

Synchrodyne Series - Wideband 400 MHz to 1500 MHz Receiver with AGC, IRIG B Synchronization, Gen 3 PCIe Bus and Kintex Ultrascale FPGA

Product Specification – October 1, 2020

P/N AD12-2000-IRIG



Ultraview Corporation
808 Gilman Street, Berkeley, CA 94710
PH:(925) 253-2960
Fax (925) 253-4894
www.ultraviewcorp.com
Copyright © 2020Ultraview Corporation

Table of Contents

| | |
|---|-----------|
| 1. WARRANTY..... | 3 |
| 2. SYNCHRODYNE RECEIVER SERIES DESCRIPTION..... | 4 |
| 2.1 OVERVIEW OF MODEL AD12-2000-IRIG 12-BIT 4GSPS A/D WITH AGC FRONT END AND IRIG B SYNCHRONIZATION AND TIME CODE REPORTING SYSTEM..... | 5 |
| 3. SPECIFICATIONS - AD12-2000-IRIG..... | 6 |
| 4. AD12-2000-IRIG RECEIVER PHYSICAL CONFIGURATION..... | 7 |
| 4.1 AD12-2000-IRIG INPUT CABLE DESCRIPTION..... | 8 |
| 4.2 IRIG-B BULKHEAD RACK PANEL ASSEMBLY INPUT CABLE DESCRIPTION..... | 9 |
| 4.3 AD12-2000-IRIG LED DESCRIPTION..... | 10 |
| 5. HARDWARE INSTALLATION AND SETUP..... | 11 |
| 6. SOFTWARE INSTALLATION AND SETUP..... | 12 |
| 6.1 SOFTWARE INSTALLATION FOR WINDOWS 10..... | 12 |
| 6.2 SOFTWARE INSTALLATION FOR LINUX (64-BIT)..... | 13 |
| 6.3 ADVANCED SOFTWARE SETUP..... | 13 |
| 7. RUNNING THE EXAMPLE PROGRAMS..... | 16 |
| 7.1 PREPARING TO RUN THE EXAMPLE PROGRAMS..... | 16 |
| 7.2 LABVIEW™ THE GRAPHICAL WAVEFORM VIEWER UNDER WINDOWS 7/10..... | 17 |
| 7.3 THE REAL-TIME GRAPHICAL RX WAVEFORM / SPECTRAL VIEWER UNDER LINUX..... | 20 |
| 7.4 THE REAL-TIME PIPES BASED ACQUISITION SOFTWARE..... | 21 |
| 7.5 CROSS PLATFORM COMMAND LINE ACQUISITION & SYNTHESIS..... | 27 |
| 7.5.1 <i>acquire - acquire data into on-board DRAM, and then store the buffer to disk</i> | 28 |
| 7.6 CROSS PLATFORM COMMAND LINE OPTIONS..... | 31 |
| 8. MICROSYNTH PROGRAMMABLE INTERNAL CLOCK..... | 33 |
| 8.1 TTL INPUT/OUTPUT LINES FOR CUSTOM FIRMWARE (PRELIMINARY)..... | 33 |
| 9. APPENDIX A – FIRMWARE UPDATE USING PROGRAMMING CABLE..... | 35 |
| 10. APPENDIX B – ADC GAIN/OFFSET/BIAS/MICROSYNTH CALIBRATION..... | 36 |
| 11. APPENDIX C – HOW TO MODIFY YOUR LABVIEW™ PROJECT..... | 38 |
| 18.1 – MODIFYING VIs..... | 38 |
| 12. APPENDIX D – DATA OUTPUT FORMAT (UVDMA.DAT)..... | 39 |
| 13. APPENDIX E. CERTIFICATE OF VOLATILITY..... | 40 |
| 14. APPENDIX F. KNOWN ISSUES..... | 41 |
| 15. APPENDIX G – EXAMPLE BUILD INSTRUCTIONS FOR DELL POWEREDGETM R940 SERIES SERVER..... | 42 |

1. Warranty

Ultraview Corporation hardware, software and firmware products are warranted against defects in materials and workmanship for a period of two (2) years from the date of shipment of the product. During the warranty period Ultraview Corporation shall at its option, either repair or replace hardware, software or firmware products which prove to be defective. Ultraview products are only supported with Ultraview provided firmware and software, any modifications made by customers are not supported and are not covered under warranty. This limited warranty does not cover damage caused by misuse or abuse by customer, and specifically excludes damage caused by dropping the unit or by the application of excessive voltages to the inputs and/or outputs of data acquisition boards. Due to the complex nature of computer systems, Ultraview boards operation should be verified in the desired host computer system prior to purchasing multiple units of host system. In some systems the PCIe bus DMA throughput is insufficient to allow the full DMA transfer rate of the board. In some systems, the reset time is too short to allow certain Ultraview FPGA-based boards to fully configure before being accessed by the system, therefore requiring a warm boot before operation is possible in these systems.

While Ultraview Corporation hardware, software and firmware products are designed to function in a reliable manner, Ultraview Corporation does not warrant that the operation of the hardware, software or firmware will be uninterrupted or error free. Ultraview products are not intended for use as critical components in life support systems, aircraft, military systems or other systems whose failure to perform can reasonably be expected to cause significant injury to humans. Ultraview expressly disclaims liability for loss of profits and other consequential damages caused by the failure of any product, and recommends that customer purchase spare units for applications in which the failure of any product would cause interruption of work or loss of profits, such as industrial, shipboard or military equipment. In no way will Ultraview Corporation's liability exceed the amount paid by the customer for the product.

This limited warranty is in lieu of all other warranties expressed or implied. The warranties provided herein are buyer's sole remedies. In no event shall Ultraview Corporation be liable for direct, special, indirect, incidental or consequential damages suffered or incurred as a result of the use of, or inability to use these products. This limitation of liability remains in force even if Ultraview Corporation is informed of the possibility of such damages.

Some states do not allow the exclusion or limitation of incidental or consequential damages, so the above limitation and exclusion may not apply to you. This warranty gives you specific legal rights, and you may also have other rights which vary from state to state.

WARNING! To avoid overheating, all Ultraview boards must be installed in a well-cooled workstation or server chassis, or alternatively in an industrial chassis PC. Installation in a PC or workstation without fans at the front end of the card cage may cause the board to overheat, and resulting damage is not covered by warranty.

2. Synchrodyne Receiver Series Description

Ultraview's PCIe Synchrodyne series of wideband receivers are complete high-speed data acquisition systems which can acquire RF signals or other analog data either independently or in synchronization with other systems located anywhere in the world, using IRIG-B satellite synchronization signals and time codes. These receivers are contained on a single full-size PCI-express (PCIe) Gen 3 card. Designed for continuous operation in military, scientific, medical and industrial applications these boards function in PCIe systems having at least one available Gen3 x16 or Gen3x8 PCIe open-end slot. Drivers and ready-to-use user programs for Linux (Centos 7.6 or Redhat 7) and 64-bit Windows 10 are provided. Windows 10 64-bit drivers are fully signed.

The analog section of the receiver is comprised of the following stages:

- 1) A 5dB 50 ohm internal attenuator/pad immediately after the RF input jack
- 2) A 17dB LNA
- 3) A 3dB attenuator/pad and frequency equalizer
- 4) A 0-30dB digital step attenuator that can be set using a software command or the board's automatic gain control (AGC)
- 5) A 3dB 50 ohm attenuator/pad and frequency equalizer.
- 6) A 20dB gain RF amplifier
- 7) A second 0-30dB software / AGC controllable digital step attenuator
- 8) An RLC frequency equalization network
- 9) A 17dB gain RF amplifier
- 10) A 3dB 50 ohm attenuator/pad
- 11) A broadband transformer balun
- 12) A 12-bit 4000MSPS analog-to-digital converter (presently operated at 3000MSPS)

The digital section of these boards includes a Xilinx Kintex 7™ FPGA, which queues up the A/D data in its FIFO registers, and outputs bursts of 128-bit wide LVDS data vectors at up to 500 MWPS (8GB/sec) that are transferred into an on-board 8GB DDR4 DIMM module. Between these forward bursts of A/D data, the FPGA can burst read data to the PCIe bus interface, allowing for uninterrupted high speed data acquisition. These devices require x16 physical or x8-open-end PCIe slots, but are internally wired for x8 lane operation. Sustained transfer rate to the host is up to 7.0 GB/s, and is determined by the speed of this 8 lane Gen 3 PCIe interface, which is dependent on the host system motherboard, the operating system and applications running.

The FPGA on all boards is an 1156 pin Xilinx Kintex 7™ Ultrascale. Most of the 1156 pin Kintex7 Ultrascale™ FPGA's may be specified by the customer at the time the board is ordered, these include the XCKU040, and XCKU060. **The default FPGA is an XCKU040™**, whose registers and BRAMS are approximately 30%/50% utilized for the AD12-2000-IRIG when the board is shipped with the default factory firmware. With the purchase of any Ultraview data acquisition system, and under the protection of a signed Non-Disclosure Agreement, Ultraview Corporation will provide the VHDL firmware source code to allow its OEM users the option of developing and loading their own firmware into the on-board FPGA.

All boards are full-size PCIe boards (4.2" x 8.625"). **Due to the excess height of the memory module on the board each board occupies the space of two slots.**

To avoid overheating, boards must be installed in a well-cooled workstation or a server chassis. Installation in a standard PC chassis is feasible as long as a minimum of 100 linear feet per minute of airflow is provided to cool the board. Further, the filler bracket for the slot in front of the Ultraview board (the slot into which the Ultraview's DIMM protrudes) must be removed or replaced with a vented filler bracket to allow additional airflow to the board.

2.1 Overview of Model AD12-2000-IRIG 12-Bit 4GSPS A/D with AGC Front End and IRIG B Synchronization and Time Code Reporting System

Models in the AD12-2000-IRIG series contain a 300MSPS - 4GSPS 12-bit A/D converter, 8GB of on-board DDR4 DRAM memory and the ability to transfer data, via the host system's PCIe bus, directly into the computer system's memory up to approximately 7 GB/s. A/D sampling may either be controlled by an external clock input between 300 MHz and 1.5 GHz (the A/D sampling rate is equal to twice the clock frequency) or the on-board programmable internal clock synthesizer. This internal synthesizer/timebase may be slaved to an externally supplied 10MHz reference clock, which can be provided by an IRIG receiver or other source. Alternatively, a 50MHz internal TCXO-based reference may be software selected as reference clock for the synthesizer. Multiple boards may be configured to acquire either concurrently, for more simultaneous acquisition channels, or sequentially, for longer record length. A 26-pin connector and TTL Input Panel will be provided which the user can drive from an SMA cable. TTL triggering is available on the GC_P pin of this panel and may be invoked by specifying TTL trigger using the software interface. Selective recording is optionally available using the GC_P by specifying "no trigger". See Appendix for additional details.

External triggering of the start of data acquisition may also be made using the IRIG-B bulkhead panel, which can accept a differential negative ECL (NECL) 1PPS IRIG B time synchronization signal. The board is also capable of appending IRIG-B time code information, supplied by the external IRIG-B receiver, into each packet of digitized data.

3. Specifications - AD12-2000-IRIG

| | |
|---|--|
| A/D Converter Resolution: | 12 Bits |
| Number of Channels: | 1 at up to 3.2GSPS (3.0GSPS max in most host systems due to PCIe bus limitations). |
| Wideband Signal-to-Noise Ratio: Noise figure: | typically 90 dB (not including spurs) 10dB max, 9dB typ, meas. at 422 and 922MHz |
| RF Input Range (DO NOT EXCEED +24dBm): | +11dBm 1dB compression point -6dBm with nominal attenuator settings -50dBm typ. with minimum attenuator settings |
| Analog Input Impedance: | 50 ohms nominal, Return loss >20dB (18dB when used with Ultraview Filter Bulkhead assy) |
| Analog Input Bandwidth | 300MHz to 1500MHz (2dB p-p variation) 1.5dB p-p variation from 400-1200MHz, 1.5dB p-p variation from 900-1500MHz |
| Sampling Rate into on-board RAM: | 3.2 GSPS Max, 600 MSPS Min. |
| 10MHz External Clock Reference Input | 10MHz 0dBm to +18dBm Sine or Square Wave |
| Internal Reference Oscillator | 50MHz TCXO |
| Alternative External Clock Input (AC Coupled) | |
| Frequency | 300MHz–3.2GHz |
| Input Impedance: | 50 ohms in series with 0.01uF |
| AC Voltage Minimum: | 0.7V min. – 3.0V Max Peak-to-Peak |
| 1PPS IRIG B Trigger Input (Negative ECL) : | Positive-going edge - rise time must be <2ns |
| (Input via two SMA jacks, differential signal must be time matched to 250ps or less) | |
| IRIG-B PPS synchronization error | +/-60ps nominal (+/-2.5ns max) |
| DMA Transfer Rate: | 6-7 GB/s max (host system dependent) |
| Sample rate for continuous stream to host: | up to 3GSPS (3.2GSPS typical) |
| Operating Temperature Range: | 0 to +50 Degrees Celsius |
| Storage Temperature Range: | -25 to +85 Degrees Celsius |
| Power Requirements (board occupies 2 slots): | +3.3V +/-5% at 3.8A Max (2.9 A typical) +12V +/-5% at 4.0A Max (2.6 A typical) |
| <p>Note: Current consumption is dependent on firmware installed on board, which may be updated from time to time, but is never expected to exceed the following levels: 4.0A from +3.3V supply, and 5.0A from +12V PCIe supplies. Current consumption verified with Ultraview PCIeEXT-16Hot Live Insertion Bus Extender at time of final production test.</p> | |

4. AD12-2000-IRIG Receiver Physical Configuration

The AD12-2000-IRIG receiver consists of a PCIe data acquisition board with variable gain / AGC front end, and an optional 19" rack-mountable bulkhead panel assembly containing a DC-1500MHz lowpass filter and input connectors that accept the RF input and IRIG-B receiver signals. Three cables connect the inside face (rear) of the bulkhead panel to the PCIe data acquisition board that is mounted in the host computer system.

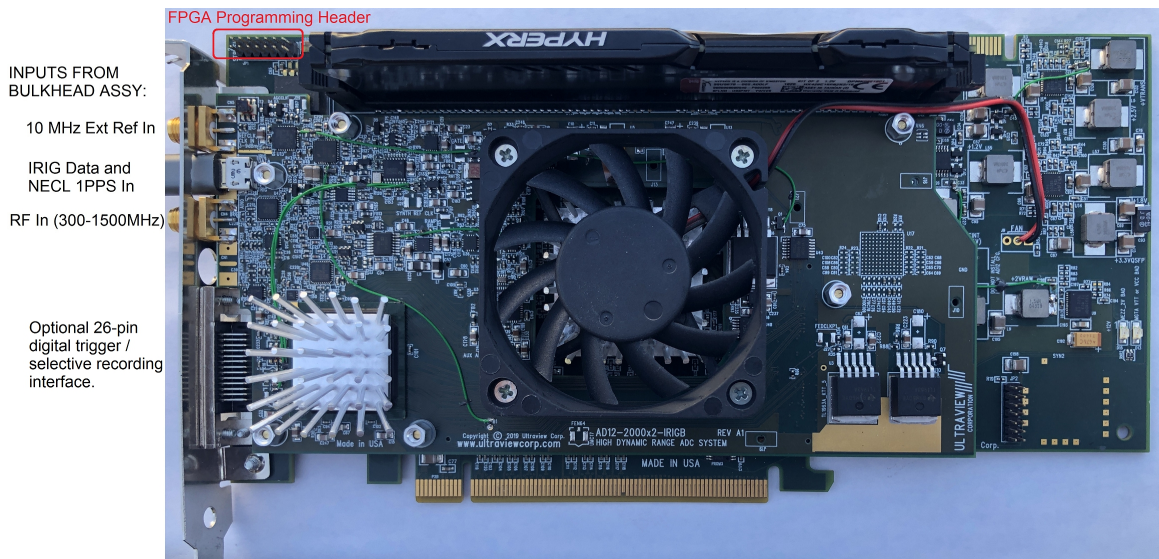


Figure 1. Input Connectors on AD12-2000-IRIG PCIe receiver board.



Figure 2. Input Connectors on IRIG-B Bulkhead rack panel assembly

4.1 AD12-2000-IRIG Input Cable Description

Figure 1 shows the input connections to the AD12-2000-IRIG A/D board. The function of these inputs is as follows:

10 MHz Ext Ref In. This input jack may be fed from either a 10MHz external reference (if selected via software), such as the 10MHz reference output from the IRIG-B Bulkhead rack panel assembly in Figure 2, or by a 300MHz to 1500MHz external clock. These signals must be stable in frequency and must be continuously supplied before any user programs are run. This external reference/clock is useful if acquisition is to be synchronized-to/driven-by an external source. For applications in which an external reference or an external clock are not supplied, the board's internal reference and timebase can be used, the software must then select internal clock mode.

IRIG Data and NECL 1PPS In. This MicroUSB jack is fed via the supplied MicroUSB cable from the IRIG-B bulkhead rack panel assembly shown in Figure 2. The 1PPS signal must be negative ECL (nominally -1.3V, with a 800mV p-p swing on each differential pin). The IRIG data input must be TTL.

RF IN This SMA input connector accepts an RF signal in the range specified in the specifications section. Under no circumstances should the signal supplied to the analog input exceed +24dBm, as damage may occur that is not covered by the warranty. For most applications, this input is fed from the filter output on the IRIG-B Bulkhead rack panel assembly

The following optional signals may be fed into the AD12-2000-IRIG board's optional 26-pin digital trigger/selective recording interface, sold separately.

Acquire Disable (Trigger) This is the optional trigger or selective recording TTL input for the AD12-2000-IRIG. The board will store data to the on-board RAM only when the acquire disable input is LOW (board has an internal pull-down resistor, leave unconnected if unused). The timing in this mode is not precise to a single sample and can vary from 0 to 3 samples if the input is not synchronized with the A/D clock.

GC_P on TTL Board (section 16) (AD12-2000-IRIG) This is the selective recording TTL input for the AD12-2000-IRIG. The board will store data to the on-board RAM only when the acquire disable input is LOW (board has an internal pull-down resistor, leave unconnected if unused). The timing in this mode is not precise to a single sample and can vary from 0 to 3 samples if the input is not synchronized with the A/D clock. Selecting TTL edge trigger will use this input as a trigger.

4.2 IRIG-B Bulkhead rack panel assembly input Cable Description

Figure 2 shows the input connections to the system, which are made by connecting cables from the external signal sources (Main RF signal, IRIG B signals from receiver) to the IRIG-B Bulkhead rack panel assembly.

RF INPUT This SMA analog input connector accepts an RF signal in the range specified in the specifications section, under no circumstances should the signal supplied to the analog input exceed +24dBm as damage may occur that is not covered by the warranty. The bulkhead rack panel assembly contains a DC-1500MHz low-pass filter, which is fed from his input jack, and whose output is then fed to the RF In jack on the AD12-2000-IRIG board.

1PPS+. 1PPS-. These jacks must be is fed from differential negative ECL outputs from an IRIG Receiver. These 1PPS signals are nominally -1.3V, with a 800mV p-p swing on each differential pin. Differential skew between these inputs must be kept below 250ps (+/-125ps).

DATA. This IRIG B data jack is fed from the TTL IRIG data output from the IRIG receiver.

10 MHz Ref. This input jack may be fed from a 10MHz external such as the 10MHz reference output from the IRIG-B Receiver. This signals must be stable in frequency and must be continuously supplied before any user programs are run.

Figure 3 shows the cable connections between the IRIG-B Bulkhead rack panel assembly and the AD12-2000-IRIG A/D board.

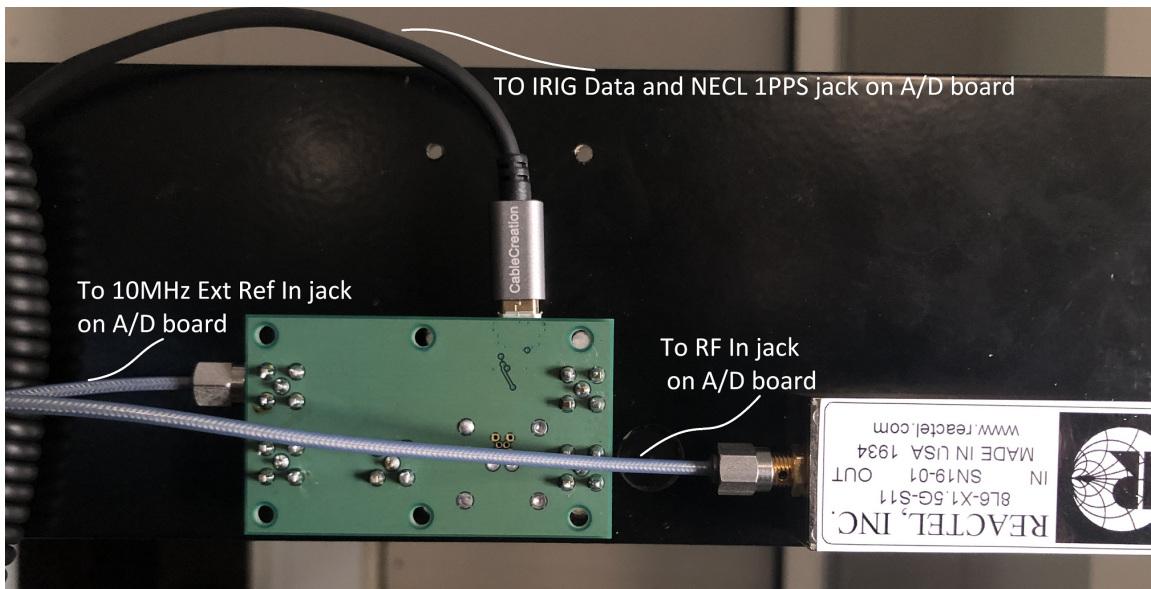


Figure 3. Cable connections from IRIG-B Bulkhead rack panel assembly and AD12-2000-IRIG PCIe receiver board.

4.3 AD12-2000-IRIG LED Description

LEDs on the AD12-2000-IRIG A/D board can be useful during system installation and initial checkout, to illustrate operating status of the AD12-2000-IRIG A/D board.

NO CLOCK Indicates the clock is operating correctly.

ENABLE ACQ Indicates that the board is armed to acquire data if provided that it receives the specified clock or trigger.

DMA Indicates that the board sees an input clock when this LED is flashing, the rate of flashing is proportional to the clock source frequency.

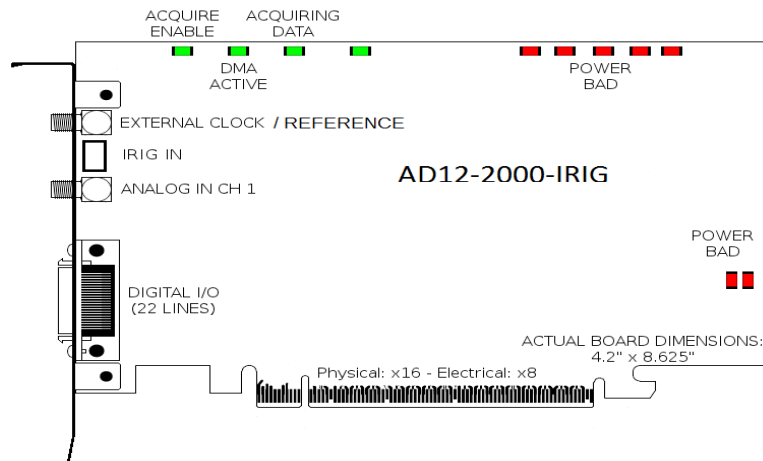
COREPWR BAD Indicates the 0.95V DC power to the FPGA is out of spec. This may occur if a fault develops in the board's power supply, or if the slot in which it is installed is incapable of supplying sufficient +3.3V PCIe current to power the board.

1.9/2.5/+VTRN BAD Indicates the specified internal board power supply voltage is out of spec. This may occur if a fault develops in the board's power supply, or if the slot in which it is installed is incapable of supplying sufficient +12V PCIe current to power the board.

1.2/1.8/3,3QSFP BAD Indicates the specified internal board power supply voltage is out of spec. This may occur if a fault develops in the board's power supply, or if the slot in which it is installed is incapable of supplying sufficient +12V PCIe current to power the board.

Mezz_2V_bad Indicates that the +2V power to the A/D converter is bad. This may occur if a fault develops in the board's power supply, or if the slot in which it is installed is incapable of supplying sufficient +12V PCIe current to power the board. This should briefly light up and then disappear when the power is powered on.

MGTA VTT or VCC BAD Indicates the specified internal board power supply voltage is out of spec. This may occur if a fault develops in the board's power supply, or if the slot in which it is installed is incapable of supplying sufficient +12V PCIe current to power the board.



5. Hardware Installation and Setup

16GB of system memory is required (with >8GB free) to properly run 8GB acquisitions. 32GB is recommended. Handle the board carefully – mechanical damage to the board is not covered by warranty. **To avoid overheating, the board must be installed in a well-cooled workstation, server or industrial chassis PC.** Installation in a PC or workstation without **fans at the front end of the card cage will cause the board to overheat, and resulting damage is not covered by warranty.** **If after 5 minutes of operation, any component on the board feels too hot to comfortably touch, a system with better cooling is required.**

1. Use the shutdown command, turn off the system power, and disconnect the power cord.

BEFORE REMOVING THE COMPUTER SYSTEM COVER OR REMOVING ANY BOARD, BE SURE THAT THE POWER TO THE COMPUTER, AS WELL AS TO ALL PERIPHERAL DEVICES IS OFF. WEAR A STATIC-DISSIPATING WRISTBAND WHICH IS GROUNDED TO THE SYSTEM CHASSIS WHILE OPENING OR WORKING ON YOUR SYSTEM.

2. Remove any screws that attach the computer system cover and remove the cover.
3. Remove the filler bracket from the PCIe slots the board will occupy **and the filler bracket from the PCIe slot below the board, to allow adequate air flow across the board's heatsink.**
4. Hold the board by the top of the metal bracket and the back of the board (**Do NOT EVER hold or exert any force on the DIMM memory module**). Carefully slide the board in so its PCIe connector mates with the motherboard PCIe connector. **Do not force the board. If there is any resistance, rock the board slightly when inserting it.** Be sure the board is seated firmly into the motherboard PCIe connector. Be sure no other PCIe/PCI boards have become unseated.
5. Plug coaxial I/O cables for the analog inputs and/or outputs into the appropriate SMA and microUSB connectors on the board. Connect the free ends of the analog input cable to the signal sources to be digitized, and connect the clock input cable to a suitable clock source, if using an external clock.
6. Replace the computer system cover. Reconnect power cord to the system. Power up and reboot the system. The system will then be ready for software installation.

In some systems the configuration time of the on-board FPGA may be long enough that the FPGA is not fully configured before the system begins accessing the board. If the board is not recognized by the system (e.g. not present in the Device Manager (Windows) or not shown using lspci (Linux)), or the system hangs when accessing the board, or other errors occur, then this may be the problem. If any of these issues are seen the system will need to be "warm restarted" before the board can be used. The easiest way to do this is to boot the system to the OS boot menu, then restart the system without powering it off, then let the system start normally. This allows the FPGA sufficient time to fully configure (a result of powering on) before it is assessed. This can be easily done by adding a "reboot" option to the GRUB boot loader in Linux, hitting this reboot option after the first cold boot, and then allowing the system to continue to reboot again with its normal default boot option. Another way to do this is issue a ctrl-alt-delete during the bios post screen.

6. Software Installation and Setup

“Newer boards may have software and/or FPGA firmware that is different from earlier firmware versions shipped with prior boards of this model. To ensure correct operation of your new board, **please download the current software package from Ultraview’s website. Do not install older versions of drivers or user software that you may have used with previously purchased boards. You may also need to recompile custom user software you may have written**, so it will correctly run with the current driver and board firmware.”

For users who modified a previously LabVIEW™ release and wish to update their projects to work with the latest software, see “APPENDEX – How to Update Your LabVIEW™ Project”.

6.1 Software Installation for Windows 10.

The Windows software release can be downloaded from the following URL:

<http://www.ultraviewcorp.com/downloads>

The Microsoft Visual Studio C++ (64 bit) runtime can be downloaded from the following URL:

<http://www.microsoft.com/en-us/download/confirmation.aspx?id=30679>

To install the software, place the Ultraview Windows release into a directory on your local drive. Run the Microsoft executable to install the runtime. Please read the README file provided in the root of the release for an understanding of the files provided.

To run the LabVIEW executables, the LabVIEW Run-Time Engine 2014 (64-bit) must be installed. If it is not installed, when running the LabVIEW programs a pop-up will direct you to the appropriate download location.

To install the driver, navigate to the device manager, and find the device “pci serial port” in unknown devices. Right click and select update driver. Select browse for driver software on your computer and browse to the driver/win64 directory of the Ultraview installation files. Click next, and when it asks if you'd like to install the device software with name: “Xilinx DMA”, Publisher: “Ultraview Corporation”, select install. When successful the device will show up under category “Xilinx Drivers”, device “Xilinx DMA”.

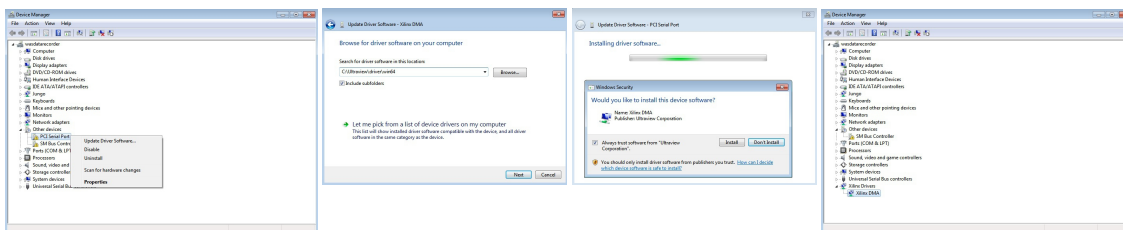


Figure 4.

To avoid data overruns, interruptions in data acquisition, and hanging of user programs, **turn off all system power management options, screen savers, etc. The system must not be allowed to go into sleep mode when the board is running.** To run the example programs for Windows 7 or 10, go to the section “Running the Example Programs under Windows 7, 10”.

6.2 Software Installation for Linux (64-bit)

Centos/RedHat Linux 7.6 are the only officially supported Linux operating system versions at the time of this writing.

To avoid data overruns, interruptions in acquisition, and hanging of user programs, **turn off all system power management options, screen savers, etc. The system must not be allowed to go into sleep mode or any reduced power modes when the board is running.**

The standard and pipes based C++ Linux software release can be downloaded from the following URL:

<http://www.ultraviewcorp.com/downloads>

For the standard release (ad12_14_16), after extracting the archive, navigate to the “driver” directory and extract Linux_driver.zip. Then navigate to the extracted driver directory and run the “make” command. You can run “make install” to permanently load the driver on startup, or you can run “load_driver” script from the tests directory. Now navigate to the _Dllsource directory, and run the “make” command. Now navigate to the src directory and run the “make” command. All of the basic programs are now usable.

In the standard release there is a folder named “Digosc7”. This is a digital oscilloscope program built with Qt that can be used to display output data from the acquire program. In order to build the program Qt (5.2 or greater) must be installed. Then open the .pro file with Qt Creator and build the project.

For the pipes based Linux software release (acq_server), after extracting the archive, navigate to the “driver” directory and extract Linux_driver.zip. Then navigate to the extracted driver directory and run the “make” command. You can run “make install” to permanently load the driver on startup, or you can run “load_driver” script from the tests directory. Instructions for how to modify and run this user program is on the next page.

6.3 Advanced software setup

The source code for both the command line Linux example programs and the command line Windows example programs share an identical section of cross platform source code; this common section of code, located in directories “AppSource” and “_DllSource” in the release, and can be compiled under Linux or Windows.

There is also a supplied multi-threaded high performance Qt based project in the “Qt” directory, however custom application support for this project is not supported without support contract due to the complexity. There is both a Qt based GUI and a Qt command line project, both located in the “Qt” directory.

For Windows:

For users who modified a previously LabVIEW™ release and wish to update their projects to work with the latest software, see “APPENDIX – How to Update Your LabVIEW™ Project”.

The DLL was compiled using Microsoft Visual Studio 2012 Professional and the user acquire programs were compiled using Microsoft Visual Studio 2017 Community.

The LabVIEW project was developed with LabVIEW 2014 64-bit Professional Development System.

To recompile the DLL:

- 1) Open "src/AcqSynth_dll/**AcqSynth.sln**" in MSVS 2012 professional.
- 2) In the "Build" tab, select "Rebuild Solution"

To recompile the Acquire program:

- 1) Open "src/command_line_utilities/Acquire/**acquire.sln**" in MSVS 2017 community.
- 2) In the "Build" tab, select "Rebuild Solution"

Do not open any ".vcproj" as they are not updated and point to old directories. The quickest way to undo is to unzip a new folder and open the ".sln" files.

The resulting executables are built into the "utils" and the "command_line_utilities" directories in the Windows release.

Other versions of Microsoft Visual studio will work as well, but may require some additional steps.

For Linux:

To ensure correct operation of the device driver with your kernel version you are required to recompile the driver module. On Centos 7, rebuilding the driver, on the actual host machine in which it is to be run, is required for proper operation. Ensure that the gcc package is installed by running "yum install gcc".

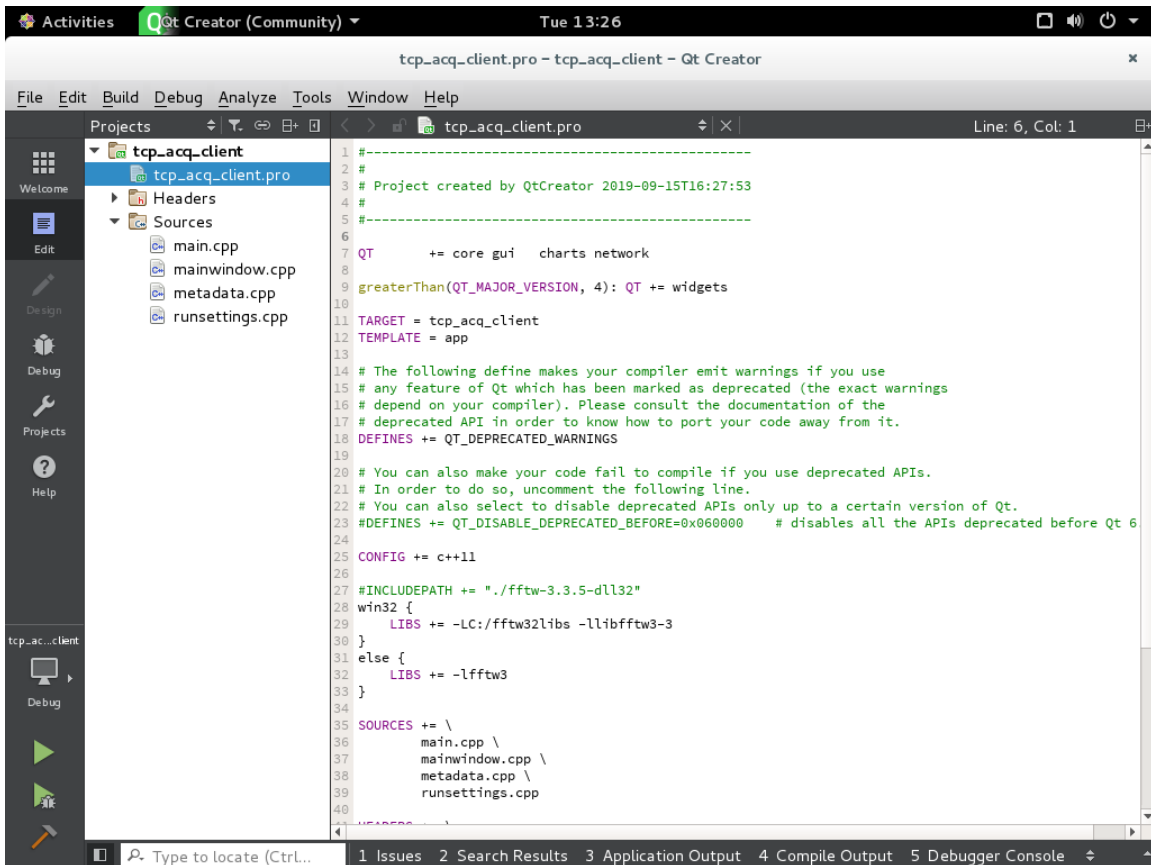
To run the Qt code, depending your version of linux, you may need to download an updated Qt package from qt.io. The real time RF viewer for linux requires Qt version 5.7.1 or greater with Qtcharts built using gcc 64 bit.

In order to build the tcp_acq_server and tcp_acq_client software, do the following:

- 1) unzip the client and server into their own directories.
- 2) Open Qt creator and open the .pro file inside both the client and server directories. (same process for both programs)
- 3) Switch to "projects mode" on the left side of the screen.
- 4) Uncheck "shadow build".
- 5) Switch back to "edit mode".
- 6) Right click on the project and run "clean", then "qmake", and finally "rebuild".
- 7) Place the files "dma_registers.txt", "fr_end_registers.txt", "fr_end_spi_registers.txt", and "pps_registers.txt" in the /opt/acq_server/bin directory.

In order to build the acq_server pipes based software, do the following:

- 1) unzip the acq_server software
- 2) Open Qt creator and open the .pro file
- 3) Switch to "projects mode" on the left side of the screen.
- 4) Uncheck "shadow build".
- 5) Switch back to "edit mode".
- 6) Right click on the project and run "clean", then "qmake", and finally "rebuild".
- 7) Place the files "dma_registers.txt", "fr_end_registers.txt", "fr_end_spi_registers.txt", and "pps_registers.txt" in the /opt/acq_server/bin directory.



The screenshot shows the Qt Creator interface with the 'tcp_acq_client.pro' file open in the editor. The left sidebar shows the project structure with 'Sources' containing 'main.cpp', 'mainwindow.cpp', 'metadata.cpp', and 'runsettings.cpp'. The main editor displays the following code:

```
1 #-----
2 #
3 # Project created by QtCreator 2019-09-15T16:27:53
4 #
5 #-----
6
7 QT     += core gui  charts network
8
9 greaterThan(QT_MAJOR_VERSION, 4): QT += widgets
10
11 TARGET = tcp_acq_client
12 TEMPLATE = app
13
14 # The following define makes your compiler emit warnings if you use
15 # any feature of Qt which has been marked as deprecated (the exact warnings
16 # depend on your compiler). Please consult the documentation of the
17 # deprecated API in order to know how to port your code away from it.
18 DEFINES += QT_DEPRECATED_WARNINGS
19
20 # You can also make your code fail to compile if you use deprecated APIs.
21 # In order to do so, uncomment the following line.
22 # You can also select to disable deprecated APIs only up to a certain version of Qt.
23 #DEFINES += QT_DISABLE_DEPRECATED_BEFORE=0x060000   # disables all the APIs deprecated before Qt 6
24
25 CONFIG += c++11
26
27 #INCLUDEPATH += "./fftw-3.3.5-dll32"
28 win32 {
29     LIBS += -LC:/fftw32libs -llibfftw3-3
30 }
31 else {
32     LIBS += -lfftw3
33 }
34
35 SOURCES += \
36     main.cpp \
37     mainwindow.cpp \
38     metadata.cpp \
39     runsettings.cpp
40
```

7. Running the Example Programs

The software provided allows the A/D boards to be run under either a Command Prompt under both Windows and Linux, as well as in LabVIEW for Windows, and Qt for Linux.

See Section 11.2 for how to run the LabVIEW project.

See Section 11.3 and 11.4 for how to run the Command Prompt project

7.1 Preparing to Run the Example Programs

The software release has both source and executables for the various example programs, which can immediately be run to demonstrate the use of the board.

Usage information for the applications is available by simply running the executable without any arguments. The on-board prom contains calibration settings (offset/gain/bias) of each board. If the user would like to change these, please see appendix 16.

7.2 LabVIEW™ The Graphical Waveform Viewer under Windows 7/10

This complex but versatile LabVIEW project incorporates most of the same features as the 'acquire.exe' command but has the advantage of displaying the data acquired in real time with a powerful user interface. To run the LabVIEW executables, the user must first download the LabVIEW 2014 64-bit Run-Time Engine, depending on the description of the software package.

Please Note: All directory references are within the current software directory that the ultraview software was extracted to.

Quick Start in Run And Display:

- 1) In the "complete_daq_utilities\LabVIEW_2014_64-bit" directory, double-click either **LabVIEW_Acquire_Data_x64.exe** or **LabVIEW_Acquire_Data_x86.bat** depending on the system OS: Windows 7: exe, Windows 10: bat.
- 2) In **ReadandDisplay.vi**, press **Run** under the **RunAndDisplay** tab.
- 3) At this point, the board should be running in Continuous mode using the internal clock with no triggering (i.e. free-run mode). Data is simply being read from the board and displayed, and is **NOT being stored to disk**.

Quick Start in Acquire And Save:

- 1) In the "complete_daq_utilities\LabVIEW_2014_64-bit" directory, double-click either **LabVIEW_Acquire_Data_x64.exe** or **LabVIEW_Acquire_Data_x86.bat** depending on the system OS: Windows 7: exe, Windows 10: bat.
- 2) In **ReadandDisplay.vi**, select the **AcquireAndSave** tab, browse to select or create the file where you wish to save the data, and press **Acquire Data**. At this point, the board should be running in Acquire Once mode using the internal clock with no triggering. Data is simply being read from the board and displayed, and **IS being stored to disc** in the default directory with the file name **uvdma.dat** with a size of **260KB** when the **# of Blocks to Acquire** is "1".

Quick Start in Read Data File:

- 1) In the "complete_daq_utilities\LabVIEW_2014_64-bit" directory, double-click either **LabVIEW_Acquire_Data_x64.exe** or **LabVIEW_Acquire_Data_x86.bat** depending on the system OS: Windows 7: exe, Windows 10: bat.
- 2) In **ReadandDisplay.vi**, select the **ReadDataFile** tab, click the button that says "**metadata file**", browse to select the file where you wish to read data from, and press **Play Whole File**. At this point, the file should be read and displayed in the graphical windows. Data is simply being read from the file and displayed, nothing is being stored or read from any external source. If the bit resolution of the file does not match the bit resolution of the board currently in the system, use the ADC res readback control at the bottom. This viewer can be used to view the amplitude of any raw binary file with 8,12,14,16, or 32 bits per sample.

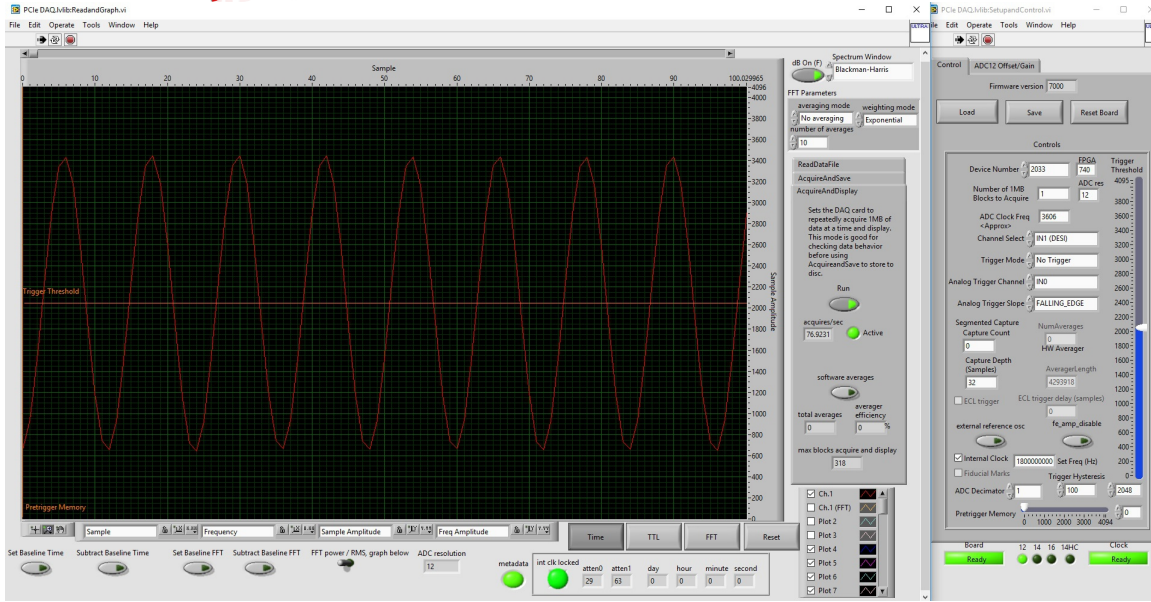


Figure 5. Time domain display of RF input. Control panel shown, which can be used for either time or frequency spectral display is shown at right.

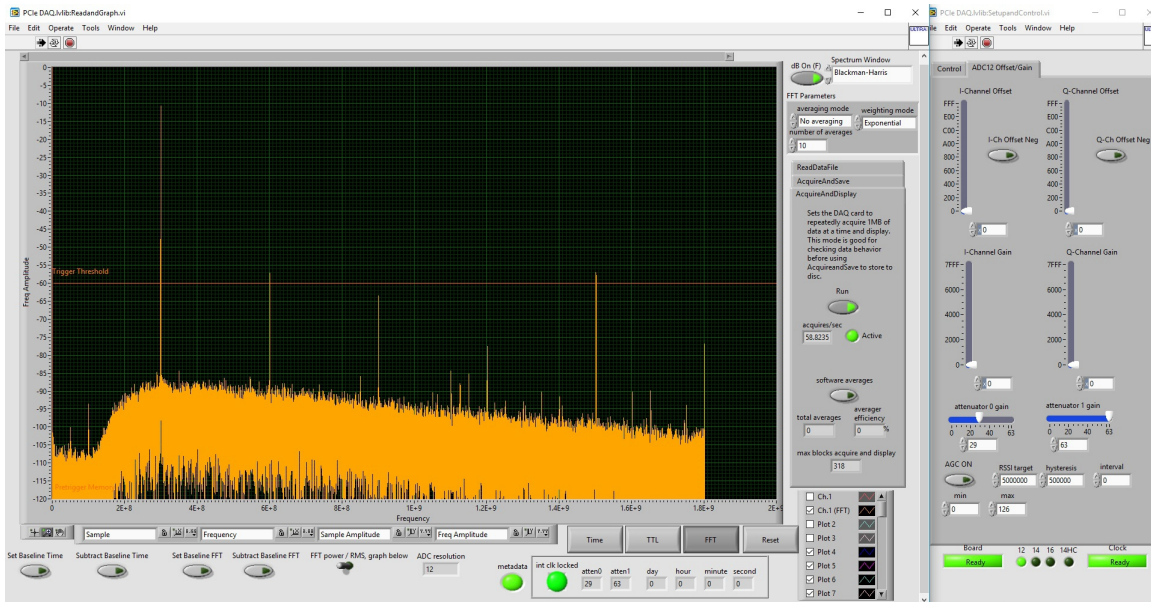


Figure 6. Frequency domain display of RF input. Control panel shown, which can be used for either time or frequency spectral display is shown at right. Acquisition is of 100mV 300MHz input with first and second RF attenuator settings of 29 and 63.

In LabVIEW, two executables communicate over TCP/IP to share important board configurations. ReadandDisplay.exe acts as the client, and SetupandControl.exe acts as the server. For the user's convenience, **LabVIEW_Acquire_data_x64.exe** and **LabVIEW_Acquire_Data_x86.bat** (Windows 10) are simple batch files to open both programs so that TCP/IP connects properly. To do this manually, the user must open SetupandControl.exe FIRST and ReadandGraph.exe SECOND, located in the "build" directory. All boards are set up with default parameters when the Labview program is opened and only if there is a lockup does the "Reset Board" button need to be

pressed.

When running **SetupandControl.vi**, the first tab displayed in the program is the "**Control**" tab. The "**Load**" button can be used to load the device and configuration settings from the previous running of the program. Modify these settings and press the "**Save**" button to save configuration settings. Next, to change boards, if multiple boards are in the system, simply change the "**Device Number**" to the proper serial number and click "**Setup Board**" to initialize the default configuration settings. **Note: These configuration settings are the same as used for the acquire program. Triggering and select recording options are explained in Section 13.4.1.**

Note: The board has not yet begun acquiring.

For all triggered modes of acquisition the AD12 adds programmable capture length. For each recognized trigger (TTL edge or analog threshold), the board will acquire "Capture Depth" number of samples. Note that if "Capture Depth" or "Capture Count" are zero then once triggered the data will be recorded until the acquisition is complete (normal mode). There is an added "Capture Count" modifier, which programs how many trigger events to record before resuming normal capture operation. Since our minimum block size to acquire is 260Kb, and a user may have a situation where only a limited number of triggers are going to be produced, using the "Capture Count" flushes out the rest of the buffer after a programmed number of triggers.

When running **ReadandGraph.vi**, the time and frequency graphs are **overlayed** onto a single graph. The **left and bottom axes** are for the **frequency domain response**, while the **top and right axes** are for the **time domain response**. Simple "**Time**" and "**Freq**" buttons are available to easily hide Time or Frequency Plots. The "**Reset**" button resets the X and Y scales to default values for both Time and Frequency.

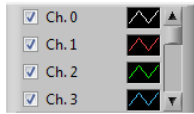
In the "**AcquireAndDisplay**" mode, use the "**Run**" button to start and stop acquisition of data. In this mode the board is used to acquire and display a small amount of data. **AcquireAndDisplay does not store data to disk.** In this mode the board starts acquisition, then after accumulating the desired number of blocks the data is displayed. The board starts again, and this cycle repeats as long as the "Run/Stop" button is pressed. This mode is useful for quick observation/verification of real-time data. When triggering, convenient cursors are shown to indicate the waveform trigger threshold and pre-trigger value. Additional cursors can be made and customized by right-clicking the cursor legend. In SetupandControl.exe the user can adjust the number of blocks for LabVIEW™ to process by increasing the "**# Blocks to Acquire**". The RunAndDisplay mode sets the board up to acquire 1 to 1+X blocks at a time, where X is the number of additional blocks supported by available memory at run time.

The "**AcquireAndSave**" tab can be used to acquire the previously specified number of blocks of data to a file and then view the data. Change the "graph interval" value to acquire data without displaying every waveform to speed up acquisition. Specify the number of blocks to acquire by changing the "**# Blocks to Acquire**" before pressing "Acquire Data". When using AcquireAndSave, the user can specify 8192 blocks guaranteed without overruns and data is stored to "uvdma.dat". **Specifying more than 8192 blocks may result in overruns if the input clock is too fast.** When the Acquire button is pressed, each block of data is displayed and stored in user defined filename in the default directory. AD12 is 12 bit data with 2 TTL bits and 2 blank bits.

AD12 only: For looking at the entire record without software averaging, select the appropriate bit resolution in the ADC res readback control and push the "file res != adc res" button. The scale can be quickly adjusted to the correct scale by pushing the Y scale button.

Built-in LabVIEW™ panning and zoom utilities are available on the **bottom and bottom-left** of the graph  (zoom palette)  (axes palette). Right-clicking the graph itself

allows for more graph options. **Built-in LabVIEW™ plot legend** are made available to the user on



the **bottom-right** of the graph (plot legend).

Note: There is a LabVIEW™ bug where the checkbox for the first channel does not work. Instead, left-click on the plot icon and deselect “Plot Visible”.

Note: There is a LabVIEW™ bug where the plot legend retains all plots in the list, even if there is no data in those channels. So ignore the long list.

At any time the user wishes to return to default window settings, the “**Reset Board**” button will do just that.

To properly close the LabVIEW software without any warnings popping up about TCP/IP error, close SetupControl.exe FIRST, then close ReadGraph.exe SECOND.

7.3 The Real-time Graphical Rx Waveform / Spectral Viewer under Linux

A simple RF receiver control panel software program which is shipped with the AD12-2000-IRIGB is shown below. This program conveniently displays RF data in both time domain and frequency spectral formats. Captured data can be aligned to the 1 PPS signal from the IRIG-B receiver in PPS mode.

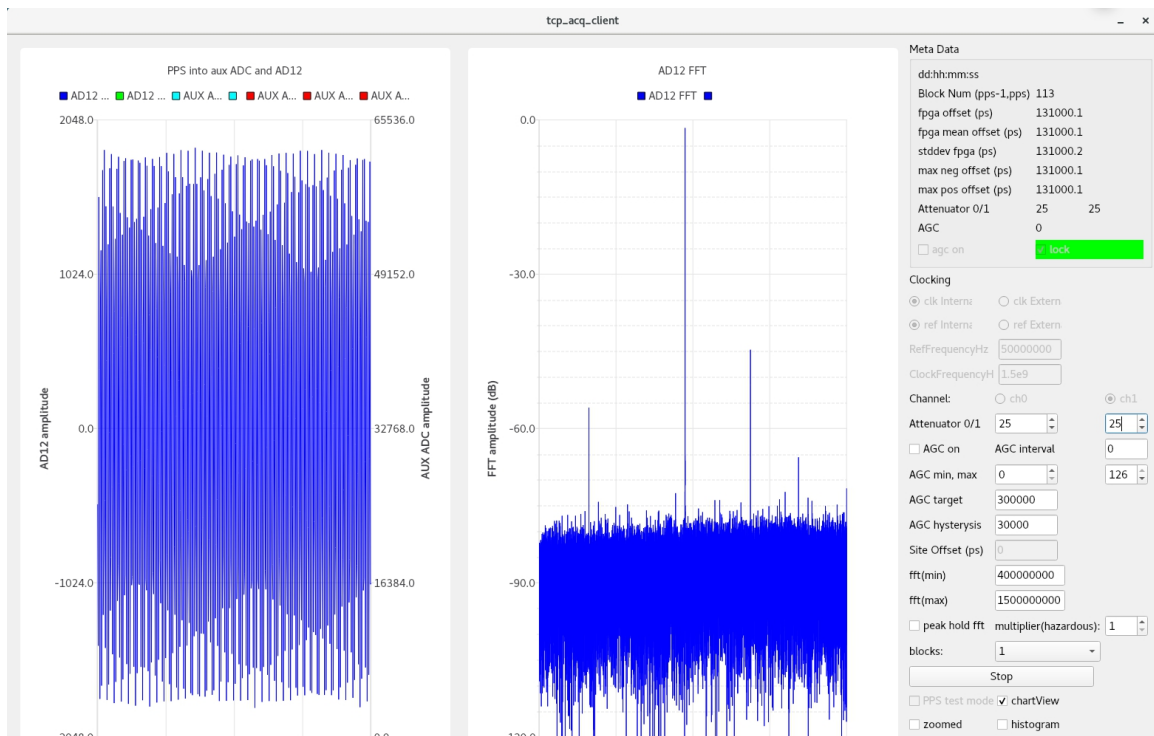
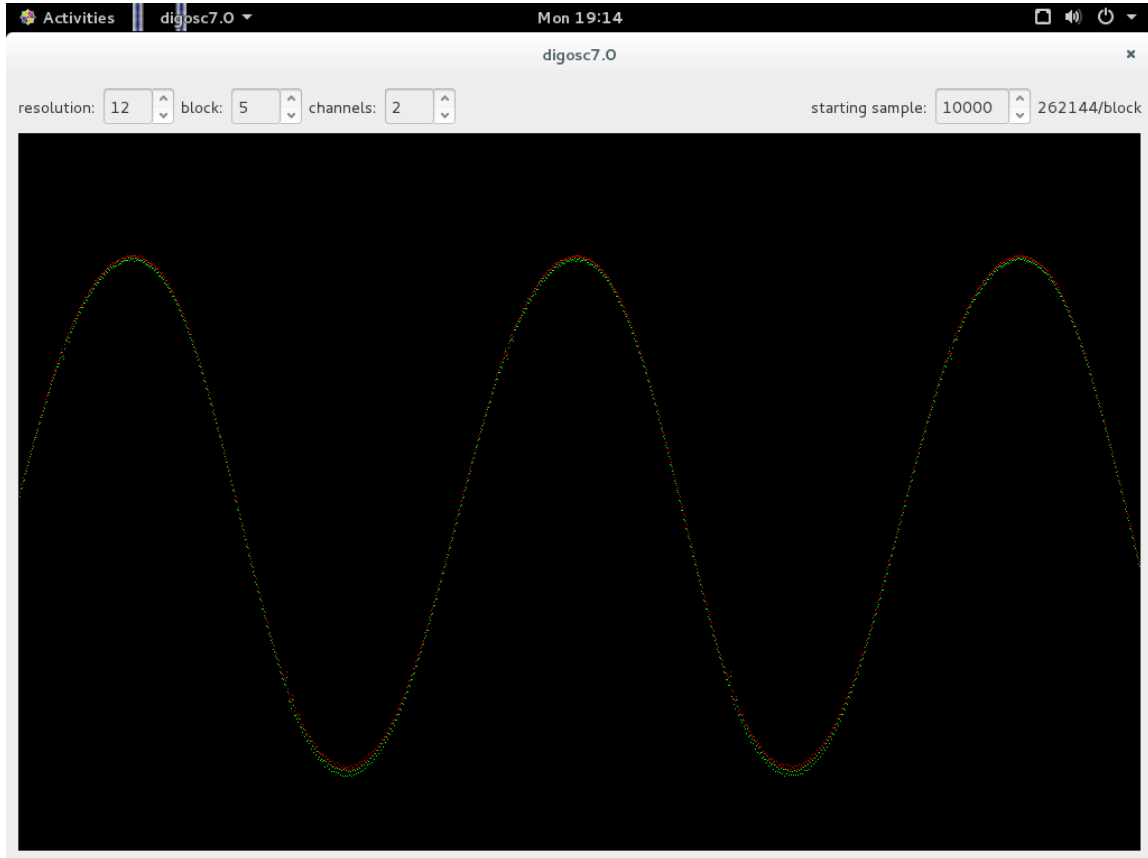


Figure 7. Linux receiver control panel program, allows full control, and displays real-time status. Left graph is time domain waveform plot, right graph is log spectral plot. Shown with a 900MHz input signal.

The additional Linux application “digosc7” displays, in waveform format, the A/D data acquired by the board. This graphical waveform viewer can be used to read data from a previously stored record on the hard disk.



When executed the program will ask for a binary file to read in, then ask for the specifications of that file. After specifications are entered, digosc will display the data. The data can be scrolled through by changing the sample number in the upper right, and the block number in the upper left. Only the current 262K block, the previous 262K block, and the next 262K block are loaded into memory at a time for fast access to large files that may be larger than system memory. As such, changing the block number re-enters data from the file specified, and if the file is changed the new file will be read in instead.

7.4 The Real-time pipes based acquisition software

For highly custom applications, a pipes based software environment is also provided. This allows end-users the flexibility of using any environment and programming language they want (that can do pipes) to connect to Ultraview software libraries and hardware. An example of the output of this program (Figure 8) and the software API is shown on the next pages.

Using this program for acquisition is done by modifying the runsettings.cpp file, which controls the acquisition parameters.

void controlThread::start (int verbose_i, int * s2m_i, int * m2s_i, int * gain_i, int ** metaDataPipe_i)

Start the main control thread

Parameters

| | | |
|-----|--|--|
| in | <i>verbose_i</i> | sets the amount of debug output. See global_settings.h |
| out | <i>s2m_i[2]</i> | is the acquisition control pipe from the controlThread to the Ultraview acquisition threads |
| in | <i>m2s_i[2]</i> | is the status pipe from the Ultraview acquisition threads back to the controlThread . See hw_wrapper thread |
| out | <i>gain_i[2]</i> | is the gain control pipe from the controlThread to the Ultraview acquisition threads |
| in | <i>metaDataPipe_i[N UM_READER_TH READS][2]</i> | is the data pipe from the Ultraview acquisition threads back to the controlThread . See demo_producers and rxDataThread |

void glitchTestThread::start (int verbose_i, int ** gitcheTestPipe_i, runSettingsStruct rss_i)

Start the receive data thread (this thread)

Parameters

| | | |
|----|--------------------------|---|
| in | <i>verbose_i</i> | sets the amount of debug output. See global_settings.h |
| in | <i>thread_num_i</i> | associated this thread to the Ultraview acquisition thread . See demo_producers |
| in | <i>metaDataPipe_i[2]</i> | is the data pipe from the associated Ultraview acquisition thread . See demo_producers |
| in | <i>num_blocks_i</i> | is number of system blocks to process. This thread converts that number to how many blocks this thread should process |

void metaData::fillStructFromMem (void * metaDataMem)

fill the basic information into the metadata structure

Parameters

| | | |
|----|----------------|-------------------|
| in | <i>pointer</i> | to metadata block |
|----|----------------|-------------------|

```
//Runsettings struct:
typedef struct {
    uint64_t magicNumber; // automatically set by runsettings.cpp
    uint64_t numBlocksToAcquire; // 1 - (2^48 - 1)
    uint64_t internalClockFrequencyHz; // internal clock mode sets
microsyth frequency, External clock mode is provided for FFT, etc.
    uint64_t refFrequencyHz; // internal clock used to inform micrsynth
what the reference freq is
    int64_t siteOffset; // used to correct 1 pps arrival time units =
ps note, this is internally converted to adc sample units before
programming the setting in the FPGA
    uint32_t sequenceNum; // used to keep synchronization
    uint32_t blockSize; // number of ADC samples per block
    uint32_t ECLTriggerDelay; // unused, there is no ECL trigger on the
RF/IRIG board
    uint32_t CaptureCount; // unused on the RF/IRIG board
    uint32_t CaptureDepth; // unused on the RF/IRIG board
    uint32_t PretriggerMemory; // unused on the RF/IRIG board
    uint32_t AGC_Target; // AGC target value in ADC counts^2
    uint32_t AGC_Hysteresis; // AGC imits = (target -
hysteresis) to (target + hysteresis)
```

```

uint16_t Decimation; // can be used to reduce data
bandwidth by sending only every (n+1)th sample
uint16_t SingleChannelSelect; // must be true
uint16_t DualChannelSelect; // must be false
uint16_t TriggerThreshold; // unused on the RF/IRIG board
uint16_t TriggerHysteresis; // unused on the RF/IRIG board
uint16_t TriggerMode; // unused on the RF/IRIG board
uint16_t TriggerCh; // unused on the RF/IRIG board
uint16_t NumReaderThreads; // should be 4
uint16_t ProcessingMode; // OFFSET_BINARY, SUBTRACTED_MEAN, or
TWOS_COMPLEMENT
uint16_t atten0; // direct setting of atten0 0-63, 0 is max atten,
63 is minimum
uint16_t atten1; // direct setting of atten1 0-63, 0 is max atten,
63 is minimum
uint16_t agc_min; // 0-126 sets AGC control limit
uint16_t agc_max; // 0-126 sets AGC control limit
uint16_t agc_interval; // 0-65535 sets how often the AGC is updated.
0 = every block, 1 , nevery other n = every (n+1)th block

bool TriggerSlopeRisingEdge; // unused on the RF/IRIG board
bool AcqDisableInv; // unused on the RF/IRIG board
bool clockInternal; // set to true for internal synth
bool refInternal; // set to true for internal 50MHz ref
bool SingleChannelMode; // set to true
bool DualChannelMode; // set to false
bool DESQ; // use the lower or ch0 input
bool DESI; // use the upper or ch1 input
bool DESIQ; // see AD12D2000RF data sheet. requires both analog
inputs to be driven by the signal of interest
bool DESCLKIQ; // see AD12D2000RF data sheet. requires both analog
inputs to be driven by the signal of interest
bool ECLTrigger; // unused on the RF/IRIG board
bool TTLInvert; // unused on the RF/IRIG board
bool ForceCal; // set to true to force clock and adc calibration
(may not be fully implemented yet, and each run already calibrtes)
bool EightBitTransfers; // set to true to reduce data to 8bit
bool IncludeMetaData; // set to true to include METADATA
bool IncludeAuxData; // leave false (not working, required aux DATA
is in METADATA)
bool GenTestRamp; // unused on the RF/IRIG board
bool InvertSpectrum; // invert every other sample

bool attenCmd; // set to true to control AGC or to set the
attenuators manually.
bool AGC_on; // set to true to use AGC (blocks all other SPI
commands, so it should be set to false when not in use)
uint16_t internalCmd; // leave false
bool PPS_testMode; // leave false
bool stopAcq; // set to true to stop acuisition
bool quit; // set to true to quit the data acquisition system
bool forceCal; // set to true to force a calibration (only valid
with a new acq)

int rf_block_interval; // unused, leave at default value
int num_rf_blocks_per_transfer; // unused, leave at default value
int ch0_ad12_threshold; // unused, leave at default value
bool ch0_ad12_is_inverted; // unused, leave at default value
} runSettingsStruct;

```

int32_t runSettings::calc_board_blocks ()

calculate the number of system blocks available in the boards memory

Returns

number of system blocks or a negative val on error **int64_t runSettings::calc_read_blocks ()**
 calculate the number of "read blocks" each dma read read in a "read block" because of peculiarities with the driver and dma-ip, we need multiples of 16k as such read blocks can consist of multiple sys blocks

Returns

number of read blocks or a negative val on error **void runSettings::debug ()**
 print the **runSettingsStruct** **void runSettings::debug (std::string srcID)**
 print the **runSettingsStruct** with the srcID string (helpful with multiple threads)

Parameters

| | | |
|----|-------|---|
| in | srcID | a text string supplied by the user to identify which thread is displaying |
|----|-------|---|

int32_t runSettings::num_sys_per_rd_blk ()

calculate the number of sys blocks per read blocks sys blocks specified as rss.numBlocksToAcquire when rss.IncludeMetaData is true, each read must be a multiple of sysBlocks such that the resulting read is a multiple of 16k

Returns

number of system blocks per read block or a negative val on error **void runSettings::quitProcess (int * s2m_i)**
 Send a quit (shutdown) acquisition command to the ultraview acquisition threads

Parameters

| | | |
|-----|----------|--|
| out | s2m_i[2] | is the acquisition control pipe from the controlThread to the Ultraview acquisition threads |
|-----|----------|--|

int64_t runSettings::read_blocks_in_pool ()

Returns

number of read blocks in the circular acquisition circular-queue or a negative val on error **bool runSettings::startAcquisition (int * s2m_i)**
 Send the **runSettingsStruct** to the ultraview acquisition system

Parameters

| | | |
|-----|----------|--|
| out | s2m_i[2] | is the acquisition control pipe from the controlThread to the Ultraview acquisition threads |
|-----|----------|--|

Returns

True if the command was sent. False if shutdown in progress. **void runSettings::stopAcquisition (int * s2m_i)**
 Send a stop acquisition command to the ultraview acquisition threads

Parameters

| | | |
|-----|-----------------|--|
| out | <i>s2m_i[2]</i> | is the acquisition control pipe from the controlThread to the Ultraview acquisition threads |
|-----|-----------------|--|

int32_t runSettings::sys_block_size ()

calculate the size (bytes) of a sys block

Returns

size of a system block incuding METADATA (if rss.IncludeMetaData) **char ***
runSettings::sysBuffer (uint64_t *blockNum*, char ** *pool*)

Returns

pointer to the buffer containing the requested system block num **bool**
runSettings::waitForServer (int * *m2s_i*)

Wait until the ultraview acquisition threads send a ready status

Parameters

| | | |
|----|-----------------|---|
| in | <i>m2s_i[2]</i> | is the status pipe from the Ultraview acquisition threads back to the controlThread . See hw_wrapper thread |
|----|-----------------|---|

Returns

True if the acuisition system is ready, False if the acquisition system returns shutting down.

void rxDataThread::start (int *verbose_i*, int *thread_num_i*, int * *metaDataPipe_i*, int * *gitchTestPipe_i*, runSettingsStruct *rss_i*)

Start the receive data thread (this thread)

Parameters

| | | |
|----|--------------------------|---|
| in | <i>verbose_i</i> | sets the amount of debug output. See global_settings.h |
| in | <i>thread_num_i</i> | associated this thread to theUltraview acquisition thread . See demo_producers |
| in | <i>metaDataPipe_i[2]</i> | is the data pipe from the associated Ultraview acquisition thread . See demo_producers |
| in | <i>num_blocks_i</i> | is number of system blocks to process. This thread converts that number to how many blocks this thread should process |

Meta Data Structure (bits):

- (31 downto 0) <= MAGIC_NUM;
- (79 downto 32) <= std_logic_vector(block_num);
- (111 downto 96) <= std_logic_vector(max_i_arr3(15 downto 0));
- (127 downto 112) <= std_logic_vector(max_q_arr3(15 downto 0));
- (151 downto 128) <= num_i_over_runs2;
- (183 downto 160) <= num_q_over_runs2;
- (215 downto 192) <= mean_sqr_i_accum3;

```
(247 downto 224) <= mean_sqr_q_accum3;
(271 downto 256) <= std_logic_vector(means_i(15 downto 0));
(287 downto 272) <= std_logic_vector(means_q(15 downto 0));
```

```
-- AGC settings -- 48 bits including padding
```

```
(311 downto 288) <= agc_target_val_i;
(317 downto 312) <= atten0_val_i;
(323 downto 318) <= atten1_val_i;
(330 downto 324) <= agc_val_i;
(337 downto 331) <= agc_min_i;
(344 downto 338) <= agc_max_i;
(368 downto 345) <= agc_hysteresis_val_i;
(384 downto 369) <= agc_interval_cnt_i;
(385) <= agc_on_i;
```

```
-- general settings -- 48 bits including padding
```

```
(396 downto 392) <= block_bits_i;
(397) <= two_ch_mode_i;
(398) <= is_8bit_i;
(399) <= invert_spectrum_i;
(401 downto 400) <= pocessing_sel_i;
(404 downto 402) <= gain_val_i_i;
(407 downto 405) <= gain_val_q_i;
(419 downto 408) <= mean_val_i_i;
(435 downto 424) <= mean_val_q_i;
(440) <= microsynth_ld;
(63 downto 0)+1* 512 <= trig_sample;
(127 downto 64)+1* 512 <= pps_siteOffset;
(159 downto 128)+1* 512 <= pps_ad12Offset;
(191 downto 160)+1* 512 <= pps_t;
(271 downto 192)+1* 512 <= pps_derived;
(275 downto 272)+1* 512 <= irig_bcd_s0;
(279 downto 276)+1* 512 <= irig_bcd_s10;
(283 downto 280)+1* 512 <= irig_bcd_m0;
(287 downto 284)+1* 512 <= irig_bcd_m10;
(291 downto 288)+1* 512 <= irig_bcd_h0;
(295 downto 292)+1* 512 <= irig_bcd_h10;
(299 downto 296)+1* 512 <= irig_bcd_d0;
(303 downto 300)+1* 512 <= irig_bcd_d10;
(307 downto 304)+1* 512 <= irig_bcd_d100;
/-- skip nibble for even byte
(317 downto 312)+1* 512 <= irig_seconds;
/-- skip unused for even byte
```

```
(325 downto 320)+1* 512 <= irig_minutes;
/-- skip unused for even byte
(332 downto 328)+1* 512 <= irig_hours;
/-- skip unused for even byte
(344 downto 336)+1* 512 <= irig_days;
```

```
(63 downto 0)+5* 512 <= pps_num_accum;
(127 downto 64)+5* 512 <= pps_denom_accum;
(191 downto 128)+5* 512 <= pps_t_num;
(255 downto 192)+5* 512 <= pps_b;
(287 downto 256)+5* 512 <= pps_m;
(295 downto 288)+5* 512 <= pps_ramp_start;
(303 downto 296)+5* 512 <= pps_ramp_n;
(311 downto 304)+5* 512 <= pps_mean_n;
(383 downto 320)+5* 512 <= pps_mean;
```

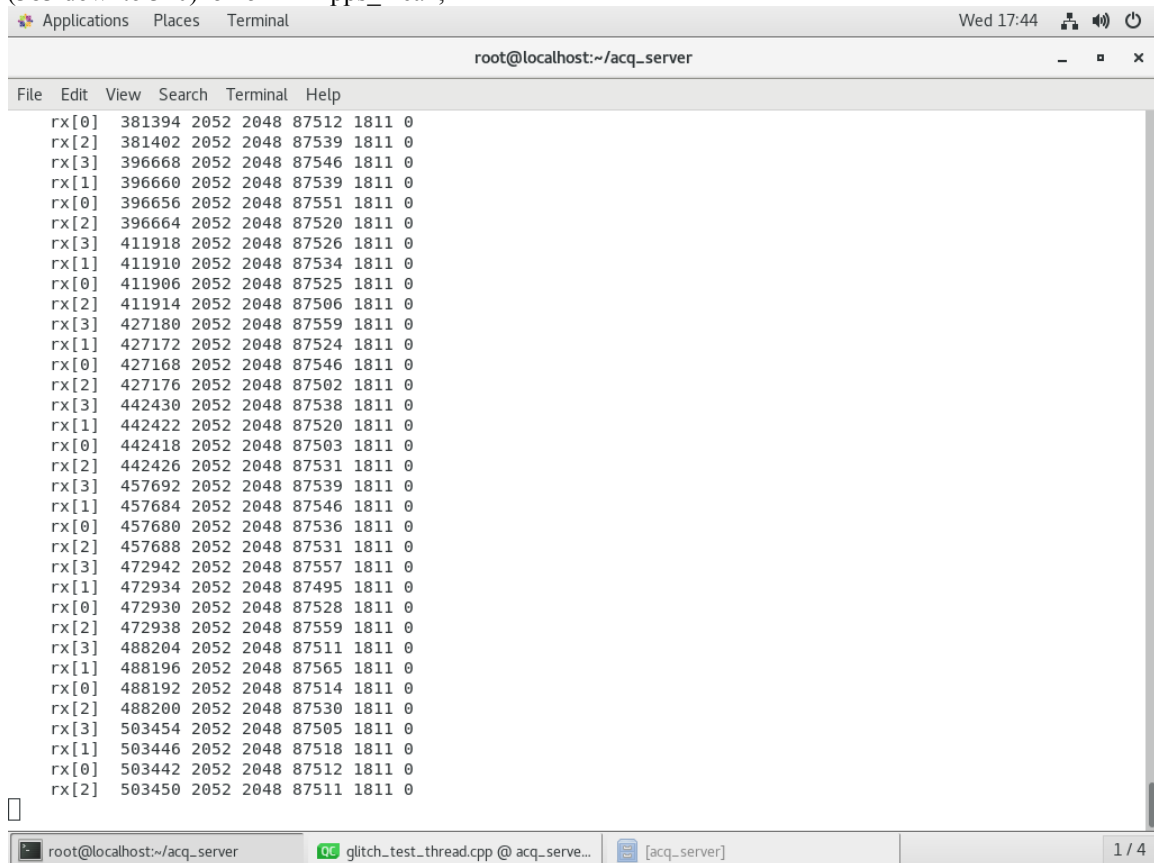


Figure 8. Example output from acq_server standalone pipes program

7.5 Cross Platform Command Line Acquisition & Synthesis

Both the Windows and Linux software releases include the same C source code files that implement a program to acquire data to disk and a program to synthesize data from disk. The Windows release includes these files as Visual Studio projects and the Linux release includes these files in /src/AppSource and /src/_DllSource, each with an included makefile. These example programs are mostly single threaded, with small sections of multithreading for DMA transfer. This is the simplest project to modify to get up and running with the board for custom applications.

There is a Qt based project in the directory “unsupportedcodeexamples” that is available on both Windows and Linux, that shows how to operate a high performance multi threaded application for controlling the board. There is both a command line and graphical Qt program. Due to the complexity, only moderately experienced or advanced programmers should attempt to make large changes to this program. This multi threaded program achieves the highest sustained transfer rates due to complex architecture designed to keep the board running as fast as possible.

7.5.1 acquire - acquire data into on-board DRAM, and then store the buffer to disk.

The program “acquire” acquires a selectable number of DMA blocks of data into the board's onboard memory and then move the data to a file on disk.

The program acquire can be used to acquire data to the board's on-board RAM, followed by an automatic storage of the board's buffer to a disk file. **For usage of acquire, perform the following commands and read all statements printed, further details can be found in the comments in acquire.cpp.** A minimum of 1 260KB block of data can be acquired, there is no maximum, but for the data to be valid through the entire record the sampling rate cannot exceed the saving to disk rate for longer than the 8GB on board memory buffer allows.

```
# cd /uvdma/example_programs
# ./acquire
```

```
Ex.  ./acquire      //Linux
      acquire      //Windows
```

```
// This will prompt a list of options the user can use to configure the board when
// acquiring
```

```
Ex.  ./acquire 1 -ic //Linux
      // acquires 260KB using internal clock with no triggering and stores
      // to disc in the build directory with file name uvdma.dat
```

Note: In Windows, the acquire program is accessed through the “Command Prompt” in the “build” directory. Simply type “acquire” in the Command Prompt for a list of operations.

```
Ex.  acquire 1      //Windows
      // acquire 260KB using external clock with no triggering and stores
      // to disc in the build directory with file name uvdma.dat
```

The acquire program has several different modes of operation. In the simplest case, the board acquires to onboard DRAM while continuously reading from the board into a small bounce buffer allocated in host memory and then writing from the bounce buffer to disk. In this case the board can acquire up to the size of the board's memory without any limitation on incoming data rate, or for a very long time if the disk DMA rate is sufficient relative to the incoming data rate. Alternatively, the acquire program can be compiled to use a large amount of system memory (typically limited to the amount of system memory minus a couple gigabytes) to increase the capture length. In this case writing to disk is delayed until the capture is completed, data is acquired into onboard DRAM and DMA'ed to the allocated system memory until this system memory is filled then the board's memory is filled until it contains only new data (data not yet transferred to the host). In this case the acquisition rate is limited to approximately the DMA rate of the system.

The acquire program offers basic triggering

- 1) “-ttlledge” allows for TTL triggering
- 2) “-analog” allows for analog waveform triggering. (Default falling edge)
 “-ttlinv” for rising edge trigger
- 3 “-analog_ch <Chan N>” to select which channel to analog trigger from. (Chan N) must be either 0,1,2, or 3.
- 4) “-thresh_a <N>” for analog trigger threshold. N=[0,65535]
- 5) “-thresh_b <N>” for analog trigger hysteresis. N=[0,65535]
 thresh_b should always be greater than thresh_a whether rising edge or falling edge.

Appending “-ttlinv” to the original acquire command will modify the trigger condition to an active high. For example, “acquire 1 -ic -ttlinv” will acquire 1 blocks with the internal 400MHz clock only when the selective recording trigger signal is HIGH.

Triggering allows for continuous acquisition, beginning at the first rising or falling edge (depending if “-ttlinv” has been asserted) of the trigger signal and stopping only when all data has been acquired.

To use an external reference for the internal clock instead of the built in 50Hz internal reference, use the flag “-extref”. This is nominally set to 10MHz in the configuration prom on the board and if a different frequency of reference is required this will need to be changed. See section appendix B for more details.

For multiple channel boards, the “-scm” extension enables single-channel mode. When using this option, also use “-scs” to specify which channel to acquire from (single-channel select) starting from “0”. For the channel not used, make sure to leave the input unconnected.

Note: Both triggering and selective recording use the “Acquire Disable (Trigger)” SMA jack as its input.

7.6 Cross Platform Command Line Options

acquire (N blocks) [OPTIONS]

Acquires specified number of blocks. Number of blocks option must always be first.

The following [OPTIONS] may follow the number of blocks option in any order:

For all boards:

"-ic" Board will use the internal clock, do not connect an external clock. If not specified board uses external clock.

"-ttlinv" Inverts the TTL trigger. Default is active low TTL trigger.

"-dec (factor)" Enables input sample decimation. (factor) can be 1,2,4,8,16,32,64, or 128.

"-ttledge" Sets acquisition to await TTL trigger edge.

"-forceCal" Forces calibration

"-capturedelay N" (delay N=0-63) Overrides adc clock to data capture delay (14 bit boards only)

"-analog" for analog waveform trigger. Use -ttlinv to specify rising or falling edge.

"-analog_ch (Chan N)" for analog waveform trigger. Used with -ttledge. Must be either 0,1,2, or 3.

"-thresh_a <N>" for analog waveform trigger threshold. <N> = [0,65535].

"-hysteresis <N>" for analog waveform trigger hysteresis. <N> = [0,65535].

"-pretrigger" for pretrigger memory after trigger is acquired. pretrigger must be 0 to 4095

"-ecltrig" Sets acquisition to await ECL trigger.

"-ecldelay" Set the ECL trigger delay. Default = 4000. See manual for details.

"-desiq" Sets the AD12-2000 to DESIQ single channel mode. Both inputs must be externally driven. In DESIQ, the I- and Q- inputs are shorted together. "-scm" must be asserted.

"-desclkiq" Sets the AD12-2000 to DESCLKIQ single channel mode. Both inputs must be externally driven. In DESCLKIQ, the I- and Q- inputs remain electrically separate, increasing input bandwidth. "-scm" must be asserted.

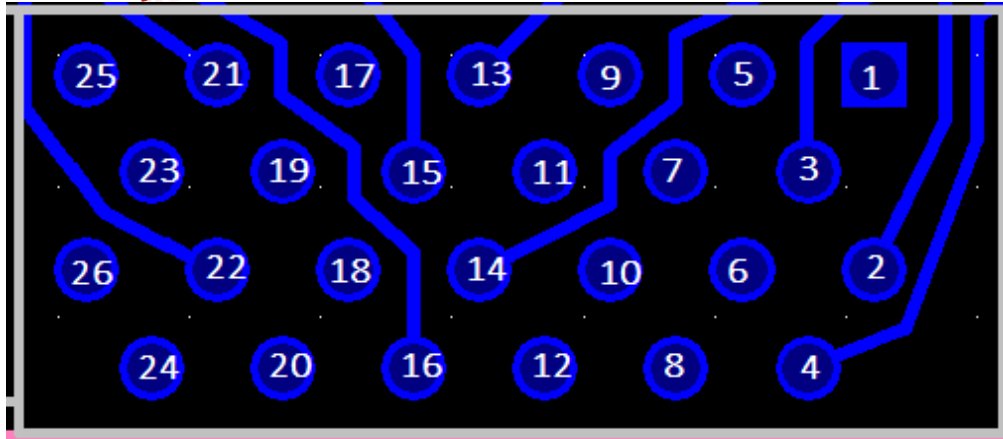
8. Microsynth programmable internal clock

The internal clock is a TI LMX2581 PLL-based synthesizer, which is programmable to any acceptable sampling frequency for the board. This can be done by setting -freq (frequency in Hz) as an acquire flag, or by typing in the desired frequency in the Labview control panel internal clock frequency control and pressing return. The internal reference oscillator is 50MHz. Optionally, users can connect their own external reference to the external clock SMA jack, and select “external reference” in Labview. The default external reference frequency is 10MHz, but any frequency between 5MHz and 900MHz is valid. If planning to use a frequency other than 10MHz, users must use the “filetoprom” program and the configuration file to properly assign the external reference frequency, or tell Ultraview the desired external reference frequency before ordering.

8.1 TTL input/output lines for custom firmware (Preliminary)

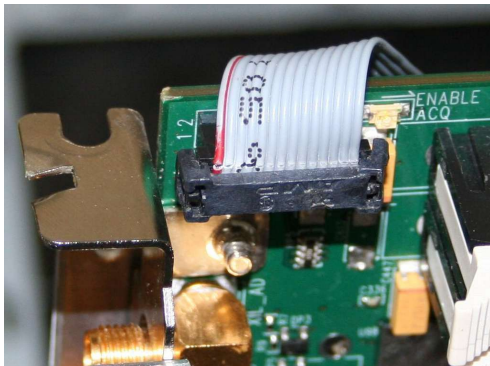
For those wishing to develop custom firmware, listed below are the TTL boardlet pins with their corresponding Xilinx pin.

| Connector Pin | Function | Xilinx Pin | SMA Jack Connector |
|---------------|----------------|------------|--------------------------------------|
| 1 | Ground | | |
| 2 | VTrans | | |
| 3 | Mezz P3 | G15 | TTL 4 |
| 4 | Mezz N3 | G14 | TTL 5 |
| 5 | Mezz P5 | K16 | |
| 6 | Mezz N5 | J16 | |
| 7 | Mezz N2 | J14 | |
| 8 | Mezz P2 | J15 | |
| 9 | Mezz P1 | K18 | |
| 10 | Mezz N1 | K17 | |
| 11 | Ground | | |
| 12 | Misc Xil L19 | A23 | |
| 13 | Mezz P7 | J19 | TTL 2 |
| 14 | Mezz N7 | J18 | TTL 3 |
| 15 | Mezz P6 | H17 | TTL 1 |
| 16 | Mezz N6 | H16 | TTL 0 |
| 17 | Mezz P4 | L19 | |
| 18 | Mezz N4 | L18 | |
| 19 | Misc Xil L15 | NA | |
| 20 | Misc Xil H14 | NA | |
| 21 | Mezz N0 | J10 | GC_N |
| 22 | Mezz P0 | K10 | Sync Select Record / TTL_TRIG / GC_P |
| 23 | Ground | | |
| 24 | Misc HDR Bank2 | AD8 | |
| 25 | VTrans | | |
| 26 | VTrans | | |

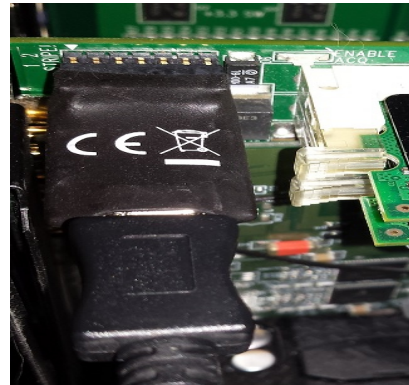


9. APPENDIX A – Firmware Update Using Programming Cable

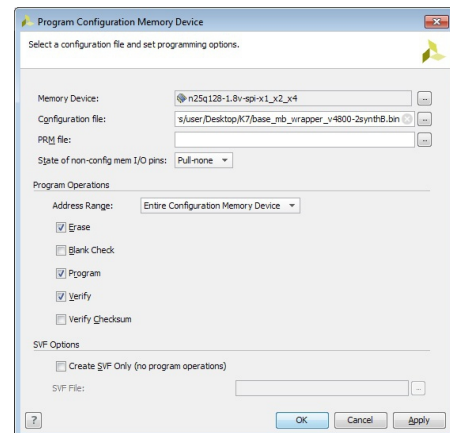
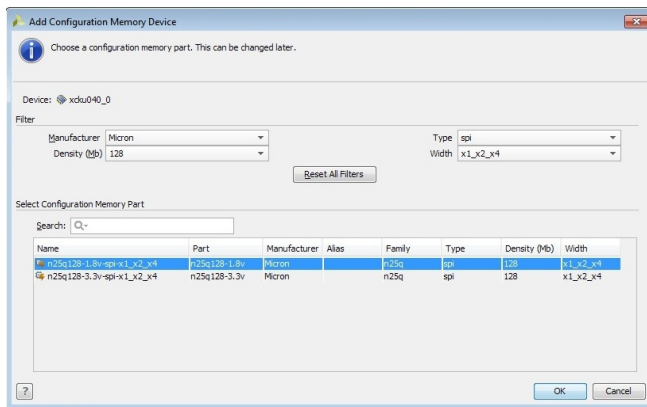
1. With the programming cable connected to the system that can run Vivado hardware manager connect the programming cable's ribbon cable to the programming header on the board. **Ensure that the red stripe on the ribbon cable is connected to pin 1 on the programming header, as shown below. For the digilent cable, ensure the alignment key is facing downwards. Improper connection could damage the board.**
2. Power up the system containing the board to be programmed. Booting the system into the BIOS is the safest thing to do.
3. In Vivado hardware manager, click “open target”, then “auto connect”.
4. If auto connect is successful the part should now show up and display the results of the DDR calibration. To program the prom, it must first be selected. Select “tools → add configuration memory device → XCKU040”
5. Select the n25q128-1.8V-spi-x1_x2_x4 part
6. Select the .bin file that contains the programming information for the prom/FPGA and click OK.



Programming cable orientation (original)



Programming cable orientation (Digilent)



Prom settings on Kintex 7 ultrascale FPGA based boards

10. APPENDIX B – ADC Gain/Offset/Bias/Microsynth Calibration

Normally, users should not need to recalibrate unless application specific precise calibration is required. The board specific calibration parameters are now contained in the on board prom on Kintex 7 Ultrascale FPGA based boards. To pull the factory values out of the prom, run the program “promptofile” and this will create a file “config.dat” that contains your board specific calibration. After using Labview to change these values, if appropriate, enter the new values into config.dat” and then run the program “filetoprom” and this will flash the prom with the new values so they will be loaded every time the board is run. In order to wipe the user settings from the prom and restore it to factory calibration, run the program “wipeuserpromsettings”. The program source for “promptofilefactory” is included, but no executable for it is included and it should not be run by users as this will wipe the factory calibration. User calibration will always supercede factory calibration.

```
--(Serial Number)
ADC_RES=          (2 digits, base 10) (required always) (ADC bit resolution)
ADC_CHAN=         (2 digits, base 10) (required always) (Number of ADC channels)
DAC_RES=          (2 digits, base 10) (required always) (DAC bit resolution)
DAC_CHAN=         (2 digits, base 10) (required always) (Number of DAC channels)

ADC3OFF=          (2-3 digits, base 16) (00-> no offset, FF-> maximum offset)
ADC3OFFNEG=      (1 digit, base 2) (1-> offset is negative, 0-> offset is positive)
ADC3FSR=          (3-4 digits, base 16) (Gain adjustment, 000 to 1FF)

ADC3OFF_Q=        (2-3 digits, base 16) (00-> no offset, FF-> maximum offset)
ADC2OFFNEG_Q=    (1 digit, base 2) (1-> offset is negative, 0-> offset is positive)
ADC3OFFNEG_Q=    (1 digit, base 2) (1-> offset is negative, 0-> offset is positive)
ADC3FSR_Q=        (3-4 digits, base 16) (Gain adjustment, 000 to 1FF)

ADC12D2000=      (2 digits, base 2)
ADC12D2000_DESI_Q_OFFSET= (3 digits, base 16)(Q offset adjustments to match when in single channel I)
ADC12D2000_DESQ_I_OFFSET= (3 digits, base 16)(I offset adjustments to match when in single channel Q)
ADC12D2000_DESIQ_I_OFFSET= (3 digits, base 16)(I offset adjust when IQ are electrically tied in SCM)
ADC12D2000_DESIQ_Q_OFFSET= (3 digits, base 16)(Q offset adjust when IQ are electrically tied in SCM)

ISLA_ADC0_IDELAY_OFFSET= (3 digits, base 10) (This is the data capture window)
ISLA_ADC1_IDELAY_OFFSET= (3 digits, base 10) (This is the data capture window)
ISLA_ADC2_IDELAY_OFFSET= (3 digits, base 10) (This is the data capture window)
ISLA_ADC3_IDELAY_OFFSET= (3 digits, base 10) (This is the data capture window)

MICROSYNTH_EXT_OSC_FREQ= (9 digits, base 10)
```

For AD12-2000-IRIG boards:

Set to single channel mode, channel 1

Adjust ADC3OFF and ADC12D2000_DESI_Q_OFFSET so that an open input gives a 50% 2047/2048 distribution. Check FFT.

Adjust ADC3FSR/ADC3FSR_Q so that an input signal gives a minimum value at FS/2 on the FFT.

11. APPENDIX C – How to Modify Your LabVIEW™ Project

18.1 – Modifying VI's

This section is for users who wish to modify their LabVIEW™ VI's from their original configuration.

1. Open the LabVIEW project Ultraview PCIe DAQ.lvproj.
2. Open and modify the VI file of interest.
3. Build the executable which is based on the modified VI. Some permissions may need to be altered depending on operating system and environment. If you are not running as administrator and the LabVIEW project explorer says you do not have access to the required folder, create the folder, go to properties, and change access on everyone group from read only to full control.
4. Copy the executable from its built location to its original location, overwriting the current file.
5. If updating to a new LabVIEW version, the data file lvanlys.dll in the /data directory may need to be copied over to the release as well.

12. APPENDIX D – Data Output Format (uvdma.dat)

Data is in little-endian format.

12-bit Single Channel

| [X][M][N] | where X = channel #,
M = sample (2 bytes),
N = byte #, 0 is LSB.

| | | | | | | |
|-----------|-----------|-----------|-----------|-----------|-----------|-----|
| [0][0][0] | [0][0][1] | [0][1][0] | [0][1][1] | [0][2][0] | [0][2][1] | ... |
|-----------|-----------|-----------|-----------|-----------|-----------|-----|

13. APPENDIX E. Certificate of Volatility

CERTIFICATE OF VOLATILITY

AD12-2000-IRIG SERIES DATA ACQUISITION BOARDS

1) All Ultraview PCIe bus data acquisition boards contain three types of memory storage – Non-Volatile Flash memory for the COTS FPGA configuration data, Volatile SRAM inside the FPGA for elasticity buffering of incoming data, and volatile DRAM, for on-board storage of acquired data until it is transferred to the host system.

2) Only the above referenced SRAM and DRAM receives acquired data, and therefore only these components could contain classified data. The most reliable way to erase 100% of this data is to run one of the active purging programs described in 3 below, which overwrite all RAM on the board with a neutral pattern, and then optionally further verify that only this neutral pattern exists in the board’s RAM. However, even if one of these active purging programs is not run, the vast majority (>99.9%) of the acquired data will alternatively be passively lost from both of these memories within one hour after power is removed from the board, such as by powering off the system. The remainder of the data in the board will disappear well within 24-hours after the board is powered down.

3) The board may alternatively be actively purged of all acquired data by executing any of the following processes:

A) **Disconnect all analog input cables from the data acquisition board, and run “acquire 8192 –ic”**. Wait until the board completes the operation and returns with a shell prompt (completion should occur within 16 seconds). All data in both the Volatile SRAM and the volatile DRAM in the board is now overwritten with baseline noise.

B) Run **“memtest –t 8192”**, and wait for completion (can take several minutes). This program writes a triangle wave pattern into all of memory, and then reads the entire memory back, verifying that only the triangle wave pattern remains in memory.

C) Run **“acquire 8192 -ic -purge”**. This will set the board into test pattern mode, then fill the ram with test data.

4) **Do not remove the DIMM memory modules from the board**, as the sockets are not designed for repeated insertions, and removal of the DIMM will void the product warranty. The procedures in (3) above will reliably remove data.

14. Appendix F. Known Issues

The following issues have been noted and may be addressed at a later time:

1. The AD12-2000-IRIG A/D boards will freeze if the PC sleeps. All power saving features MUST be disabled.
2. The Linux GUI program tcp_acq_client has a bug in that acquisition must be stopped (via the stop button in the GUI) before changing most operating parameters, otherwise the program may hang.

15. Appendix G – Example build instructions for Dell PowerEdge™ R940 series server

Steps to build/rebuild Synchrodyne IRIG receiver system

- 1) Insert usb stick with RHEL 7.6
- 2) In the bios, select single shot UEFI boot from USB
- 3) During installation of RHEL 7.6, change software installation from minimal install to server with GUI.
- 4) Select 6 additional package options: hardware monitor, large systems, performance tools, compatibility libraries, development tools, and system administration tools.
- 5) Create user <username>.
- 6) After installation has finished, log in as both <username> and root and turn off power options: no blank screen, no suspend.
- 7) Log in as <username>, and copy Ultraview provided archive of installation files to the server.
- 8) Extract all ultraview software from .zip archives and place it in /home.
- 9) Extract and install Vivado Lab tools to /home/<username>/Xilinx
- 10) Install Qt 5.12, include options "desktop gcc 64-bit" and "qt charts".
- 11) Log out as <username>, and log in as root
- 12) Place etc/udev files (2) in /etc/udev/rules.d
- 13) Navigate to /home/<username>/tcp_acq_client/fftw3.5
- 14) Run ./configure then make then make install.
- 15) Navigate to /usr/lib64 and run "ln -s libGL.so.1 libGL.so". For some reason the OpenGL linking works by default on Centos 7 but not on Redhat 7.
- 16) Navigate to /home/<username>/Xilinx/Vivado/2019.1/data/xicom/cable_drivers/linux64/install_script/install_drivers and run ./install_drivers.sh
- 17) Power off system and install Ultraview Synchrodyne IRIG receiver card.
- 18) Power on system (warm boot in bios may be required) and log in as root.
- 19) Navigate to /home/<username>/AD12_14_14HC_16-K7-Win7-Win10-Centos7_64bit_Software_R4.3/Driver/Xilinx_Answers_65444/driver.
- 20) Run "make clean", "make", "insmod xdma.ko", "make install".
- 21) Create directories /opt/acq_server/bin.
- 22) Place 4 .txt files from /home/<username>/tcp_acq_server to opt/acq_server/bin
- 23) Log out as root and log in as <username>.
- 24) Build all user programs (steps below):
- 25) AD12_14_14HC_16-K7-Win7-Win10-Centos7_64bit_Software_R4.3/src/_DllSource/ make clean, make.
- 26) AD12_14_14HC_16-K7-Win7-Win10-Centos7_64bit_Software_R4.3/src/ make clean, make
- 27) acq_server,tcp_acq_server,tcp_acq_client: Open Qt creator, navigate to the .pro file contained in each directory and open each project.
- 28) Select left tab 'project' and make sure "shadow build" is unchecked.
- 29) Select left tab 'edit', then right click on the project and run clean, run qmake, rebuild.
- 30) The system is now set up.