).

TriggerLevel

Setzt die Polarität des Triggersignales

0 -> low aktiv
1 -> high aktiv

Beschreibung:

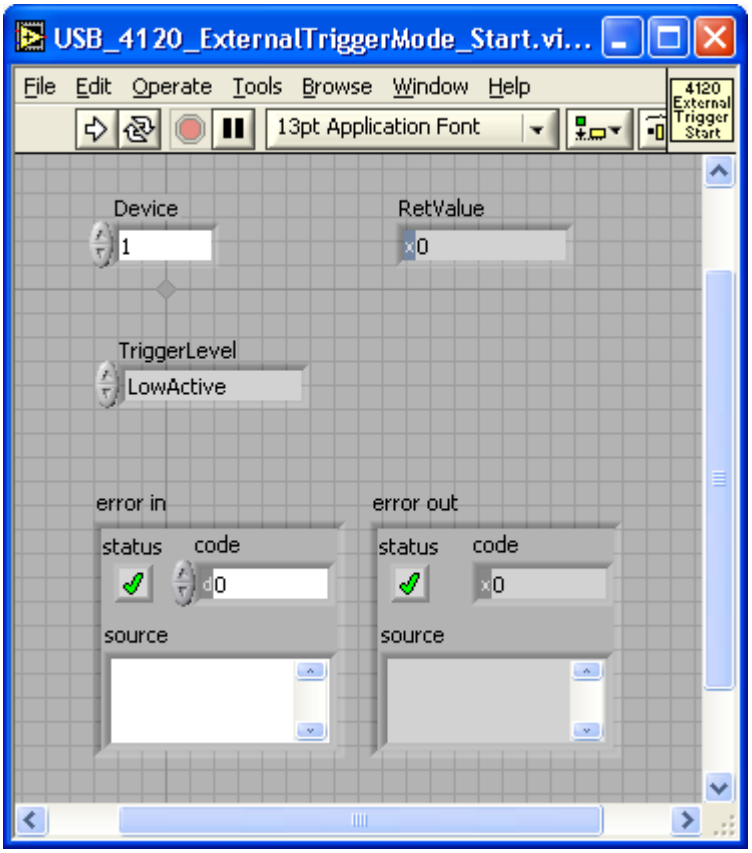
Der Framegrabber kann über D-Sub Pin 3 extern getriggert werden.

Nach der externen Triggerung wird das nächste Bild abgegriffen, auf dem Board gespeichert und mit dem ebenfalls gespeicherten Referenzbild entsprechend der konfigurierten BitMaske verglichen (siehe Set BitMask).

Vorraussetzung ist die fehlerfreie Konfiguration (GUSB_Platform_4120_Init()) und das Speichern eines Referenzbildes auf dem Board (GUSB_Platform_4120_LoadReferenceFromBuffer(), GUSB_Platform_4120_LoadReferenceFromFile())

Vorteil dieses Modus ist die Triggerung in Echtzeit und die Möglichkeit, das Board ohne PC oder Laptop zu betreiben. Dabei muss die Stromversorgung aber weiterhin gewährleistet werden (siehe Kapitel Hardware/ Digitale I/O Schnittstelle).

Zugehöriges LabVIEW – VI:



3.2.15 External TriggerMode_Stop

Die Funktion GUSB_Platform_4120_ExternalTriggerMode_Stop deaktiviert den externen Triggermodus des mit DeviceNumber indizierten USB 4120-Boards.

Format:

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

Parameter:

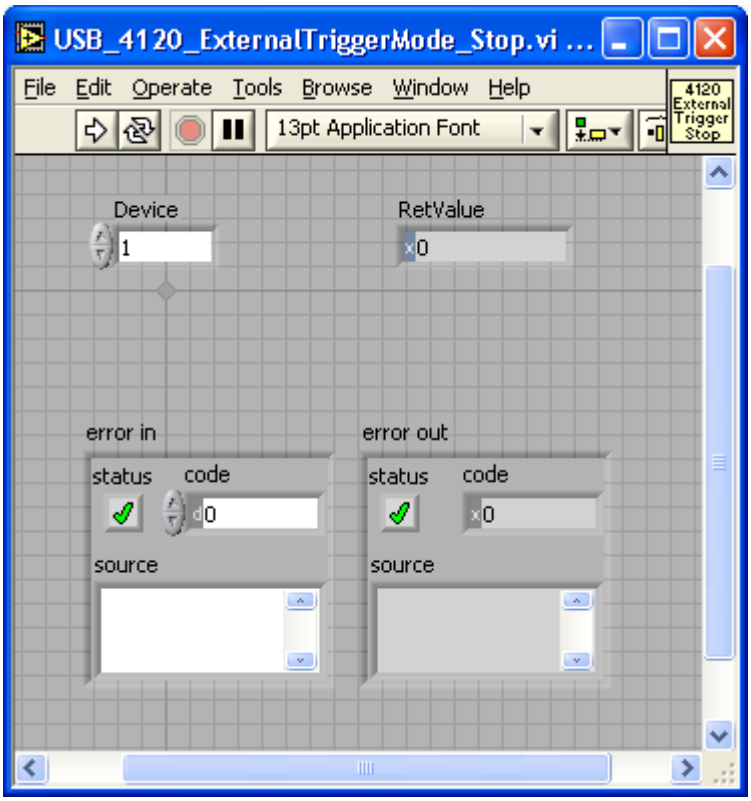
DeviceNumber

Nummer des adressierten Gerätes. Wenn mehrere Geräte gleichen Typs angeschlossen sind, erfolgt die Nummerierung in aufsteigender Reihenfolge der Seriennummern (das Gerät mit der NIEDRIGSTEN Seriennummer hat immer DeviceNumber 1).

Beschreibung:

Stoppt den externen Triggermodus

Zugehöriges LabVIEW – VI:



3.2.16 External TriggerMode_ GetState

Die Funktion GUSB_Platform_4120_ExternalTriggerMode_GetState zeigt den Status der externen Triggerungsfunktionalität des mit DeviceNumber indizierten USB 4120-Boards an.

Format:

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

Parameter:

DeviceNumber

Nummer des adressierten Gerätes. Wenn mehrere Geräte gleichen Typs angeschlossen sind, erfolgt die Nummerierung in aufsteigender Reihenfolge der Seriennummern (das Gerät mit der NIEDRIGSTEN Seriennummer hat immer DeviceNumber 1).

RspFlags

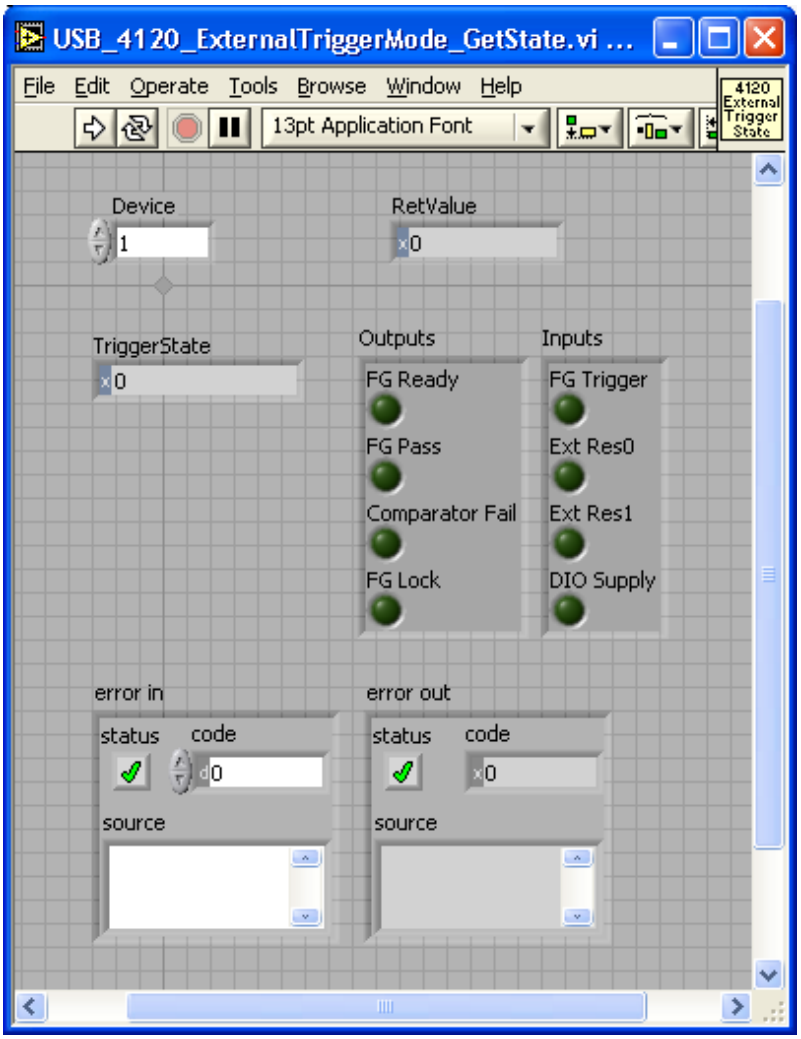
zeigt den aktuellen Zustand der acht DIO-Pins des D-Sub Steckers an:

Bit	Bezeichnung	Bedeutung
0	Framegrabber Ready Output (D-Sub Pin 1)	Bit 0 = 1 -> Framegrabber bereit (Externer Trigger Mode aktiv, Referenzbild geladen)
1	Framegrabber Pass Output (D-Sub Pin 6)	Bit 1= 1 -> Bild fehlerfrei abgetastet und verglichen
2	Comparator Failure Output (D-Sub Pin 2)	Bit 2= 1 -> Comparator hat mindestens einen Bildfehler erkannt
3	Framegrabber Lock Output (D-Sub Pin 7)	Bit 3= 1 -> Framegrabber auf LVDS-Signal gelockt
4	Externer Trigger Input (D-Sub Pin 3)	Bit 4= 1 -> Externer Trigger aktiv (Bildempfang und automatischer Vergleich werden gestartet)
5	Externer Input Reserve 0 (D-Sub Pin 8)	Reserve 0
6	Externer Input Reserve 1 (D-Sub Pin 4)	Reserve 1
7	Externer Vcc Input (D-Sub Pin 5)	Bit 7= 1 -> Externe Spannungsversorgung aktiv (notwendig, wenn der entsprechende Schalter auf DIO gestellt worden ist (siehe Abbildung 2-5)
8..31	not used	Nicht belegt

Beschreibung:

Die Funktion zeigt den Status der externen Triggerfunktionalität an.

Zugehöriges LabVIEW – VI:



3.2.17 Capture

Die Funktion GUSB_Platform_4120_Capture löst die Abtastung eines Bildes in dem mit DeviceNumber indizierten USB 4120-Board aus.

Format:

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

Parameter:

DeviceNumber

Nummer des adressierten Gerätes. Wenn mehrere Geräte gleichen Typs angeschlossen sind, erfolgt die Nummerierung in aufsteigender Reihenfolge der Seriennummern (das Gerät mit der NIEDRIGSTEN Seriennummer hat immer DeviceNumber 1).

ErrorLine

Zeigt die Nummer der Bildzeile an, in der ein Synchronisationsfehler aufgetreten ist

0 -> kein Fehler

Synchronisationsfehler können durch falsche Initialisierung oder fehlerhafte HSync, VSync und DE-Signale auftreten.

ErrorColumn

Zeigt die Nummer der Bildspalte an, in der ein Synchronisationsfehler aufgetreten ist

0 -> kein Fehler

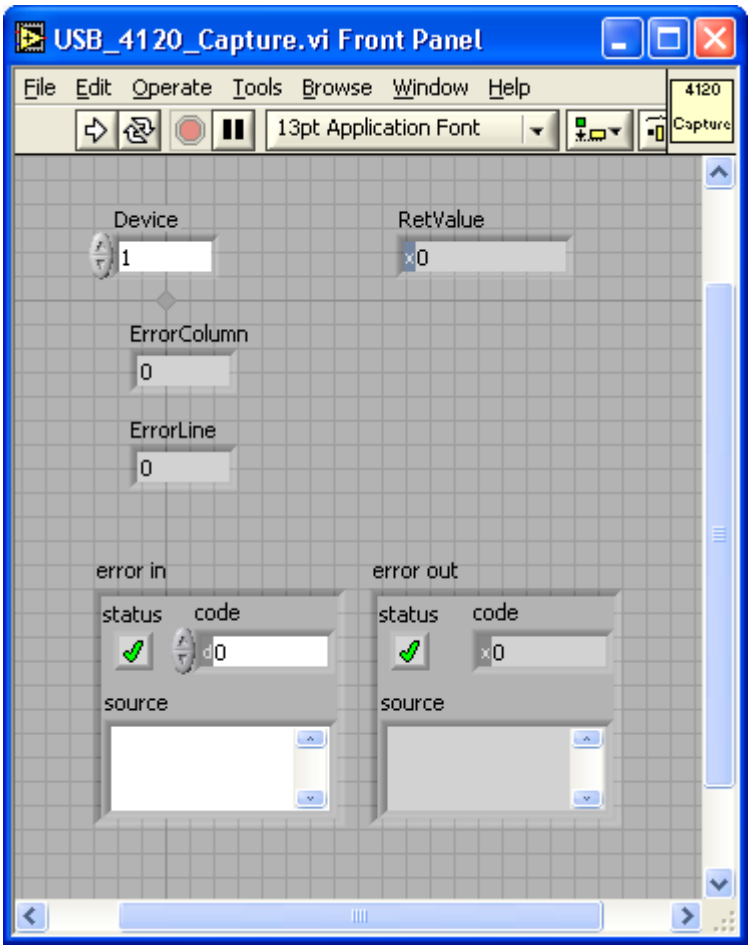
Synchronisationsfehler können durch falsche Initialisierung oder fehlerhafte HSync, VSync und DE-Signale auftreten.

Beschreibung:

Die Funktion löst die Abtastung eines Bildes aus.

Die Daten werden im Framegrabber gespeichert und stehen somit zum internen Vergleich mit einem Referenzbild zur Verfügung (Funktion GUSB_Platform_4120_Compare())
oder können mit GUSB_Platform_4120_ReadCapturedFrameToBuffer() oder GUSB_Platform_4120_ReadCapturedFrameToFile() ausgelesen werden (siehe auch Compare, Read CapturedFrame ToBuffer und Read CapturedFrame ToFile).

Zugehöriges LabVIEW – VI:



3.2.18 Compare

Die Funktion GUSB_Platform_4120_Compare vergleicht das zuletzt abgetastete Bild mit dem Referenzbild in dem mit DeviceNumber indizierten USB 4120-Board.

Format:

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

Parameter:

DeviceNumber

Nummer des adressierten Gerätes. Wenn mehrere Geräte gleichen Typs angeschlossen sind, erfolgt die Nummerierung in aufsteigender Reihenfolge der Seriennummern (das Gerät mit der NIEDRIGSTEN Seriennummer hat immer DeviceNumber 1).

ErrorCount

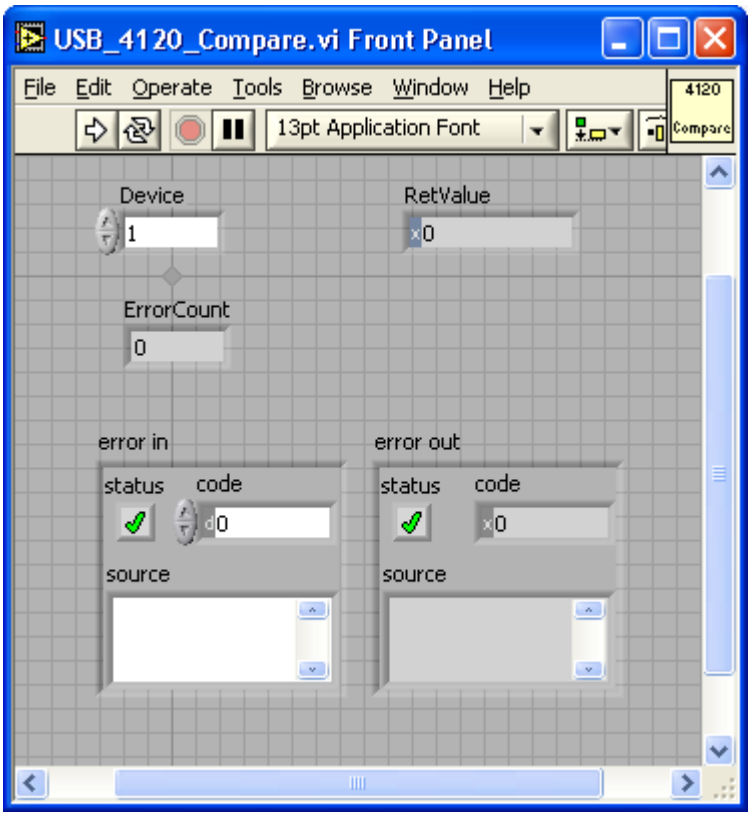
Zeigt die Anzahl der gefundenen Pixelfehler an

0 -> kein Fehler aufgetreten

Beschreibung:

Die Funktion vergleicht das letzte fehlerfrei abgetastete Bild mit dem auf dem Framegrabber gespeicherten Referenzbild.
Dabei wird die unter GUSB_Platform_4120_SetBitMask() geladene Bitmaske (siehe Set BitMask) berücksichtigt.

Zugehöriges LabVIEW – VI:



3.2.19 Capture ToBuffer

Die Funktion GUSB_Platform_4120_CaptureToBuffer löst die Abtastung eines Bildes in dem mit DeviceNumber indizierten USB 4120-Board aus und überträgt die Bilddaten in den bereit gestellten Datenpuffer.

Format:

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

Parameter:

DeviceNumber

Nummer des adressierten Gerätes. Wenn mehrere Geräte gleichen Typs angeschlossen sind, erfolgt die Nummerierung in aufsteigender Reihenfolge der Seriennummern (das Gerät mit der NIEDRIGSTEN Seriennummer hat immer DeviceNumber 1).

Data

Zeiger auf einen Datenpuffer
Der von der Applikation bereit gestellte Speicher muss mindestens eine Größe von K_MAX_BYTE_RD_EXTENDED_4120 aufweisen
(siehe Datei *GUSB_Platform.h*).

Ein Pixel besteht aus 32 Bit.
Der Framegrabber tastet zeilenweise von links oben nach rechts unten ab. Entsprechend sind die Daten im Buffer abgelegt.

Aufbau der 32 Datenbits eines Pixels:

Bits	31..27	26	25	24	23..16	15..8	7..0
Inhalt	C7..C3	DE	HS	VS	R7..R0	G7..G0	B7..B0
	Steuersignale				Farbinformationen		

Length

Größe des Datenpuffers, auf den Data zeigt, in Bytes

Nach Funktionsausführung: Anzahl der tatsächlich in den Datenpuffer eingetragenen Bytes

ErrorLine

Zeigt die Nummer der Bildzeile an, in der ein Synchronisationsfehler aufgetreten ist

0 -> kein Fehler

Synchronisationsfehler können durch falsche Initialisierung oder fehlerhafte HSync, VSync und DE-Signale auftreten.

ErrorColumn

Zeigt die Nummer der Bildspalte an, in der ein Synchronisationsfehler aufgetreten ist

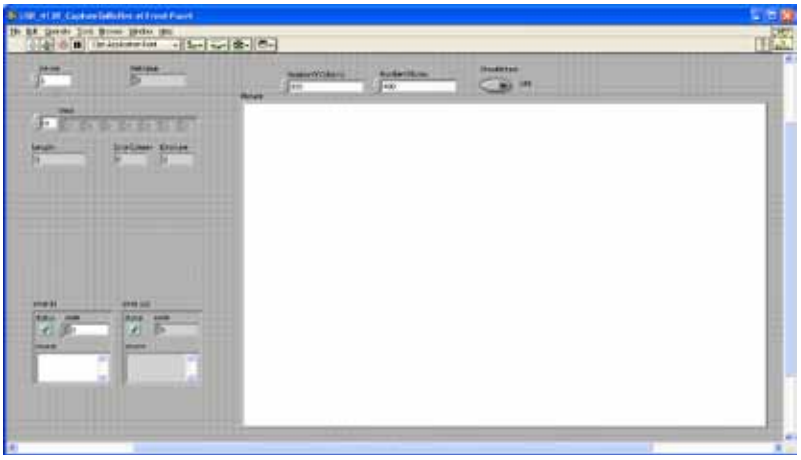
0 -> kein Fehler

Synchronisationsfehler können durch falsche Initialisierung oder fehlerhafte HSync, VSync und DE-Signale auftreten.

Beschreibung:

Die Funktion löst die Abtastung eines Bildes aus.
Die Daten werden im bereit gestellten Datenpuffer gespeichert.

Zugehöriges LabVIEW – VI:



Hier gibt es zusätzlich die Möglichkeit, die vom Framegrabber bereit gestellten Daten im Feld Picture als Bild anzuzeigen.

Für ShowPicture = ON müssen die korrekten Werte des abgetasteten Bildes in die Felder NumberOfColums und NumberOfLines eingetragen werden.

3.2.20 Capture ToFile

Die Funktion GUSB_Platform_4120_CaptureToFile löst die Abtastung eines Bildes in dem mit DeviceNumber indizierten USB 4120-Board aus und liefert die Bilddaten in einer BMP-Datei zurück.

Format:

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

Parameter:

DeviceNumber

Nummer des adressierten Gerätes. Wenn mehrere Geräte gleichen Typs angeschlossen sind, erfolgt die Nummerierung in aufsteigender Reihenfolge der Seriennummern (das Gerät mit der NIEDRIGSTEN Seriennummer hat immer DeviceNumber 1).

Width

Anzahl der Spalten des Bildes (horizontale Auflösung)

(notwendig für die interne Umwandlung der abgetasteten Bilddaten in eine BMP-Datei)

Height

Anzahl der Zeilen des Bildes (vertikale Auflösung)

(notwendig für die interne Umwandlung der abgetasteten Bilddaten in eine BMP-Datei)

Path

Dateipfad, in dem die Bilddaten abgelegt werden sollen

ErrorLine

Zeigt die Nummer der Bildzeile an, in der ein Synchronisationsfehler aufgetreten ist

0 -> kein Fehler

Synchronisationsfehler können durch falsche Initialisierung oder fehlerhafte HSync, VSync und DE-Signale auftreten.

ErrorColumn

Zeigt die Nummer der Bildspalte an, in der ein Synchronisationsfehler aufgetreten ist

0 -> kein Fehler

Synchronisationsfehler können durch falsche Initialisierung oder fehlerhafte HSync, VSync und DE-Signale auftreten.

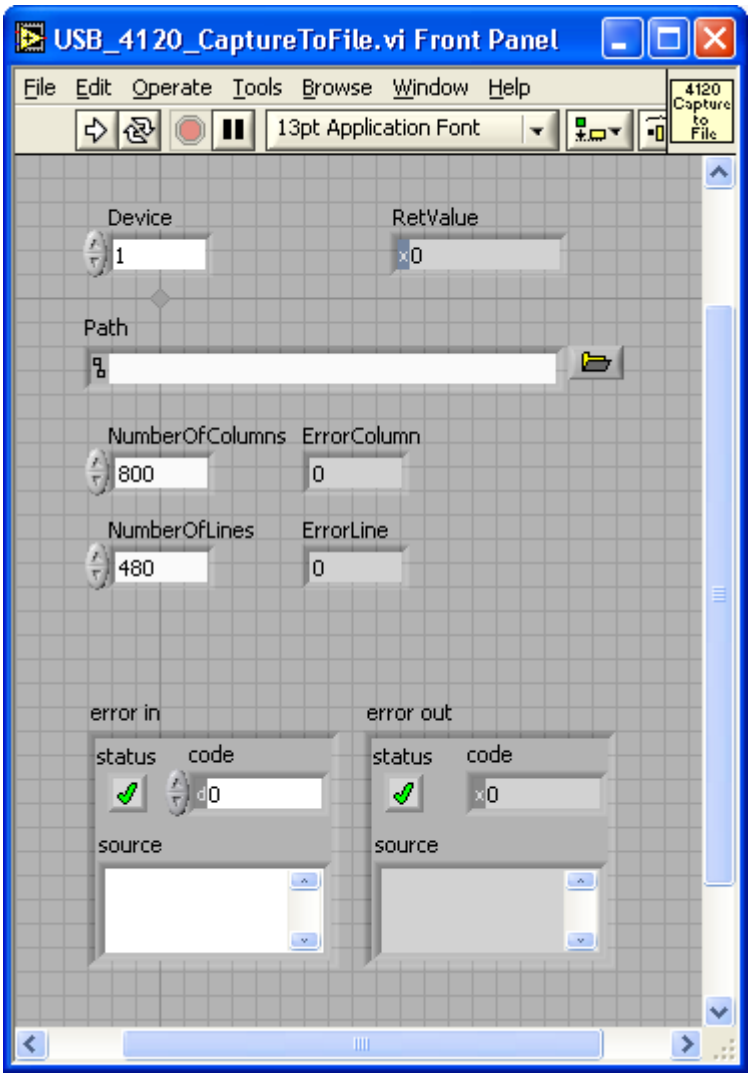
Beschreibung:

Die Funktion löst die Abtastung eines Bildes aus. Die Daten werden im bereit gestellten Pfad als BMP-Datei gespeichert.

Format der BMP-Datei:

- 32 Bit pro Pixel
- 24 Bit Farbtiefe

Zugehöriges LabVIEW – VI:



3.2.21 Read CapturedFrame ToBuffer

Die Funktion GUSB_Platform_4120_ReadCapturedFrameToBuffer liefert die Bilddaten des letzten abgetasteten Bildes des mit DeviceNumber indizierten USB 4120-Boards und speichert sie in einem Puffer.

Format:

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

Parameter:

DeviceNumber

Nummer des adressierten Gerätes. Wenn mehrere Geräte gleichen Typs angeschlossen sind, erfolgt die Nummerierung in aufsteigender Reihenfolge der Seriennummern (das Gerät mit der NIEDRIGSTEN Seriennummer hat immer DeviceNumber 1).

Data

Zeiger auf einen Datenpuffer

Der von der Applikation bereit gestellte Speicher muss mindestens eine Größe von K_MAX_BYTE_RD_EXTENDED_4120 aufweisen (siehe Datei *GUSB_Platform.h*).

Ein Pixel besteht aus 32 Bit.
Der Framegrabber tastet zeilenweise von links oben nach rechts unten ab. Entsprechend sind die Daten im Buffer abgelegt.

Aufbau der 32 Datenbits eines Pixels:

Bits	31..27	26	25	24	23..16	15..8	7..0
Inhalt	C7..C3	DE	HS	VS	R7..R0	G7..G0	B7..B0
	Steuersignale				Farbinformationen		

Length

Größe des Datenpuffers, auf den Data zeigt, in Bytes

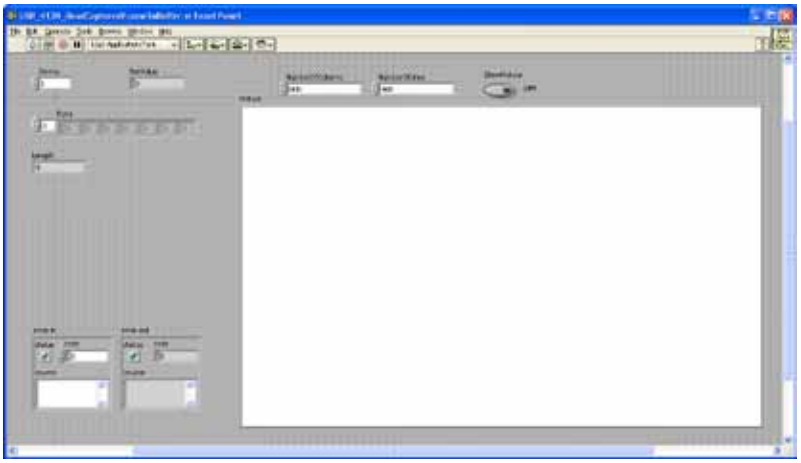
Nach Funktionsausführung: Anzahl der tatsächlich in den Datenpuffer eingetragenen Bytes

Beschreibung:

Die Funktion speichert die Daten des zuletzt abgetasteten Bildes im bereit gestellten Datenpuffer.

Um aktuelle Bilddaten zu erhalten, muss zuvor mit GUSB_Platform_4120_Capture() eine Bildabtastung ausgelöst werden.

Zugehöriges LabVIEW – VI:



Hier gibt es zusätzlich die Möglichkeit, die vom Framegrabber bereit gestellten Daten im Feld Picture als Bild anzuzeigen.

Für ShowPicture = ON müssen die korrekten Werte des abgetasteten Bildes in die Felder NumberOfColums und NumberOfLines eingetragen werden.

3.2.22 Read CapturedFrame ToFile

Die Funktion GUSB_Platform_4120_ReadCapturedFrameToFile liefert die Bilddaten des letzten abgetasteten Bildes des mit DeviceNumber indizierten USB 4120-Boards und speichert diese in einer BMP-Datei.

Format:

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

Parameter:

DeviceNumber

Nummer des adressierten Gerätes. Wenn mehrere Geräte gleichen Typs angeschlossen sind, erfolgt die Nummerierung in aufsteigender Reihenfolge der Seriennummern (das Gerät mit der NIEDRIGSTEN Seriennummer hat immer DeviceNumber 1).

Width

Anzahl der Spalten des Bildes (horizontale Auflösung)

(notwendig für die interne Umwandlung der abgetasteten Bilddaten in eine BMP-Datei)

Height

Anzahl der Zeilen des Bildes (vertikale Auflösung)

(notwendig für die interne Umwandlung der abgetasteten Bilddaten in eine BMP-Datei)

Path

Dateipfad, in dem die Bilddaten abgelegt werden sollen

Beschreibung:

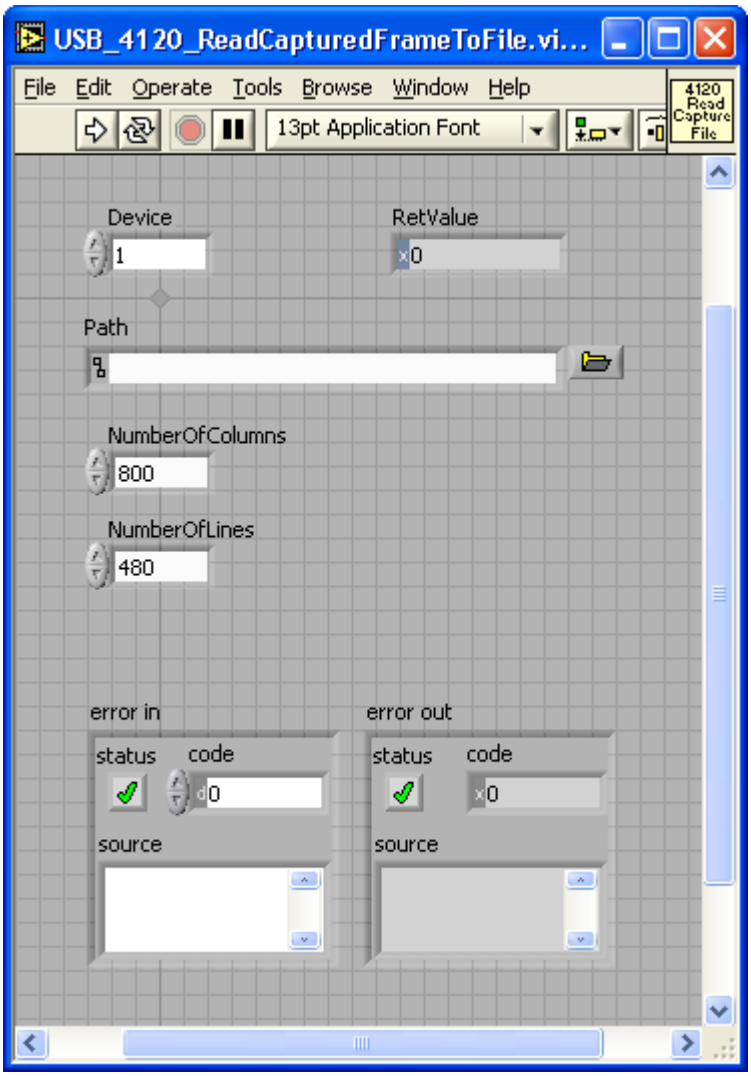
Die Funktion speichert die Daten des zuletzt abgetasteten Bildes als BMP im bereit gestellten Pfad.

Um aktuelle Bilddaten zu erhalten, muss zuvor mit GUSB_Platform_4120_Capture() eine Bildabtastung ausgelöst werden.

Format der BMP-Datei:

- 32 Bit pro Pixel
- 24 Bit Farbtiefe

Zugehöriges LabVIEW – VI:



3.2.23 Load Reference FromBuffer

Die Funktion GUSB_Platform_4120_LoadReferenceFromBuffer lädt Daten eines Referenzbildes in das mit DeviceNumber indizierte USB 4120-Board.

Format:

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

Parameter:

DeviceNumber

Nummer des adressierten Gerätes. Wenn mehrere Geräte gleichen Typs angeschlossen sind, erfolgt die Nummerierung in aufsteigender Reihenfolge der Seriennummern (das Gerät mit der NIEDRIGSTEN Seriennummer hat immer DeviceNumber 1).

Data

Zeiger auf einen Datenpuffer

Die Bilddaten dürfen dabei eine Größe von K_MAX_BYTE_RD_EXTENDED_4120 – 12 nicht überschreiten (siehe Datei *GUSB_Platform.h*).

Ein Pixel besteht aus 32 Bit.
Der Framegrabber tastet zeilenweise von links oben nach rechts unten ab. Entsprechend sind die Daten im Buffer abgelegt.

Aufbau der 32 Datenbits eines Pixels:

Bits	31..27	26	25	24	23..16	15..8	7..0
Inhalt	C7..C3	DE	HS	VS	R7..R0	G7..G0	B7..B0
	Steuersignale				Farbinformationen		

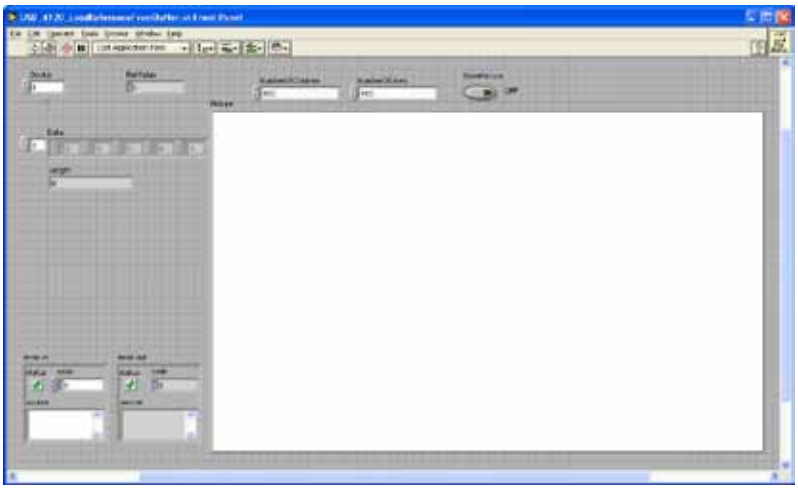
Length

Größe des Datenpuffers, auf den Data zeigt, in Bytes

Beschreibung:

Die Funktion lädt Bilddaten in den Framegrabber. Dieser legt die Daten in einem internen Speicher ab.
Damit können Vergleiche zwischen diesem Referenzbild und einem abgetasteten Bild direkt auf dem Board erfolgen.

Zugehöriges LabVIEW – VI:



Hier gibt es zusätzlich die Möglichkeit, die vom Framegrabber bereit gestellten Daten im Feld Picture als Bild anzuzeigen.

Für ShowPicture = ON müssen die korrekten Werte des abgetasteten Bildes in die Felder NumberOfColums und NumberOfLines eingetragen werden.

3.2.24 Load Reference FromFile

Die Funktion GUSB_Platform_4120_LoadReferenceFromFile lädt Daten eines Referenzbildes aus einer BMP-Datei in das mit DeviceNumber indizierte USB 4120-Board.

Format:

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

Parameter:

DeviceNumber

Nummer des adressierten Gerätes. Wenn mehrere Geräte gleichen Typs angeschlossen sind, erfolgt die Nummerierung in aufsteigender Reihenfolge der Seriennummern (das Gerät mit der NIEDRIGSTEN Seriennummer hat immer DeviceNumber 1).

Path

Dateipfad, in dem die BMP-Bilddaten abgelegt sind

Die Bilddaten dürfen dabei eine Größe von K_MAX_BYTE_RD_EXTENDED_4120 – 12 nicht überschreiten (siehe Datei *GUSB_Platform.h*).

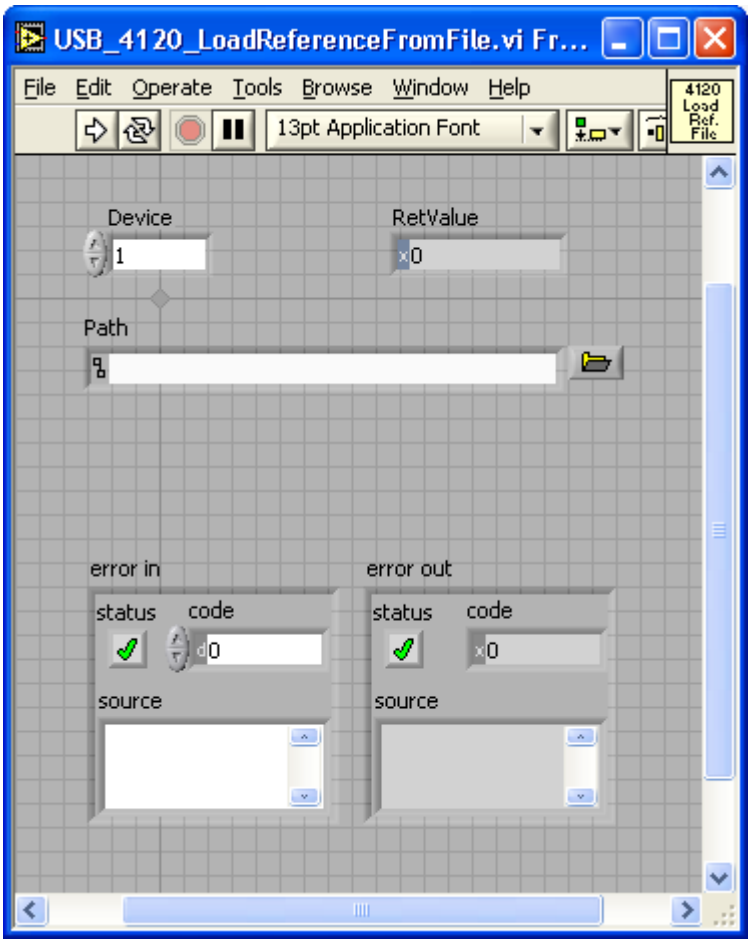
Beschreibung:

Die Funktion lädt Bilddaten aus einer BMP-Datei in den Framegrabber. Dieser legt die Daten in einem internen Speicher ab.
Damit können Vergleiche zwischen diesem Referenzbild und einem abgetasteten Bild direkt auf dem Board erfolgen.

Format der BMP-Datei:

- 32 Bit pro Pixel
- 24 Bit Farbtiefe

Zugehöriges LabVIEW – VI:



3.2.25 Read Reference ToBuffer

Die Funktion GUSB_Platform_4120_ReadReferenceToBuffer liefert die gespeicherten Referenz-Bilddaten des mit DeviceNumber indizierten USB 4120-Boards.

Format:

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

Parameter:

DeviceNumber

Nummer des adressierten Gerätes. Wenn mehrere Geräte gleichen Typs angeschlossen sind, erfolgt die Nummerierung in aufsteigender Reihenfolge der Seriennummern (das Gerät mit der NIEDRIGSTEN Seriennummer hat immer DeviceNumber 1).

Data

Zeiger auf einen Datenpuffer

Der von der Applikation bereit gestellte Speicher muss mindestens eine Größe von K_MAX_BYTE_RD_EXTENDED_4120 aufweisen (siehe Datei *GUSB_Platform.h*).

Ein Pixel besteht aus 32 Bit. Der Framegrabber tastet zeilenweise von links oben nach rechts unten ab.
Entsprechend sind die Daten im Buffer abgelegt.

Aufbau der 32 Datenbits eines Pixels:

Bits	31..27	26	25	24	23..16	15..8	7..0
Inhalt	C7..C3	DE	HS	VS	R7..R0	G7..G0	B7..B0
	Steuersignale				Farbinformationen		

Length

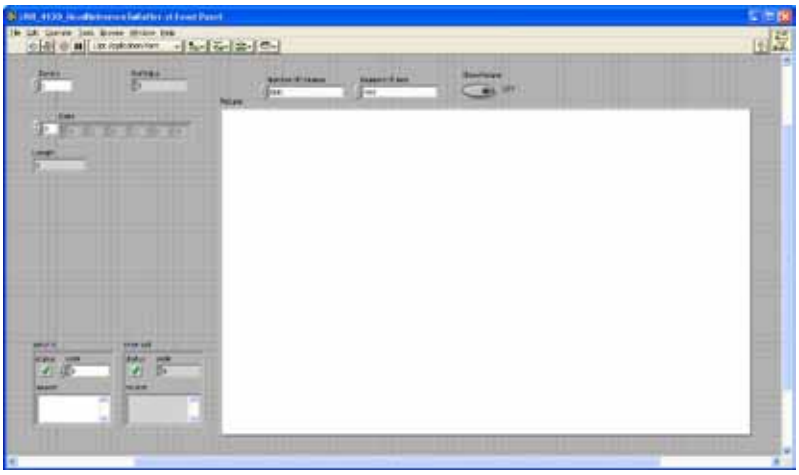
Größe des Datenpuffers, auf den Data zeigt, in Bytes

Nach Funktionsausführung: Anzahl der tatsächlich in den Datenpuffer eingetragenen Bytes

Beschreibung:

Die Funktion speichert die im Framegrabber abgelegten Referenzdaten eines Bildes in den bereit gestellten Datenpuffer.

Zugehöriges LabVIEW – VI:



Hier gibt es zusätzlich die Möglichkeit, die vom Framegrabber bereit gestellten Daten im Feld Picture als Bild anzuzeigen.

Für ShowPicture = ON müssen die korrekten Werte des abgetasteten Bildes in die Felder NumberOfColums und NumberOfLines eingetragen werden.

3.2.26 Read Reference ToFile

Die Funktion GUSB_Platform_4120_ReadReferenceToFile liefert die gespeicherten Referenz-Bilddaten des mit DeviceNumber indizierten USB 4120-Boards in einer BMP-Datei zurück.

Format:

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

Parameter:

DeviceNumber

Nummer des adressierten Gerätes. Wenn mehrere Geräte gleichen Typs angeschlossen sind, erfolgt die Nummerierung in aufsteigender Reihenfolge der Seriennummern (das Gerät mit der NIEDRIGSTEN Seriennummer hat immer DeviceNumber 1).

Width

Anzahl der Spalten des Bildes (horizontale Auflösung)

(notwendig für die interne Umwandlung der abgetasteten Bilddaten in eine BMP-Datei)

Height

Anzahl der Zeilen des Bildes (vertikale Auflösung)

(notwendig für die interne Umwandlung der abgetasteten Bilddaten in eine BMP-Datei)

Path

Dateipfad, in dem die Bilddaten abgelegt werden sollen

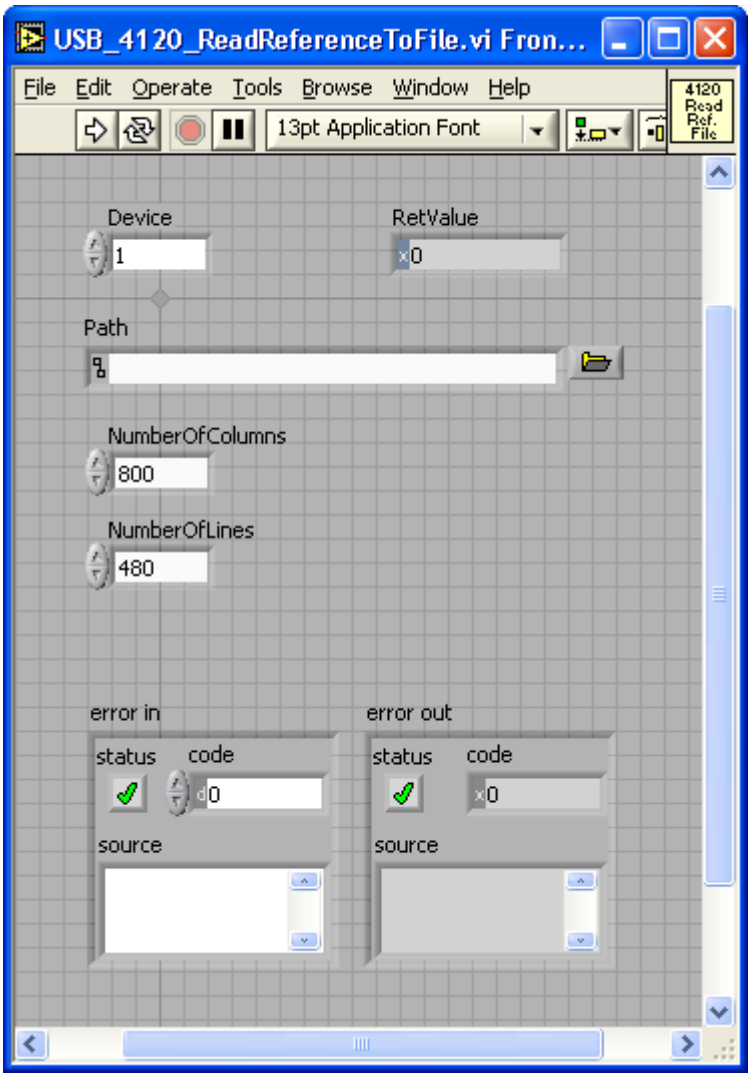
Beschreibung:

Die Funktion speichert die im Framegrabber abgelegten Referenzdaten eines Bildes als BMP-Datei im bereit gestellten Pfad.

Format der BMP-Datei:

- 32 Bit pro Pixel
- 24 Bit Farbtiefe

Zugehöriges LabVIEW – VI:



3.2.27 Deserializer Configuration Vector

Die Funktion GUSB_Platform_4120_DeserializerConfigurationVector konfiguriert den Deserializer des mit DeviceNumber indizierten USB 4120-Boards.

Format:

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

Parameter:

DeviceNumber

Nummer des adressierten Gerätes. Wenn mehrere Geräte gleichen Typs angeschlossen sind, erfolgt die Nummerierung in aufsteigender Reihenfolge der Seriennummern (das Gerät mit der NIEDRIGSTEN Seriennummer hat immer DeviceNumber 1).

Mode

Setzt den Zugriffsmodus auf die Parameter Byte1...Byte4 fest (siehe Datei *GUSB_Platform.h*)

K_4120_READ_DESERIALIZER_VECTOR - > Lesen
K_4120_WRITE_DESERIALIZER_VECTOR -> Schreiben

Byte1

Wert für das Konfigurationsregister des Deserializers
INAP -> ADR 1; DS90 -> ADR 0

Byte2

Wert für das Konfigurationsregister des Deserializers
INAP -> ADR 2; DS90 -> ADR 1

Byte3

Wert für das Konfigurationsregister des Deserializers
INAP -> ADR 3; DS90 -> ADR 2

Byte4

Wert für das Konfigurationsregister des Deserializers
INAP -> ADR 4; DS90 -> ADR 3

Beschreibung:

Die Konfiguration ist nur für folgende Deserializertypen erforderlich:

- APIX INAP125R24_V10
- APIX INAP125R24_V11
- DS90UR906

Die genaue Beschreibung der Konfigurationsregister finden Sie in den jeweiligen Datenblättern der Deserializer.

Beispiel 1 INAP125R24:

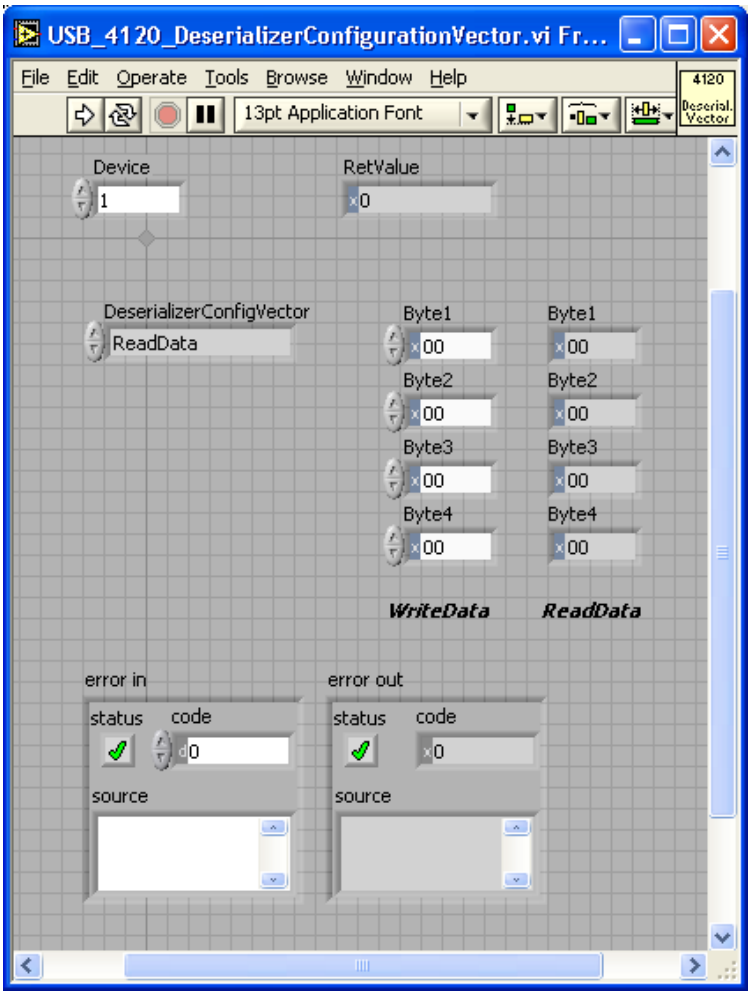
Byte1:	C5	(dedicated upstream=disabled, Rest=Default)
Byte2:	7F	(pixel date width=24, Rest=Default)
Byte3:	28	(Reserved=2, Rest=Default)
Byte4:	0C	(Default)

Die Bits 7..4 im Byte 3 müssen auf 0x2 gestellt werden, da sonst das LVDS-Lock nicht korrekt angezeigt wird.

Beispiel 2 DS90UR906:

Byte1:	51	(OSSel=1, rfb=1, UserReg1=1)
Byte2:	70	(Default)
Byte3:	00	(Default)
Byte4:	00	(Default)

Zugehöriges LabVIEW – VI:



3.3 Programmieren mit LabVIEW

LLB unter Verwendung des Windows Device Treibers

Auf der mitgelieferten CD befindet sich eine VI-Sammlung, mit deren Hilfe die USB 4120-Boards oder basicCON 4120-Geräte direkt unter LabVIEW angesprochen werden können.

Dabei werden die Funktionen genutzt, die im Abschnitt Programmieren über DLL-Funktionen beschrieben worden sind.

3.4 Nutzung weiterer GÖPEL Software

PROGRESS, Programm-Generator und myCAR der GÖPEL electronic GmbH sind komfortable Programme zur Prüfung mit GÖPEL-Hardware.

Weitere Informationen zur Nutzung dieser Programme finden Sie in den entsprechenden Softwarebeschreibungen.

3.5 Steuerbefehle USB-Controller

Der USB-Controller ist für die Anbindung der USB 4120-Baugruppe an den PC/ Laptop über USB 2.0 zuständig.

An diesen USB-Controller können Nachrichten (i. Allg. USB-Befehle) gesendet werden, die für Konfigurationszwecke benötigt werden.

3.5.1 USB-Befehlsaufbau

Ein USB-Befehl besteht aus vier Bytes Header und den Daten (aber nicht alle USB-Befehle benötigen Daten!).

Der Header eines USB-Befehls ist folgendermaßen aufgebaut:

Bytenummer	Bedeutung	Inhalt
0	StartByte	0x23 (ASCII-Zeichen „#“)
1	Command	(0x..) Verwendete Codes entsprechend USB-Befehle
2	reserviert	0x00
3	reserviert	0x00

3.5.2 USB-Antwortaufbau

Genau wie der USB-Befehl, ist auch die USB-Antwort in vier Bytes Header und die Daten unterteilt (aber nicht alle USB-Befehle senden Daten zurück!).

Der Header einer USB-Antwort ist folgendermaßen aufgebaut:

Bytenummer	Bedeutung	Inhalt
0	StartByte	0x24
1	Command	(0x..) Verwendete Codes entsprechend USB-Befehle
2	Length	Vom Befehl abhängige Länge
3	ErrorCode	Gibt den Fehlercode des Befehls zurück

3.5.3 USB-Befehle

Gegenwärtig steht nur der USB-Befehl READ_SW_VERSION zur Verfügung.

Command	Bezeichnung	Bedeutung
0x04	READ_SW_VERSION	Liefert die Version des USB-Controllers Antwort: Byte 4: low Byte generic Softwareversion Byte 5: high Byte generic Softwareversion Byte 6: low Byte Softwareversion des funktionellen Teiles Byte 7: high Byte Softwareversion des funktionellen Teiles

B

basicCON 4120
Anschlüsse 2-3
Bild
Abtasten..... 3-31
BMP 3-37
Puffer..... 3-35
Vergleichen 3-33
Bitmaske..... 3-24

D

Deserializer
Konfiguration 3-51
Typ 2-5, 2-10, 3-12
DIO 2-8
DLL-Funktionen 3-1

G

G-API 3-1

I

Installation..... 1-1

K

Kennwerte 2-4

L

LEDs 2-9
LVDS.....2-7
Takt.....3-22

R

Reset
Board.....3-18
FPGA 3-10

S

Stromversorgung 2-6

T

Trigger
Modus.....3-29
Start 3-26
Stop 3-28

U

USB
Antwort.....3-55
Befehl.....3-55
USB 4120
Bestimmung 2-1
USB-Rack 2-1