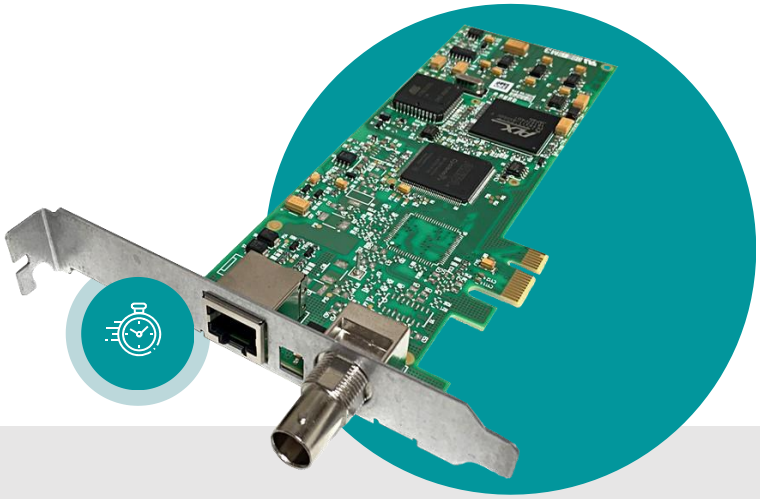
plura

TIMING SOLUTIONS

PCIe PC Cards



PCIe 3G
PCIe LV
PCIe L

Time Code Reader Board

PCIe 3G

Operating Manual
Version: 5.39
April 20, 2023

TABLE OF CONTENTS

A1 Revision History

No.	Date	Subject
3.37	December 3, 2008	
4.67	November 29, 2011	Added PCIe 3G board. Added firmware update procedure.
4.68	January 12, 2012	Added Windows 7 support. Added functions pclGetVideoStandard, pclGetGpi, pclGetPayloadIdentifier and pclSetUsePayloadIdentifier.
4.72	August 27, 2012	Added Windows 8 support.
4.75	October 9, 2012	Added function pclGetAncDataTab.
4.77	March 20, 2015	Changed PCL PCI / PCL PCIe to PCI / PCIe.
5.34	November 16, 2017	Added Windows 10 support. Added Linux installation. Corrected RJ45 orientation in PCIe drawing.
5.35	September 25, 2019	Changed address of Plura Europe GmbH.
5.36	December 3, 2020	Re-formatted in new design.
5.37	October 14, 2021	Added product names to CE Declaration.
5.38	April 4, 2023	Added Windows 11 and Server 2022 support.
5.39	April 20, 2023	Fixed mising PCI / PCIe product name.

A2 Safety Instructions

General rules

Only use the device as directed in a dry atmosphere. Treat the PCI / PCIe board with the same care as other PC boards. Please follow the advice in the following operator's manual.

Damages in transit

In case the device shows obvious damages from transit the shipper in question must be notified and the dealer must be informed.

Repairs

The PCI / PCIe board does not require any extra maintenance. There are no user serviceable parts at the device. Repairs should be sent to an authorized service partner.

EMC

The EMC regulations are observed only under the following condition: use high quality shielded cables at data inputs and outputs.

A3 Copyright

For further information please contact your local dealer or:

Plura Europe GmbH
Binger Weg 12
D- 55437 Ockenheim
Phone: +49 6725 918 006-70
Fax: +49 6725 918 006-77
E-Mail: info@plurainc.com
Internet: http://www.plurainc.com

A4 CE Declaration

We,

Plura Europe GmbH
Binger Weg 12
D- 55437 Ockenheim

declare under our sole responsibility that the

PCIe 3G
PCIe LV
PCIe L

meet the intent of the following directives, standards and specifications:

89/336/EEC Electromagnetic Compatibility

EN 50081-1 Emissions
- EN 55022
- EN 55103-1

EN 50082-1 Immunity
- EN 55024
- EN 55103-2

B Functions

B1 Overview

Models

The PCI / PCIe are Time Code reader boards for PC's with a PCI Bus. This is an easy, fast, professional, and technically advanced way for the user to integrate Time Code into its applications. The board has its own processor system with a register set for data transfer. Thus, the critical time routines are completely decoupled from the PC's CPU.

The PCI / PCIe board range consists basically of the following models:

PCIe 3G	High-Speed LTC, DVITC and ATC reader, 3G/HD/SD video, PCIe bus
PCIe LV	High-Speed LTC and VITC (CVBS video) reader, PCIe bus
PCIe L	High-Speed LTC reader, PCIe bus

And the discontinued models:

PCIe HD	High-Speed LTC, DVITC and ATC reader, HD/SD video, PCIe bus
PCIe D	High-Speed LTC and DVITC (SD video) reader, PCIe bus
PCI HD	High-Speed LTC, DVITC and ATC reader, HD/SD video, universal 3.3V / 5V PCI bus
PCI D	High-Speed LTC and DVITC (SD video) reader, universal 3.3V / 5V PCI bus
PCI LV	High-Speed LTC and VITC (CVBS video) reader, universal 3.3V / 5V PCI bus
PCI L	High-Speed LTC reader, universal 3.3V / 5V PCI bus

Variants of the models are available upon request.

LTC reader

Reads Linear Time Code according to ANSI/SMPTE 12M-1995. Reader accepts slow and high speed from 1 to 2500 frames per second (fps), in forward and reverse directions. Simultaneous decoding of time and user information. Input 100mV to 5Vpp, balanced or unbalanced, 2 x RCA connector. Automatic detection of the frame rate (can be switched off).

VITC reader

Reads Vertical Interval Time Code according to ANSI/SMPTE 12M-1995. Reader accepts VITC from still picture to search speed. Simultaneous decoding of time and user information. Automatic adaptation at the VITC data level. 1 x BNC connector, 75 Ω termination can be switched off. Automatic VITC lines detection, but a single line, 2 lines or a range of lines can be selected as well. Automatic detection of the frame rate (can be switched off).

DVITC reader

Reads Digital Vertical Interval Time Code according to SMPTE 266M-1994. Automatic level matching at DVITC data level can be selected. Simultaneous decoding of time and user information. Input SD video (4:2:2 component, SMPTE 259M-1997), 1 x BNC connector with 75 Ω termination. Automatic DVITC lines detection, but a single line, 2 lines or a range of lines can be selected as well. Automatic detection of the frame rate (can be switched off).

ATC reader

Reads Ancillary Time Code (SMPTE RP 188-1999) and HANC Time Code (SMPTE RP 196-1997). Simultaneous decoding of time and user information. Input SD video (4:2:2 component, SMPTE 259M-1997), HD video (SMPTE 292M-1998), or 3G Video (SMPTE 424M-2006), 1 x BNC connector with 75 Ω termination. Automatic detection of the frame rate (can be switched off).

PC interface

32-Byte-register set for data transfer to PCI interface. Parallel operating of several PCI / PCIe boards is possible. Selectable interrupt control. Board for the PCI local bus, 32Bit / 33 MHz slot.

Driver support

Drivers and example programs are included for Windows Vista / 7 / 8 / 10 / 11, Server 2008 / 2012 / 2016 / 2019 / 2022, and Linux, in 32-bit and 64-bit versions, respectively. DLL functions for reading Time Code rates and configuration are given. C/C++, Delphi and Visual Basic are supported.

Linux driver is included in source code.

B2 Time Code features

Time Code register

Time Code data transfer from the PCI / PCIe to the PC takes place by using a register (command 0x43: pclGetRegister).

The LTC register holds the data from the LTC reader. The reader checks for plausible time information, detects the direction (forward or reverse) or checks for a still LTC. In case of a still LTC the time values do not change frame by frame, this data is then transferred to the register. In all other cases the time will be counted + 1 frame during forward, - 1 frame during reverse, and then transferred to the register. A flag bit indicates the direction of the LTC.

The VITC register holds the data of the VITC/DVITC reader. Having passed the plausibility check the data is transferred to the register immediately. This takes place every field. A flag bit indicates the first and second field.

The mixed register holds LTC or VITC/DVITC data, controlled by the commands 0x20 (pclMixedEnable) and 0x21 (pclPriority). Command 0x20 enables LTC or VITC/DVITC for using the mixed register. Command 0x21 sets the priority, if both LTC and VITC/DVITC is read simultaneously.

The ATC reader receives data packets and checks the type of Time Code. The decoded data will then be transferred to the special registers, as there are ATC LTC, ATC VITC, HANC LTC and HANC VITC. These Time Codes cannot be transferred to the mixed register.

Frame rate

The automatic frame rate detection sets the frame rate to 24, 25, 30 (without drop) or 30 drop mode, individual for LTC and VITC/DVITC. After power-on the frame rate is pre-set to 25 and

the automatic frame rate detection is turned on. The automatic frame rate detection can be switched off by setting the frame rate manually. Command 0x30 (pclTcFrames) does this for LTC and VITC/DVITC simultaneously. Command 0x31 alters the LTC frame rate mode, command 0x32 the VITC frame rate mode.

Error checks

The read Time Code values are checked of plausible time data and of a correct time sequence without breaks or drop-outs. Plausible data will be transferred to the Time Code registers and announced as new data. Any errors are indicated with an error counter. This enables you to verify that a Time Code source is free of errors.

The LTC error counter can be read by command 0x17 (pclLtcError). Errors counted are: implausible time data and/or a discontinuity of the time sequence (new frame should be ± 1 frame of previous frame).

The VITC/DVITC error counter can be read by command 0x18 (pclVitcError). Errors counted are: implausible time data and/or a discontinuity of the time sequence (check at every 1st field: time of new frame should be + 1 of previous frame, check at every 2nd field: time should be equal to 1st field). In case the automatic level matching is switched on (see chapter below), further errors will be counted while the automatic is active (after a loss of video or a time-out of VITC/DVITC).

The error check with the flying wheel feature (see next chapter) enabled at same time does not give a valid result with regard to the continuity check of the time sequence.

The ATC error counters can be read by commands 0x22 (pclAtcLtcError), 0x23 (pclAtcVitcError), 0x25 (pclHancLtcError) and 0x26 (pclHancVitcError). Errors counted are: implausible time data and/or a discontinuity of the time sequence (in case of LTC: new frame should be + 1 frame of previous frame; in case of VITC: new field should be equal to or + 1 frame of previous field).

The following access modes are provided for the commands:

- CMDD0 = $00: request and automatic reset and initialization of the error counter. After this command the error counter will be set to zero and the first Time Code will not count an error. This command should be used to start with a new error check.
- CMDD0 = $01: request only. The error counter has a maximum value of 255, at further errors the counter will remain at this maximum value. This command may be used to verify that Time Code can be read without errors. The exact amount of errors may not be of special interest.
- CMDD0 = $02: request and automatic reset to zero. If a large number of errors is expected and the amount of errors is of interest, the application software should use this command and create an own error counter.

Flying wheel

The flying wheel serves as a drop-out compensation for a continuous up-counting Time Code. Command 0x19 (pclFlyWheel) enables or disables this feature for LTC and VITC/DVITC separately. It is not available for any ATC Time Code. After power-on the flying wheel is disabled. If enabled, an internal counter can continue to transfer new time data to the Time Code registers if the Time Code source has drop-outs. The flying wheel only works if the last read Time Code

was free of errors, i.e. in a correct timing sequence and within a specific frequency range. The flying wheel is not a precise clock; its goal is to overcome a few seconds of missing Time Code.

With LTC the flying wheel is able to synchronize to the incoming LTC frequency in a range of 15 - 60 frames/second. The accuracy will be ≤ ±1 frame each minute.

With VITC/DVITC the flying wheel transfers data corresponding to the first field only. The current frame rate determines the television system: 25 = 625/50 (PAL), 30 = 525/60 (NTSC). The accuracy will be better than 1 frame in ten minutes.

VITC level matching

After power-on the automatic level matching is switched on. The VITC reader analyzes the VITC level and adjusts the threshold to an optimal value. This assures readability of VITC over a wide range of video or VITC levels. In case of a lack of VITC or of cumulative errors the process starts again. This procedure requires a few seconds to find the best value. If the video signal is very bad (for example working in the jog or shuttle mode with a VHS recorder), the automatic level matching feature on one hand can help make it possible to read the VITC, or on the other hand it can produce additional errors during adjustment. So, whether the automatic mode gives better results or not depends on the application and should be tested. Using command 0x16 (pclVitcLevelControl) the automatic can be switched on and off, furthermore the current threshold value can be requested and can be set explicitly. The values must be within range 0x00 to 0x5A, this corresponds to a voltage level from -100mV to +1400mV approximately (0V = black level of the video). The default value is 0x16 (≈ 270mV).

DVITC level matching

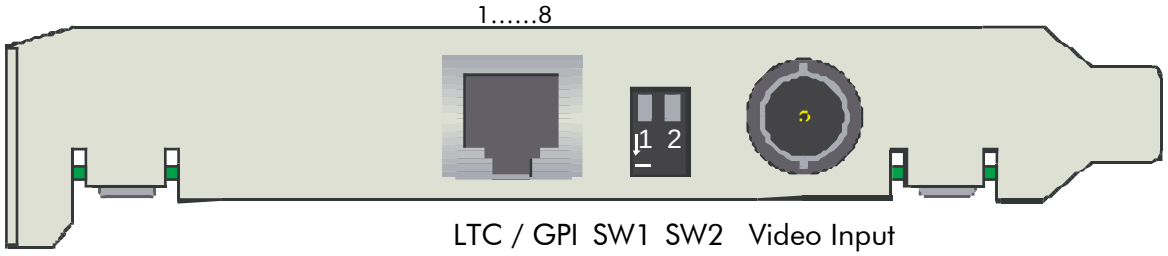
Similar to the analogue VITC, the best threshold value for the DVITC can be adjusted automatically as well. But after power-on the automatic level matching is switched off, and an average value is pre-set. Using command 0x16 (pclVitcLevelControl) the automatic can be switched on and off, furthermore the current threshold value can be requested and can be set explicitly. The input values must be within range 0x00 to 0x5A; this will be transformed to values of the digital video from 0x10 to 0xC4. The default value is 0x32 (corresponding to 0x74).

B3 Connections and Technical Data

Connections

Input	Connector	Signal description
LTC-1 LTC-2 (PCI Boards)	2 x RCA jack	LTC input, balanced or unbalanced, 100 mV_{pp} to 5 V_{pp} , frequency 1 to 2500 frames/sec.
LTC / GPI (PCIe Boards)	RJ45	LTC input, balanced or unbalanced, 100 mV_{pp} to 5 V_{pp} , frequency 1 to 2500 frames/sec.
Video Input (PCI LV, PCIe LV)	BNC, 75 Ω (IEC 169-8)	Composite video input, 75 Ω termination can be switched off. Television system 525/60 (NTSC) or 625/50 (PAL). DC offset: < ± 12V, sync amplitude = 300mV ± 6dB (150mV-600mV). VITC data level: V_{min} (“1”) = 200mV, V_{max} (“0”) = 500mV, V_{1-0} (“1” - “0”) ≥ 200mV.
Video Input (PCI D, PCIe D)	BNC, 75 Ω (IEC 169-8)	SD input, 8/10 bit according to SMPTE 259M-C: 270 Mb/s, 525/625 components.
Video Input (PCI HD, PCIe HD)	BNC, 75 Ω (IEC 169-8)	SD input, 8/10 bit according to SMPTE 259M-C: 270 Mb/s, 525/625 components. HD input, 8/10 bit according to SMPTE 292M-1998: 1.485 Gb/s.
Video Input (PCIe 3G)	BNC, 75 Ω (IEC 169-8)	SD input, 8/10 bit according to SMPTE 259M-C: 270 Mb/s, 525/625 components. HD input, 8/10 bit according to SMPTE 292M-1998: 1.485 Gb/s. 3G input, 8/10 bit according to SMPTE 422M-2006: 2.970 Gb/s.

PCIe L, LV, D, HD, 3G



LTC / GPI

Pin	Signal
3	LTC Input LTC-1
6	LTC Input LTC-2
4	LTC GND
5	

Pin	Signal
1	GPI 1 GND
2	GPI 1 I/O
7	GPI 2 GND
8	GPI 2 I/O

Switches

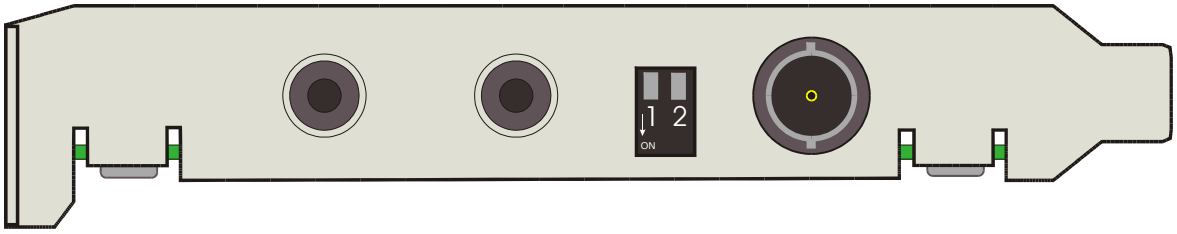
SW1	LTC Input
ON	Unbalanced LTC input to LTC-2
OFF	Balanced LTC input to LTC-1 and LTC-2

SW2	PCI LV: Video Input Termination	PCIe 3G PCIe HD PCIe D
ON	75 Ω termination	n/a
OFF	no termination	n/a

Technical data

Dimensions over all (Length x Height x Width)	**PCIe 3G:** 150 x 120 x 22 mm (standard profile slot) 150 x 80 x 22 mm (low profile slot) **PCIe L / LV / D / HD:** 191 x 120 x 22 mm (standard profile slot) 191 x 80 x 22 mm (low profile slot)
Weight	≈ 80 g
Operating voltage	3.3 VDC and 12 VDC
Power consumption	3 W
Ambient temperature	5 – 40 °C
Relative humidity	35 – 85 %

PCI L, LV, D, HD



LTC-1 LTC-2 SW1 SW2 Video Input

Switches

SW1	LTC Input
ON	Unbalanced LTC input to LTC-2
OFF	Balanced LTC input to LTC-1 and LTC-2

SW2	PCI LV: Video Input Termination	PCI D: Equalization	PCI HD:
ON	75 Ω termination	Gain equalization 270 Mb/s automatic up to 100m (Belden 8281, PSF 1/3 or equivalent)	n/a
OFF	no termination	Equalization bypass, used for short cable length (< 10m)	n/a

Technical data

Dimensions over all (Length x Height x Width)	141 x 120 x 22 mm
Weight	≈ 110 g
Operating voltage	3.3 VDC and 5 VDC
Power consumption	2 W
Ambient temperature	5 – 40 °C
Relative humidity	35 – 85 %

B4 Installation

For an installation in a PC with a Windows operating system please follow these instructions:

- Log in as Administrator and run the script "Install Driver.bat" from the CD-ROM shipped with the board. This installs the device driver for the PCI / PCIe board.
- Set the DIP switch to match the input signals of the PCI / PCIe board according to the previous passage. You can change the setting if required, even after the board has been installed.
- Shut down the PC and switch it off.
- Slide the PCI / PCIe board into a free slot. Ensure a proper fit in the PCI slot and then tighten the board into the slot, with the existing screw.
- Switch on the PC, boot it up and log in as Administrator.
- Windows will detect the new board automatically. If it asks for the driver disk, insert the CD-ROM shipped with the PCI / PCIe board. Alternatively, you may download the latest version from https://www.plurainc.com/downloads, unpack it and choose it as source for the driver.
- With this step the installation of the driver is complete. You may now test the PCI / PCIe board using PclTest.exe or IntTest.exe.

For an installation in a PC with Linux operating system please follow these instructions:

- Download and unzip the "PCI SDK and Linux Driver" (file avpcisdk.zip) to a new folder on your hard disk.
- Change to folder "Linux". Check that there are no spaces (blanks) in the path to this folder.
- The Linux driver is shipped as C source code and has to be compiled before first use.
- Open file "README" with a text editor and follow the instructions.

B5 Firmware Update

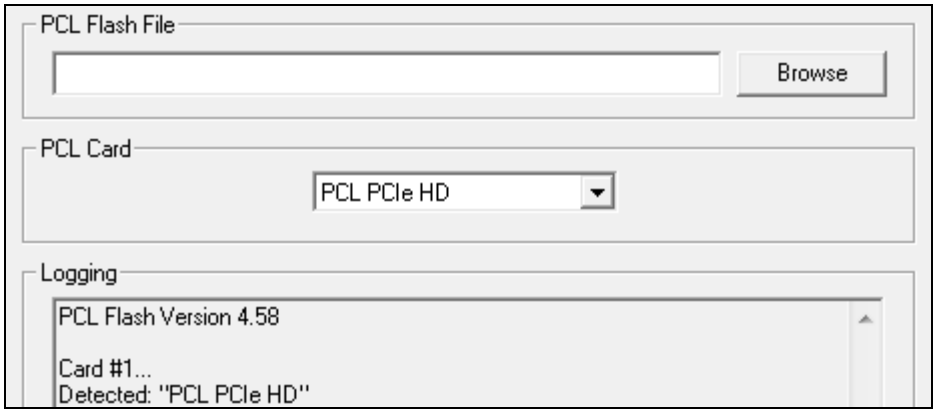
The PCIe and the PCI HD cards come with flash memory and can get firmware updates. That is described in this chapter. With the PCI LV, PCI L, PCI V and PCI TS cards a chip change is needed in to do that. Please contact Plura in this case.

Firmware updates require a computer with Windows or Linux operating system and the **PCLFlash** program. You can download the latest version of the program from https://plurainc.com/downloads.

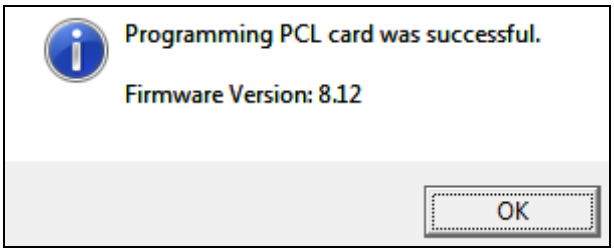
Store the new firmware as a **.tcf** file on a local hard drive of your computer.

For Windows operating system, please follow these steps:

1. Start the computer with the PCI / PCIe card plugged in, log in as Administrator and install the current driver (if not already done).
2. Execute **PCLFlash** on your computer. The program scans the computer for PCI / PCIe cards and gives a list of all devices found. Select the PCI / PCIe card you with to update from the drop-down list.



3. Open the **.tcf** file with "Browse". A click to "Flash!" starts the update. The program checks whether the new firmware matches the correct type of the device. In case there is no match an error message appears: "Incompatible Flash Update File". Update starts automatically if everything is ok. Click the OK button at the end.



4. The Update is finished now. You may flash other cards or close **PCLFlash**.

For Linux operating system, open the "Linux" folder and follow the instructions in the "README" file.

C Programming

C1 AVPCL32 Driver

Windows Driver Structure

The Windows driver for the PCI / PCIe consists of two parts:

- **AvPcl32.dll** is the interface for the user program. It implements functions like pclOpen(), pclClose(), pclGetTc(), etc.
- **Avpcl6.sys** is the Low-Level-driver. It processes the access to the PCI bus.

Driver Files

The following directories and files are in the file "avpci.zip":

Root Directory

The root contains test programs that check the function of the PCI / PCIe as well as the driver installation.

- **PclTest.exe** is a simple WIN32 console application, that displays the PCI / PCIe Time Code. The source code is in the SDK in the „Samples" directory.
- **IntTest.exe** is a simple WIN32 console application, that displays the PCI / PCIe Time Code. Opposite to PclTest.exe it uses interrupts. The source code is in the SDK in the „Samples" directory.

"Driver" Directory

This directory and its sub directories contain all files needed for an installation of the PCI / PCIe on a PC with Windows operating system.

- **Avpcl6.inf** is the description of the device driver for installation.
- **Avpcl6.cat** is the digital signature of avpcl6.inf for 32-bit Windows operating systems.
- **Avpcl6.ntamd64.cat** is the digital signature of avpcl6.inf for 64-bit Windows operating systems.
- **Install Driver.bat** is a script to install the device driver. It checks if it is running on a 32-bit or 64-bit Windows operating system and, depending on the result, calls one of the scripts in the i386 or amd64 directory.
- **Pcl-Pci.inf** is the description of the PCI / PCIe driver for installation.
- **Pcl-Pci.cat** is the digital signature of pcl-pci.inf for 32-bit Windows operating systems.
- **Pcl-Pci.ntamd64.cat** is the digital signature of pcl-pci.inf for 64-bit Windows operating systems.
- **Update Driver.bat** is a script to update the driver. It checks if it is running on a 32-bit or 64-bit Windows operating system and, depending on the result, calls one of the scripts in the i386 or amd64 directory.

"Driver \ i386" Directory

This directory contains parts of the driver for 32-bit Windows operating systems.

- **AvPcl32.dll** is the interface between user programs and the Low-Level-driver avpcl6.sys. During installation it will be copied to the \Windows\system32 directory.
- **Avpcl6.sys** is the device driver. During installation it will be copied to the \Windows\system32\driver directory.
- **Difxapi.dll** is used by wdreg_gui.exe and wdreg.exe.
- **Install Driver.bat** is a script to install the device driver. It calls wdreg_gui.exe with appropriate parameters.
- **Update Driver.bat** is a script to update the driver. It calls wdreg_gui.exe with appropriate parameters.
- **Wdreg_gui.exe** is a program that installs device drivers.
- **Wdreg.exe** is the command line version of wdreg_gui.exe.

"Driver \ amd64" Directory

This directory contains parts of the driver for 64-bit Windows operating systems.

- **AvPcl32.dll** is the interface between 64-bit user programs and the Low-Level-driver avpcl6.sys. During installation it will be copied to the \Windows\system32 directory.
- **AvPcl64.dll** is the interface between 32-bit user programs and the Low-Level-driver avpcl6.sys, by using the WoW64 subsystem (Windows on Windows 64-bit). During installation it will be copied to the \Windows\sysWow64 directory and renamed to avpcl32.dll.
- **Avpcl6.sys** is the device driver. During installation it will be copied to the \Windows\system32\driver directory.
- **Difxapi.dll** is used by wdreg_gui.exe and wdreg.exe.
- **Install Driver.bat** is a script to install the device driver. It calls wdreg_gui.exe with appropriate parameters.
- **Update Driver.bat** is a script to update the driver. It calls wdreg_gui.exe with appropriate parameters.
- **Wdreg_gui.exe** is a program that installs device drivers.
- **Wdreg.exe** is the command line version of wdreg_gui.exe.

SDK Files and Linux Driver

The SDK files are located separate from the driver files in the file "avpcisdk.zip". It contains all necessary files to write PCI / PCIe user programs as well as the Linux driver.

Root Directory

- **History.txt** contains a brief history of the driver.

"Lib" Directory

This directory contains libraries shared by the sample code.

- **AvPcl.h** contains all declarations of the library AvPcl32.dll. The file is extensively commented and serves as a reference for the user.
- **AvPcl32.def** is the module definition file to AvPcl32.dll.

"Lib \ i386" Directory

This directory contains libraries to write 32-bit application programs.

- **AvPcl32.lib** is the 32-bit version of the import library to AvPcl32.dll.

"Lib \ amd64" Directory

This directory contains libraries to write 64-bit application programs.

- **AvPcl32.lib** is the 64-bit version of the import library to AvPcl32.dll.

"Samples" Directory

This directory contains the source code of simple PCI / PCIe test programs. They are written as WIN32 console applications and explain the general use of the PCI / PCIe.

- **AvPcl.h** is a link to the file AvPcl.h in the directory above.
- **PclTest.cpp** is the source code of the test program PclTest.exe.
- **IntTest.cpp** is the source code of the test program IntTest.exe.

"Samples \ msdev_5" Directory

This directory contains workspace and project files for Microsoft Visual C++ 5.0.

- **Sample.dsw** is the workspace file for both sample programs.
- **PclTest \ PclTest.dsp** is the project file for the PclTest sample program.
- **IntTest \ IntTest.dsp** is the project file for the PclTest sample program.

"Samples \ msdev_2008" Directory

This directory contains workspace and project files for Microsoft Visual Studio 2008.

- **Sample.sln** is the solution file for both sample programs.
- **PclTest \ PclTest.vcproj** is the project file for the PclTest sample program.
- **IntTest \ IntTest.vcproj** is the project file for the PclTest sample program.

"Pascal" Directory

This directory contains the necessary files to access the PCI / PCIe with Turbo/Borland Pascal or Delphi.

- **AvPcl.pas** contains the declarations for the access to AvPcl32.dll.
- **PclTest.pas** is a simple test program as WIN32 console application.

"Basic" Directory

This directory contains the information required to use the PCI / PCIe with Visual Basic.

- AvPci32.vb contains all declarations for 32-Bit Visual-Basic-Programs.

"Linux" Directory

This directory contains the Linux driver in source code. Please read the file "README" for additional information.

C2 Data access

This chapter explains the access to the PCI / PCIe on a hardware level. As far as one of the supported operating systems is used, this information is not needed, since the board will be accessed by the driver shipped with the board, as described in the chapter above.

Memory vs. I/O access

There are three blocks inserted in the memory resp. I/O address space by the PCI / PCIe:

Block	Memory or I/O	Size	Description
BAR0	Memory	PCI: 0x80, PCIe: 0x100	Local configuration register
BAR1	I/O	PCI: 0x80, PCIe: 0x100	Local configuration register
BAR2	Memory	0x100	PCI / PCIe register set

The local configuration registers are used to configure the PCI interface chips. They should not be modified.

The PCI register set is mapped memory address space only.

Register set

Only the least significant byte of all the 64 long words is used. Therefore, only addresses, which can be divided by four, can be accessed with one byte cycles. This results in a maximum of 64 bytes, but presently only 21 bytes are used.
The PCI / PCIe register set is defined as following:

PCI / PCIe		Register	Description
Offset	r/w		
0x00	r	DATA0	Data from PCI / PCIe
0x04	r	DATA1	"
0x08	r	DATA2	"
0x0C	r	DATA3	"
0x10	r	DATA4	"
0x14	r	DATA5	"
0x18	r	DATA6	"
0x1C	r	DATA7	"
0x20	r	DATA8	"
0x24	r	DATA9	"

PCI / PCIe		Register	Description
0x28	r	DATAA	"
0x2C	r	INTFLGL	Interrupt flags L
0x30	r	CMDR	Command response
0x34	r	ACK	Acknowledge from PCI / PCIe
0x38	w	CMD	Command to PCI / PCIe
0x3C	w	CMDD0	Data to PCI / PCIe
0x40	w	CMDD1	"
0x44	w	CFG	PCI / PCIe configuration
0x48	r	INTFLGH	Interrupt flags H
0x4C	w	INTACK	Interrupt acknowledge
0x7C	r	VERSION	PCI / PCIe chip version

Every register permits data transfers only in one direction, either reading or writing. A read only register must not be written to, a write only register must not be read from.

Register

The registers in particular:

DATA0 to DATAA

The commands will return **data** via these registers.

INTFLGL

This register contains the **interrupt flags**. For every interrupt source the corresponding flag indicates, that an interrupt condition has occurred (see also INTFLGH). The bits are coded as following:

Bit	Hex	Interrupt Request
0	0x01	Mixed register
2	0x04	LTC register
3	0x08	VITC register
6	0x40	Odd video field (1st field)
7	0x80	Even video field (2nd field)

E.g. if bit '2' is set, a new LTC has been read. It doesn't matter if the interrupt request was masked (enabled) or not (see command pclIntMask), i.e. if it initiated a hardware interrupt or not. With that, it's possible to use this register for polling mode.

Interrupt requests have to be acknowledged. This is done by reading the appropriate register for Mixed-, LTC- and VITC-bits. The other bits can be acknowledged by the pclIntAck command.

CMDR

After servicing the command, PCI / PCIe returns the command number into this register. With that it can be verified, that the last command has been executed correctly. With the **command response** it is possible to decide how the data in DATA0 to DATAA have to be interpreted.

If an unknown command has been detected, 0xFF will be written to this register.

ACK

After executing any command, this register will be incremented = **acknowledge**.

CMD

To send a **command** to the PCI / PCIe, the command number has to be written to this register.

CMDD0 and CMDD1

With this register **command data** can be transferred.

CFG

This register serves diagnostic purposes only. It should not be written to.

INTFLGH (PCI / PCIe HD and 3G only)

This register contains the **interrupt flags**. For every interrupt source the corresponding flag indicates, that an interrupt condition has occurred (see also description of INTFLGL). The bits are coded as following:

Bit	Hex	Interrupt Request
0	0x01	ATC LTC
1	0x02	ATC VITC
3	0x08	HANC LTC
4	0x10	HANC VITC
5	0x20	VBI
6	0x40	ANC

INTACK

Writing to this register will **acknowledge** a hardware **interrupt** of the PCI / PCIe. Additionally, the interrupt request has to be acknowledged by a suitable command (e.g. pclIntAck).

VERSION

The chip **version** of the PCI / PCIe can be retrieved from this register. The chip type is coded in the high nibble (presently 0xA), the version number in the low nibble (presently 0x2). The chip type provides information about the type of PCI / PCIe, i.e. it has a LTC and/or a VITC reader. The version number will increment in future chip versions of the PCI / PCIe.

Commands

Almost every access to the PCI / PCIe is handled by commands. These are instructions to the PCI / PCIe, like setting a parameter or reading Time Code. A command is sent to the PCI / PCIe, explained below:

- If needed for the command, write command data to CMDD0 and CMDD1.
- Read ACK.
- Write command number to CMD.
- Wait, until ACK has changed. If this doesn't happen within 40ms, a timeout error has occurred.

- Verify if CMDR is equal to the command number. If not, a command error has occurred.
- If defined to the command, read the result of the command from DATA0 to DATAA.

The following commands are valid. The default values, valid after power-on, are put in parentheses. All what is described for VITC is valid for DVITC as well, unless differences have been shown explicitly. ATC and HANC are available only with PCI / PCIe HD and 3G.

Command	No.	Parameter	Result
pclReset	0x01	CMDD0: Operating mode (0x00) • 0x00: Normal mode • 0x80: Hardware reset • 0xF0: Reserved (test mode)	
	PCI / PCIe will be reset. All settings will be reset to the factory settings, i.e. the values that were present after power-on.		
pclIntMask	0x03	CMDD0: Interrupt mask L (0x00) • 0x00: All interrupts disable • 0x01: Mixed register • 0x04: LTC register • 0x08: VITC register • 0x40: Odd video field • 0x80: Even video field • 0xFF: All interrupts enable CMDD1: Interrupt mask H (PCI / PCIe HD and 3G only) (0x00) • 0x00: All interrupts disable • 0x01: ATC LTC • 0x02: ATC VITC • 0x08: HANC LTC • 0x10: HANC VITC • 0xFF: All interrupts enable	DATA0: Interrupt mask L • 0x01: Mixed register • 0x04: LTC register • 0x08: VITC register • 0x40: Odd video field • 0x80: Even video field DATA1: Interrupt mask H (PCI / PCIe HD and 3G only) • 0x01: ATC LTC • 0x02: ATC VITC • 0x08: HANC LTC • 0x10: HANC VITC
	Interrupts are enabled or disabled. CMDD0 and CMDD1 (PCI / PCIe HD and 3G) contain a bit mask, every bit set to "1" enables an interrupt source, every bit set to "0" disables it. It's possible to enable several interrupt sources simultaneously. The result is the current interrupt mask in DATA0 and DATA1 (PCI / PCIe HD and 3G).		

Command	No.	Parameter	Result
pclSetIntMask	0x04	CMDD0: Interrupt mask • 0x01: Mixed register • 0x04: LTC register • 0x08: VITC register • 0x40: Odd video field • 0x80: Even video field • 0xFF: All interrupts enable CMDD1: Interrupt mask H (PCI / PCIe HD and 3G only) (0x00) • 0x01: ATC LTC • 0x02: ATC VITC • 0x08: HANC LTC • 0x10: HANC VITC • 0xFF: All interrupts enable	DATA0: Interrupt mask • 0x01: Mixed register • 0x04: LTC register • 0x08: VITC register • 0x40: Odd video field • 0x80: Even video field DATA1: Interrupt mask H (PCI / PCIe HD and 3G only) • 0x01: ATC LTC • 0x02: ATC VITC • 0x08: HANC LTC • 0x10: HANC VITC
	Every bit set to "1" enables the corresponding interrupt source. The result is the current interrupt mask in DATA0 and DATA1 (PCI / PCIe HD and 3G).		
pclResetIntMask	0x05	CMDD0: Interrupt mask L • 0x01: Mixed register • 0x04: LTC register • 0x08: VITC register • 0x40: Odd video field • 0x80: Even video field • 0xFF: All interrupts disable CMDD1: Interrupt mask H (PCI / PCIe HD and 3G only) (0x00) • 0x01: ATC LTC • 0x02: ATC VITC • 0x08: HANC LTC • 0x10: HANC VITC • 0xFF: All interrupts disable	DATA0: Interrupt mask L • 0x01: Mixed register • 0x04: LTC register • 0x08: VITC register • 0x40: Odd video field • 0x80: Even video field DATA1: Interrupt mask H (PCI / PCIe HD and 3G only) • 0x01: ATC LTC • 0x02: ATC VITC • 0x08: HANC LTC • 0x10: HANC VITC
	Every bit set to "1" disables the corresponding interrupt source. The result is the current interrupt mask in DATA0 and DATA1 (PCI / PCIe HD and 3G).		
pclReaderEnable	0x10	CMDD0: Reader mask (0x06) • 0x00: LTC/VITC off • 0x02: LTC on • 0x04: VITC on CMDD1: Reader mask (0x1F) (PCI / PCIe HD and 3G only) • 0x00: ATC/HANC all off • 0x01: ATC LTC on • 0x02: ATC VITC on • 0x08: HANC LTC on • 0x10: HANC VITC on	

Command	No.	Parameter	Result
	Every bit set to "1" switches on the corresponding reader, every bit set to "0" switches it off. "Switched off" means, that there won't be new data from the reader, even if its own register is requested.		
pclVitcSetupLines	0x14	CMDD0: First line • Range: 6 to 26 CMDD1: Last line • Range: 6 to 26 CMDD0 must be less or equal to CMDD1.	DATA0: Error code • 0x00: Success • 0x01: Range error in first line • 0x02: Range error in last line • 0x03: First line > last line
	The video lines for the VITC reader are set. For PAL video: lines 6 to 22 are specified. For NTSC video: lines 10 to 20 are specified. After power-on the maximum range is selected.		
pclVitcSetupMode	0x15	CMDD0: VITC reader mode (0x01) • 0x00: Two lines • 0x01: Block of lines	
	The mode of the VITC reader is set. VITC can be read from two separate lines or from a block of lines. In case first line = last line (see command 0x14), only one line is selected.		
pclVitcLevelControl	0x16	CMDD0: VITC/DVITC level control • 0x00: Automatic mode on • 0x01: Automatic mode and set threshold value from CMDD1 • 0x02: Automatic mode and initialise the adjustment • 0x03: Manual mode, keeps the current value • 0x04: Manual mode and set threshold value from CMDD1 CMDD1: Threshold value, if CMDD0 = 0x01 or 0x04. Range 0x00 - 0x5A.	DATA0: Current value of threshold • Range: 0x00 to 0x5A
	The automatic level matching mode of the VITC/DVITC reader and the threshold value itself can be set. See corresponding chapter for details. The default for VITC is with automatic, for DVITC without automatic. As the result the current threshold value is returned in DATA0.		
pclLtcError	0x17	CMDD0: Access mode • 0x00: Get error counter and initialise it afterwards • 0x01: Get error counter only • 0x02: Get error counter and reset it afterwards	DATA0: Current error counter
	Request LTC error counter: counts errors like not plausible time and time discontinuities.		

Command	No.	Parameter	Result
pclVitcError	0x18	CMDD0: Access mode • 0x00: Get error counter and initialise it afterwards • 0x01: Get error counter only • 0x02: Get error counter and reset it afterwards	DATA0: Current error counter
	Request VITC error counter: counts errors like not plausible time and time discontinuities.		
pclFlyWheel	0x19	CMDD0: Switch fly wheel function on or off (0x00) • 0x00: Fly wheel off • 0x02: LTC fly wheel on • 0x04: VITC fly wheel on • 0x06: Both VITC and LTC fly wheel on	
	The fly wheel function serves as drop-out compensation, see corresponding chapter for details. This is available for LTC and VITC Time Codes, not for ATC or HANC Time Codes.		
pclMixedEnable	0x20	CMDD0: Reader mask (0x06) • 0x00: Switch all reader off • 0x02: Switch on LTC reader • 0x04: Switch on VITC reader	
	Every bit set to "1" enables the corresponding reader for the mixed register. This is done by a priority scheme. Every bit set to "0" will be ignored, but it is still possible to get the reader's data from its own reading register. This is available for LTC and VITC Time Codes, not for ATC or HANC Time Codes.		
pclPriority	0x21	CMDD0: Priority (0x01) • 0x01: VITC before LTC • 0x02: LTC before VITC	
	The priority defines, which Time Code data are transferred to the mixed register, if Time Code is read by both enabled readers.		
pclAtcLtcError	0x22	CMDD0: Access mode • 0x00: Get error counter and initialise it afterwards • 0x01: Get error counter only • 0x02: Get error counter and reset it afterwards	DATA0: Current error counter
	Request ATC LTC error counter: counts errors like not plausible time and time discontinuities.		

Command	No.	Parameter	Result
pclAtcVitcError	0x23	CMDD0: Access mode • 0x00: Get error counter and initialise it afterwards • 0x01: Get error counter only • 0x02: Get error counter and reset it afterwards	DATA0: Current error counter
	Request ATC VITC error counter: counts errors like not plausible time and time discontinuities.		
pclHancLtcError	0x25	CMDD0: Access mode • 0x00: Get error counter and initialise it afterwards • 0x01: Get error counter only • 0x02: Get error counter and reset it afterwards	DATA0: Current error counter
	Request HANC LTC error counter: counts errors like not plausible time and time discontinuities.		
pclHancVitcError	0x26	CMDD0: Access mode • 0x00: Get error counter and initialise it afterwards • 0x01: Get error counter only • 0x02: Get error counter and reset it afterwards	DATA0: Current error counter
	Request HANC VITC error counter: counts errors like not plausible time and time discontinuities.		
pclGpi1H:M (Option GPI OUT)	0x28	CMDD0: Hours CMDD1: Minutes	
	Transfer the event time of GPI_1. See command 0x2A as well.		
pclGpi1S:F (Option GPI OUT)	0x29	CMDD0: Seconds CMDD1: Frames	
	Transfer the event time of GPI_1. See command 0x2A as well.		

Command	No.	Parameter	Result
pclGpi1Init (Option GPI OUT)	0x2A	CMDD0: Mode • Bits 0..2: Signal Output 0 = static High 1 = static Low 2 = Low pulse, recurrent 3 = High pulse, recurrent 4 = Low pulse, one shot 5 = High pulse, one shot 6 = ↓ edge, one shot 7 = ↑ edge, one shot • Bit 3: Time Comparator without (=0) or with (=1) Frames • Bits 4..5: Time Comparator 0 = "=" (LTC forward only) 1 = ">=" (LTC forward only) 2 = "=" 3 = ">=" • Bits 6..7: Pulse Width 0 = 10 ms approx. 1 = 40 ms approx. 2 = 500 ms approx. 3 = 1000 ms approx. CMDD1: Initialisation • = 0x00: GPI_1 initialisation, i.e. the output returns to its initial state and the next event triggers the chosen function. • <> 0x00: Initialisation plus the time transferred by commands 0x28 and 0x29 is set to the event register.	
	Select mode/function of GPI_1 and initialise.		
pclGpi1Status (Option GPI OUT)	0x2B		DATA0: no event (= 0) or event (<> 0) detected since last initialisation. DATA1: <> 0, if GPI_1 signal pulse still is active (= represents the timer of the pulse width).
	Status request of GPI_1.		
pclGpi2H:M (Option GPI OUT)	0x2C	CMDD0: Hours CMDD1: Minutes	
	Transfer the event time of GPI_2. See command 0x2E as well.		
pclGpi2S:F (Option GPI OUT)	0x2D	CMDD0: Seconds CMDD1: Frames	
	Transfer the event time of GPI_2. See command 0x2E as well.		

Command	No.	Parameter	Result
pclGpi2Init (Option GPI OUT)	0x2E	CMDD0: Mode (same as for GPI_1, see command 0x2A for description). CMDD1: Initialisation (same as for GPI_1, see command 0x2A for description).	
	Select mode/function of GPI_2 and initialise.		
pclGpi2Status (Option GPI OUT)	0x2F		DATA0: no event (= 0) or event (<> 0) detected since last initialisation. DATA1: <> 0, if GPI_1 signal pulse still is active (= represents the timer of the pulse width).
	Status request of GPI_2.		
pclTcFrames	0x30	CMDD0: Frame rate (0x00) • 0x00: Auto • 0x01: 24 fps • 0x02: 25 fps • 0x03: 30 fps with drop • 0x04: 30 fps without drop • 0x05: Initialize to „Unknown“, then Auto	
	The frame rate of the Time Code can be detected automatically from the reader values, or it can be set to a fixed value. If you set the frame rate with this command, it fixes the frame rate for all Time Code readers. It is possible to run both Time Codes (LTC or video Time Code) at different frame rates, using the two following functions.		
pclLtcFrames	0x31	CMDD0: Frame rate (0x00) • 0x00: Auto • 0x01: 24 fps • 0x02: 25 fps • 0x03: 30 fps with drop • 0x04: 30 fps without drop • 0x05: Initialize to „Unknown“, then Auto	
	The frame rate of the LTC can be detected automatically from the reader values, or it can be set to a fixed value. For the PCI / PCIe HD and 3G board this command sets the frame rate of the ATC LTC and HANC LTC as well. The ‘pclGetRegister’ command returns the current frame rate at DATA9.		

Command	No.	Parameter	Result
pclVitcFrames	0x32	CMDD0: Frame rate (0x00) • 0x00: Auto • 0x01: 24 fps • 0x02: 25 fps • 0x03: 30 fps with drop • 0x04: 30 fps without drop • 0x05: Initialize to „Unknown“, then Auto	
	The frame rate of the video Time Code (VITC/DVITC) can be detected automatically from the reader values, or it can be set to a fixed value. For the PCI / PCIe HD and 3G board this command sets the frame rate of the ATC VITC and HANC VITC as well. The ‘pclGetRegister’ command returns the current frame rate at DATA9.		
pclSendIdata	0x38	CMDD0: IDATA Address	DATA0: IDATA Data
pclSetIdata	0x39	CMDD0: IDATA Address CMDD1: IDATA Data	
pclSendPdata	0x3A	CMDD0: PDATA Address	DATA0: PDATA Data
pclSetPdata	0x3B	CMDD0: PDATA Address CMDD1: PDATA Data	
pclSendXdata	0x3C	CMDD0: XDATA Address H CMDD1: XDATA Address L	DATA0: XDATA Data
	The Idata- Pdata- and Xdata-Commands are designed for diagnostic purposes only.		
pclGetTc	0x41		DATA0: Frames DATA1: Seconds DATA2: Minutes DATA3: Hours
	Read the time information of the Time Code from mixed register. The data will be in packed BCD format.		
pclGetUser	0x42		DATA0: U1/2 (“frames”) DATA1: U3/4 (“seconds”) DATA2: U5/6 (“minutes”) DATA3: U7/8 (“hours”)
	Read the user bits of the Time Code from mixed register		

Command	No.	Parameter	Result
pclGetRegister	0x43	CMDD0: Register number • 0x00: Mixed-Register • 0x02: LTC-Register • 0x03: VITC-Register • 0x11: ATC LTC • 0x12: ATC VITC • 0x14: HANC LTC • 0x15: HANC VITC	DATA0: Frames DATA1: Seconds DATA2: Minutes DATA3: Hours DATA4: Flag bits LTC • 0x01: LTC bit 10 (30 Drop) • 0x02: LTC bit 11 (CF) • 0x04: LTC bit 27 • 0x08: LTC bit 43 • 0x10: LTC bit 58 • 0x20: LTC bit 59 • 0x40: 0 = read from LTC reader, 1 = counted from fly wheel. • 0x80: 0 = LTC forward, 1 = LTC reverse or DATA4: Flag bits VITC • 0x01: VITC bit 14 (30 Drop) • 0x02: VITC bit 15 (CF) • 0x04: VITC bit 35 (30 Field) • 0x08: VITC bit 55 • 0x10: VITC bit 74 • 0x20: VITC bit 75 (25 Field) • 0x40: 0 = read from VITC reader, 1 = counted from fly wheel. DATA5: U1/2 (“frames”) DATA6: U3/4 (“seconds”) DATA7: U5/6 (“minutes”) DATA8: U7/8 (“hours”) DATA9: Frame rate • 0x24: 24 fps • 0x25: 25 fps • 0x30: 30 fps without drop • 0xB0: 30 fps with drop DATAA: New data flags • 0x00: No new data • 0x01: ATC LTC • 0x02: ATC VITC • 0x08: HANC LTC • 0x10: HANC VITC • 0x40: LTC • 0x80: VITC/DVITC
	Utilize this command to read the time, the user bits, the flag bits and status information out of the various Time Code registers.		

Command	No.	Parameter	Result
pclGetRegStatus	0x44	CMDD0: Register number • 0x11: ATC LTC • 0x12: ATC VITC • 0x14: HANC LTC • 0x15: HANC VITC	DATA0: DID1 (ATC) DATA1: DID 2 (ATC) or Line Select (HANC VITC)
	Read extended register status.		
pclSetDBB1	0x45	CMDD0: ATC register CMDD1: DBB1	
	Set DBB1 of ATC register		
pclGetVersion	0x46		DATA0: Product identification • 0x00: Normal firmware • other: Special version DATA1: Version number • Bit 7..6: 0x40 = PCI / PCIe • Bit 5..4: Main version • Bit 3..0: Sub version
	Read firmware version number and product type.		
pclIntAck	0x48	CMDD0: Interrupt mask L • 0x01: Mixed register • 0x04: LTC register • 0x08: VITC register • 0x40: Odd video field • 0x80: Even video field CMDD1: Interrupt mask H (PCI / PCIe HD and 3G only) • 0x01: ATC LTC • 0x02: ATC VITC • 0x08: HANC LTC • 0x10: HANC VITC	
	Acknowledge interrupt request for ever “1” in interrupt mask.		
pclSetDID	0x49	CMDD0: ANC register • 0x00: ATC • 0x01: HANC LTC • 0x02: HANC VITC CMDD1: DID	
	Set DID of ANC register		
pclSetSDID	0x4A	CMDD0: ANC register • 0x00: ATC • 0x01: HANC LTC • 0x02: HANC VITC • CMDD1: SDID	
	Set SDID of ANC register		

Command	No.	Parameter	Result
pclGetVide-oStandard	0x4B		• DATA0: Video System 0x00: Unknown SD Video 0x01: 625i25 0x02: 525i29.97 0x03: 1080i25 0x04: 1080i29.97 or 30 0x05: 1080i24 0x06: 720p50 0x07: 720p59.94 or 60 0x08: 1080p25 0x09: 1080p29.97 or 30 0x0A: 1080p/24 0x0B: 720p24 0x0C: 720p25 0x0D: 720p29.97 or 30 0x0E: 1035i29.97 or 30 0x0F: Unknown HD Video 0x10: Unknown 3G Video 0x11: 1080p50 0x12: 1080p59.94 or 60 The following video standards can only be detected with the help of SMPTE 352M Payload Identifier packets: 0x13: 720p23.98 0x14: 720p24 0x15: 720p29.97 0x16: 720p30 0x17: 720p59.94 0x18: 720p60 0x19: 1080i25 0x1A: 1080i29.97 0x1B: 1080i30 0x1C: 1080psF23.98 0x1D: 1080psF24 0x1E: 1080psF25 0x1F: 1080psF29.97 0x20: 1080psF30 0x21: 1080p23.98 0x22: 1080p24 0x23: 1080p29.97 0x24: 1080p30 0x25: 1080p59.94 0x26: 1080p60 • DATA1: Format 0x00: Interlaced 0x01: Progressive
	Read information about connected video signal (D / HD / 3G only)		

Command	No.	Parameter	Result
pclGetVideoRaster	0x4C		• DATA0: Words / active line (low byte) • DATA1: Words / active line (high byte) • DATA2: Total words / line (low byte) • DATA3: Total words / line (high byte) • DATA4: Active lines / field (low byte) • DATA5: Active lines / field (high byte) • DATA6: Total lines / frame (low byte) • DATA7: Total lines / frame (high byte)
	Read information about connected video signal (D / HD / 3G only)		
pclSetAncDID	0x4D	CMDD0: User ANC DID CMDD1: User ANC SDID	
	Set DID and SDID of the user defined ANC reader		
pclGetAncData	0x4E		• DATA2: descriptor address pointer (low byte) DATA3: descriptor address pointer (high byte)
	Read first address pointer of the user defined ANC reader		
pclGetGpi	0x4F		• DATA0: GPI IN Status 0x01: GPI1 active 0x02: GPI2 active DATA1: GPI OUT Status 0x01: GPI1 active 0x02: GPI2 active
	Get status of GPI In/Out		
pclGetPayload-Identifier	0x50		• DATA0: S352 Byte 1 • DATA1: S352 Byte 2 • DATA2: S352 Byte 3 • DATA3: S352 Byte 4 • DATA4: Flags 0x01: A valid S352M packed is present in the video. 0x02: The S352M packet conflicts with the detected video. 0x04: Use S352M packets to detect video system
	Get payload identifier (SMPTE 352M)		
pclSetUsePayload-Identifier	0x51	CMDD0: • 0x00: Ignore S352M packets • 0x04: Detect video system from S253M packets	

Command	No.	Parameter	Result
	Set use of Payload Identifier (SMPTE 352M)		
pclGetAncDataTab	0x52	CMDD0: Number of 1st ANC packet	• DATA0: Number of 1st ANC packet • DATA1: Number of ANC de-scriptors following • DATA2: 1st descriptor address pointer (low byte) • DATA3: 1st descriptor address pointer (high byte) DATA4: 2nd descriptor address pointer (low byte) • DATA5: 2nd descriptor address pointer (high byte) DATA6: 3rd descriptor address pointer (low byte) • DATA7: 3rd descriptor address pointer (high byte) • DATA8: 4th descriptor address pointer (low byte) • DATA9: 4th descriptor address pointer (high byte)
	Read multiple address pointers of the user defined ANC reader		
pclSetGpiEvent	0x53	CMDD0: • 0x00: Disable all GPI1 events • 0x01: Enable GPI1 event with falling edge • 0x02: Enable GPI1 event with rising edge • 0x10: Disable all GPI2 Events • 0x11: Enable GPI2 event with falling edge • 0x12: Enable GPI2 event with rising edge CMDD1: Register number • 0x00: Mixed register • 0x02: LTC register • 0x03: VITC register • 0x11: ATC LTC • 0x12: ATC VITC • 0x14: HANC LTC • 0x15: HANC VITC	
	Set or clear GPI input 1 or 2 events		

Command	No.	Parameter	Result
pclGetGpiEvent	0x54	CMDD0: • 0x01: Get GPI1 event • 0x02: Get GPI2 event	DATA0: Frames DATA1: Seconds DATA2: Minutes DATA3: Hours DATA4: Flag bits DATA5: U1/2 („Frames“) DATA6: U3/4 („Seconds“) DATA7: U5/6 („Minutes“) DATA8: U7/8 („Hours“) DATA9: Register number • 0x00: Mixed register • 0x02: LTC register • 0x03: VITC register • 0x11: ATC LTC • 0x12: ATC VITC • 0x14: HANC LTC • 0x15: HANC VITC
	Get data of GPI1 or GPI2 events		
pclGetGpiEvents	0x55		• DATA0: GPI IN Status 0x01: GPI1 active 0x02: GPI2 active • DATA1: GPI IN change 0x01: GPI1 had changed 0x02: GPI2 had changed
	Get status of GPI1 and GPI2 events		

D Options

D1 “GPI OUT“: Two Signal Outputs

Description

For this option PCI needs an additional assembly and a special firmware. The two signal outputs, called GPI_1 and GPI_2 (GPI = General Purpose Interface), are independent from each other. For each output you can select (by commands) one of the following functions:

Output Signal:

- Static High (“H”: see technical data);
- Static Low (“L”: see technical data);
- Low pulse, recurrent with each event;
- High pulse, recurrent with each event;
- Low pulse, one shot, with the next event. After the event a new initialisation is required;
- High pulse, one shot, with the next event. After the event a new initialisation is required;
- Falling edge (level changes from H to L), one shot, with the next event. After the event a new initialisation is required;
- Rising edge (level changes from L to H), one shot, with the next event. After the event a new initialisation is required.

Pulse Width, four steps:

- 10 ms approx.
- 40 ms approx.
- 500 ms approx.
- 1000 ms approx.

An event will be defined by a comparative time (time code) and a condition. The time (HH:MM:SS:FF) is transferred to PCI by commands for each signal output separately. Is the output signal not selected as a static signal, the PCI compares the time with the time code of the Mixed Register and checks the condition. One of the following conditions can be selected by command:

- read time of the time code = comparative time (in case of LTC: time code has to be in ascending order);
- read time of the time code > or = comparative time (in case of LTC: time code has to be in ascending order);
- read time of the time code = comparative time;
- read time of the time code > or = comparative time.

Additionally, it can be selected to compare the time with or without frames.

Commands 0x2A and 0x2B set the corresponding signal output to its initial state. The GPI now is ready for the next event. After this initialisation and after each event at least one frame of the time code input must not match to the condition, otherwise this event will be lost. In a Time

Table application, a "recurrent pulse mode" should be selected. After an event the next comparative time should be transferred as fast as possible (within one frame). Two frames distance will be the maximum rate of recurrent events of one GPI. The signal outputs are independent from each other; thus, it is possible to indicate an event by GPI_2 in a one frame distance to an event by GPI_1.

The following remark describes the difference between time compare with or without frames and condition "=" or ">=":

The output signal will be triggered frame accurate if the time is compared with frames. But the time code reader must read this particular frame, i.e., the time value has to be within the valid range (dependent on the frame rate) and no drop-out should occur (bad time code or NTSC drop frame). The ">=" condition may avoid a drop-out problem. With this condition the application has to be started correctly, i.e., after initialisation a time which is < the comparative time has to be read. Time compare without frames lets an event occur at a new second. In this mode there should be no drop-out problem.

After power-on the PCI sets itself to:

- static "H";
- pulse width 10 ms;
- read time of the time code = comparative time (in case of LTC: time code has to be in ascending order);
- time compare without frames.

Commands

This option requires new commands:

No. of Command	Command
0x28	Hours:Minutes of the comparative time GPI_1
0x29	Seconds:Frames of the comparative time GPI_1
0x2A	Mode of GPI_1 + initialisation
0x2B	GPI_1 status request
0x2C	Hours:Minutes of the comparative time GPI_2
0x2D	Seconds:Frames of the comparative time GPI_2
0x2E	Mode of GPI_2 + initialisation
0x2F	GPI_2 status request

For a detailed description see chapter C.

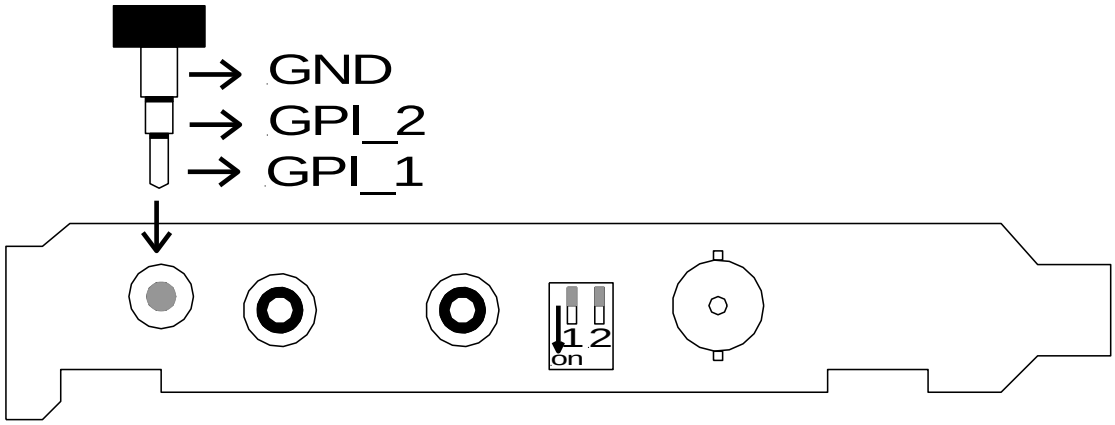
Connections and Technical Data

PCIe

The connections are described in chapter B3.

PCI

GPI_1 = Signal 1, GPI_2 = Signal 2 are connected at a stereo jack socket. The corresponding stereo jack plug is part of the delivery.



Technical data

General	Open-collector output of a npn transistor. Maximum power dissipation: 200 mW.
H-state	Open-collector output floating (off-state). To drive a logic High add an external pull-up resistor. Typical values of the pull-up: 0.3 kΩ - 6 kΩ @3V 0.5 kΩ - 10 kΩ @5V 1.2 kΩ - 24 kΩ @12V 1.5 kΩ - 30 kΩ @15V 2.4 kΩ - 48 kΩ @24V Maximum off-state voltage: 30 V DC.
L-state	Output driven to ground (on-state). Switching current (sinking): max. 100 mA DC. On-state voltage: @100 mA: typical 200 mV (≤ 600 mV) @10 mA: typical 90 mV (≤ 250 mV)

Availability

PCI / PCIe Board	GPI_1	GPI_2	Option / Standard
PCI L	Input / Output	Input / Output	Option
PCI LV	Input / Output	Input / Output	Option
PCI D	Input / Output	Input / Output	Option
PCI D (v2)	Input / Output	Input only	Option
PCI HD	Input / Output	Input only	Option
PCIe L	Input / Output	Input only	Standard
PCIe L (v2)	Input / Output	Input / Output	Standard
PCIe LV	Input / Output	Input only	Standard
PCIe LV (v2)	Input / Output	Input / Output	Standard
PCIe D	Input / Output	Input only	Standard
PCIe HD	Input / Output	Input only	Standard
PCIe 3G	Input / Output	Input / Output	Standard
PCIe 3G (v2)	Input / Output	Input / Output	Standard

Contact Us

plura®

Corporate Offices:
Plura Broadcast, Inc.
Ph: +1-602-944-1044
Sales@plurainc.com

Plura Europe GmbH
Ph: +49-6725-918006-70
Sales@plurainc.com

Plura MEA
Ph: +971-50-715-9625
Sales@plurainc.com

Plura Asia
Ph: +82-10-6688-8826
Sales@plurainc.com