UniversalPps Analyzer

Reference Manual

Product Info	
Product Manager	Sven Meier
Author(s)	Sven Meier
Reviewer(s)	-
Version	1.6
Date	31.03.2023

Copyright Notice

Disclaimer

Overview

NetTimeLogic's PPS Analyzer is specifically designed for (PTP) Plugfests where multiple devices are synchronizing each other, and the accuracy of the individual devices shall be measured via PPS (offset from reference PPS). The device has 8 PPS inputs that are measured simultaneously, and it synchronizes itself to an additional reference PPS input. Additionally, it has a PPS output of the synchronized clock which is used for PPS measurement and Threshold signals which can be set when a measurement exceeds. If NetTimeLogic's PPS Analyzer Shield is used, it can read calibration values from an EEPROM and has a calibration PPS. Multiple PPS Analyzers can be connected to the same host and are all discovered automatically. It uses a Serial interface (mostly over USB) or Ethernet (UDP/IP) to access the registers in the FPGA. In the FPGA it uses NetTimeLogic's configuration IP which represents an AXI Master to the other IP cores. It uses a proprietary protocol to convert the Serial or Ethernet data stream from/to AXI register access. The core part consists of the following NetTimeLogic IP cores: PPS Slave IP core, PPS Master IP core, Adjustable Counter Clock IP core and multiple instances of the Signal Timestamper IP core. The tool needs no configuration and self-discovers all cores available in the design.

The PPS analyzer is implemented on an Arty Development Board from Digilent © with a specific shield which contains termination resistors and 3.3V buffers which allow PPS inputs of up to 15V. NetTimeLogic also sells its ready to use PPS Analyzer schield which gives you the most functionality:

Key Features:

- 8 PPS Inputs
- 1 Reference PPS Input
- 1 Reference PPS Output
- Serial Connection over USB
- Ethernet Connection (requires EEPROM or otherwise fixed IP is used)
- Powered via USB or via 7-15V Powerjack (~1A)
- Auto discovery of connected PPS Analyzers
- Multi Analyzer capable
- Multi User capable (Ethernet Connection only)
- Optional 1 Calibration PPS Output (requires NTL Shield)
- Optional 2 Threshold Outputs (High Low) (requires NTL Shield)

- Optional EEPROM for Delay compensation and configuration values (requires NTL Shield)
- Synchronized Clock via PPS
- Timestamp resolution 4ns
- PPS compensated for synchronization error introduced by the reference PPS
- Delay compensation of input and output buffer delays (requires EEPROM)
- Cable Compensation individually on each PPS input
- Save Screen as PNG, JPG or TIFF
- Sliding Window Screen shows up to the last 100000 measurements
- Calibration Mechanism (requires extra cabling or calibration shield)
- Log values as CSV

Revision History

This table shows the revision history of this document.

Version	Date	Revision
0.1	16.04.2018	First draft
1.0	21.06.2018	Added logging
1.1	15.02.2019	Added delay screen
1.2	10.02.2020	Added threshold screen
1.3	06.04.2020	Added protocol and Register map
1.4	09.07.2020	Added ethernet connection
1.5	06.11.2020	Added selective interfaces
1.6	31.03.2023	Added Electrical Specification, added Shield Revision B, added ArtyA7-100T and ArtyS7-50 support

Table 1: Revision History

Content

Definitions

Definitions	
PPS Slave Clock	A clock that can synchronize itself to a PPS input
PPS Master Clock	A clock that generates a PPS to synchronize other nodes via a PPS output
Signal Timestamper	A core which takes timestamps on a edge of a signal and can compensate delays
Offset	Phase difference between clocks
Drift	Frequency difference between clocks

Table 2: Definitions

Abbreviations

Abbreviations	
AXI	AMBA4 Specification (Stream and Memory Mapped)
CSV	Character separated value
PPS	Pulse Per Second
PS	PPS Slave
PM	PPS Master
TS	Timestamp
FPGA	Field Programmable Gate Array
VHDL	Hardware description Language for FPGA's

Table 3: Abbreviations

1 Introduction

1.1 Context Overview

The PPS Analyzer is designed to compare PPS references from multiple sources simultaneously and check their accuracy against a reference PPS. It synchronizes itself to the reference PPS input and generates a reference PPS from this synchronized clock. The PPS is compensated for the input and output delays introduced by the 3.3V buffers. The 3.3V buffers are optional but it has to be taken care of, that the PPS input and output pins are directly connected to the FPGA input via a 200 Ohm serial resistor. Also the design compensates the delay of the 3.3V buffer which means that if the buffers are not inserted the reference clock has an offset of the 3.3V buffers. This has a minimal influence on the accuracy measurement since the delay applies to all Inputs including the reference PPS but the buffers might have different delays. If not compensated the output PPS is then shifted by the sum of output and input delay.
All PPS cables shall be of the same length, but can be compensated for in the application.

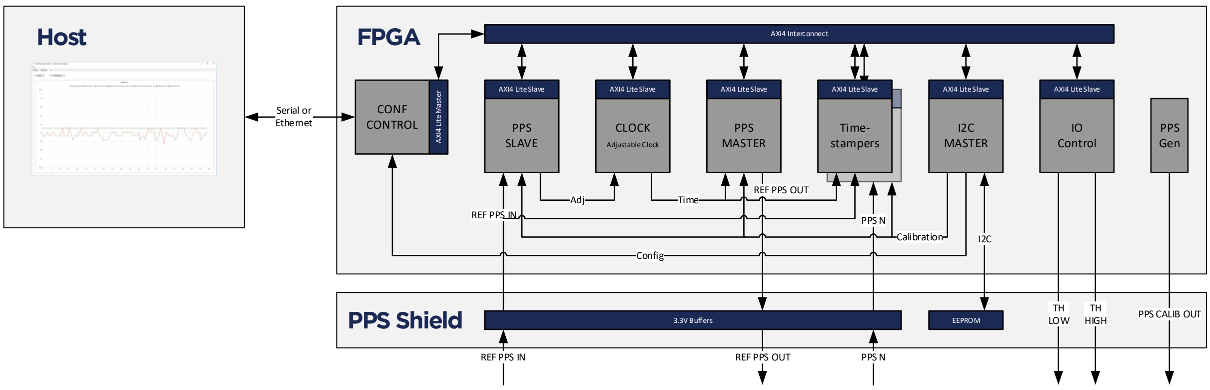


Figure 1: Context Block Diagram

1.2 Function

The PPS Analyzer synchronizes itself to the reference PPS via the NetTimeLogic PPS Slave IP core and generates a reference PPS based on the synchronized clock via the NetTimeLogic PPS Master IP Core. The 8 PPS inputs are timestamped via the NetTimeLogic Signal Timestamper IP core on the rising edge of the PPS input. The timestamps are then passed to the application where the difference to the next or last second overflow of the reference clock which is synchronized to the reference PPS is calculated. The current error of the reference PPS is also measured via an additional Timestamper and can be used to compensate the error introduced by the synchronization in the application. Additionally, the input and output buffer delays are compensated by calibration values stored in the EEPROM (if available). The application also checks for threshold values and asserts them (exceed High or Low) if measurements are exceeded

2 PPS Analyzer

2.1 Electrical Specification

Powered via USB or via 7-15V Powerjack (~1A)

PPS Input (PPS-IN-1 to 8):

- 3.3V TTL (Allowed input range: Low: 0V – 1V, High 2.5V - 15V)
- 4.7kOhm terminated
- Active high
- Min pulse width 1 ms

Reference PPS Input (REF-IN-PPS)

- 3.3V TTL (Allowed input range: Low: 0V – 1V, High 2.5V - 15V)
- 4.7kOhm terminated or no termination
- Active high
- Min pulse width 10 ms

Reference PPS Output (REF-OUT-PPS)

- 3.3V TTL,
- Max output current 25mA
- Active high
- Pulse width ~500 ms

2.2 Hardware

The PPS Analyzer hardware is built out of two components:

- The FPGA Module
- A custom Arduino Shield which contains the 3.3V buffers (M74HC4050B1R), the 4.7kOhm termination resistors to protect the FPGA from to high input voltages and the BNC connectors and optional threshold signals, a calibration PPS and threshold signals.

The ArtyA7-35T or ArtyA7 100T board is an FPGA board from Digilent Inc. with an Artix7 FPGA from AMD/Xilinx. (http://store.digilentinc.com/arty-board-artix-7-fpga-development-board-for-makers-and-hobbyists/).
If no Ethernet is required (communication only via USB-UART interface) it is also possible to use the ArtyS7-50 board from Digilent Inc. with an Spartan7 FPGA from

AMD/Xilinx. (https://digilent.com/shop/arty-s7-spartan-7-fpga-development-board/).

Since the shield contains no logic (EEPROM is optional) except of ensuring correct voltage levels and protecting the FPGA IO pins it can be easily built yourself, the only requirement is that the pins are connected to the correct IOs (See 0 for details). The shield can be bought as full assembled and calibrated kit or as a DIY kit: https://www.nettimelogic.com/tools-pps-analyzer.php

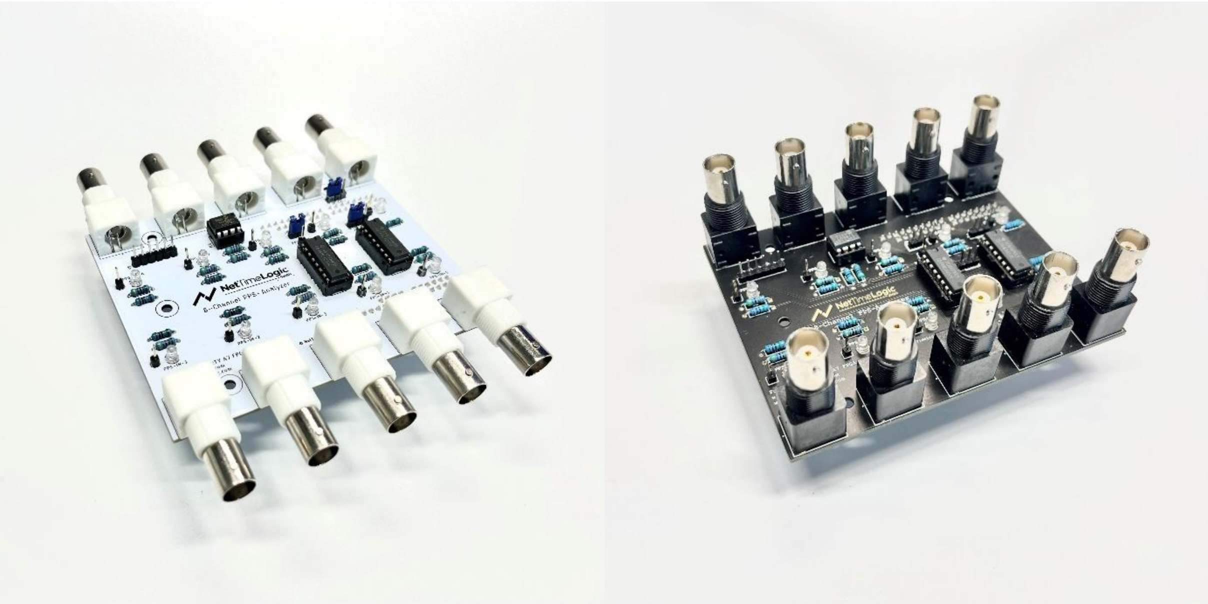


Figure 2: PPS Analyzer Shield (left: Rev A; right Rev B)

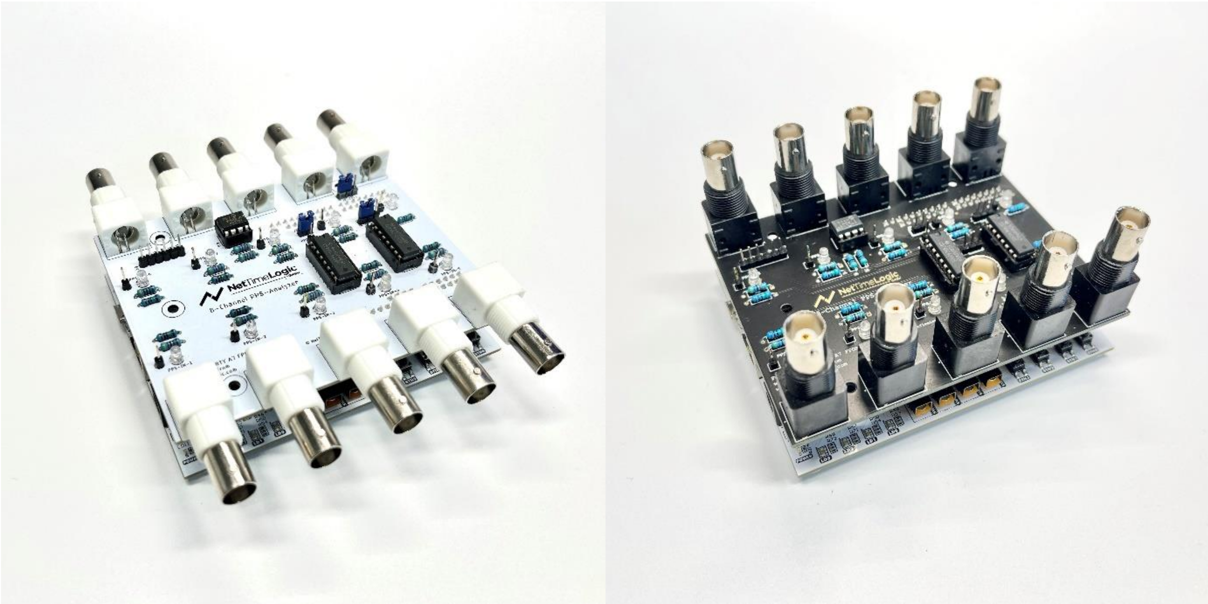


Figure 3: PPS Analyzer Shield with Base Board (left Rev A; right Rev B)

2.2.1 Pinout ArtyA7

The signals are connected to the following pins on the ArtyA7-35T/100T board, naming is according to printing on the board.

Signal Name	Arty Pin	FPGA Pin	Description
REF_PSS_IN	IO26	U11	Reference PPS input
PPS1	IO27	V16	Measure PPS input 1
PPS2	IO28	M13	Measure PPS input 2
PPS3	IO29	R10	Measure PPS input 3
PPS4	IO30	R11	Measure PPS input 4
PPS5	IO31	R13	Measure PPS input 5
PPS6	IO32	R15	Measure PPS input 6
PPS7	IO33	P15	Measure PPS input 7
PPS8	IO3	T11	Measure PPS input 8
REF_PPS_OUT	IO6	T15	(Reference) PPS output
PPS_CALIB_OUT	IO7	T16	Calibration PPS output (not aligned in any way)
TH_LOW	IO40	P18	Threshold Low output
TH_HIGH	IO41	N17	Threshold High output
I2C_DATA	IO11	U18	I2C Data in/output
I2C_CLK	IO12	R17	I2C Clock output
I2C_WP	IO13	P17	I2C Write Protect output

Table 4: Pinout A7

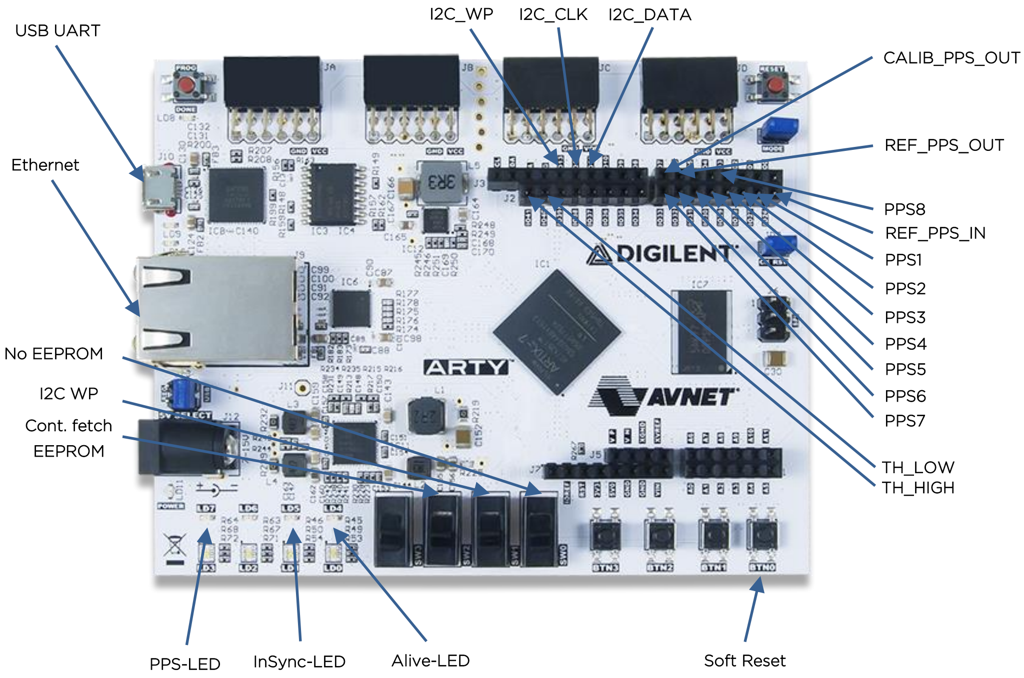


Figure 4: ArtyA7 (source Digilent Inc)

2.2.2 Pinout ArtyS7

The signals are connected to the following pins on the ArtyS7-50 board, naming is according to printing on the board.

Signal Name	Arty Pin	FPGA Pin	Description
REF_PSS_IN	IO26	U11	Reference PPS input
PPS1	IO27	T11	Measure PPS input 1
PPS2	IO28	R11	Measure PPS input 2
PPS3	IO29	T13	Measure PPS input 3
PPS4	IO30	T12	Measure PPS input 4
PPS5	IO31	V13	Measure PPS input 5
PPS6	IO32	U12	Measure PPS input 6
PPS7	IO33	V15	Measure PPS input 7
PPS8	IO3	R14	Measure PPS input 8
REF_PPS_OUT	IO6	R17	(Reference) PPS output
PPS_CALIB_OUT	IO7	V17	Calibration PPS output (not aligned in any way)
TH_LOW	IO40	V16	Threshold Low output
TH_HIGH	IO41	U15	Threshold High output
I2C_DATA	IO11	H17	I2C Data in/output
I2C_CLK	IO12	K14	I2C Clock output
I2C_WP	IO13	G16	I2C Write Protect output

Table 5: Pinout S7

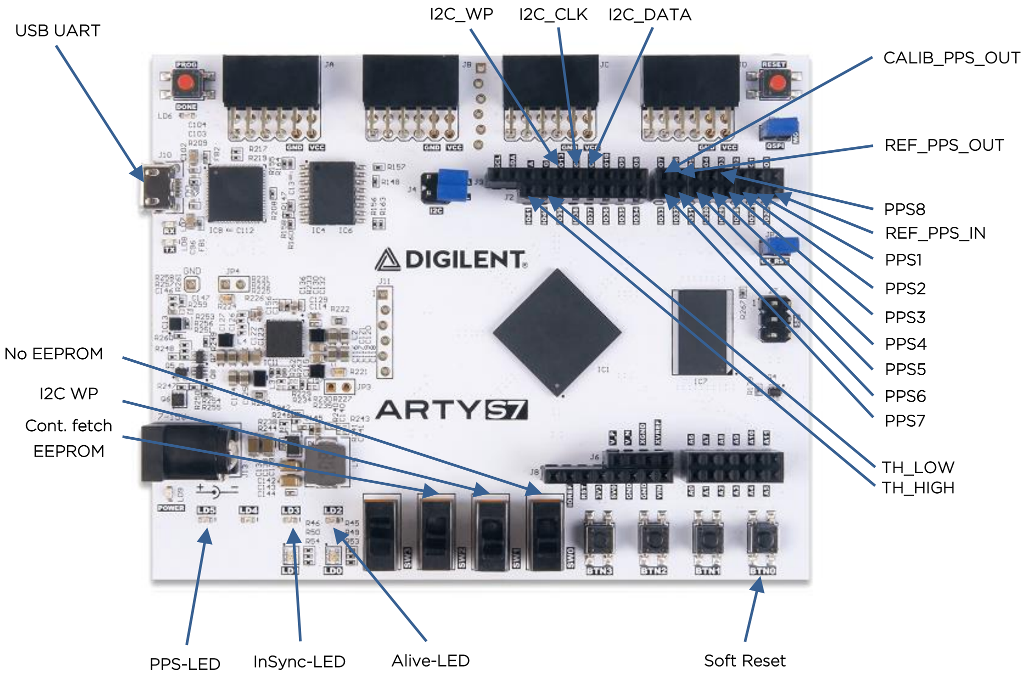


Figure 5: ArtyS7 (source Digilent Inc)

2.3 Software

The PPS Analyzer software is a simple Qt application, discovering and connecting to all available PPS Analyzer hardware (via USB/UART and Ethernet) connected and starting to show the PPS measurements.

The measurement screen (only current) can be saved, cleared and it can be chosen to use compensated or raw values. In general, the compensated values should be used since this removes much of the synchronization error introduces by the reference PPS.

In addition to saving the screen the measurement values can be logged as CSV file (separator “;”). Once the logging is started all new values are logged to a file until logging is stopped or the application closed. Every time a log is started a new file with the date is created (can be changed).

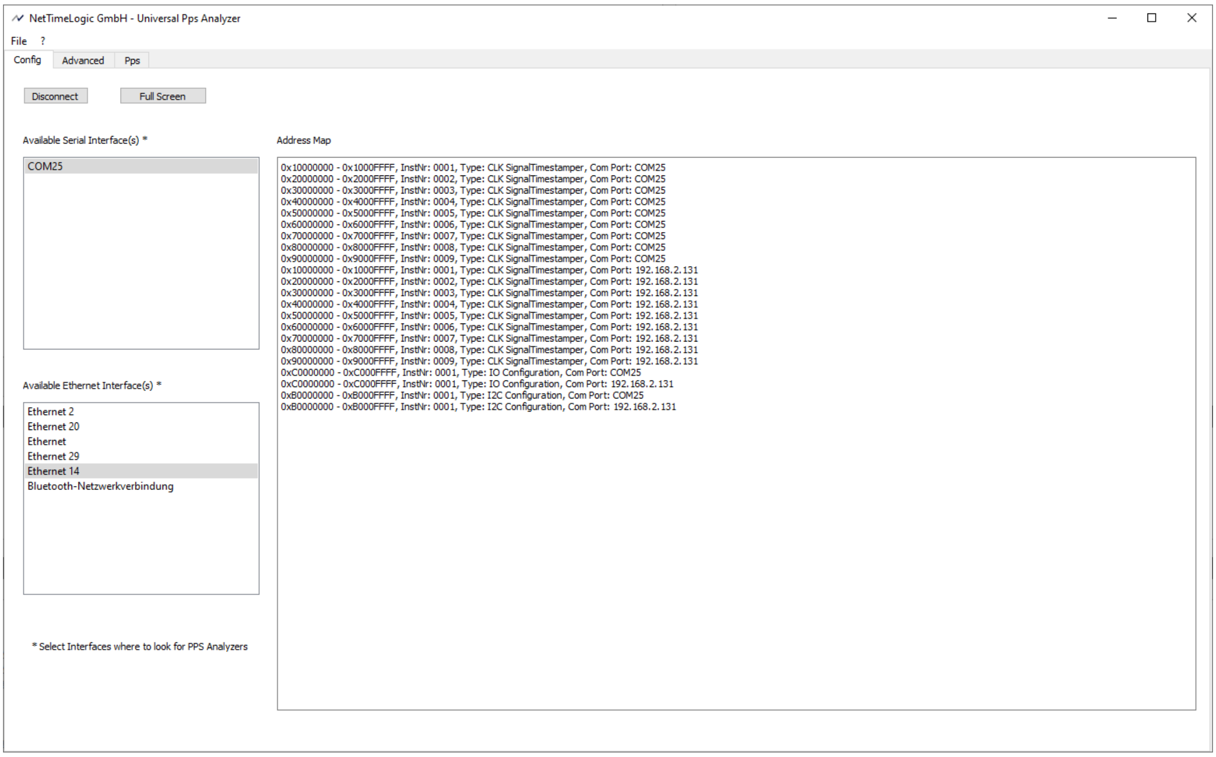


Figure 6: PPS Analyzer Software Connected

Figure 7: PPS Analyzer Software PPS

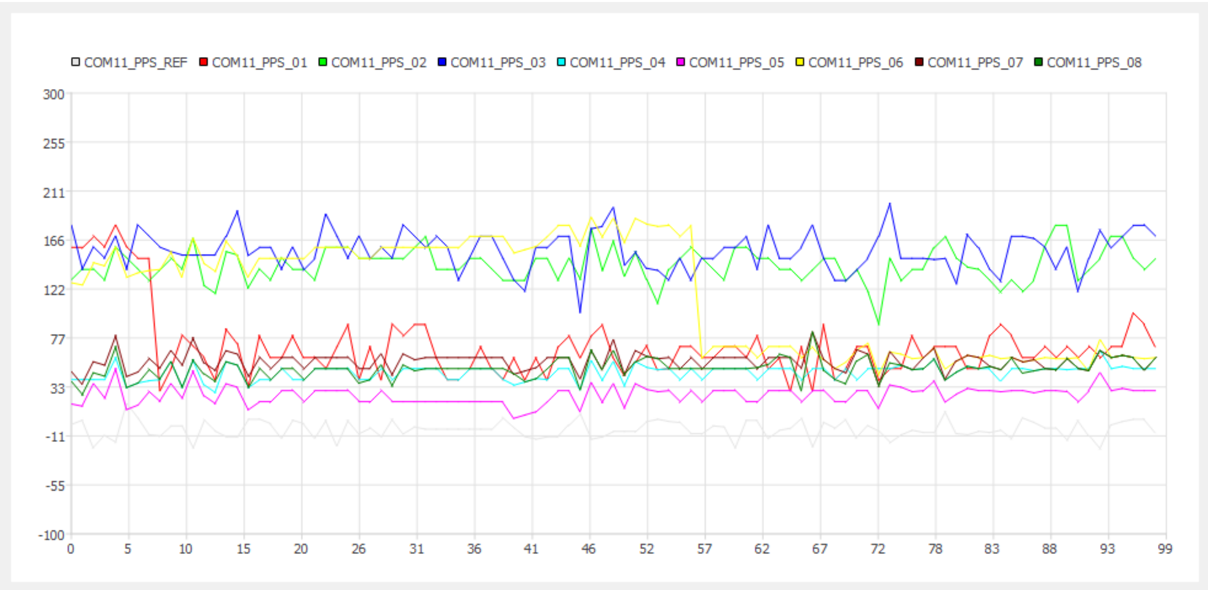


Figure 8: PPS Analyzer Saved Screenshot

Also, the cable delays and names of the individual PPS can be changed and enabled to be shown in the graph. The delay can be to compensate different delays on input buffers (if no EEPROM) or different delays due to different cable lengths from the source to the PPS inputs. The configured delay is subtracted from the measured offset for each PPS individually, positive and negative values are possible,

changing delays in software will not have any influence on the PPS Output behavior, this must come from a calibration value.
First choose the PPS Analyzer on which you want to configure the delays then choose the PPS input to correct. If you are done with an Analyzer press "Change Delays", only then the values will have an effect.
Note: Be aware, that changing the delay on the reference PPS has an influence on all PPS belonging to the same analyzer.

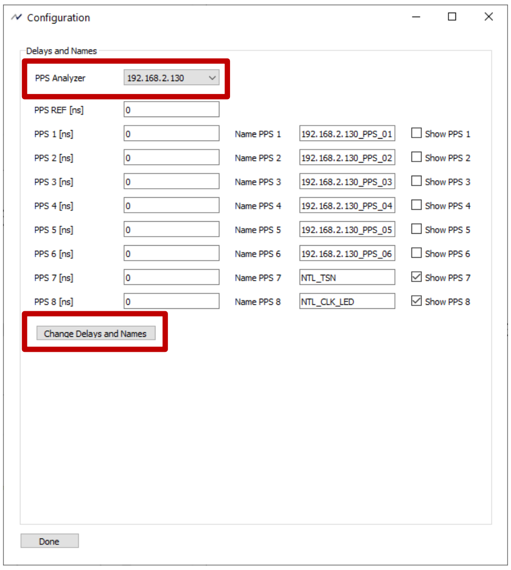


Figure 9: PPS Analyzer Software Delay Screen

In addition you can set the Threshold per PPS (need to be also enabled to be shown). When a specific signal exceeds the configured Threshold level the corresponding Threshold signal is set and the Status pictures changes. First choose the PPS Analyzer on which you want to configure the thresholds, change the values and press "Change Thresholds", only then the values will have an effect.

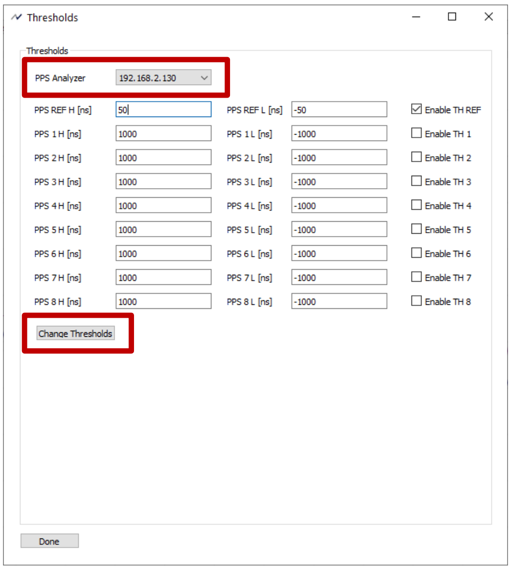


Figure 10: PPS Analyzer Software Threshold Screen

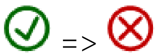


Figure 11: PPS Analyzer Threshold State

2.4 Run

1. Connect the Arty board via USB (which is combined Power, JTAG and UART)
2. Download the bitstream to your Arty (flash via JTAG over USB), this needs to be done only once
3. Make sure SW0-SW3 are all OFF
4. Powercycle the board
5. Connect the Arty board via USB (which is combined Power and UART) or Ethernet (requires external Power, default IP is 192.168.1.128)
6. Connect a PPS to the REF_PPS_IN input
7. Connect the other PPS inputs (if not all are used make sure they are on a stable 0 logic level, there are weak pull downs on the pins).
8. Open the Universal PPS Analyzer application
9. Slect the interfaces where the PPS Analyzer is connected to
10. Press Connect
11. Wait until the grey signal which is the reference PPS error is close to zero and the InSync LED is lit on the Arty, then clear the screen.
12. Check your PPS signals

Steps 1-4 need to be done only once when the FPGA is not programmed yet.

2.5 Download and buy your Shield

If you want to buy a PPS shield from us, either as fully assembled and calibrated kit or as a Do-It-Yourself PCB without Components, contact us via contact@nettime-logic.com or visit our web shop https://www.nettimelogic.com/shop.php.

The bitstream and application can be downloaded from http://www.nettime-logic.com/downloads.php

The source code of the GUI can be found here:
https://github.com/NetTimeLogic/UniversalPpsAnalyzer
The download page is password protected, contact us via contact@nettime-logic.com to get access.

The PPS Analyzer software is free of charge and available under the LGPL 3.0 license. The FPGA part is available as freeware, binary only.

2.6 Calibration and setting MAC and IP

For Calibration, an extra calibration shield or special cabling is required. Contact us (contact@nettimelogic.com) if you need instructions on how to build or purchase a calibration shield. Our pre-assembled and calibrated modules do not need additional calibration.

WARNING, Do not start a calibration if you have no shield, you will lose the pre-calibration!

When Ethernet shall be used to connect and an EEPROM is in place (e.g. our shield) then the MAC and IP of the device can be configured.

1. Make sure SW0-SW3 are all OFF
2. Connect via UART or the default IP 192.168.1.128
3. Go to the Advanced tab
4. Press Calibrate

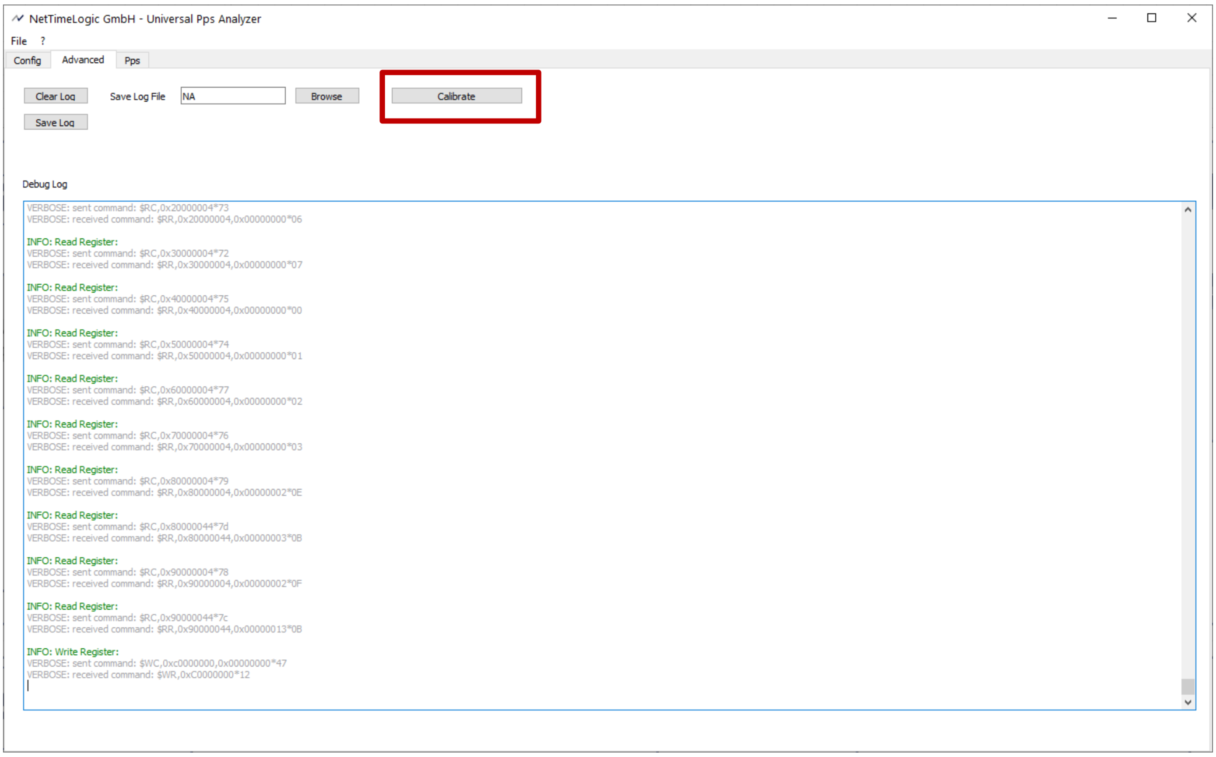
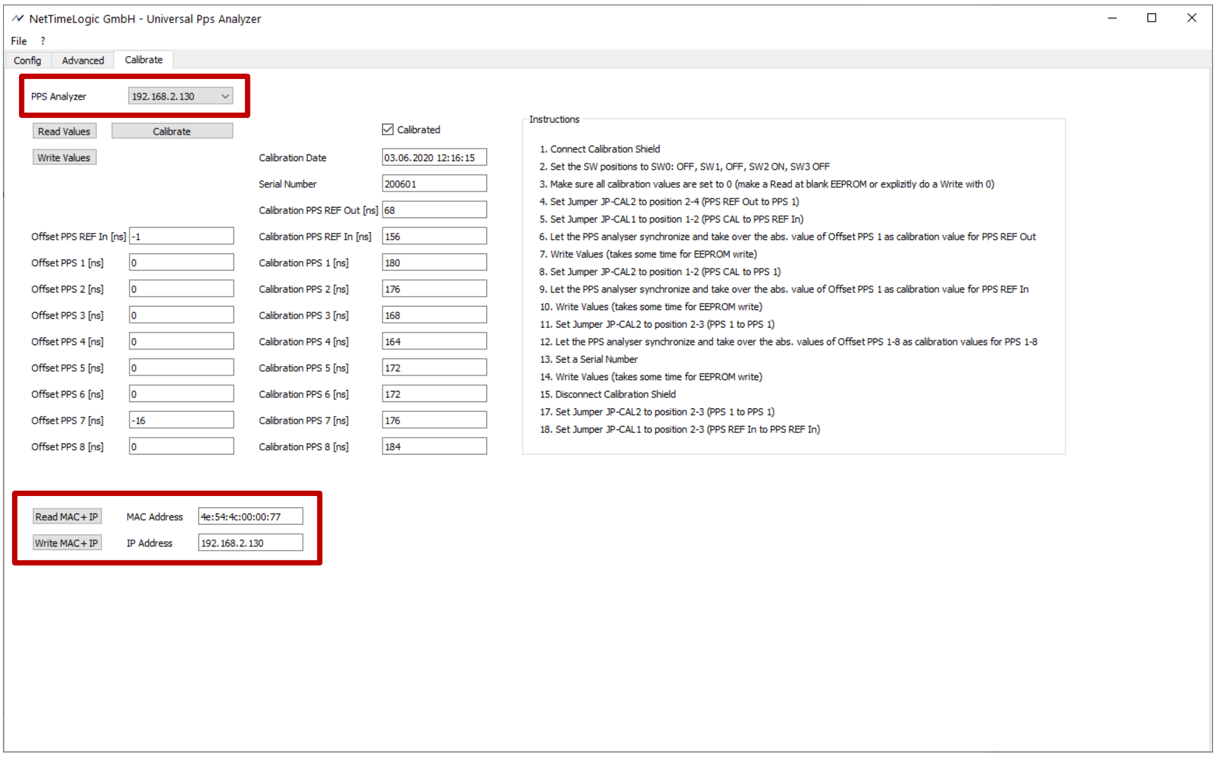


Figure 12: Advanced tab

5. Press Calibrate
6. Go to the Calibrate tab



7. Choose the PPS Analyzer which you would like to change
8. Press the Read MAC + IP button
9. Change the values as desired
10. Press the Write MAC + IP button
11. Powercycle the board, restart the application and connect again

3 Interface and Protocol Basics

3.1 UART Interface

For the communication between the FPGA and the Host a UART interface can be used. Often this UART interface is done via an USB UART (like on the Arty board via an FTDI® chip). The following parameters are used:

- 1 Start bit
- 8 Data bits
- 1 Stop bit
- No Parity bit
- Baudrate 115200

3.2 ETHERNET Interface

For the communication between the FPGA and the Host also an 100Mbit Ethernet interface can be used (only on ArtyA7-35T/100T). The following parameters are used:

- 100Mbit only
- Access via Broadcast or Unicast MAC and IP
- IPv4
- TTL: 128
- UDP Port: 0xBEEF
- Default IP (after production or without EEPROM) is 192.168.1.128
- Default MAC (after production or without EEPROM) is 4E:54:4C:10:00:00

Data is encapsulated into a UDP/IPv4 frame as one command per frame. It also expects ASCII character and does the padding and cut off the padding.

3.3 Protocol

The protocol run on the UART and Ethernet is a proprietary protocol defined by NetTimeLogic.
It is a simple protocol with no retransmission and therefore also not failsafe. The protocol uses ASCII characters, so it can also be entered directly from a terminal if UART is used (e.g. TeraTerm).
A couple of extra characters are used in the Data stream to allow synchronization of start and end of the commands as well as separation of the individual fields.
The command always starts with a '$' character followed by a two-character command code. Then individual fields can follow, each field is separated by a ','

character. After the fields a ‘*’ character indicates the end of the command and that a checksum is followed, the two characters of checksum are followed. The command is ended with a <CR><LF> (carriage return and line feed) combination. The checksum is optional for the host and can be left away, in this case the ‘*’ character is also left away. The checksum XOR combines all received bytes between the ‘$’ and ‘*’ characters (not including) starting with 0x00 as starting value. If a checksum is present, the checksum is checked and an error is signaled by the FPGA to the host if the checksum is not correct and the command ignored.
The protocol engine in the FPGA allows empty lines and comments be transferred also via UART. A comment line starts with “--“ characters. This functionality, and the fact that the checksum is optional can be used if the whole content of a file containing not only commands but also comments is copied to a terminal. The PPS Analyzer GUI will always send only commands from the host to the FPGA and always with a checksum.

3.3.1 Write Command and Write Response

The two messages described here are used for writing a register.

The format of the write command looks the following:

$WC,<ADDRESS>,<DATA>*<CHECKSUM><CR><LF>

e.g. : $WC,0x50000000,0x40000001*14

A write command is always issued by the host.
The write command starts with the command identifier of the two characters “WC”. Following the identifier, the 32bit AXI address to be written in hexadecimal format is added. Following the address, 32bit of write data in hexadecimal format is added. Both address and data have to start with “0x” followed by 8 hexadecimal characters.

A write command will always trigger a write response in the FPGA. If something goes wrong an error response is sent containing an error code.

The format of the write response looks the following:

$WR,<ADDRESS>*<CHECKSUM><CR><LF>

e.g. : $WR,0x50000000*64

A write response is always issued by the FPGA.
The write response starts with the command identifier of the two characters “WR”. Following the identifier, the 32bit AXI address written in hexadecimal format is added which is the address which was written in the FPGA (as in the examples, 0x50000000). The address has to start with “0x” followed by 8 hexadecimal characters.

3.3.2 Read Command and Write Response

The two messages described here are used for reading a register.

The format of the read command looks the following:

$RC,<ADDRESS>*<CHECKSUM><CR><LF>

e.g. : $RC,0x50000000*70

A read command is always issued by the host.
The read command starts with the command identifier of the two characters “RC”. Following the identifier, the 32bit AXI address to be read in hexadecimal format is added. The address has to start with “0x” followed by 8 hexadecimal characters.

A read command will always trigger a read response in the FPGA. If something goes wrong an error response is sent containing an error code.

The format of the read response looks the following:

$RR,<ADDRESS>,<DATA>*<CHECKSUM><CR><LF>

e.g. : $RR,0x50000000,0x00000001*04

A read response is always issued by the FPGA.
The read response starts with the command identifier of the two characters “RR”. Following the identifier, the 32bit AXI address read in hexadecimal format is added which is the address which was read in the FPGA (as in the examples, 0x50000000). Following the address, 32bit of read data read in hexadecimal format is added. Both address and data have to start with “0x” followed by 8 hexadecimal characters.

3.3.3 Connect Command and Connect Response

The two messages described here are used for testing the connection, for e.g. to figure out if a system is connected that supports this protocol.

The format of the connect command looks the following:

$CC*<CHECKSUM><CR><LF>

e.g. : $CC*00

A connect command is always issued by the host.
The connect command starts with the command identifier of the two characters “CC”.

A connect command will always trigger a connect response in the FPGA. If something goes wrong an error response is sent containing an error code.

The format of the connect response looks the following:

$CR*<CHECKSUM><CR><LF>

e.g. : $CR*11

A connect response is always issued by the FPGA.
The connect response starts with the command identifier of the two characters “CR”.

3.3.4 Error Response

The error messages described here is used when something goes wrong. It is always issued as reaction to another command.

The format of the error response looks the following:

$ER,<ERROR CODE>*<CHECKSUM><CR><LF>

e.g. : $ER,0x00000003*70

An error response is always issued by the FPGA.
The error response starts with the command identifier of the two characters “ER”.
Following the identifier, a 32bit error code in hexadecimal format is added.
The enumeration of the errors as of today is as following:

- 0x00000000: Checksum error
- 0x00000001: Unknown command (or error in command)
- 0x00000002: Read error on AXI
- 0x00000003: Write error on AXI
- 0x00000004: Access timeout error on AXI (illegal address, no answer)

4 Register Map

The PPS Analyzer FPGA Design contains 9 PPS Timestamper, one for the Reference PPS Input (REF_PPS_ IN) and 8 for the DUT PPS Inputs (PPS1-PPS8) and the IO module to set the threshold outputs:

- 0x10000000 - 0x1000FFFF: Timestamper REF_PPS_ IN
- 0x20000000 - 0x2000FFFF: Timestamper PPS1
- 0x30000000 - 0x3000FFFF: Timestamper PPS2
- 0x40000000 - 0x4000FFFF: Timestamper PPS3
- 0x50000000 - 0x5000FFFF: Timestamper PPS4
- 0x60000000 - 0x6000FFFF: Timestamper PPS5
- 0x70000000 - 0x7000FFFF: Timestamper PPS6
- 0x80000000 - 0x8000FFFF: Timestamper PPS7
- 0x90000000 - 0x9000FFFF: Timestamper PPS8
- 0xC0000000 - 0xC000FFFF: Threshold IO for TH_LOW and TH_HIGH

Each Timestamper has its own Registerset at the defined base addresses above, the register description of the Timestamper registers and functionality can be found here: https://www.nettimelogic.com/resources/Clk_SignalTimestamper_ReferenceManual.pdf Chapter 3.

For the Threshold the following register exist

- 0xC0000000: Data Output Register (RW)
  - Bit0: Threshold Low (TH_LOW) Output state (0=low, 1=high)
  - Bit1: Threshold High (TH_HIGH) Output state (0=low, 1=high)

A List of tables

B List of figures