



Instruments That Advance The Art

Pixie-16 MZ-TrigIO

User Manual

Version 0.20

January 17, 2019

Hardware Revisions: A

Software Revision: 0x1000

Firmware Revision and Variants:
0x0001 (standard)

XIA LLC

31057 Genstar Rd

Hayward, CA 94544 USA

Email: support@xia.com

Tel: (510) 401-5760; Fax: (510) 401-5761

<http://www.xia.com/>

Information furnished by XIA LLC is believed to be accurate and reliable. However, no responsibility is assumed by XIA for its use, or for any infringements of patents or other rights of third parties which may result from its use. No license is granted by implication or otherwise under any patent or patent rights of XIA. XIA reserves the right to change hardware or software specifications at any time without notice.

Contents

| | |
|---|----|
| Safety | 4 |
| Specific Precautions | 4 |
| Power Source | 4 |
| User Adjustments/Disassembly | 4 |
| Voltage Ratings..... | 4 |
| Servicing and Cleaning..... | 4 |
| Linux Passwords..... | 4 |
| Linux Backup..... | 4 |
| Warranty Statement | 5 |
| Contact Information: | 5 |
| Manual Conventions | 6 |
| 1 Introduction..... | 7 |
| 1.1 Pixie-16 MZ-TrigIO Features | 7 |
| 1.2 Specifications..... | 8 |
| 1.3 System Requirements..... | 9 |
| 1.3.1 Drivers and Software | 9 |
| 1.3.2 Input Signals | 9 |
| 1.3.3 Power Requirements | 9 |
| 1.3.4 Connectors and Cabling | 10 |
| 1.4 Software and Firmware Overview | 10 |
| 1.5 Support..... | 10 |
| 2 Setup | 11 |
| 2.1 Power | 11 |
| 2.2 Serial Port (USB-UART)..... | 11 |
| 2.3 SSH login..... | 12 |
| 2.4 Web Interface..... | 12 |
| 2.5 SMB (Samba)..... | 13 |
| 2.6 Required Initial Linux Commands..... | 13 |
| 2.7 Useful Linux Commands | 13 |
| 2.8 Direct Network Connection between Pixie-16 MZ-TrigIO and a Windows PC | 14 |
| 3 Pixie-16 MZ-TrigIO Operation..... | 16 |
| 3.1 Adjust Parameters | 16 |
| 3.2 Data Acquisition Monitoring | 17 |
| 3.3 User Interface Options | 17 |
| 3.3.1 Terminal | 17 |
| 3.3.2 Terminal and Webpages..... | 17 |
| 3.3.3 Terminal, SMB and Windows programs | 18 |
| 3.3.4 Webpage-Only Operation | 18 |
| 3.3.5 On-board ROOT Graphical User Interface | 19 |
| 4 API functions | 20 |
| 4.1 progfippi..... | 20 |
| 4.2 runstats, cgstats | 20 |
| 4.3 clockprog..... | 21 |
| 5 Web Pages..... | 22 |

| | | |
|--------|---|-------------------------------------|
| 5.1 | index.html | 22 |
| 5.2 | rspage.html, cgistats.cgi | 23 |
| 6 | Parameters in the Settings Files | 24 |
| 6.1 | System Parameters | 24 |
| 6.2 | Local Control | 25 |
| 6.3 | Trigger Control Parameters..... | 26 |
| 7 | Data Formats..... | 28 |
| 7.1 | Status Files (RS.csv)(..... | 28 |
| 8 | Registers Visible to Linux | 30 |
| 8.1 | Input Registers | 30 |
| 8.2 | Output Registers..... | 32 |
| 9 | Linux Configuration | 34 |
| 9.1 | Licensing Information..... | 34 |
| 9.2 | Software Information | 34 |
| 10 | Board Architecture..... | 38 |
| 10.1 | Connections..... | 38 |
| 10.1.1 | Backplane Connections..... | 38 |
| 10.1.2 | Front Panel Connections | 39 |
| 10.1.3 | Zynq SOM Connections..... | 39 |
| 10.1.4 | Custom Connections | 40 |
| 10.2 | Power | 40 |
| 10.3 | Clocks | 40 |
| 10.4 | IEEE 1588 Precision Timing Protocol..... | 41 |
| 10.4.1 | Overview | 41 |
| 10.4.2 | PTP Stack Software: LinuxPTP..... | 42 |
| 10.4.3 | PTP PHY Control Software: ptp-mii-tool..... | 43 |
| 10.4.4 | PTP Clock PLL Control Software: clockprog | 43 |
| 10.4.5 | Reduced General Purpose Functionality | 44 |
| 11 | Hardware Information..... | 45 |
| 11.1 | Jumpers | 45 |
| 12 | Firmware Information..... | 46 |
| 13 | Software Information..... | Error! Bookmark not defined. |

Safety

Please take a moment to review these safety precautions. They are provided both for your protection and to prevent damage to the Pixie module and connected equipment. This safety information applies to all operators and service personnel.

Specific Precautions

Power Source

The Pixie-16 MZ-TrigIO module is powered through an AC/DC wall adapter or a PXI backplane. The default adapter has a variety of AC plug attachments for different localities. Please remember to shut down the Linux OS before removing the power plug from the Pixie-16 MZ-TrigIO or powering down the PXI chassis.

User Adjustments/Disassembly

To avoid personal injury, and/or damage, always disconnect power before accessing the module's interior. There are a few jumpers related to clocking on the board that experienced users may want to use.

Voltage Ratings

Signals on the inputs and outputs must not exceed $\pm 3.3V$. Please review the pinout in the appendix before making any connections.

Servicing and Cleaning

To avoid personal injury, and/or damage to the Pixie module or connected equipment, do not attempt to repair or clean the inside of these units.

Linux Passwords

The Pixie-16 MZ-TrigIO Linux OS comes with default user IDs and passwords for 1) SSH login, 2) SMB file sharing, and 3) Web Operations as described below. Users should immediately change these passwords, especially when the Pixie-16 MZ-TrigIO is connected to external networks. Don't let hackers take over your Pixie-16 MZ-TrigIO!

Linux Backup

The Pixie-16 MZ-TrigIO Linux OS is stored on a removable SD card. SD cards' file systems can become corrupted, which would crash the Linux system and make the Pixie-16 MZ-TrigIO unable to operate. Therefore periodic backup of the SD card is recommended, for example using Win32DiskImager. (Byte for byte copy is required). Note that all Linux passwords are stored on the SD card.

Warranty Statement

XIA LLC warrants that this product will be free from defects in materials and workmanship for a period of one (1) year from the date of shipment. If any such product proves defective during this warranty period, XIA LLC, at its option, will either repair the defective products without charge for parts and labor, or will provide a replacement in exchange for the defective product.

In order to obtain service under this warranty, Customer must notify XIA LLC of the defect before the expiration of the warranty period and make suitable arrangements for the performance of the service.

This warranty shall not apply to any defect, failure or damage caused by improper uses or inadequate care. XIA LLC shall not be obligated to furnish service under this warranty a) to repair damage resulting from attempts by personnel other than XIA LLC representatives to repair or service the product; or b) to repair damage resulting from improper use or connection to incompatible equipment.

THIS WARRANTY IS GIVEN BY XIA LLC WITH RESPECT TO THIS PRODUCT IN LIEU OF ANY OTHER WARRANTIES, EXPRESSED OR IMPLIED. XIA LLC AND ITS VENDORS DISCLAIM ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. XIA'S RESPONSIBILITY TO REPAIR OR REPLACE DEFECTIVE PRODUCTS IS THE SOLE AND EXCLUSIVE REMEDY PROVIDED TO THE CUSTOMER FOR BREACH OF THIS WARRANTY. XIA LLC AND ITS VENDORS WILL NOT BE LIABLE FOR ANY INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES IRRESPECTIVE OF WHETHER XIA LLC OR THE VENDOR HAS ADVANCE NOTICE OF THE POSSIBILITY OF SUCH DAMAGES.

Contact Information:

XIA LLC
31057 Genstar Rd.
Hayward, CA 94544 USA

Telephone: (510) 401-5760
Downloads: <http://support.xia.com>
Hardware Support: support@xia.com
Software Support: support@xia.com

○

Manual Conventions

The following conventions are used throughout this manual

| Convention | Description | Example |
|--|--|---|
| » | The » symbol leads you through nested menu items and dialog box options. | The sequence File»Page Setup»Options directs you to pull down the File menu, select the Page Setup item, and choose Options from the sub menu. |
| Bold | Bold text denotes items that you must select or click on in the software, such as menu items, and dialog box options. | ...click on the MCA tab. |
| [Bold] | Bold text within [] denotes a command button. | [Start Run] indicates the command button labeled Start Run. |
| Monospace | Items in this font denote text or characters that you enter from the keyboard, sections of code, file contents, and syntax examples. | Setup.exe refers to a file called “setup.exe” on the host computer. |
| “window” | Text in quotation refers to window titles, and quotations from other sources | “Options” indicates the window accessed via Tools»Options . |
| <i>Italics</i> | Italic text denotes a new term being introduced , or simply emphasis | <i>peaking time</i> refers to the length of the slow filter. ...it is important first to set the energy filter Gap so that SLOWGAP to <i>at least one unit greater than</i> the preamplifier risetime... |
| <Key> <Shift-Alt-Delete> or <Ctrl+D> | Angle brackets denote a key on the keyboard (not case sensitive). A hyphen or plus between two or more key names denotes that the keys should be pressed simultaneously (not case sensitive). | <W> indicates the W key <Ctrl+W> represents holding the control key while pressing the W key on the keyboard |
| <i>Bold italic</i> | Warnings and cautionary text. | <i>CAUTION: Improper connections or settings can result in damage to system components.</i> |
| CAPITALS | CAPITALS denote DSP parameter names | SLOWLEN is the length of the slow energy filter |
| SMALL CAPS | SMALL CAPS are used for panels/windows/graphs in the GUI. | ...go to the MCADISPLAY panel and you see... |

1 Introduction

The Pixie-16 MZ-TrigIO is a Zynq System on Module¹ (SoM) combined with an input/output carrier board. The Zynq is a combination of an FPGA (Programmable Logic, PL) with an ARM processor (Processing System, PS). The PL captures the ADC data and applies digital pulse processing. The PS runs a basic Linux OS with gcc, webserver, etc; it has USB and Ethernet peripherals and 1GB of memory.

The P16 MZ TrigIO carrier board is primarily a 6U breakout board for the Pixie-16 custom PXI chassis backplane. It connects ~48 lines from the backplane to one of the SOM headers. Another ~48 lines from the second SOM header connect to front panel I/O via LVDS I/O buffers and RJ-45 connectors for CAT-5 cables. The backplane and front panel connections are also connected to 0.1" headers to allow direct access via cables or custom signal I/O via daughtercards. The SOM can be accessed via USB/UART on its native connectors (via cabling though an empty slot) but mainly via a PTP² compatible 10/100M Ethernet port on the front panel.

Besides being operated in the P16 PXI chassis, the P16 MZ TrigIO can also be operated as a standalone desktop unit. The front panel LVDS connectors can be directly used in desktop mode; the backplane connections are only available with custom daughtercards or cables. When operated in a standard 6U PXI chassis, connectors J3 and J4 have to be removed and no backplane I/O is available.

1.1 Pixie-16 MZ-TrigIO Features

The Pixie-16 MZ-TrigIO is designed to route signals from the backplane (rear connectors) to the front panel (front connectors) and make logical combinations between them in FPGA fabric. It has the following features and capabilities:

- Ethernet programmable trigger/coincidence control module for the Pixie-16
- 48+ Pixie-16 backplane trigger connections to local Zynq processor
- 48 front panel LVDS connections to local Zynq processor
- MicroZed Zynq processor with embedded Linux, acting as a standalone PC with built-in SD card drive, USB host, 10/100 Ethernet, webserver, etc
- 1588 PTP and SyncE clock synchronization
- Open source user access to software and firmware
- Use as standalone desktop unit or in 6U PXI chassis
- Custom I/O standards via daughtercards

¹ Model Avnet MicroZed 7020

² IEEE 1588 Precision Time Protocol

1.2 Specifications

| Front Panel I/O | |
|-------------------------------------|---|
| LVDS Input/Output (48x) | 12 RJ-45 connectors for general purpose LVDS I/O with CAT-5 cables 4 signal pairs each programmable input or output direction for each signal individually Vio H/L: +/- 100 mV |
| Ethernet | 1 RJ-45 connector for 10/100M Ethernet PTP and SyncE compatible |
| Clock Input/Output | 1 MMCX coaxial connector jumper selectable input or output 3.3V signaling standard |
| Power (desktop mode only) | 12V DC in. AC adapter requirements: 12V, 10W Barrel Plug 2.1mm I.D. x 5.5mm O.D., center positive |
| Rear Panel I/O | |
| PXI J1 | Used only for power, 3.3V and 5V 5V ~ 1A 3.3V ~ 1 A |
| PXI J2 (part) PXIe XJ4 | Used only for PXI clock input to PXI_CLKIN (if located in PXI slot 2) output from PXI_CLK typically 50 MHz in Pixie-16 chassis, 10 MHz default chassis |
| PXI J3 | 18 backplane 3.3V trigger signals (Pixie-16 EB_Data and EB_Control) 1 clock output to right neighbor |
| PXI J4 | 34 backplane 3.3V trigger signals (Pixie-16 TriggerAll and Control) |
| On-Board I/O | |
| Zynq Processor | 1 USB 2.0 1 USB-UART 1 RJ45 Ethernet (10/100/1000M) 1 SD card slot |
| Other I/O | 6 34-pin 0.1" headers for cables or daughtercards, connecting to front/rear I/O signals (no cable drivers) |
| Embedded Processing | |
| Processor | Xilinx Zynq |
| Data Interfaces | 10/100/1000 Ethernet USB 2.0 (host) USB-UART |

| | |
|--|---|
| Memory | 1 GB of DDR3 SDRAM 128 Mb of QSPI Flash 16 GB micro SD card (removable) |
| Operating System | Linux (Xillinux – based on Ubuntu 15) Operates from Linux partition on SD card |
| Digital Controls | |
| Trigger Routing and Coincidence | Firmware demo code to <ul style="list-style-type: none"> - control connections between backplane and front panel - compute multiplicity - test for coincidence |
| FPGA register programming | C software to <ul style="list-style-type: none"> - program FPGA registers that enable or disable connections. - read FPGA status registers |
| Time synchronization | C software (LinuxPTP) to operate IEEE 1588 PTP timers and adjust local clock to network clock master C software to define PTP timed triggers |

Table 1-1. Specifications for the Pixie-16 MZ-TrigIO

1.3 System Requirements

The system considered here consists of a Pixie-16 MZ-TrigIO and Pixie-16 pulse processor system for radiation detectors in a PXI chassis. Optional connections can be made to external electronics. A PC, smartphone or tablet is required to communicate with the Pixie-16 MZ-TrigIO, but trigger routing and coincidence logic is fully contained in the Pixie-16 MZ-TrigIO itself.

1.3.1 Drivers and Software

The Pixie-16 MZ-TrigIO operates with an embedded Linux system that includes all software and drivers to communicate with external devices via Ethernet or USB. It can

- Make status information available via webserver
- Read and write USB drives for data exchanges
- Share files over a Windows network

1.3.2 Input Signals

The Pixie-16 MZ-TrigIO is designed for fast rising, exponentially decaying signals. Step pulses and short non-exponential pulses can be accommodated with specific parameter settings. Staircase type signals from reset preamplifiers generally need to be AC coupled.

Input signals must not exceed $\pm 3.3V$. The LVDS signaling standard is ± 100 mV.

1.3.3 Power Requirements

The Pixie-16 MZ-TrigIO consumes roughly 10 W. It requires following currents

In PXI mode from the backplane:

3.3V up to 1.0 A

5.0V up to 1.0 A

In desktop mode from the DC adapter

12V up to 1.0 A

1.3.4 Connectors and Cabling

The Pixie-16 MZ-TrigIO uses 12 RJ-45 connectors for the LVDS signal I/O. These connectors are not capable of Ethernet connections.

The Pixie-16 MZ-TrigIO uses 1 RJ-45 connector for the Ethernet connection. Please verify labels on the board to ensure using the correct connector for Ethernet.

A MMCX connectors are used for clock input/outputs. MMCX to BNC adapter cables are provided with the module.

1.4 Software and Firmware Overview

The software of the Pixie-16 MZ-TrigIO consists of a small set of C programs (API functions) applying settings to the PL, reading data from the PL, and storing it on the SD card (or network drives). The API functions are called from the Linux command line or as CGI scripts from a web page. Results can be viewed or downloaded via the web page, or copied over the network. Configuration parameters are stored in an .ini file that can be edited by the user to adjust parameter settings.

Firmware code for the PL on-board pulse processing functions is loaded to the PL as part of the powerup boot sequence.

Users may modify the API functions (source code and gcc compiler is included in the Linux environment on the SD card). Applications can be executed from the SD card or a mounted USB drive.

More sophisticated graphical user interfaces can be developed, providing control buttons and data analysis functions. Please contact XIA for details.

1.5 Support

A unique benefit of dealing with a small company like XIA is that the technical support for our sophisticated instruments is often provided by the same people who designed them. Our customers are thus able to get in-depth technical advice on how to fully utilize our products within the context of their particular applications.

Please read through the following sections before contacting us. Contact information is listed in the first few pages of this manual.

2 Setup

When powered up, the Pixie-16 MZ-TrigIO automatically boots the FPGA configuration and starts the Linux OS. There are several ways for a user to connect to the Pixie-16 MZ-TrigIO Linux OS:

1. Serial port via USB-UART
2. Network SSH terminal
3. Web server
4. SMB (Samba) file sharing

The typical procedure for setup would be to first to power up the Pixie-16 MZ-TrigIO (2.1), then install and configure drivers for the serial port on a USB master PC (2.2). Log on via serial port terminal, find the Pixie-16 MZ-TrigIO's IP address, and execute a few setup programs (2.5). After that, the Pixie-16 MZ-TrigIO can be operated through terminal, web interface and/or SMB.

2.1 Power

To power up the Pixie-16 MZ-TrigIO in a PXI chassis, first verify power selection jumpers are set to "PXI" and install the module in an available PXI slot³. Then turn on the chassis power.

To power up the Pixie-16 MZ-TrigIO in desktop mode, simply connect the 12V DC power plug from the AC adapter. The center pin must be positive and the adapter must be rated for **18W** or more.

Several outputs and internal settings are undefined after powerup, so it is recommended to log in quickly via serial port or SSH and apply the settings.

To power down the Pixie-16 MZ-TrigIO, first shut down the Linux OS (type `halt`), then turn off chassis power or remove the 12V DC power plug (and UART cable).

2.2 Serial Port (USB-UART)

The serial port connection on the Zynq SOM is made via a micro USB cable. It requires a driver to map the UART to a Windows serial port as well as a terminal running on the USB master PC. Windows installation consists of the following steps:

1. Download and extract/install the Silicon Labs CP210x USB-to-UART driver from www.silabs.com/products/development-tools/software/usb-to-uart-bridge-vcp-drivers. For more information, see microzed.org/sites/default/files/documentations/CP210x_Setup_Guide_1_2.pdf
2. Download and install Tera Term (or other suitable terminal program). See <http://ttssh2.osdn.jp/>
3. Connect USB cable between Zynq SOM board and PC, then power up the Pixie-16 MZ-TrigIO (see 2.1)

³ Note that the connections in J3 (EB_Data) connect within a PXI segment only (slots 1-7 or 8-14) while connections in J4 connect to all slots (TrigegrAll).

(PC: any USB port, Pixie-16 MZ-TrigIO: micro USB port on [red] Zynq SOM board)

4. From the Silicon Labs installation, run CP210xVCPInstaller_x64.exe to create a COM port
 - Find the new COM port's number in the Windows device manager
 - The COM port assignment is device specific, so this has to be repeated every time switching to a different Pixie-16 MZ-TrigIO unit. For the same Pixie-16 MZ-TrigIO, this is a one-time operation.
5. Open Tera Term.
 - Connect via the serial port showing the COM number above
 - Select Setup > Serial port. Defaults are ok, except baud rate must be 115200
 - Adjust the font and size if desired
 - Currently, no login credentials are required for Linux login via serial port

On a Linux system, it is possible to use the Minicom utility as for serial port I/O. Usually no installation or drivers are required. It can be configured via

```
sudo minicom -s
```

and by specifying `ttyUSB0` as the port, with *no* HW flow control. See also <https://help.ubuntu.com/community/Minicom>.

6.

2.3 SSH login

Tera Term or another suitable program can be used to log in via the network once the Pixie-16 MZ-TrigIO is powered, connected to a network, and its IP address is known. The IP address can be found by

- connecting via serial port terminal as described above, and typing `ifconfig`
- log on to router, see list of connected devices
- just try the one it had before, IP addresses seem fairly persistent through power cycles

To connect, open a terminal and make connection to the Pixie-16 MZ-TrigIO's IP address. Default ID/PW is **root/xia17pxn** (*please change*).

Two possible free SSH terminal program for Android are "SSH Client" and "JuiceSSH", allowing login from a tablet or smartphone.

2.4 Web Interface

Any web browser can be used to log in via the network once the Pixie-16 MZ-TrigIO is powered, connected to a network, and its IP address is known. The IP address can be found by

- connecting via serial port terminal as described above, and typing `ifconfig`
- log on to router, see list of connected devices
- just try the one it had before, they seem fairly persistent through power cycles

To connect, enter the Pixie-16 MZ-TrigIO's IP address as the web address in your browser. Any browser should work; XIA uses Firefox and does not test for compatibility with other browsers. No password is required for the Pixie-16 MZ-TrigIO home page, however, login

is required for web operations duplicating terminal access (**not yet implemented**). The default ID/PW is **webops/xia17pxn** (*please change*).

For proper operation of most links in the home page, prior execution of a function in the Linux terminal is required. See below for details.

2.5 SMB (Samba)

The Pixie-16 MZ-TrigIO Linux OS is running Samba, a “Windows interoperability suite of programs for Linux and Unix”. It allows the files in the Linux partition of the Pixie-16 MZ-TrigIO SD card to be shared with Windows networking. By default, only one folder is shared (/var/www) which is the location of the API functions.

To connect, type \\192.168.1.xxx\PNvarwww in the Windows Explorer location bar, where 192.168.1.xxx is the IP Address of the Pixie-16 MZ-TrigIO found as above and PNvarwww is the name given to /var/www for file sharing purposes in Samba. Windows will prompt for login; the default ID/PW is **root/xia17pxn** (*please change*).

The settings file settings.ini, output data files, and all (source and executable) files of the API functions can now easily be copied or edited with Windows tools and programs. However, to *execute* the API functions, serial port or SSH login is required (or in some cases, API functions can be executed from the web interface)

2.6 Required Initial Linux Commands

Once logged on via the Linux terminal or the web operations page, the following steps **must** be performed:

1. Change directory: `cd /var/www`
This is the directory visible via the web server. Data created by execution of XIA API functions can be read via the web browser from this directory. For convenience, it contains a “release” of all XIA SW functions and is used as the default working directory. (Not required for web operations page)
2. Apply settings to FPGA: `./progfippi`

2.7 Useful Linux Commands

The following commands may be helpful

1. Type `ifconfig` to find the IP address
The network does not come up by itself occasionally. Type `ifconfig eth1 up xxx.xxx.xxx.xxx` with a known allowed IP number to restart. The IP number can be omitted in some cases. Or, for a change of restart, type `sudo /etc/init.d/networking restart` else power cycle.
2. The temperature is reported as a raw number in
`/sys/devices/soc0/amba/f8007100.adc/iio:device0/in_temp0_raw`; the actual temperature can be computed by subtracting 2219 and multiplying with 0.12304. Board and Zynq temperatures are also reported in the status information and by `./progfippi`
3. To change a parameter value in the settings file without opening/editing/closing the file, a sed command as follows can be used:

```
sed -i `'/RUN_TYPE/c RUN_TYPE      1281' settings.ini
```

This replaces (-i = in the file) the line containing the string “RUN_TYPE” with the text “RUN_TYPE 1281”.

4. To mount a USB drive (e.g. to copy data or SW updates), type

```
mount /dev/sda1 /mnt/usb
```

 /var is not a suitable directory to mount the USB stick, as it confuses the web server
5. Type `date` to verify Linux time automatically updated to UTC
6. Use the `mount` command to mount external network drives. For example, to mount NASdrive/data from a PC with IP address 192.168.1.123, type

```
mount //192.168.1.123/data /mnt/data -o
```

```
"username=[user],password=[passwd]"
```

 with the appropriate values filled in for [user] and [passwd]
7. To mount the SD card boot partition to a folder /mnt/sd, execute

```
mount /dev/mmcblk0p1 /mnt/sd
```

 this is useful to update the boot files without removing the SD card. The Pixie-16 MZ-TrigIO has to be rebooted before the new boot files become effective.
8. To clear the command line history, type

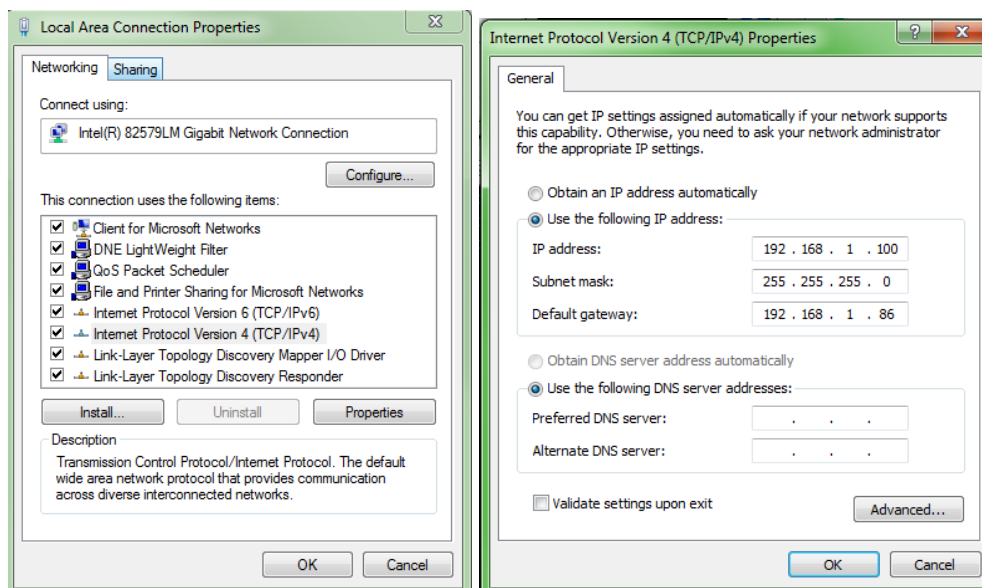
```
history -c
```

```
history -w
```

2.8 Direct Network Connection between Pixie-16 MZ-TrigIO and a Windows PC

The above description of setup and operation of the Pixie-16 MZ-TrigIO assumes both Pixie-16 MZ-TrigIO and the user device (PC, tablet, smartphone) are connected to a local network with DHCP server and gateway. Instead, it is possible to directly connect the Pixie-16 MZ-TrigIO to a laptop or desktop with a standard Ethernet cable (no crossover cable required). However, a number of configurations have to be set manually:

1. On the Pixie-16 MZ-TrigIO terminal, type `ifconfig eth1 up 192.168.1.86` to bring up the Ethernet connection and assign the IP number. In some cases, `eth0` or `eth2` has to be used instead of `eth1`.
2. On the Windows PC (here Windows 7), go to Start > Control Panel > Network and Sharing Center > Change Adapter Settings > Local Area Connection



In the Local Area Connection “Properties” dialog, select “Internet Protocol Version 4 (TCP/IPv4)” and click “Properties”.

In the properties dialog, set IP address, subnet mask, and gateway as shown above. (The gateway is equal to Pixie-16 MZ-TrigIO’s IP address.)

Note: The network will not be able to connect to the internet. The date/time of the Pixie-16 MZ-TrigIO will likely be initialized to 1970.

3 Pixie-16 MZ-TrigIO Operation

Basic operation of the Pixie-16 MZ-TrigIO uses a combination of terminal commands and web interface. The former controls all parameter settings (and possible DAQ monitoring) and requires login to the system. The latter displays data and is accessible to anyone. This provides a measure of security in instrument control while making it convenient to view data.

3.1 Adjust Parameters

The FPGA parameters control the connections between backplane and front panel, for example direction (front to back vs back to front) and output enable. In addition they control the “processing” of the inputs, for example building multiplicity sums and applying coincidence tests.

All settings are stored in an .ini file. A default settings file, *defaults.ini*, contains all the parameter settings as described in section 6. Settings are arranged into system (e.g. requested run time), local logic, and triggers (e.g. enable front panel output). In the triggers sections, each parameter is repeated 5 times; 3 times for the front panel groups (FRONT A-C, 16 signals each, in 4 RJ-45 connectors) and the 2 times for the backplane groups (TRIGGERALL with 32 signals and EBDATA with 16 signals). A condensed settings file, *settings.ini*, contains the most important parameters and **will override the defaults**. The normal method to change settings is to edit *settings.ini* and then execute `./progfippi` to apply them to the FPGA. If necessary, lines with additional parameters can be copied from *defaults.ini* into *settings.ini*; also lines can be deleted from *settings.ini* if there is no need to vary specific parameters. The file *defaults.ini* should be considered a read-only file. Editing can be accomplished with a built-in Linux editor through the terminal (for example VI) or by opening the file in a Windows editor through the SMB file sharing.

The most important parameters for initial setup are

FRONT_A_OUTENA Enable outputs on front panel, one bit set or cleared for each signal in group A.
equivalent for group B, C and the backplane groups

FRONT_A_OUTPUT_SELECT A number between 0 and 5 to choose which signal to output

After execution of `./progfippi`, you can also open/refresh the Status page in the web browser or execute `./runstats` and read the output parameters in the resulting RS.csv file. The current state of the inputs, results of coincidence tests, temperatures, and FPGA system time will be reported. In a Windows Explorer window pointing to `\\<Pixie_Net_IP>\PNvarwww`, the file RS.csv can be copied or opened by Windows tools and programs.

The typical startup sequence (see also section 2.6) is to run `./progfippi` pretty much as soon as the modules is powered up to initialize all parameters in the FPGA logic. Every time there is a change in *settings.ini*, run `./progfippi` to apply the changes to the pulse processing logic.

3.2 Data Acquisition Monitoring

The C function “monitordaq” is a template demonstrating how to periodically read FPGA status registers and print them to file.

3.3 User Interface Options

There are a variety of interface options to operate the Pixie-16 MZ-TrigIO. They range from basic terminal and file I/O to plug-ins for data acquisition or data analysis software. Fundamentally, the Pixie-16 MZ-TrigIO communicates through generic interfaces such as serial port and Ethernet, so that any program emulating a serial port and reading files from a network drive can be used to operate the system. A few examples are listed in the following sections

3.3.1 Terminal

```

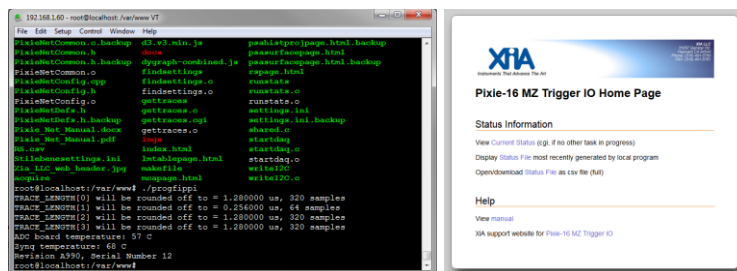
root@localhost: /usr/local
./progfippi
TRACE_LENGTH(0) will be rounded off to = 1.280000 us, 320 samples
TRACE_LENGTH(1) will be rounded off to = 0.256000 us, 64 samples
TRACE_LENGTH(2) will be rounded off to = 1.280000 us, 320 samples
TRACE_LENGTH(3) will be rounded off to = 1.280000 us, 320 samples
ADC board temperature: 57 C
Syng temperature: 68 C
Revision A990, Serial Number 12
root@localhost: /usr/local

```

At the most basic level, users can log in to the Pixie-16 MZ-TrigIO with a terminal program and execute the C programs to set up and read status (e.g. ./progfippi and ./runstats). Settings are modified by editing the settings file settings.ini with a basic text editor like VI. Results are files on the Pixie-16 MZ-TrigIO’s SD card, which can be viewed with a text editor or copied to mounted network drives.

This environment may appeal most to users familiar with Linux, and also allows modification and recompilation of the C programs with the built-in gcc compiler.

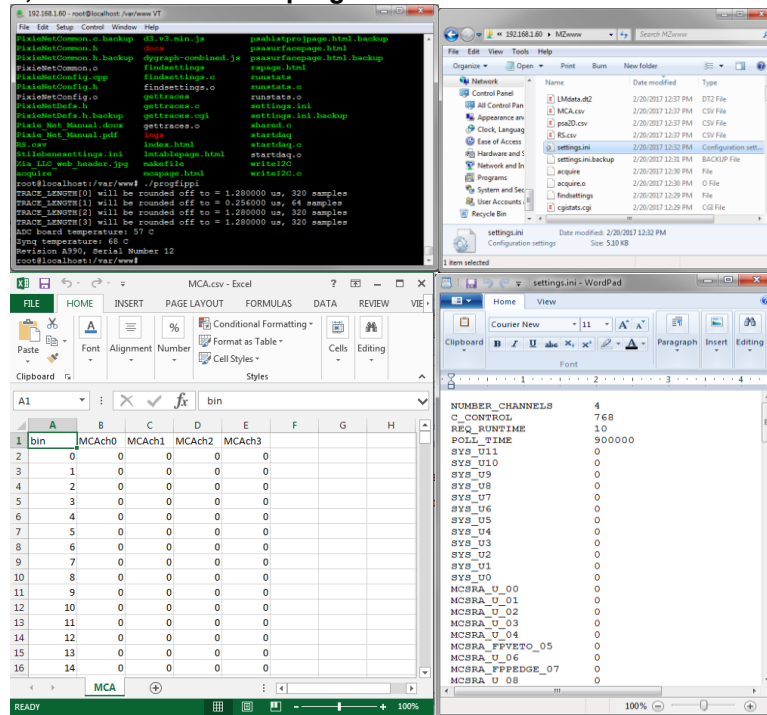
3.3.2 Terminal and Webpages



For direct graphical feedback, users can log in via terminal as in the preceding section, but view results as a webpage. The Pixie-16 MZ-TrigIO is running a web server, and the output files are formatted as plots or tables on webpages with a variety of java scripts.

While still requiring familiarity with Linux, the graphical feedback makes for an easier setup and a better presentation of the results.

3.3.3 Terminal, SMB and Windows programs



With SMB file sharing, Windows programs can be used to directly view, edit, and analyze settings and results. For example, the workflow could be

- 1) Using WordPad, open, modify, and save settings.ini
- 2) Execute `./progfippi` and `./runstats` in the terminal
- 3) Open RS.csv in Excel or other program to view, analyze, and plot results

3.3.4 Webpage-Only Operation

Not yet implemented

As an alternative to the terminal entry and execution of the C programs to set up and read status (e.g. `./progfippi` and `./runstats`), there is a [moderately] secure web page that allows execution of the C programs as a cgi script through a browser. The Web Operation page can be reached via a link in the “CGI Current Data” section of the Pixie-16 MZ-TrigIO Home page. It essentially lists the C functions that would normally be called by typing in the terminal; clicking on the function name executes the function. Data is created in a separate subdirectory with Linux permissions for data write access for the webserver (`/var/www/webops`). The Web Operation page shows the equivalent links from the Pixie-16 MZ-TrigIO home page to download and display the data.

Settings files can currently not be modified through the Web Operation page.

As this webpage allows generation and execution of files from any remote user, the webserver has been set up to require authentication to access this webpage. Details of the setup are described in section 10. The default user ID/password is **webops/xia17pxn**, currently specified as plain text in the file `/var/www/webopspasswords`. Obviously, this is

only moderately secure and should be a) changed by the user as soon as possible and b) upgraded to encrypted password storage for sensitive environments.

3.3.5 On-board ROOT Graphical User Interface

Not yet implemented

A further alternative to the terminal entry and execution of the C programs is available on Pixie-16 MZ-TrigIO systems with the Ubuntu 15 operating system, which comes with the ROOT software package installed. A ROOT GUI can be developed that can configure the Pixie-16 MZ-TrigIO (`./progfippi` etc), view ADC traces, read the status information (`./startdaq`), and display the results.

To use this option, two steps are required:

1) Graphical log on

The Pixie-16 MZ-TrigIO processor is capable of generating a graphical desktop, but since there is no video output, direct connection of a monitor is not possible. Instead, users can log on remotely with either

- a. Windows Remote Desktop: Start Remote Desktop on a remote Windows PC and enter the Pixie-16 MZ-TrigIO's IP address, then log on (default user/password **root/xia17pxn**)
- b. Linux SSH X11 forwarding: `ssh -Y` and `-X` let you run an X11 program on a remote machine, with its windows appearing on the local X monitor. For example, in a terminal window of the remote Linux PC, type `ssh -Y root@192.168.1.74` (with appropriate IP address)

2) Start ROOT and Pixie-16 MZ-TrigIO application

- Open a terminal on the graphical desktop
- Type `cd /var/www` to change to the main DAQ directory
- Type `root` to start ROOT
- Type `./x pixieNet.C` to start the Pixie-16 MZ-TrigIO ROOT application

4 API functions

4.1 progfippi

Functions performed

- Parse .ini file, extract FPGA parameters
- Convert into “FPGA units”, e.g. number of samples instead of us (where necessary)
- Apply limits and dependencies, if not ok, give feedback and abort
- Read data from PROM and verify the current version of the Pixie-16 MZ-TrigIO hardware is supported by the software
- Write to FPGA registers to apply settings
- Toggle I2C lines (0x002) to set LVDS direction etc.

Arguments: fixed to “settings.ini”

Output: Errors for bad settings, prints hardware info from PROM and temperatures

Restrictions: Do not execute during a data acquisition

4.2 runstats, cgistats

These functions are used to read and output the status registers. runstats writes them to a local file RS.csv. cgistats prints them as a webpage to std out, which can be piped into the webserver; this is displayed as the “Status” webpage. These functions can be used for development, setup, and diagnostics, for example the run statistics include module revision and serial number, temperature, ICR and OOR.

Status registers can also be read and written to the file RS.csv by data acquisition monitoring routines (to be implemented). During a data acquisition, do not execute runstats and cgistats (i.e. do not load webpage “Current status” = cgistats.cgi) to avoid conflicting access to the run status registers and possible delays and dead times. Instead monitor the file RS.csv (i.e. load webpage “Status File”= rspage.html)

Functions performed

- Read Runstats registers in PL
- Read I2C info from PROM
- write to file

Arguments: none

Output: RS.csv

Restrictions: Do not execute during a data acquisition

Notes: The cgi program requires presence of the web page template rspage.html in the same folder.

4.3 clockprog

Programs the clock PLL that buffers PTP or external clocks for FPGA and backplane. A number of modes are hardcoded in the program; others can be added with the CDCE813-Q1 data sheet and TI's ClockPro utility as a reference. This allows for example using an external clock of arbitrary frequency. See also description of PTP variant.

Functions performed

- Program PLL for PTP and external clock

Arguments: Mode (read and display, write to EEPROM, program PLL registers)

Mode 0: read and display registers

Mode 1: write current register contents to PROM memory

Mode 2: Input 25 MHz, output 25 MHz on Y1 (FPGA)

Mode 3: Input 25 MHz, output 25 MHz on Y1 (FPGA) and Y3 (External)

Mode 4: Input 25 MHz, output 50 MHz on Y1 (FPGA), Y2 (backplane), and Y3 (External)

Mode 5: Input 50 MHz, output 50 MHz on Y1 (FPGA), Y2 (backplane), and Y3 (External)

Output: display current register settings

Restrictions: Do not execute during a data acquisition

4.1 monitordaq

Template for monitoring Pixie-16 data acquisition. At this point, it runs for the time specified in the settings file and periodically writes the status registers to file.

Functions performed

- Read settings files
- Reset counters in FPGA
- Loop for requested time, save status to RS.csv periodically

Arguments: None

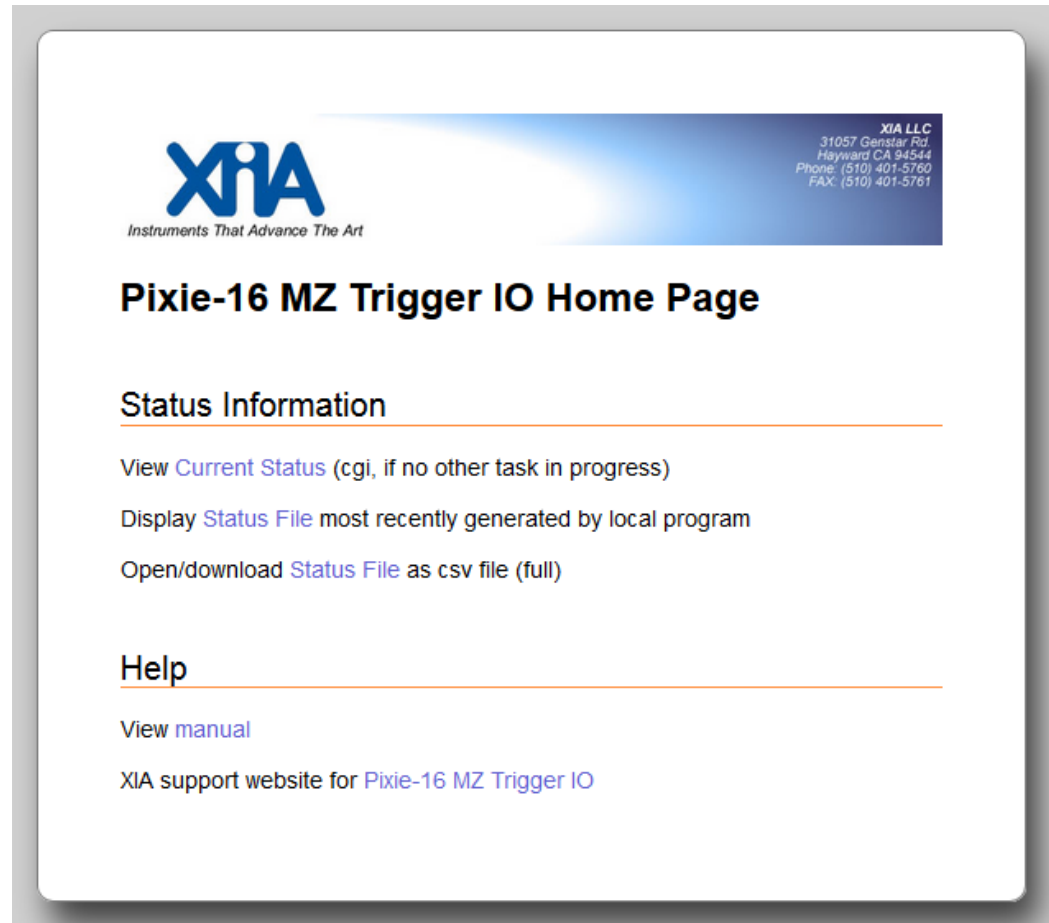
Output: RS.csv

Restrictions: Do not execute runstats or cgstats while monitordaq is in progress

5 Web Pages

The Pixie-16 MZ-TrigIO web pages are located in /var/www; this is the default directory for the installed lighttpd webserver. Browsing to the Pixie-16 MZ-TrigIO's IP address will bring up index.html, from which all other pages can be accessed.

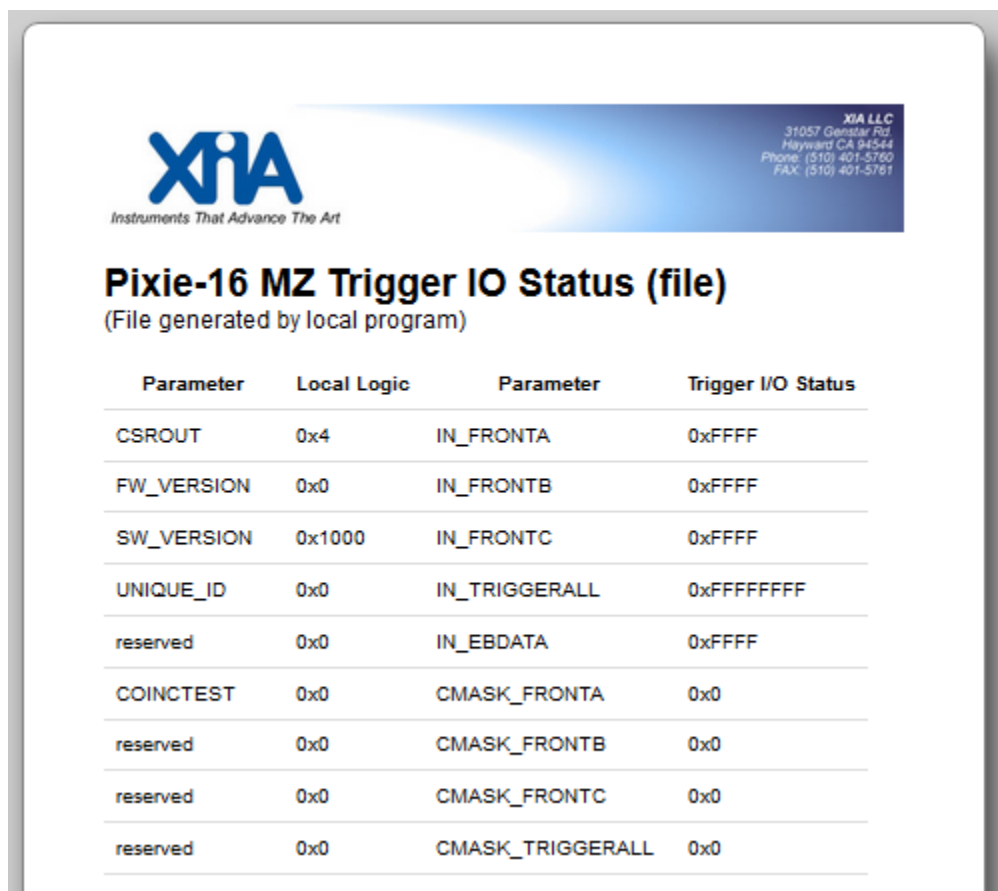
5.1 index.html



This is the home page for the Pixie-16 MZ-TrigIO.

At the top of the page, there are a few shortcuts to common tasks or pages. Do not use the Current Status link while another task is in progress, instead use the Status File link. You can also download the status file

5.2 rspage.html, cgistats.cgi



XIA
Instruments That Advance The Art

XIA LLC
31057 Genstar Rd
Hayward CA 94544
Phone: (510) 401-5760
FAX: (510) 401-5761

Pixie-16 MZ Trigger IO Status (file)

(File generated by local program)

| Parameter | Local Logic | Parameter | Trigger I/O Status |
|------------|-------------|------------------|--------------------|
| CSROUT | 0x4 | IN_FRONTA | 0xFFFF |
| FW_VERSION | 0x0 | IN_FRONTB | 0xFFFF |
| SW_VERSION | 0x1000 | IN_FRONTC | 0xFFFF |
| UNIQUE_ID | 0x0 | IN_TRIGGERALL | 0xFFFFFFFF |
| reserved | 0x0 | IN_EBDATA | 0xFFFF |
| COINTEST | 0x0 | CMASK_FRONTA | 0x0 |
| reserved | 0x0 | CMASK_FRONTB | 0x0 |
| reserved | 0x0 | CMASK_FRONTC | 0x0 |
| reserved | 0x0 | CMASK_TRIGGERALL | 0x0 |

The rspage contains a table showing the Run Status information: the inputs from front and rear, version numbers, temperatures. Internally, this is a d3 javascript that links to the file RS.csv. The file is generated by executing `./runstats` on the Pixie-16 MZ-TrigIO terminal.

A similar page reading and displaying the data can be generated by the script `cgistats.cgi` from the web browser. This should only be used when no other task is in progress.

Units in the pages (and the underlying csv file) are in seconds or counts/s, or raw internal units (numbers, counts and ns).

6 Parameters in the Settings Files

The ini files contain the parameter settings for the data acquisition. There are two files, *defaults.ini* and *settings.ini*. The file *defaults.ini* should be considered read-only; it contains defaults for all parameters. The file *settings.ini* contains a subset of most relevant parameters, which will overwrite the defaults. Parameter lines can be added or removed from *settings.ini* to focus better on parameters relevant for a certain application. The .ini files are ASCII text files, one line per parameter. Parameter name spelling must be maintained, but the order is not essential. (Users customizing code can add parameters at the end of the file.) The parameter values are in physical units, and will be translated by the `progfippi` routine into values and bit patterns and then written into PL input registers.

Several control bit patterns are broken out as one bit per line. For other, less commonly used bit patterns, their value can be written as decimals or hexadecimals, for example 65536 or 0xFFFF.

The following sections describe the parameters. Unused parameters are shown in gray. Key parameters for *settings.ini* are shown in bold. NYI = not yet implemented.

6.1 System Parameters

Reserved for system wide parameters that do not need to be programmed into the FPGA. For example, when monitoring a Pixie-16 data acquisition, one might want to specify the interval in which to check run status using `POLL_TIME`.

| Parameter Name | Units Limits | Description |
|-----------------|--------------------------------|--|
| NUMBER_CHANNELS | 4 | Number of channels |
| C_CONTROL | Bits, 0x0-FFFF | unused |
| REQ_RUNTIME | Seconds, 5..2 ³² | Requested run time in (full) seconds |
| POLL_TIME | Tbd, 100..2 ³² | Number of internal polling loops between updates of csv output files. In the order of microseconds |

6.2 Local Control

Local Control parameters are intended for settings that affect the local logic in the FPGA. Examples include which reset signals to use for internal counters (PTP, Pixi-16 backplane, external) or what to count. Currently all parameters are unused.

| Parameter Name | Units Limits | Description |
|----------------------|--------------|---|
| LOCAL_CONTROL_00_B00 | 0 or 1 | Example of a single control bit from parameter LOCAL_CONTROL_00 |
| LOCAL_CONTROL_00_B01 | 0 or 1 | Example of a single control bit from parameter LOCAL_CONTROL_00 |
| LOCAL_CONTROL_00_B02 | 0 or 1 | Example of a single control bit from parameter LOCAL_CONTROL_00 |
| LOCAL_CONTROL_00_B03 | 0 or 1 | Example of a single control bit from parameter LOCAL_CONTROL_00 |
| LOCAL_CONTROL_00_B04 | 0 or 1 | Example of a single control bit from parameter LOCAL_CONTROL_00 |
| LOCAL_CONTROL_00_B05 | 0 or 1 | Example of a single control bit from parameter LOCAL_CONTROL_00 |
| LOCAL_CONTROL_00_B06 | 0 or 1 | Example of a single control bit from parameter LOCAL_CONTROL_00 |
| LOCAL_CONTROL_00_B07 | 0 or 1 | Example of a single control bit from parameter LOCAL_CONTROL_00 |
| LOCAL_CONTROL_00_B08 | 0 or 1 | Example of a single control bit from parameter LOCAL_CONTROL_00 |
| LOCAL_CONTROL_00_B09 | 0 or 1 | Example of a single control bit from parameter LOCAL_CONTROL_00 |
| LOCAL_CONTROL_00_B10 | 0 or 1 | Example of a single control bit from parameter LOCAL_CONTROL_00 |
| LOCAL_CONTROL_00_B11 | 0 or 1 | Example of a single control bit from parameter LOCAL_CONTROL_00 |
| LOCAL_CONTROL_00_B12 | 0 or 1 | Example of a single control bit from parameter LOCAL_CONTROL_00 |
| LOCAL_CONTROL_00_B13 | 0 or 1 | Example of a single control bit from parameter LOCAL_CONTROL_00 |
| LOCAL_CONTROL_00_B14 | 0 or 1 | Example of a single control bit from parameter LOCAL_CONTROL_00 |
| LOCAL_CONTROL_00_B15 | 0 or 1 | Example of a single control bit from parameter LOCAL_CONTROL_00 |
| LOCAL_CONTROL_01 | 0-64K | 16bit local control variable, unused |
| LOCAL_CONTROL_02 | 0-64K | 16bit local control variable, unused |
| LOCAL_CONTROL_03 | 0-64K | 16bit local control variable, unused |
| LOCAL_CONTROL_04 | 0-64K | 16bit local control variable, unused |
| LOCAL_CONTROL_05 | 0-64K | 16bit local control variable, unused |
| LOCAL_CONTROL_06 | 0-64K | 16bit local control variable, unused |
| LOCAL_CONTROL_07 | 0-64K | 16bit local control variable, unused |

6.3 Trigger Control Parameters

Trigger Control parameters define the routing of backplane and front panel signals.

| Parameter Name | Type | Description |
|---------------------------|---------|---|
| FRONT_A_OUTENA | 16 bits | Bit pattern to enable outputs to the 16 front panel LVDS signals. |
| FRONT_B_OUTENA | 16 bits | Bit pattern to enable outputs to the 16 front panel LVDS signals. |
| FRONT_C_OUTENA | 16 bits | Bit pattern to enable outputs to the 16 front panel LVDS signals. |
| TRIGGERALL_OUTENA | 32 bits | Bit pattern to enable outputs to the 32 backplane trigger signals. Currently must be 0, no output to backplane |
| EBDATA_OUTENA | 16 bits | Bit pattern to enable outputs to the 16 backplane trigger signals. Currently must be 0, no output to backplane |
| FRONT_A_COINC_MASK | 16 bits | Inputs are bitwise ANDed (masked) with this pattern before being used in the coincidence test. |
| FRONT_B_COINC_MASK | 16 bits | |
| FRONT_C_COINC_MASK | 16 bits | |
| TRIGGERALL_COINC_MASK | 32 bits | |
| EBDATA_COINC_MASK | 16 bits | |
| FRONT_A_MULT_MASK | 16 bits | Inputs are bitwise ANDed (masked) with this pattern before being used in the multiplicity sum. |
| FRONT_B_MULT_MASK | 16 bits | |
| FRONT_C_MULT_MASK | 16 bits | |
| TRIGGERALL_MULT_MASK | 32 bits | |
| EBDATA_MULT_MASK | 16 bits | |
| FRONT_A_COINC_PATTERN | 16 bits | Masked inputs are compared to this bit pattern, if equal, a coincidence test bit is set to 1 A – bit 0 TrigAll - bit 3 B – bit 1 EBData – bit 4 C – bit 2 |
| FRONT_B_COINC_PATTERN | 16 bits | |
| FRONT_C_COINC_PATTERN | 16 bits | |
| TRIGGERALL_COINC_PATTERN0 | 32 bits | |
| EBDATA_COINC_PATTERN | 16 bits | |
| FRONT_A_MULT_THRESHOLD | 0-16 | Masked inputs are summed and the sum is compared to this threshold. If greater or equal, a coincidence test bit is set to 1 A – bit 8 TrigAll - bit 11 B – bit 9 EBData – bit 12 C – bit 10 |
| FRONT_B_MULT_THRESHOLD | 0-16 | |
| FRONT_C_MULT_THRESHOLD | 0-16 | |
| TRIGGERALL_MULT_THRESHOLD | 0-32 | |
| EBDATA_MULT_THRESHOLD | 0-16 | |
| FRONT_A_OUTPUT_SELECT | 0-5 | Defines the signal to output on Front A panel LVDS pins 0 – TriggerAll [15:00] 1 - TriggerAll [15:00] & coinc mask 2 - TriggerAll [15:00] & multiplicity mask 3 – coincidence test bit 3 (all 16 lines) 4 – coincidence test 11 (all 16 lines) 5 – test signal, periodic toggle |

| | | |
|--------------------------|-----|---|
| FRONT_B_OUTPUT_SELECT | 0-5 | Defines the signal to output on Front B panel LVDS pins 0 – TriggerAll [31:16] 1 - TriggerAll [31:16] & coinc mask 2 - TriggerAll [31:16] & multiplicity mask 3 – coincidence test bit 3 (all 16 lines) 4 – coincidence test 11 (all 16 lines) 5 – test signal, periodic toggle |
| FRONT_C_OUTPUT_SELECT | 0-5 | Defines the signal to output on Front C panel LVDS pins 0 – EB_Data [31:16] 1 - EB_Data [31:16] & coinc mask 2 - EB_Data [31:16] & multiplicity mask 3 – coincidence test bit 4 (all 16 lines) 4 – coincidence test 12 (all 16 lines) 5 – test signal, periodic toggle |
| TRIGGERALL_OUTPUT_SELECT | 0-5 | Ignored. Backplane output is always 0 |
| EBDATA_OUTPUT_SELECT | 0-5 | Ignored. Backplane output is always 0 |

7 Data Formats

7.1 Status Files (RS.csv)(

Status files contain comma separated values. The first row lists the column headers. The columns are

| | |
|------------|--------------------------------|
| ParameterL | Name of local result/status |
| LocalLogic | Value of local result/status |
| ParameterT | Name of trigger status/result |
| Channel# | Value of trigger status/result |

The units of the reported values are generally in seconds, nanoseconds, or counts per seconds. For full definition and explanation of the values, please see section 8.

For example, the line

```
FW_VERSION,0x0,IN_FRONTB,0xFFFF
```

reports that firmware version 0x0000 is used and that the front panel inputs in group B are all logic high.

```
ParameterL,LocalLogic,ParameterT,TriggerIOStatus
CSROUT,0x4,IN_FRONTA,0xFFFF
FW_VERSION,0x0,IN_FRONTB,0xFFFF
SW_VERSION,0x1000,IN_FRONTC,0xFFFF
UNIQUE_ID,0x0,IN_TRIGGERALL,0xFFFFFFFF
reserved,0x0,IN_EBDATA,0xFFFF
COINCTEST,0x0,CMASK_FRONTA,0x0
reserved,0x0,CMASK_FRONTB,0x0
reserved,0x0,CMASK_FRONTC,0x0
reserved,0x0,CMASK_TRIGGERALL,0x0
reserved,0x0,CMASK_EBDATA,0x0
RUNTICKS,0,MMSUM_FRONTA,0
RUNTICKS,0,MMSUM_FRONTB,0
NUMTRIGGERS,0,MMSUM_FRONTC,0
NUMTRIGGERS,0,MMSUM_TRIGGERALL,0
RUNTIME,0.000000,MMSUM_EBDATA,0
TRIGGERRATE,0.000000,reserved,0
T_ZYNQ,54,reserved,0
T_BOARD,28,reserved,0
reserved,0,reserved,0
```

RunTime is a result computed by the .runstats routine, converting the number of clock cycles in the RUNTICKS counter into seconds.

TriggerRate is a is a result computed by the .runstats routine, dividing the number of pulses counted in NUMTRIGGERS by the runtime. The result is counts/seconds.

8 Registers Visible to Linux

The FPGA (PL) registers described below are used to apply settings to the firmware and read back event data. This is generally handled by XIA's API functions and users do not need to go in depth to understand this functionality. The primary way to set registers is by specifying parameters in the .ini file and call XIA's API function `progfippi` to convert and apply them as appropriate. XIA may change register addresses and bits in future code revision, but will try to keep the settings file format constant.

FPGA (PL) I/O registers are organized into 2 blocks. All registers are 32bit wide. Only registers in block 0 can be written to, and the block number is ignored. To read registers, the output block number has to be written to address 0x003; block 0 reads back the values written and block 1 reads status registers that are defined in the logic.

8.1 Input/Output Registers

Address range is 0x000 – 0x1FF. Can be read back to verify I/O with `OUTBLOCK = 0`.

At this time, the input register definitions copy exactly the settings file parameters. More generally, the settings files may specify a time in microseconds (e.g. a coincidence window) and the input register. Also, note that the addresses have gaps.

| Register | Address | R/W | Description |
|-------------------|-------------|-----|---|
| CSRIN | 0x000 | R/W | Run Control Register bits 0 RunEnable (set to start DAQ run, unused) |
| unused | 0x001 | R/W | unused |
| I2C | 0x002 | R/W | Control the SDA and SCL lines 0 SDA 1 SCL 2 SDA ENA (SDA output enable) |
| OUTBLOCK | 0x003 | R/W | Specifies address range for reads If 0: read back I/O If 1: read FPGA results |
| unused | 0x004-0x008 | | unused |
| COUNTER_CLR | 0x009 | W | Writing to this register issues a pulse to clear counters. |
| unused | 0x00A-0x0FF | R/W | unused |
| FRONT_A_OUTENA | 0x100 | R/W | Bit pattern to enable outputs to the 16 front panel LVDS signals. |
| FRONT_B_OUTENA | 0x101 | R/W | Bit pattern to enable outputs to the 16 front panel LVDS signals. |
| FRONT_C_OUTENA | 0x102 | R/W | Bit pattern to enable outputs to the 16 front panel LVDS signals. |
| TRIGGERALL_OUTENA | 0x103 | R/W | Bit pattern to enable outputs to the 32 backplane trigger signals. |

| | | | |
|---------------------------|-------|-----|--|
| EBDATA_OUTENA | 0x104 | R/W | Bit pattern to enable outputs to the 16 backplane trigger signals. |
| FRONT_A_COINC_MASK | 0x108 | R/W | Inputs are bitwise ANDed (masked) with this pattern before being used in the coincidence test. |
| FRONT_B_COINC_MASK | 0x109 | R/W | |
| FRONT_C_COINC_MASK | 0x10A | R/W | |
| TRIGGERALL_COINC_MASK | 0x10B | R/W | |
| EBDATA_COINC_MASK | 0x10C | R/W | |
| FRONT_A_MULT_MASK | 0x110 | R/W | Inputs are bitwise ANDed (masked) with this pattern before being used in the multiplicity sum. |
| FRONT_B_MULT_MASK | 0x111 | R/W | |
| FRONT_C_MULT_MASK | 0x112 | R/W | |
| TRIGGERALL_MULT_MASK | 0x113 | R/W | |
| EBDATA_MULT_MASK | 0x114 | R/W | |
| FRONT_A_COINC_PATTERN | 0x118 | R/W | Masked inputs are compared to this bit pattern, if equal, a coincidence test bit is set to 1 A – bit 0 TrigAll - bit 3 B – bit 1 EBData – bit 4 C – bit 2 |
| FRONT_B_COINC_PATTERN | 0x119 | R/W | |
| FRONT_C_COINC_PATTERN | 0x11A | R/W | |
| TRIGGERALL_COINC_PATTERN | 0x11B | R/W | |
| EBDATA_COINC_PATTERN | 0x11C | R/W | |
| FRONT_A_MULT_THRESHOLD | 0x120 | R/W | Masked inputs are summed and the sum is compared to this threshold. If greater or equal, a coincidence test bit is set to 1 A – bit 8 TrigAll - bit 11 B – bit 9 EBData – bit 12 C – bit 10 |
| FRONT_B_MULT_THRESHOLD | 0x121 | R/W | |
| FRONT_C_MULT_THRESHOLD | 0x122 | R/W | |
| TRIGGERALL_MULT_THRESHOLD | 0x123 | R/W | |
| EBDATA_MULT_THRESHOLD | 0x124 | R/W | |
| FRONT_A_OUTPUT_SELECT | 0x128 | R/W | Defines the signal to output on Front A panel LVDS pins 0 – TriggerAll [15:00] 1 - TriggerAll [15:00] & coinc mask |

| | | | |
|--------------------------|-------|-----|---|
| | | | 2 - TriggerAll [15:00] & multiplicity mask 3 – coincidence test bit 3 (all 16 lines) 4 – coincidence test 11 (all 16 lines) 5 – test signal, periodic toggle |
| FRONT_B_OUTPUT_SELECT | 0x129 | R/W | Defines the signal to output on Front B panel LVDS pins 0 – TriggerAll [31:16] 1 - TriggerAll [31:16] & coinc mask 2 - TriggerAll [31:16] & multiplicity mask 3 – coincidence test bit 3 (all 16 lines) 4 – coincidence test 11 (all 16 lines) 5 – test signal, periodic toggle |
| FRONT_C_OUTPUT_SELECT | 0x12A | R/W | Defines the signal to output on Front C panel LVDS pins 0 – EB_Data [31:16] 1 - EB_Data [31:16] & coinc mask 2 - EB_Data [31:16] & multiplicity mask 3 – coincidence test bit 4 (all 16 lines) 4 – coincidence test 12 (all 16 lines) 5 – test signal, periodic toggle |
| TRIGGERALL_OUTPUT_SELECT | 0x12B | R/W | Ignored. Backplane output is always 0 |
| EBDATA_OUTPUT_SELECT | 0x12C | R/W | Ignored. Backplane output is always 0 |

8.2 Output Registers

Address range is 0x000 – 0x1FF. Read only. Must set OUTBLOCK = 1 to read

| Register | Address | R/W | Description |
|-------------|-------------|-----|---|
| CSROUT | 0x000 | R | Run Status info bits 0 RunEnable (unused) 2 SDA readback 3 PTP trigger 3 4 PTP trigger 4 |
| SYSREVISION | 0x001 | R | Firmware version number |
| reserved | 0x002-0x004 | R | Placeholder for C variables |
| COINCRESULT | 0x005 | R | 0 – masked FRONT A == COINC PAT A 1 – masked FRONT B == COINC PAT B 2 – masked FRONT C == COINC PAT C 3 – masked TRIGGERALL == COINC PAT TA. 4 – masked EB_DATA == COINC PAT EB_D. 5-7 unused 8 – masked sum of FRONT A >= threshold 9 – masked sum of FRONT B >= threshold 10 - masked sum of FRONT C >= threshold 11 - masked sum of TriggerAll >= threshold |

| | | | |
|------------------|-------------|---|--|
| | | | 12 - masked sum of EB_DATA >= threshold 13-15 - unused |
| unused | 0x006-0x009 | R | unused |
| NUMTRIG_L | 0x00A | R | Number of pulses on TriggerAll[7] (low, high) Reset by writing to 0x009 TriggerAll[7] is the Pixie-16 chassis fast trigger. |
| NUMTRIG_H | 0x00B | R | |
| RUNTICKS_L | 0x00C | R | Number of clock tick since last reset (low, high) Reset by writing to 0x009 One tick is 10ns |
| RUNTICKS_H | 0x00D | R | |
| reserved | 0x00E-0x011 | R | Reserved for temperatures and C results |
| SNUM | 0x012 | R | Serial number from PROM |
| unused | 0x013-0x0FF | R | unused |
| IN_FRONTA | 0x100 | R | Status of the IO pins. If a pin is defined as an output, the IO pin is equal to the output value. If defined as input, the IO pin is equal to the external input value. |
| IN_FRONTB | 0x101 | R | |
| IN_FRONTC | 0x102 | R | |
| IN_TRIGGERALL | 0x103 | R | |
| IN_EBDATA | 0x104 | R | |
| unused | 0x105-0x107 | R | unused |
| CMASK_FRONTA | 0x108 | R | Status of the IO pins masked with the coincidence mask |
| CMASK_FRONTA | 0x109 | R | |
| CMASK_FRONTA | 0x10A | R | |
| CMASK_TRIGGERALL | 0x10B | R | |
| CMASK_EBDATA | 0x10C | R | |
| unused | 0x10D-0x10F | R | unused |
| MMSUM_FRONTA | 0x110 | R | Sum of the IO pins after masking with the multiplicity mask |
| MMSUM_FRONTA | 0x111 | R | |
| MMSUM_FRONTA | 0x112 | R | |
| MMSUM_TRIGGERALL | 0x113 | R | |
| MMSUM_EBDATA | 0x114 | R | |
| unused | 0x115-0x1FF | R | unused |

9 Linux Configuration

9.1 Licensing Information

The Pixie-16 MZ-TrigIO software includes several components that are licensed with the GNU general public license (GPL). For most of them, the software is unchanged, and the source code is provided on the Pixie-16 MZ-TrigIO SD card with the corresponding GPL license files. A small portion of the software is modified by XIA and the modified source code is provided both on the SD card and on XIA's website. The source code for the Linux kernel built by XIA for the Pixie-16 MZ-TrigIO and for the full collection of Ubuntu Linux programs is too large for the SD card or to host by XIA, but we can provide it upon request. XIA strives to keep those programs up to date and our own code compatible with the latest version, however, the best source of the programs may be their official repository.

List of key GPL programs:

- LinuxPTP source code on SD card or
 at <http://linuxptp.sourceforge.net/>
- ptp-mii-tool source code on SD card or
 at <https://github.com/giftnuss/net-tools> (original mii-tool) or
 at <http://support.xia.com/default.asp?W772> (XIA modification)
- Linux kernel source code per request or
 at <https://github.com/Xilinx/linux-xlnx>
- Ubuntu files source code and configuration files on SD or
 at [https://releases.linaro.org/ubuntu/images/](https://releases.linaro.org/ubuntu/images/developer/latest/linaro-vivid-developer-20151215-714.tar.gz)
 developer/latest/linaro-vivid-developer-20151215-714.tar.gz
 or per request

includes

- Lighttpd
- Samba
- g++
- i2c-tools

9.2 Software Information

The Pixie-16 MZ-TrigIO Linux system was originally based on Xillinux (1.3), which is based on Ubuntu LTS 12.04. The initial distribution can be downloaded from the Xillinux website (<http://xillybus.com/xillinux>). In what is distributed by XIA, the initial setup steps as described in the Xillinux documentation have been applied, including copying the Linux OS image on an SD card and increasing the “disk space” to 16GB.

(http://xillybus.com/downloads/doc/xillybus_getting_started_zynq.pdf)

In the current software, Ubuntu 15 with kernel version 4 is used. This has no major effect on the Pixie-16 MZ-TrigIO operation, but more apps from the Linux universe are available.

The SD card contains mainly the Linux OS (not visible to Windows) and 4 boot files (visible) in a small FAT partition. FPGA configuration updates have to be copied to that partition.

Notes for the particular configuration of Ubuntu 15 for Pixie-16 MZ-TrigIO:

- Installed Lighttpd web server
The default webpage is in /var/www. Need to modify /etc/lighttpd/lighttpd.conf as follows: add mod_cgi under server.modules and

```
$HTTP["url"] =~ "/cgi-bin/" {
    cgi.assign = ( "" => "" )
}

cgi.assign      = (
    ".cgi"      => ""
)
```

Further modify for authentication of the webops section: see below

- sudo apt-get install g++
- sudo apt-get install i2c-tools
- install boost libraries 1.61 in /usr/local and compiled all binaries (Xilinx) or installed boost version 1.55 from Ubuntu
- Installed Samba

Following Ubuntu help.” How to Create a Network Share Via Samba ...”⁴
About This Guide

In this text, I teach how to create a network share via Samba using the CLI (Command-line interface/Linux Terminal) in an uncomplicated, simple and brief way targeting Windows users.

Procedures

All commands must be done as root (precede each command with 'sudo' or use 'sudo su').

1. Install Samba

```
sudo apt-get update
sudo apt-get install samba
```

2. Set a password for your user in Samba

```
sudo smbpasswd -a <user_name>
```

Note: Samba uses a separate set of passwords than the standard Linux system accounts (stored in /etc/samba/smbpasswd), so you'll need to create a Samba password for yourself. This tutorial implies that you will use your own user and it does not cover situations involving other users passwords, groups, etc...

Tip1: Use the password for your own user to facilitate.

Tip2: Remember that your user must have permission to write and edit the folder you want to share.

Eg.:
sudo chown <user_name> /var/opt/blah/blahblah
sudo chown :<user_name> /var/opt/blah/blahblah

Tip3: If you're using another user than your own, it needs to exist in your system beforehand, you can create it without a shell access using the following command :

```
sudo useradd USERNAME --shell /bin/false
```

You can also hide the user on the login screen by adjusting lightdm's configuration, in /etc/lightdm/users.conf add the newly created user to the line :
hidden-users=

Contents

1. About This Guide
2. Procedures
3. Source

⁴ <https://help.ubuntu.com/community/How%20to%20Create%20a%20Network%20Share%20Via%20Samba%20Via%20CLI%200%28Command-line%20interface/Linux%20Terminal%29%20-%20Uncomplicated,%20Simple%20and%20Brief%20Way!>

3. Create a directory to be shared

```
mkdir /home/<user_name>/<folder_name>
```

4. Make a safe backup copy of the original smb.conf file to your home folder, in case you make an error

```
sudo cp /etc/samba/smb.conf ~
```

5. Edit the file "/etc/samba/smb.conf"

```
sudo nano /etc/samba/smb.conf
```

Once "smb.conf" has loaded, add this to the very end of the file:

```
[<folder_name>]
path = /home/<user_name>/<folder_name>
valid users = <user_name>
read only = no
```

Tip: There Should be in the spaces between the lines, and note que also there should be a single space both before and after each of the equal signs.

6. Restart the samba:

```
sudo service smbd restart
```

7. Once Samba has restarted, use this command to check your smb.conf for any syntax errors

```
testparm
```

8. To access your network share

```
sudo apt-get install smbclient
# List all shares:
smbclient -l //<HOST_IP_OR_NAME>/<folder_name> -U <user>
# connect:
smbclient //<HOST_IP_OR_NAME>/<folder_name> -U <user>
```

To access your network share use your username (<user_name>) and password through the path "smb://<HOST_IP_OR_NAME>/<folder_name>/" (Linux users) or "\\<HOST_IP_OR_NAME>\<folder_name>\\" (Windows users). Note that "<folder_name>" value is passed in "[<folder_name>]", in other words, the share name you entered in "/etc/samba/smb.conf".

Note: The default user group of samba is "WORKGROUP".

Source

1. http://www.hardcode.nl/archives_147/article_548-samba-quick-setup-on-ubuntu-1004.htm

How to Create a Network Share Via Samba Via CLI (Command-line interface/Linux Terminal) - Uncomplicated, Simple and Brief Way! (last edited 2015-10-06 20:46:34 by r2buntu-d)

The material on this wiki is available under a free license, see [Copyright / License](#) for details
You can contribute to this wiki, see [Wiki Guide](#) for details

- Set up lighttpd for authentication for web ops ⁵
 - 1) Modify /etc/lighttpd/lighttpd.conf file

```
server.modules += ( "mod_auth" )
auth.debug = 2
auth.backend = "plain"
auth.backend.plain.userfile = "/var/www/webopspasswords"

auth.require = ( "/webops/" =>
(
"method" => "basic",
"realm" => "Password protected area",
"require" => "user=webops"
)
)

2) Create "/var/www/webopspasswords"
webops:xia17pxn
This file should be modified to change the password. For sensitive environments, change to encrypted passwords.

3) Create folder /var/www/webops
change owner to www-data (=lighttpd)
in webops, make links to ini, jpg, js, html files from /var/www:
```

⁵ <https://www.cyberciti.biz/tips/lighttpd-setup-a-password-protected-directory-directories.html>

```

root@pixie-net:/var/www/webops# ls
ADC.csv          coincdaq.cgi      psahistprojpage.html
LMdata.txt       d3.v3.min.js      psasurfacepage.html
MCA.csv          defaults.ini       rspage.html
RS.csv           dygraph-combined.js runstats.cgi
Xia_LLC_web_header.jpg findsettings.cgi   settings.ini
acquire.cgi      gettraces.cgi     startdaq.cgi
adcpage.html     mcacpage.html      webopsindex.html
cgistats.cgi     pashistpage.html   webopsindex.html.backup
cgitraces.cgi    plotly-latest.min.js
cgiwaveforms.cgi progfippi.cgi

```

(white: data files created by webops, blue: links, green: webop specific html page)

- Zync temperature

For kernel 2.x (Xillinux), temperature in Celsius can be read from file
 /sys/devices/amba.0/f8007100.ps7-xadc/temp

For kernel 4.x, raw files are /sys/devices/soc0/amba/f8007100.adc/iio:device0,
 which is difficult to open/read from a program because of the colon.

So we made a symbolic link :

```
ln -s /sys/devices/soc0/amba/f8007100.adc/iio\:device0/in_temp0_raw temp0_raw
```

to access it from PixieNetCommon.c and compute T in Celsius.

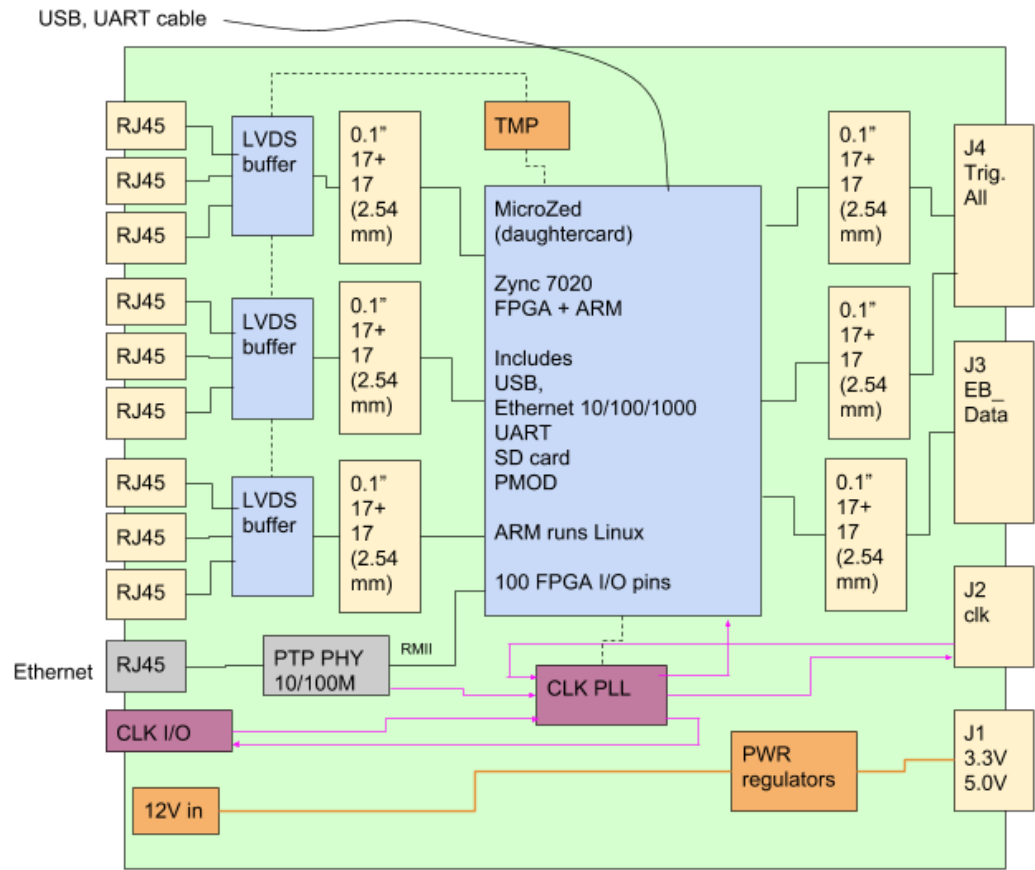
The temperature readings in the second method are typically ~5 degrees higher.

- To automatically give permission to all users to /dev/uis0 (avoiding the initial chmod command):

insert the following line in the file /etc/udev/rules.d/10-local.rules:

```
KERNEL=="uis0", MODE=="666"
```

10 Board Architecture



The P16 MZ TrigIO carrier board is primarily a 6U breakout board for the Pixie-16 custom PXI chassis backplane. It connects ~48 lines from the backplane to the FPGA fabric of a Zynq SOM⁶. Another ~48 lines connect from the panel I/O via LVDS I/O buffers to the SOM. The logic in the Zynq FPGA can be programmed to route signals from front or back to back or front, and to compute results from the signals. User control is via C programs executed on the ARM section of the Zynq, with UART or Ethernet access from a host PC or other device on the network.

10.1 Connections

10.1.1 Backplane Connections

The Pixie-16 backplane uses the “upper” PXI connectors J3, J4, J5 for power and over 200 trigger lines. The P16 MZ TrigIO module connects to

- All 32 “TriggerAll” lines in J4 (bussed to all slots in the PXI backplane)

⁶ Avnet MicroZed

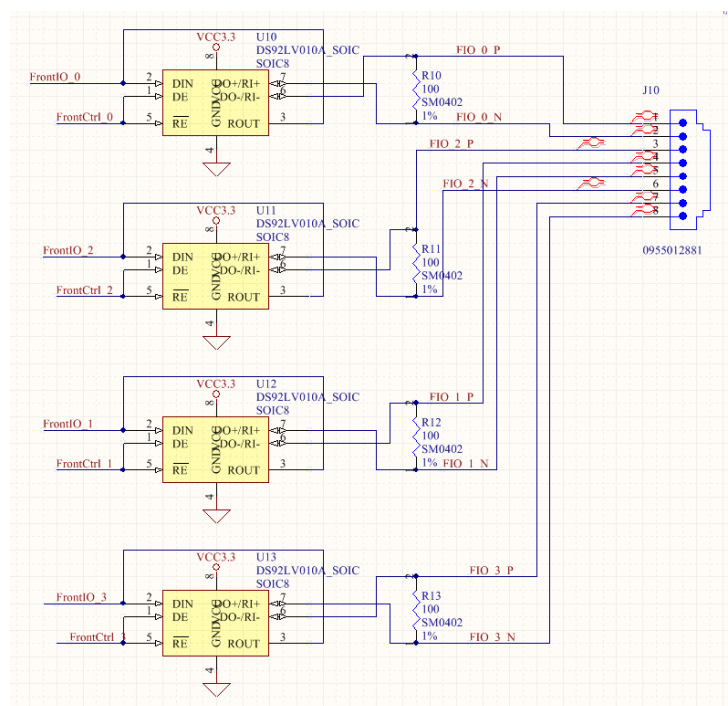
- 16 “EB_Data” lines [31:16] in J3 (bussed to slots in the same PCI backplane segment)
- 2 “Control” lines [5:4] (with pullups, bussed to all slots in the PXI backplane,
- 2 “EB_Ctrl” lines [1:0] (with pullups, bussed to all slots in the PXI backplane,

For Pixie-16 trigger and coincidence signals using these lines, please see the Pixie-16 manuals

The P16 MZ TrigIO module does not connect to the signals in J5, with is not stuffed. It uses only power from J1. The high current pins in J4 are not stuffed. It connects to clock inputs/outputs in XJ4 (J2 in plain PXI). Only XJ4 is to be stuffed, never the full J2 connector. For power and clock, see further below.

All backplane signaling is 3.3V I/O

10.1.2 Front Panel Connections



The P16 MZ TrigIO module has 12 RJ-45 connectors on the front panel for general purpose LVDS I/O with CAT-5 cables. The overall 48 signals in these connections are organized as 3 groups, FrontIO_A/B/C [15:0]. Each LVDS pair is connected to a LVDS/CMOS bidirectional buffer. The direction of the buffer is set by the MicroZed via I2C fanout chips – 6 chips with 8 outputs each that can be individually programmed high/low for in/out.

The pinout of the RJ-45 connector is shown above. The pattern is the same for every block of 4 signals from group X = A, B and C.

Some of the inputs can be used to clock the MicroZed Zynq FPGA (see below)

10.1.3 Zynq SOM Connections

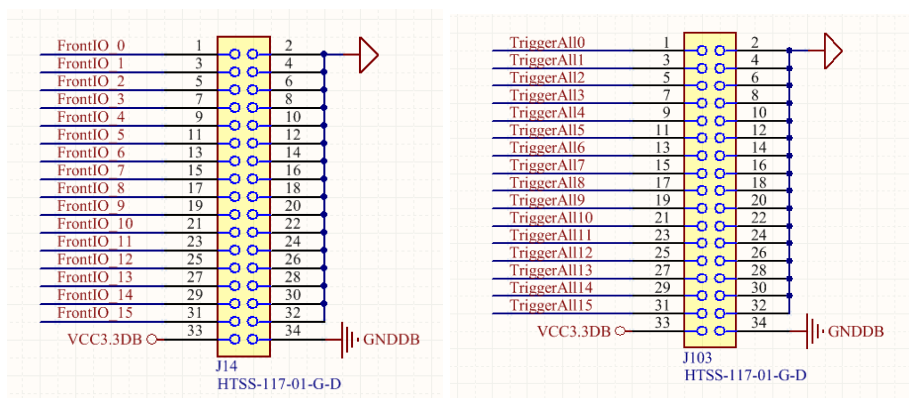
The MicroZed Zynq SOM itself has connections for Ethernet, USB, SD card, PMOD, and USB/UART. These signals are not brought out to the front panel. Instead, cables can be connected and brought out through an empty PXI slot. There is a PROM/thermometer (TMP116) on the P16 MZ TrigIO module that is connected to the same I2C bus as the front panel buffer direction control chips.

The Ethernet connector on the MicroZed itself is not used (but can be with different firmware). Instead, the Zynq RMI interface is connected to a DP83640 Ethernet PHY, which bring the Ethernet connection to the 13th RJ45 connector on the front panel. This PHY implements 10/100M Ethernet with PTP clock synchronization. Two “PTP triggers” to/from the DP83640 are connected to the MicroZed to allow generation of PTP related signals, such as 1 PPS or “go high at <time/date>” and bringing them out on the front panel or backplane.

10.1.4 Custom Connections

The 48 backplane connections and the 48 front panel connections of the MicroZed (in CMOS format) are also connected to six 34pin 0.1” headers. The headers include 16 signal, 16 GNDs, and optional 3.3V power and GND. The headers can either connect to cables to/from external electronics, or hold custom daughtercards with application specific connectors (e.g. har link matching Pixie-16 front panel I/O) and/or application specific buffer circuitry. Note that there are no buffers to drive long cables for these connectors; signals are direct inputs/outputs to the FPGA.

All 0.1” headers signaling is 3.3V I/O



The above figure shows the pinout for these headers. Power and ground are connected to the main 3.3V and GND of the board with jumpers JP504 and J505. Each connector also has a mounting hole near pin 34 for attachment of a daughtercard. Connectors for front and back are the same distance from the board edge.

Todo: list matching cable connectors

10.2 Power

The P16 MZ TrigIO module is either powered from the PXI backplane (5.0V and 3.3V) or from a 12V AC wall adapter. Jumpers JP500/501 and JP502/503 on the board select the power source.

10.3 Clocks

The MicroZed FPGA's most direct clock source is its integrated Zynq clocking. This is sufficient for software/firmware I/O and non-critical connections of backplane signals to front panel signals (either direct assignments or re-clocking with Zynq clocks).

The FPGA fabric can also be clocked from

1. the MMCX front panel I/O
2. the PXI backplane clock (J2/XJ4), nominally 10 MHz but increased to 50 MHz for P16 chassis
3. the PTP or syncE derived clock from the MZ Ethernet connection
4. FrontIO_B4 or FrontIO_C12

One of options 1-3 is selected by Jumpers JP600-602 which define the input to the clock PLL. In option 2-4, the MMCX connector can also be an output. The clock PLL has 3 outputs, one for the FPGA, one for the front panel, and one for the backplane. They can be individually programmed in frequency with the "clockprog" utility, see section 4.3. Further jumpers near the backplane (JP700, JP701) disconnect the backplane clock pins from the PLL output.

10.4 IEEE 1588 Precision Timing Protocol

While traditionally detector data acquisition electronics use dedicated clock and trigger connections to synchronize multiple units, the Pixie-16 MZ-TrigIO has the option to instead use the IEEE 1588 Precision Timing Protocol (PTP) to synchronize modules. Each Pixie-16 MZ-TrigIO operates from its local clock, but clock frequencies and timestamps are adjusted by software that uses special PTP messages exchanged through the data network. With suitable network configurations, this can achieve sub-nanosecond precision.

10.4.1 Overview

The Zynq SoC used as the Pixie-16 MZ-TrigIO's processor has a number of PTP functions built into its standard Ethernet controller. With suitable Linux kernel options and PTP stack software (see below), the Zynq PTP clock can be synchronized to other PTP nodes in the network with a precision of ~1500 ns. However, as there are no useful connections between the Zynq PTP clock controller and FPGA fabric, this is of limited use for the detector pulse processing.

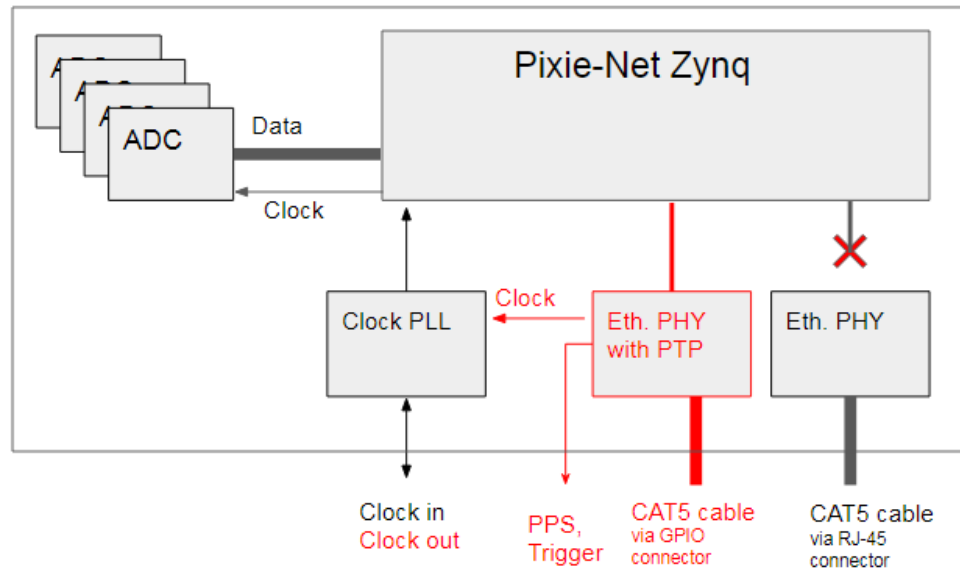


Figure 10-1: Block diagram of the clocking of the Pixie-16 MZ-TrigIO, with PTP features shown in red.

Therefore, the Pixie-16 MZ-TrigIO implements the following alternate PTP functions:

- Redirect Ethernet data to a secondary Ethernet PHY.
- Perform PTP time stamping in the secondary Ethernet PHY.
- Generate a programmable PTP clock that can be used to clock the FPGA processing.
- Generate programmable PTP triggers for internal and external use, e.g. a pulse per second signal or an enable signal that allows data acquisition to begin at a specific time.
- Connect PTP PHY to CAT-5 cable through the 13th front panel RJ45 connector.

These hardware features require a number of (open source) control software for operation, as described in the following sections. Timing resolutions between 300 ns and 200 ps have been achieved with this technique, depending strongly on the network capabilities.

10.4.2 PTP Stack Software: LinuxPTP

The PTP stack software manages the clock adjustments based on PTP messages exchanged between the PTP nodes. The PTP PHY is supported by the open source software LinuxPTP⁷, which is installed in the Pixie-16 MZ-TrigIO's Ubuntu 15 operating system. To work with the PTP PHY, LinuxPTP needs specific kernel drivers and firmware, which is part of the boot files provided for the PTP variant. (The standard boot files will let LinuxPTP communicate only with the built-in Zynq PTP functions).

To start LinuxPTP in the foreground with status messages, type

```
ptp4l -i eth0 -m -q
```

in the Pixie-16 MZ-TrigIO terminal session. If another PTP node is present in the network, the Pixie-16 MZ-TrigIO will try to link with it. The master clock node is determined automatically; for a defined setup we recommend start ptp4l in one Pixie-16 MZ-TrigIO

⁷ <http://linuxptp.sourceforge.net/>

as shown above and wait until it assigns itself as the clock master, then start `ptp4l` in the second/third Pixie-16 MZ-TrigIO with the added `-s` option to force it to clock slave mode.

It is often more convenient to start LinuxPTP as a service, or a background task and the Pixie-16 MZ-TrigIO data acquisition as a foreground task. Please see the LinuxPTP documentation for the various options. As long as LinuxPTP is running, the clock generated by the PTP PHY is synchronized to the network's PTP master. The Linux system clock can be additionally synchronized to the PTP clock with a second LinuxPTP utility, `phx2sys`.

10.4.3 PTP PHY Control Software: `ptp-mii-tool`

The secondary Ethernet PHY with PTP function has a number of registers that specify additional functions, such as the frequency of the generated PTP clock and the timing of GPO triggers. To program these registers, XIA adapted the *mii-tool* Linux utility to specifically communicate with the PTP PHY. The adapted program, *ptp-mii-tool*, is invoked by typing:

- `ptp-mii-tool` print basic status information, including PTP time
- `ptp-mii-tool -v` print more status information
- `ptp-mii-tool -v -v` print even more status information
- `ptp-mii-tool -s` toggle on/off synchronous Ethernet mode (PTP clock derived from upstream Ethernet device)
- `ptp-mii-tool -p` turn on the pulse per second (PPS) signal on the Pixie-16 MZ-TrigIO's PPS output 3 seconds from "now"
- `ptp-mii-tool -t` print current PTP and system time
- `ptp-mii-tool --enable=TON --duration=DUR`
create PTP triggers at PTP time TON and TON+DUR which are used by the Pixie-16 MZ-TrigIO to enable data acquisition starting at TON and ending at TON+DUR. The Pixie-16 MZ-TrigIO time stamp counter is reset to 0 at TON.

The functions provided by *ptp-mii-tool* are useful but optional for the PTP synchronization and data acquisition process, and more functions will be added as necessary.

10.4.4 PTP Clock PLL Control Software: `clockprog`

By default, the Pixie-16 MZ-TrigIO's input clock PLL buffers the PTP clock generated by the PTP PHY (25 MHz) for the FPGA and optionally also for the "CLK" output on the Pixie-16 MZ-TrigIO rear panel. However, this chip is very flexible and can be programmed to divide or multiply the input clock frequencies to generate frequencies other than 25 MHz for the "CLK" output. Such changes can be applied with the *clockprog* utility. A few standard modes are hardcoded in the program; expert users can modify the register settings with the help of the PLL's data sheet (CDCD813-Q1) and TI's ClockPro utility. The program is invoked by typing

```
./clockprog m
```

where `m` stands for one of the following mode numbers:

| Mode | clockprog function |
|----------------|--|
| 0 | Report the register values |
| 1 | Program the current register values into the PLL's EEPROM to make them the new power-up defaults. |
| 2 | Enable the clock output for the FPGA (25 MHz), assuming a 25 MHz input signal from the PTP PHY or MMCX. |
| 3 | Enable the clock output for the FPGA (25 MHz) and the MMCX connector (25 MHz), assuming a 25 MHz input signal from the PTP PHY or MMCX. |
| 4 (default) | Enable the clock output for the FPGA (50 MHz), the backplane (50 MHz), and the MMCX connector (50 MHz), assuming a 25 MHz input signal from the PTP PHY or MMCX. |
| 5 | Enable the clock output for the FPGA (50 MHz), the backplane (50 MHz), and the rear panel connector (50 MHz), assuming a 50 MHz input signal from the backplane or MMCX. |

Note: Using MMCX (or backplane) as output prohibits using it as input.

To use the clock PLL with an external clock source (backplane or MMCX), jumpers JP600-602 are set to use either one as input to the PLL (instead of the PTP PHY). We recommend a 3.3V, 25 or 50 MHz clock signal; otherwise the PLL has to be programmed to work with the alternate input clock frequency.

The functions provided by clockprog are optional for the PTP synchronization and data acquisition process, and more modes will be added as necessary. A possible cause of missing Ethernet connection is a wrong or missing clock from the PLL; in case of problems try execution of

```
./clockprog 4
./ifconfig eth1 up
```

and if the Ethernet connection is successfully restored, execute `./clockprog 1` to write the working PLL configuration to PROM for the next powerup cycle.

10.4.5 Reduced General Purpose Functionality

Due to resource limitations in the Zynq, the following features are not available when the PTP Etherney PHY is used:

- Gigabit Ethernet
The secondary Ethernet PHY operates only at 10/100 Ethernet.

11 Hardware Information

11.1 Jumpers

Pixie-16 MZ-TrigIO boards have the following internal jumpers

- **JP500-JP503**

When a shunt is connecting pin 1 and pin 2 in each jumpers JP501 and JP503, the board's 3.3V and 5.0V power planes are connected to the PXI backplane power.

When a shunt is connecting pin 1 and pin 2 in each jumpers JP500 and JP502, the board's 3.3V and 5.0V power planes are connected to the regulators powered by the 12V DC input.

Do not set all 4 shunts at the same time, or in any mixed configuration!

- **JP504 JP505**

When a shunt is connecting pin 1 and pin 2 in each jumper, pins 33 and 34 on the 0.1" headers are connected to 3.3V and GND, respectively

- **JP600, JP601, J602**

When a shunt is placed on JP600 ("PTP"), the output of the PTP PHY ptpclock signal is connected to the **input** of the clock PLL, which in turn generates clocks for the FPGA, backplane and/or the MMCX.

When a shunt is placed on the on JP601 ("EXT"), the **output** of the clock PLL is connected to the MMCX connector "CLK".

When a shunt is placed on the on JP602 ("BP"), the backplane pin "PXI_CLK" on J2 is connected to the **input** of the clock PLL, which in turn generates clocks for the FPGA, backplane and/or the MMCX.

When a shunt connects the neighboring pins of JP600 and JP601, the MMCX connector is connected to the **input** of the clock PLL, which in turn generates a clock for the FPGA and backplane

- **JP700, JP701**

When a shunt is connecting pin 1 and pin 2 in jumper JP700, the PLL clock output is connected to pin "PXI_CLKIN" on J2. This is only allowed if the module is placed in slot 2, and will drive the chassis' clock distribution network with the clock from the Pixie-16 MZ TrigIO.

When a shunt is connecting pin 1 and pin 2 in jumper JP701, the PLL clock output is connected to pin "CLK_RIGHT" on J3.

12 Information for Programmers

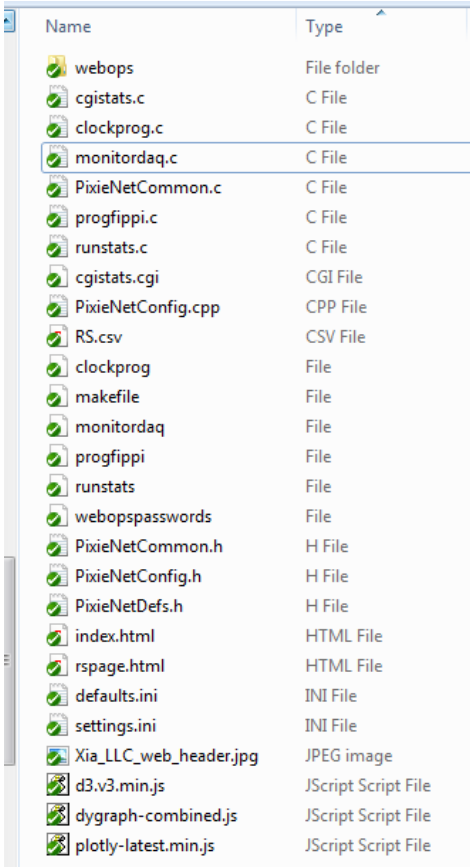
12.1 Software Information

The Pixie-16 MZ TrigIO software consists of a number of C routines located in /var/www. They include the API functions listed in section 4, plus a number of support code files for reading parameters from the settings file or utilities for I2C I/O. All code I provided as open source; users are free to modify and expand the functions. XIA appreciates feedback and new ideas.

Experienced programmers will find it relatively straightforward to understand the code (we hope). Below we point out a number of concepts for general guidance

12.1.1 File structure

1. C and H files
Besides the C programs listed in section 4, the files “PixieNetCommon.c/h” include I2C subroutines, functions to read temperature, and a function to write status registers to file or standard output (for webpages). The files “PixieNetConfig.cpp/h” include the definitions of the parameters in the settings file. PixieNetDefs.h defines global constants.
2. HTML, JS and JPG files
These files are the webpages, or contain the scripts and pictures for the webpages (tables and plots).
3. webopspasswords
is a test file containing the password for the operation through the webpages, not currently implements
4. makefile
defines the compile options for the C programs. Type make to compile.
5. RS.csv
output of the status registers
6. CGI files
Executable for webpage
7. [no extension]
Executables for command line



| Name | Type |
|------------------------|---------------------|
| webops | File folder |
| cgistats.c | C File |
| clockprog.c | C File |
| monitordaq.c | C File |
| PixieNetCommon.c | C File |
| progfippi.c | C File |
| runstats.c | C File |
| cgistats.cgi | CGI File |
| PixieNetConfig.cpp | CPP File |
| RS.csv | CSV File |
| clockprog | File |
| makefile | File |
| monitordaq | File |
| progfippi | File |
| runstats | File |
| webopspasswords | File |
| PixieNetCommon.h | H File |
| PixieNetConfig.h | H File |
| PixieNetDefs.h | H File |
| index.html | HTML File |
| rspage.html | HTML File |
| defaults.ini | INI File |
| settings.ini | INI File |
| Xia_LLC_web_header.jpg | JPEG image |
| d3.v3.min.js | JScript Script File |
| dygraph-combined.js | JScript Script File |
| plotly-latest.min.js | JScript Script File |

12.1.2 Parameter Definitions

In “PixieNetConfig.h” we define the struct for the parameters in the ini file. The struct element is named the same as the line in the ini file. “PixieNetConfig.cpp” contains functions to find a line with a certain name, read in the values as bits, integer or floats (possibly arrays if multiple values are in the line) and assign the values to the struct element. A higher level function such as progfippi should thus call the routine “init_PixieNetFippiConfig_from_file” from PixieNetConfig.cpp to load all parameter values and then use those relevant for the program by name.

12.1.3 FPGA registers

The FPGA registers are mapped through the xillybus lite firmware core as a kind of “file” in the Linux OS. I/O with the registers thus consists of opening that “file” and reading from / writing to an address in the file. Up to 4096 registers are supported by the core; the current firmware implements 512 of them and uses only a few.

12.1.4 Status Registers – cgi vs file vs page

The status registers can be read and reported in different ways:

1. Read registers and write to file
By executing ./runstats, the status registers are captured at this point in time and written to file RS.csv.
Similarly, ./monitordaq periodically reads the status registers and writes them to file.
2. Show RS.csv on webpage
The webpage rspage.html displays the contents of the file RS.csv as a table. The information is from the last time the file was written
3. Execute cgistats on webpage
Calling a cgi function in the web browser executes the function. This function reads the status register and “prints” a webpage with *current* information.

Essentially method (3) is a combination of (1) and (2), executed through the webpage. However, it should not be performed when another task (e.g. monitordaq) is reading FPGA registers, as it may lead to contention of which program has access.

12.1.5 Other Notes

- For simple desktop programming and debugging, the MicroZed can be removed from the Pixie-16 MZTrigIO board and operated as a standalone unit, powered through the USB-USAT port.

12.2 Firmware Information

Please see separate document in the firmware distribution