

ARTISAN®
TECHNOLOGY GROUP

Your **definitive** source for quality pre-owned equipment.

Artisan Technology Group
(217) 352-9330 | sales@artisantg.com | artisantg.com

Full-service, independent repair center
with experienced engineers and technicians on staff.

We buy your excess, underutilized, and idle equipment
along with credit for buybacks and trade-ins.

Custom engineering
so your equipment works exactly as you specify.

- Critical and expedited services
- In stock / Ready-to-ship
- Leasing / Rentals / Demos
- ITAR-certified secure asset solutions

Expert team | Trust guarantee | 100% satisfaction

All trademarks, brand names, and brands appearing herein are the property of their respective owners.

Visit our website - *Click HERE*

Reference Manual for the GIGExD Data Conversion Device

Innovative Computer Engineering

I.C.E, Inc.
www.ice-online.com
703-934-4879

November 2012
For Software Version 2.20

Contents

1. Quick Start....4

2. Introduction....6
   2.1. Product Notation....6
   2.2. Description....6
   2.3. Operation....7
   2.4. Configuration....7
   2.5. Programming....8
   2.6. Specialized A/D Interface....8
   2.7. Mechanical....9

3. Software....10
   3.1. Build Script and the gigexd Executable....10
   3.2. The gigexd Executable....12

4. Commands....13
   4.1. program and prgclrcfg Commands....13
   4.2. flashcfg and rtcfg Commands....17
   4.3. Configuration File Syntax....21
   4.4. Configuration File Parameter Description....21
   4.5. Real Mode Tuner....23
   4.6. query and qrywcfg Commands 31
   4.7. a2dgain Command 33
   4.8. acquire, acqall, and acqwtc Commands....35

5. Communications....40
   5.1. "Catch-22"....41
   5.2. Maximizing Host Computer GIGExD Port Data Transfer Rates....42

6.I/O Modules....43
   6.1. A2DR7....43
   6.2. A2DR7 Clocking For GIGExDR2....44
   6.3. A2DR9 Clocking For GIGExDR2....45
   6.4. A2DR10 Clocking for GIGEXDR2....46
   6.5. A2DR9/A2DR10/A2DR13 GAIN....47
   6.6. A2DR13 I/O Module....47

7. Technical Specifications....48

1. Quick Start

GIGExD Quick Start Guide

1.) The most up-to-date software can be obtained at:

http://www.ice-online.com/ftp

2) Attach data, clock and time code cables (if time code is to be used)

3) Connect power supply and Network cable to GIGExD for Host machine communication.

(GIGExD may be connected directly to the host machine or to a router network where the host machine resides.)

4) GIGExD IP address when delivered is 192.9.200.105. See the section titled **Communications** for information on how to set the host computer route to identify the address of 192.9.200.105

5) On Host computer execute: ping 192.9.200.105 <return> to be sure communications are established with the GIGExD.

6) Refer to the section: **Build Script & *gigexd* Executable** in the User Guide on how unzip/build the *gigexd* executable.

7) Refer to the section: **Commands** for generic help on the commands.

8) Under the **Commands** Section refer to the ***program* And *prgclrcfg* Commands** on a choice between firmware The GIGExD**R2** comes programmed with gigexdr2_79d2.rpd.

9) If the user selects gigexd_127d4.rpd, or gigexd_255d8.rpd, update the GIGExD firmware to gigexd_127d4.prm or gigexd_255d8.prm using the ***program* And *prgclrcfg* Commands** .

10) Refer to the section: ***flashcfg* And *rtcfg* Commands**, **Configuration File Syntax,** and **Configuration File Parameter Description** on how to configure the GIGExD for your applications.

11) Execute the *flashcfg* command to configure the GIGExD with a new IP address and configuration parameters.

12) Refer to the section: ***query* And *qrywcfg* Commands** on how to query configuration setting for step 11.

13) Execute *rtcfg* command (Section:***flashcfg* and *rtcfg* Commands)** to change parameters if necessary.

14) Refer to the section: ***acquire, acqall,* and *acqwtc* Commands** on how to acquire data to the Host computer from the GIGExD.

15) Go back to step 11 to reconfigure the GIGExD with new parameters that are non-volatile. Go back to step 13 to reconfigure the GIGExD with new parameters that are volatile.

16) Refer to section: "Catch-22" for information on setting the GIGExD back to its factory default settings.(Important)

17) All new features for this version of firmware/software are denoted by the text: *****New Feature*****

Other points:

A. The GIGExD should only be connected directly to gigabit Ethernet interfaces. It does NOT communicate with100Base-T or 10Base-T interfaces.

B. Firmware & Software updates will be posted on the web when new features are added. Some of the new features to be added are detailed in the GIGExD User Guide.

C. Finally, any questions can be directed to:

Bill Owen
ICE, Inc.
wao@ice-online.com
540-338-3252 or 703-934-4879

2. Introduction

2.1. Product Notation

****Important NEW Information****

As of version 2.0 of the GIGEXD software and firmware the GIGEXD**R1** (rev 1) is no longer supported. Software and firmware support for the GIGEXD**R1** ended with version 1.9. The GIGEXD**R2** contains more sophisticated circuitry that allows for faster and larger data manipulation. As development goes forward with more complex data manipulation algorithms, the GIGEXD**R1**'s older technology is unable to meet the requirements. Therefore, in this document, all discussions of version 2.0 and higher of software and firmware apply ONLY to the GIGEXD**R2**. Use of the GIGEXD**R1** has to be done using software/firmware versions 1.9 or earlier.

2.2. Description

The GIGExD is a self-contained data conversion device. Input data type interfaces include A/D, ECL, PECL, LVDS, SONET, etc that are converted to output gigabit Ethernet data. Once digital data enters the GIGExD it can be directly sent out via gigabit Ethernet or, optionally, pass through embedded Digital Tuners for Tune, Filter, Decimate, and Gain operations before output to gigabit Ethernet. The output gigabit Ethernet interface employs the use of the gigabit SFP (Small Form-factor Pluggable) modules which allow for either Copper or Fiber network connections. Commonly available module types are:

Copper “T” for use with copper wiring up to 100 meters

850nm fiber “SX” for use with multi-mode fiber up to 550 meters

1310nm fiber “LX” for use with single-mode fiber up to 10,000 meters

1550nm fiber “ZX” for use with single-mode fiber up to 70,000 meters

1310/1550nm “BX” single-optical fiber CWDM

The GIGExD's modular design allows for the use of standard ICE I/O modules. The I/O module data is converted to gigabit Ethernet traffic using multicast UDP packets that conform to the SDDS format-UDP protocol with 56 bytes of header in the UDP portion for time code, etc and 1024 bytes of data. The output SDDS packets are sent to a user defined multicast address. Once in UDP/SDDS packet form, that data can be transferred over commercial network frameworks then acquired into a host computer via a network port, or via an ICE PIC card using an SDDS I/O module.

rumeL Related Products

Part Number	Description
A2DR7-D	100Mhz 14 Bit, A/D
A2DR9-D	200MHz 12 Bit, Variable Gain Front End, A/D Converter
A2DR10/13-D	200MHz 12 Bit, Variable Gain Front End, A/D Converter
DR2D-D	Differential Receiver to Digital Converter Module
LBND2D	L-Band Receiver Module (Available December 1, 2007)
GIGExDR0	GIGExD Data Conversion Device, Prototype
GIGExDR1	GIGExD Data Conversion Device, Revision 1
GIGExDR2	GIGExD Data Conversion Device, Revision 2
RI-CAB-SA5	5-foot SMA-to-BNC Cable

2.3. Operation

The GIGExD is programmable and configurable via its gigabit Ethernet link. The device contains an IP address that is used for configuration or programming. A small C program is provided that will configure or program the module via a host systems' gigabit Ethernet port. Or, configuration may be done using an ICE PIC card and UDP/SDDS module that is connected to the same network framework that the GIGExD occupies. (Note: configuration of the GIGExD may be done using an ICEPIC card and UDP/SDDS interface module, but programming has not been implemented as of yet. At a later date programming of the GIGExD via the ICEPIC/SDDS module interface will be available.)

Once configured via the Ethernet, the device outputs data until reset or reconfigured again. Many options are available during configuration including the use of time code to "time stamp" packets. For A/D data, the time code source takes the form of IRIG-B time code that is converted inside the GIGExD to standard SDDS format and placed in packet headers. For digital modules, such as ECL, time code is extracted from the serial stream of the data and converted to standard SDDS format and placed in the SDDS headers. Time stamping may also be turned off for simple data conversion with no SDDS time code in the SDDS packet headers.

2.4. Configuration

As mentioned above each GIGExD has it's own IP and it's own output multicast address. The IP address is used for setting Data Type parameters within the GIGExD. Such parameters include time code type, samples size (8 bit or 16 bit), and clock rate. Along with Data Type parameters, the GIGExD can have the multicast address to which it sends data changed. Finally, the IP address of the GIGExD may be changed to allow

for multiple GIGExD units with unique IP and multicast addresses to operate in a single network framework.

On each GIGExD module a "default button" is provided. This button when pressed and held for 2 seconds will return the GIGExD to its factory default IP address and multicast output address. This allows the user to return a GIGExD to a default IP address for updating configuration or programming.

All configuration and programming is stored on the GIGExD in flash memory. This allows for Data Type configuration, IP addresses, and multicast addresses to remain through power cycles-whether the power cycles are intentional or not. Also, the use of flash memory to hold configuration means that a user can configure the parameters of the module, and these parameters are now in use until reprogrammed again. Once configured a GIGExD will run in its programmed mode until re-configured again. This allows for lower configuration cycles and maintenance. See the description of the gigexd.c program and the gigexdusr.cfg configuration file below for more details on module configuration.

2.5. Programming

The GIGExD contains on-board FPGAs, clock synthesizers, and EPROMs. As updates to the programming of these devices are made available (via the rumeL, Inc. web site or ICE software tree downloads), the GIGExD can be "re-programmed" to take advantage of these updates over any network. In other words, the GIGExD receives its programming over gigabit Ethernet. As new updates become available, updating the GIGExD is simply a matter of executing the C program to reprogram the device with these new features.

2.6. Specialized A/D Interface

Three specialized A/D interfaces has been designed to work with the GIGExD.

The first I/O module type name is A2DR7-D The A2DR7-D has an on-board clock synthesizer chip that can be set to the range of 90-100MHz sample rate that is locked to an external 10MHz reference. From the 90-100MHz sample rate other frequencies can be provided that are divisions of 90-100MHz by a power of 2-for example: 45-50MHz, and 22.5-25MHz. The A2DR7-D also allows for direct clocking at any rate via an on-board oscillator or via an external clock input. User selection between the clock sources is accomplished via software and the use of a configuration file.

The A2DR7-D also provides IRIG time code and PPS inputs. These inputs are then used to time stamp the analog data. Past projects have employed the use of GPS receivers (EndRun Technologies Meridian Model Number 3019-0102-000) that output a 10MHz reference clock, IRIG time code and a 1 PPS signal. The use of the on-board clock synthesizer was then used to digitize a 70MHz IF at 93MHz. The use of the

A2DR7-D with the GIGExD in this configuration provided highly accurate time stamped data across multiple GIGExD units.

The second module type is the A2DR9-D. Like the A2DR7-D, the A2DR9-D has an on-board clock synthesizer chip. However, the A2DR9-D,s frequency range can be set within the range of 180-200Mhz that is locked to an on-board or external reference 10Mhz reference. From the 180-200MHz sample rate other frequencies can be provided that are divisions of 180-200MHz by a power of 2-for example: 90-100MHz, 45-50MHz, and 22.5-25MHz.

The third module type is the A2DR10-D. Like the A2DR9-D the A2DR10-D has an on-board clock synthesizer device. This device uses either a on-board or external 10 Mhz reference to derive a sample clock. Unlike, the A2DR9-D the clock synthesizer can produce frequencies for the full sampling range of the A/D device-40 to 200 Mhz. Using this new clock synthesizer, a user can derive the exact frequency needed to clock tha A/D device.

As with the A2DR7-D, the A2DR9-D and A2DR10-D also provide IRIG time code and PPS input. These inputs are then used to time stamp the analog data. Using a GPS receiver that outputs a 10MHz reference clock, IRIG time code and a 1 PPS signal, would allow a user to acquire a 70 or 140MHz IF signal and accurately time stamp the data.

NOTE A2DR13-D

The A2DR10-D was renamed to the A2DR13-D module. The only changes to the A2DR13-D module compared to the A2DR10-D module is in name alone. Circuitry and functionality are identical. The name change was needed to keep module naming and functionality identical between the modules that are used on the GIGEXDR2 and the ICE PCI card I/O modules.

2.7. Mechanical

Both the Prototype and Production units come with mounting screw holes on the bottom of the housing. The Prototype unit comes with an internal fan for cooling. The Production unit does not have an internal fan but instead has its internal circuitry cooled by conduction to its metal housing using a heat sink. Schematics for mounting holes are located at the end of this document.

3. Software

A simple C program is used to program the GIGExD via a host computer's network port. A Linux build script is included with the software. Currently, the GIGExD software zip package gigexdv*A*_*B*.zip, (Where *A* and *B* represent the major and minor version numbers of the associated GIGExD software package) resides on the rumeL, Inc. web site at:

http://www.rumel-online.com/software.htm

(Note that the above extension is .htm NOT .html)

3.1. Build Script and the gigexd Executable

The software and firmware for the GIGExD is bundled into a single zip package of the form gigexdv*A*_*B*.zip. Where *A* and *B*, as detailed above, represent the major and minor version numbers of the associated GIGExD software package. When unzipped to a host computer, a directory of gigexdv*A*_*B* is created and all source/firmware files are unzipped into the directory of gigexdv*A*_*B* -this prevents the mixing of old and new software when packages are unzipped.

Once the package is unzipped, the directory gigexdv*A*_*B* will have a number of files:

File Name	Description
gigexd.c	C program source code
buildgigexd.lnx	Linux Build Script
gigexdusr.cfg	GIGExD Configuration File
gigexdr2_1023d32.rpd	GIGExD**R2** Firmware (1023 Tap By 32 Decimation Filter)
gigexdr2_255d8.rpd	GIGExD**R2** Firmware (255 Tap By 8 Decimation Filter)
gigexdr2_127d4.rpd	GIGExD**R2** Firmware (127 Tap By 4 Decimation Filter)
gigexdr2_79d2.rpd	GIGExD**R2** Firmware (79 Tap By 2 Decimation Filter)
gigexd79coef.txt	Example Of A Coefficients Data File
initgigexd	Production Initialization Script (NOT FOR END USER USE)
a2dr9dr1.pof	Programming Firmware (NOT FOR END USER USE)
gigexdr2.jic	Production Testing/Initialization Firmware (NOT FOR END USER USE)

The C program gigexd.c is the command/control and data acquisition source code for the GIGExD. This program uses standard C programming to configure and acquire data from the GIGExD using the gigabit Ethernet of a host computer. The host computer to GIGExD connection, may be a direct connection, or may be separated by several levels of routers.

The file buildgigexd.lnx is a build script for Linux that will compile the file gigexd.c into the executable *gigexd*.

The configuration file gigexdusr.cfg is a text file that can be edited to reset the IP address, multicast address, and data type configuration (Sample rate, number of bits, etc.)

The program download/firmware files of **gigexdr*rev*_*Taps*d*Decim*.rpd** are binary files that should NOT be edited. (Where *rev* is the hardware revision. For the embedded tuner, *Taps* represents the number of taps in the tuner filter and *Decim* represents the base decimation value) These files are used to reprogram the FPGAs on the GIGExD as new features are added and all end in the extension of ***.rpd***.

gigexd79coef.txt is an example file that holds the tuner filter coefficients for the embedded tuner of the firmware/download of gigexd_79d2.rpd. This file serves as an example of how users would setup a file of coefficients of their own creation to be downloaded to the GIGExD tuner FIR Filter when programmed with the download gigexd_79d2.rpd

initgigexd, gigexdr2.jic – Production testing and programming files. DO NOT ATTEMPT to use these files.

The C program gigexd.c provides the user with three tools.

1) Ability to reprogram the GIGExD firmware with new updates.

2) Ability to reconfigure the GIGExD with new configuration settings for different data manipulation.

3) The ability to acquire data to the host.

The gigexd.c program can be built under Linux by using the build script in buildgigexd.lnx.

ALL terminal line commands listed in this document are performed in the directory gigexdv*A*_*B unless otherwise specified*.

To build the gigexd.c source into the executable *gigexd* type:

>>**source buildgigexd.lnx <return>**

The source file gigexd.c is compiled and the executable *gigexd* is produced.

3.2. The gigexd Executable

Once built the executable gigexd provides several commands for the user. At the command line type:

>> **./gigexd <return>**

The program returns a listing of available commands and their syntax:

```
gigexdv1_8> ./gigexd
Syntax: gigexd <Operation Type> <Host IP Addr> <GIGEXD IP Addr>  <Path & Filename> <MC Address <file size>

               <Operation Type>       = program, prgclrcfg, flashcfg, rtcfg, query, qrywcfg, acquire, acqwtc, or
acqall
               <Host IP Addr>         = Dot Notation Of Host Machine IP Addr, Ex. 192.9.200.110
               <GIGEXD IP Addr>       = Dot Notation Of GIGEXD IP Addr, Ex. 192.168.200.1
               <path & filename>      = Path & Filename Of Program or Config File(OPTIONAL) or Data File(MANDATORY)
               <MC Address>           = Multicast Address Of GiGExD To Acquire Data From-OPTIONAL For Prog or Config
               <file size>            = Size Of File To Acquire Data (In KBytes)-OPTIONAL For Prog or Config

Typing: gigexd <Operation Type> help
Will display help for specific commands
gigexdv1_8>
```

Depending on the command used, the parameters to the executable *gigexd* can number between 3 and 6. **All** commands require the <command> , <Host IP Addr> and <GIGE IP Addr> parameters. The parameters <command> , <Host IP Addr> and <GIGE IP Addr> are the command to be executed, the host IP address and the GIGExD IP address, respectively. The parameters <path & filename>, <MC Address>, and <file size> are sometimes optional, depending on the command used. <path and filename> is the path and filename for the config text file or program download file or acquisition filename. <MC Address> and <file size> are used only for the acquire commands. For acquiring data from the GIGExD, <MC Address> is the multicast address from which the Host computer should receive data. <file size> is used for acquiring data from the GIGExD and represents the size of the file to be acquired in 1k blocks.

4. Commands

From the *./gigexd <return>* listing, the available commands are *program, prgclrcfg, flashcfg, rtcfg, query, qrywcfg, acquire, acqwtc, or acqall*.

To get syntax specific help on a command type:

>> **./gigexd <Command> help <return>**

4.1. *program* and *prgclrcfg* Commands

At the command line, help on the *program* or *prgclrcfg* commands is available by typing:

>> **./gigexd program help <return> OR >>./gigexd prgclrcfg help <return>**

Will display the following:

```
gigexdv1_8> ./gigexd program help

Syntax: gigexd program <Host IP Addr> <GIGEXD IP Addr>  <Path & Filename>

                  program OR prgclrcfg   = firware update utility (re-programs GIGEXD with new firmware)
                  <Host IP Addr>         = Dot Notation Of Host Machine IP Addr, Ex. 192.9.200.110
                  <GIGEXD IP Addr>       = Dot Notation Of GIGEXD IP Addr, Ex. 192.168.200.1
                  <path & filename>      = Path & Filename Of Programming File That Holds Firmware


        The 'program' operation is used to update the firmware of the GIGEXD.
For the GIGEXDR1, once programming is finished, the unit must be power cycled for the
new firmware to be loaded into the FPGA. The GIGEXDR2 does NOT need a power cycle after
programming.  If programming is successfully verified, the GIGEXDR2 will automatically reboot
with the new programming.
        Programming only overwrites the 'firmware' portion of flash memory.
All configuration that is stored in Flash (done with flashcfg operation) remains
intact. Since the configuration is intact, all user settings still remain and are
loaded at the power up sequence or reboot-along with the new firmware. Firmware
files for the GIGEXD can be identified in the zip sofware packages as having
the filename extension of .rpd


The 'prgclrcfg' operation performs the same function as the 'program' operation but
in addition erases all of the Flash configuration data that was stored using the 'flashcfg'
operation.  The 'prgclrcfg' operation is necessary as firmware/software evolve in complexity
and old 'flashcfg' values become obsolete.

IMPORTANT--Upon completion of the 'prgclrcfg' operaton, and a power cycle for the GIGEXDR1
or rebott for the GIGEXDR2, the GIGEXDR1 or GIGEXDR2 will default to IP address 192.9.200.105
gigexdv1_8>
```

For the GIGExD**R2, upon completion of the 'prgclrcfg' operation the new firmware will be AUTOMATICALLY reloaded without the need for a power cycle. Once reloaded, the** GIGExD**R2 will default to IP address 192.9.200.105**

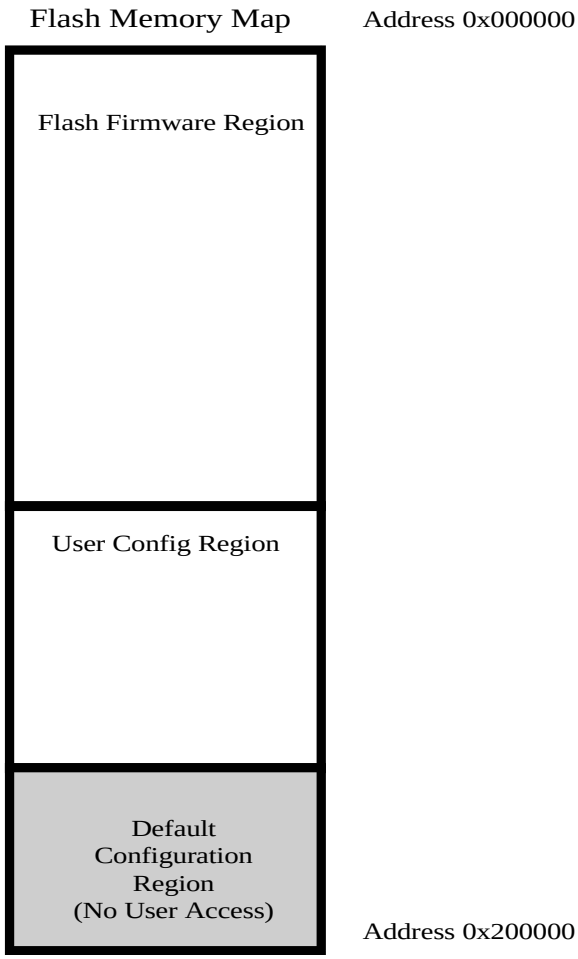
Example usage:

./gigexd program 192.9.200.110 192.9.200.105 ./gigexd_255d8.rpd <return>

As the previous software description explains, the *program* command downloads new firmware to the GIGExD. In essence, as new features are developed, this is the way in

which this new firmware with new features can be put into the GIGExD. The *program* command simply overwrites the the old firmware in the flash memory of the GIGExD. For the GIGExD**R2**, the new firmware takes effect upon completion of programming-a power cycle is not needed. The *program* command ONLY overwrites the firmware section of the flash memory. The configuration section of the flash memory (that was loaded using the *flashcfg* command) stays intact and retains its previous configuration settings-including previously programmed IP and multicast address. In other words, after using the *program* command the GIGExD should be accessed using its current IP address.

The *prgclrcfg* (program and clear configuration) command differs from the *program* command in that it overwrites the firmware section of the flash memory with new firmware AND clears ALL the configuration sections of flash memory and initializes the IP address of the GIGExD to 192.9.200.105. Upon completion of the *prgclrcfg* command, the new firmware is loaded and any previous IP address that was set using the *flashcfg* command is gone and the GIGExD now responds to the IP address of 192.9.200.105.



There are currently 4 downloads available to the user for the GIGExD**R2**-gigexdr2_79d2.rpd, gigexdr2_127d4.rpd, gigexdr2_255d8.rpd, and gigexdr2_1023d32.rpd. (These firmware downloads are numbered numbered 0, 1, 2, and 3, respectively.) The similarity of these downloads is they each are capable of raw wide band data transfer from an I/O source (A/D, ECL) to gigabit Ethernet at up to 100MBytes/Sec. For wide band data (no Digital Tuner use) the data formats that can be chosen are 8 or 16 bit data. For 100MBytes/Sec gigabit Ethernet limit this provides for 50MHz sampling using 16 samples and 100MHz sampling using 8 bit samples. Each of the 4 firmware downloads have the same wide band data rate limits. The difference of each is how they implement the embedded Digital Tuner FIR Filter.

For GIGExD**R2**:

Firmware 0 - gigexdr2_79d2.rpd provides a Tuner with a 79 Tap output FIR Filter and base/minimum decimation of 2.

Firmware 1 - gigexdr2_127d4.rpd provides a Tuner with a 127 Tap output FIR Filter and base/minimum decimation of 4.

Firmware 2 - gigexdr2_255d8.rpd provides a Tuner with a 255 Tap output FIR Filter and base/minimum decimation of 8.

Firmware 3 - gigexdr2_1023d32.rpd provides a Tuner with a 1023 Tap output FIR Filter and base/minimum decimation of 32

The user can choose a download that best suites the signal specific needs of their application. (At the time of delivery the GIGExD**R2** is programmed with gigexdr2_79d2.rpd) More taps will provide for a sharper "roll off" to the Fir Filter edges (smaller transition band) but will create a narrower Filter because of a larger minimum decimation after the Filter. Less taps will not have as sharp a roll off, but will allow for a wider Filter and thereby acquisition of greater bandwidth from a signal.

The user has 4 options for generating the FIR Filter coefficients. The options are FILE based, DEFAULT, COMPUTE, or OFF. OFF turns the Tuner capabilities off and allows for wide band data operation. FILE, DEFAULT, and COMPUTE turn the embedded Digital Tuner ON causing all data to pass through the tuner. The option FILE causes user supplied coefficients from a text file to be read and loaded into the FIR Filter. The option DEFAULT causes the *gigexd* program to compute the widest Filter possible based on rate, decimation, sample size etc. Finally, COMPUTE allows the user to specify a bandwidth and the *gigexd* program will compute the coefficients and download them to the Filter. All of these options are explained in detail in the **Config File** section.

Each of the Tuner implementations has the ability to set a user supplied center frequency and a decimation scalar. The decimation scalar is simply a factor between 1 and 255 that will be multiplied by the base/minimum decimation value of 2, 4, 8, or 32 for the respective downloads.

(Ex. Min decimation for gigexdr2_79d2.rpd is 2 * 1, for gigexdr2_127d4.rpd is 4 * 1, for gigexdr2_255d8.rpd is 8 * 1, and for gigexdr2_1023d32.rpd is 32 * 1)

(Ex. Max decimation for gigexdr2_79d2.rpd is 2 * 255, for gigexdr2_127d4.rpd is 4 * 255, and for gigexdr2_255d8.rpd is 8 * 255, for gigexdr2_1023d32.rpd is 32 * 255)

Tuner output to GIgabit Ethernet is either Complex or Real format. The user has the choice of selecting either complex 8 bit samples, complex 16 bit samples, 8 bit samples or 16 bit samples. Once again, selection of either of these is dependent upon bandwidth of the signal of interest, data rate, decimation, etc. and is at the user discretion (without violating the 100MBytes/Sec gigabit Ethernet Rate) for their particular application.

After the Tune, Filter, Decimate stages of the Digital Tuner the user then has the option of variable Gain. The variable gain is available from 0 to 30dB in 6dB steps.

Finally, all Tuned data, as with wide band data can be time code stamped using the IRIG interface before exiting via gigabit Ethernet.

4.2. *flashcfg* and *rtcfg* Commands

At the command line, help on the *flashcfg* command is available by typing:

>> **./gigexd flashcfg help <return>**

Will display the following:

```
gigexdv1_8> ./gigexd flashcfg help

Syntax: gigexd flashcfg <Host IP Addr> <GIGEXD IP Addr>  <Path & Filename>

                  flashcfg               = writes user cfg info to flash memory for retention between power cycles
                  <Host IP Addr>         = Dot Notation Of Host Machine IP Addr, Ex. 192.9.200.110
                  <GIGEXD IP Addr>       = Dot Notation Of GIGEXD IP Addr, Ex. 192.168.200.1
                  <path & filename>      = Path & Filename Of Config File That Holds Config Parameters

The 'flashcfg' operation is used to write user config info to flash memory of the GIGEXD.
This config info is used at 'power up' to configure GIGEXD parameters such as IP address,
Multicast Address, Data Rate, Sample Size, Sample Rate, and Digital Tuning parameters.
Writing config info to flash allows the user to remove power from the GIGEXD and upon
next power cycle retain configuration previous to the power cycle.
If the flash config area happens to be empty, (normally not the case, all GIGEXD units
are programmed with default config before shipment) the GIGEXD will default to IP address
192.9.200.105
gigexdv1_8>
```

Example usage:

./gigexd flashcfg 192.9.200.110 192.168.200.4 ./gigexdusr.cfg <return>

At the command line, help on the *rtcfg* command is available by typing:

```
gigexdv1_8> ./gigexd rtcfg help

Syntax: gigexd rtcfg <Host IP Addr> <GIGEXD IP Addr>  <Path & Filename>

                  rtcfg                  = writes user cfg info to RAM memory for immediate GIGEXD config
                  <Host IP Addr>         = Dot Notation Of Host Machine IP Addr, Ex. 192.9.200.110
                  <GIGEXD IP Addr>       = Dot Notation Of GIGEXD  IP Addr, Ex. 192.168.200.1
                  <path & filename>      = Path & Filename Of Config File That Holds Config Parameters

The 'rtcfg' operation is used to write user config info to RAM of the GIGEXD.
This config info is used to change parameters during testing of the GIGEXD
IP Address, Multicast Address, VLAN, and Module Type CANNOT be changed with 'rtcfg'.
All other parameters, Sample Rate, Sample Size, Digital Tuner Parameters, etc can
be changed 'on the fly'.  This allows for quick testing of the GIGEXD capabilities in
an 'engineering' environment.  Once the settings have been tested they can be committed
to flash memory on the GIGEXD (using the flashcfg operation) for permanent retention
through power cycles.  In short, 'rtcfg' provides for quick prototyping and analysis
by engineers. Removal of power erases the parameters set by rtcfg in memory and
power up causes parameters set by 'flashcfg' to be loaded and active.
gigexdv1_8>
```

Example usage:

./gigexd rtcfg 192.9.200.110 192.168.200.4 ./gigexdusr.cfg <return>

The commands *program* and *prgclrcfg* are used to download firmware to the GIGExD. The commands *flashcfg* (flash config) and *rtcfg* (realtime config) are used to configure

the firmware with user parameters for signal data manipulation. Configuration is either done to internal memory (volatile) or to external flash memory (non-volatile). The *rtcfg* command performs configuration on internal memory(RAM). The volatile nature of the internal memory means that configuration is NOT retained between power cycles. The *flashcfg* command performs configuration on the external flash memory. The non-volatile nature of the external flash means configuration is retained between power cycles. Because the on-board flash memory is external, and because the access speed for clearing and then writing flash memory, is slower then internal memory (RAM), the *flashcfg* command can take up to 90 seconds to complete. The rtcfg command is virtually instantaneous, and is provided for a quick way of testing various configuration settings in an engineering environment. Typically, various values are tried using the *rtcfg* command, once a set of values is determined to fit the user's needs then they are committed to flash memory using the *flashcfg* command for non-volatile storage.

As previously mentioned, new GIGExD units when delivered will arrive programmed with the firmware download gigexd_79d2.rpd and have the IP address 192.9.200.105. If a user does not wish to use the gigexd_79d2.rpd firmware download as it is delivered in the GIGExD, a user can change the delivered firmware in the GIGExD to some other desired download using the *program* or *prgclrcfg* commands. Once the desired firmware download is in place, the user can choose to change the IP address of the GIGExD by using the *flashcfg* command only. (The *rtcfg* command is not able to change the IP address or multicast Address of the GIGExD.) Referring back to the flash memory map diagram, each firmware download uses the same flash memory configuration area. One configuration area means that when the *flashcfg* command is issued, it is updating the values in this single flash configuration region.

Because the flash configuration values for each firmware download are unique, if a new firmware download is chosen for programming, a user MUST issue the *prgclrcfg* command to erase the old configuration values.

In summary, when delivered, the GIGExD has its IP address set equal to 192.9.200.105. A user may change this IP address (and other parameters) using the *flashcfg* command. If the user then changes firmware downloads by using the *prgclrcfg* command, the flash configuration area is erased. In erasing the configuration area, the GIGExD will again default back to IP address 192.9.200.105.

Example Listing of Configuration Text FIle gigexdusr.cfg (Included With zip Software Package)

```
192.9.200.104                                   -- A. GIGExD IP Address
224.1.2.104                                     -- B. GIGExD multicast Address
0                                               -- C. GIGExD VLAN Address
A2DR7                                           -- D. Attached Module Type
100000000                                       -- E. Data Clock Frequency
1                                               -- F. Input Data Clock Decimation Value
-16                                             -- G. Number Of Data Bits
0                                               -- H. Bit Shift Toward MSBit
TX                                              -- I. GiGExD Input/Output Mode
TC=ON                                           -- J. Insert TimeCode ON/OFF/SYN
TUNER=DEFAULT                                   -- K. Tuner Operation/Coefficients Dwnld
1e6                                             -- L.Coefficent Location Parameter/Value
2                                               -- M. Tuner Decimation Scalar
11000000                                        -- N. Center Frequency For Tuner
0                                               -- O. Tuner Gain in dB
Flags=None                                      -- P. User Flags For Special Ops

****Nothing Read From File Below This Line (Documentation)****
--A. GIGExD IP Address, Sets IP Address Of GIGExD For Communication, Programming, Configuration, Etc.
--B. GIGExD multicast Address, For TX/RX Operation, multicast Address To Send Data To (TX) or Receive Data From (RX)
        --Note This Address MUST Be Between 224.1.0.0 and 239.255.255.255
--C. GIGExD VLAN Address For VLAN Support, Set=0 For Non-VLAN Support & No 802.1Q Extensions, Otherwise VLAN Number
--D. Attached Module Type, A2DR7, A2DR9, or DR2D.
--E. Data Clock Frequency, Input Clock Frequency To GIGExD From Module (A2DR7,DR2D,etc)
--F. Input Clock Decimation Value, Decimates Incoming Data Clock Freq. From Module (Not Tuner Decimation) 1 <= valid <= 8
--G. Number Of Data Bits, Negative Number Means Complex, Ex -8=Complex 8 Bit Samples (Tuner Use Requires Complex Data Type)
--H. Bits To Shift Toward MSBit,LSBits Zero Filled, 0<=valid<=7 (Bit Shift Toward MSBit = Data x2 For Each Bit Shift)
--I. GiGExD Input/Output Mode, To Transmit(TX)-Send To MC Adr B. Or To Receive(RX), Get Data From MC Adr. B.
--J. For TX Operation, GIGExD Does TimeCode Insertion (No Spaces Around "="),OPTIONS TC=ON, TC=OFF, or TC=SYN
        - TC=ON Timecode inserted using IRIG and PPS signal inputs
        - TC=OFF No Timecode Insertion
        - TC=SYN Timecdoe set using flag TCSYNSEC=<value> from user, PPS signal used to increment seconds, IRIG signal unused
--K. OPTIONS: TUNER=OFF or DEFAULT or FILE or COMPUTE, TUNER=
        -- OFF Turns Off Tuner Brings In Wide Band Data
        -- DEFAULT Loads The Default Filter Coefficient Values From gigexd79coef.txt, gigexd127coef.txt, gigexd255coef.txt
        -- FILE Loads User Supplied Coefficients From File (Coefficient Format Is Unscaled Real Number (Floating Point))
        -- COMPUTE Computes Coefficients For Filter Width From Value Specified For K.
--L. For DEFAULT or OFF This Field Is "Don't Care"
        -- For FILE, Path & Filename For Location Of Coef For Loading
        -- For COMPUTE, This Is A Real Value (In MHz) For Width Of Filter Ex. 0.500e6 = 500KHz Wide Filter
--M.  Tuner Decimation Scalar, Tuner Base Decimation Multiply By This Number = Total Decimation, 1 <= valid <= 255
--N. Center Frequency To Which Digital Tuner Will Tune
--O. Tuner Gain in dB, 0<=valid<=30, In Increments Of 6 dB, ex. 0,6,12,18...
--P. User Flags For Special Ops, Comma Separated (NO Spaces) Ex. Flags=LEND,CLKI  For Little Endian Data & Clock Invert
```

In each of the command line examples above, both the *flashcfg* and the *rtcfg* commands retrieve user values from a configuration file (gigexdusr.cfg), then "packetize" the configuration data contained in the file and then send the packetized data to the GIGExD via the Host machine Ethernet interface. When the configuration file (gigexdusr.cfg) is read by the *gigexd* program, error checking is done on user values as it pertains to violations of the GIGExD **HARDWARE** capabilities and a report is listed. After the report, the user is queried with a "y" or "n" (yes or no) response for whether the parameters read should be committed to the GIGExD internal memory of external flash. The gigexd program does not contain several "layers" of syntax checking. Therefore, it is IMPERATIVE that the user enter configuration file values carefully and check the report that is listed before configuration is sent/committed to the GIGExD. If the configuration in the report does not match what was intended by the user, the user should answer "n" for a reply of "no" so that the values are not sent to the GIGExD and the GIGExD is not configured with the faulty values.

Below is an example of the information that is returned to a user when the *flashcfg* command is executed using the gigexdusr.cfg file that is included in the zip package as an example configuration file.

```
>> ./gigexd flashcfg 192.9.200.110 192.9.200.105 ./gigexdusr.cfg <return>

Start Communications With GigE...Done
Starting Operation: FLASHCFG
Operating On User Config Space
Config File ./gigexdusr.cfg Info :
        GigE IP Addr = 192.9.200.104
        GigE MC Addr = 224.1.2.104
        VLAN = 0
        GigE Module Type = A2DR7
        Data Rate = 100000000
        Input Decimation = 1
        Number Of Bits = -16
        Number Of Bits To Shift = 0
        GIGE Transfer Direction = TX
        Timecode Generation = ON
        Tuner = ON, Default Coefficients Filter Set To First Nyquist Zone
        Decimation Scalar = 2
        Center Frequency = 12000000
        Tuner Gain = 0
        Flags = None
        Firmware  Version = 1.3

You Are About To Re-Configure The GigExD Internal Settings To The Above Configuration!!
These Settings Will Overwrite Previous Configuration Settings!!
Contnue? ('y' or 'n') y

Turning Off Incoming Data Stream ...Done
Reading Configuration Section (30 Secs)
Clearing EPROM Cfg Sector...Done
Programming Cfg Area Of FLASH (45 Secs)
Restarting Data With New Cfg Parameters ... Done
Config Complete-New Config Parmeters Take Effect Immediately!!
```

Note that the *flashcfg* command is sent to a GIGExD with the IP address of **192.9.200.105** but after the *flashcfg* command the GIGExD has a new IP address and can only be accessed via the NEW IP address of **192.9.200.104**.

4.3. Configuration File Syntax

From the listing above, there are 15 parameters (A-O) that are used to control/configure the GIGExD for user specific operation. The first 4 items (A-D) IP address, multicast address, VLAN address and Attached Module Type are ignored when using the *rtcfg* command. In other words, changing any of these values and executing a "realtime" configuration (rtcfg) command will not change these values within the GIGExD. In order to change these values in the GIGExD, the *flashcfg* command must be used. The last 11 configuration values (E-O) are used for both *rtcfg* and *flashcfg* commands.

The syntax for the configuration file gigexdusr.cfg is fairly straight forward. All 14 values appear 1 per line LEFT justified. Where an equal sign ("=") appears there should be NO spaces between the values on either side of the equal sign. Items A and B are Network addresses and should be written as such-"dot notation". Items E, K and M may be written in exponential notation or as an integer. All of the numbers should be written as integers.

It is in the user's "best interest" to take the config gigexdusr.cfg that comes in the zip package and modify it for their use rather than creating their own. The *rtcfg* and *flashcfg* commands take as their last entry <path and filename>. Therefore, the file gigexdusr.cfg can be copied to unique names, with each file containing different configuration values (ie gigexdusr_wideband.cfg, gigexdusr_tuner.cfg, etc) and passed in to the *rtcfg* and *flashcfg* commands as the last parameter <path and filename>.

4.4. Configuration File Parameter Description

In the configuration file gigexdusr.cfg which comes with the software zip package there are 15 values. These values are listed as items A through O At the bottom of the config file gigexdusr.cfg is a description of each of these parameters for quick reference when editing the file. Below is a more detailed explanation for each parameter

A. IP Address.

This is the IP address that the GIGExD responds to for "pings" configuration and programming. Upon shipment of the GIGExD, rumeL, Inc defaults this address to 192.9.200.105. Using the *flashcfg* command, a user can change this address to any valid IP address they desire. In the example configuration file gigexdusr.cfg, included with the zip package, this address (as in the example above) has been changed to 192.9.200.104. After editing and saving the file, run the gigexd program with the *flashcfg* command to reset the IP address (and all other parameters) to those listed in the configuration file.

B. multicast Address

This is the address to which the GIGExD sends the UDP/SDDS packets when in Tx mode. In Rx mode, this is the address the GIGExD listens to when receiving data. (Rx mode is currently under development and won't be discussed in this document.) The valid range of multicast addresses include the IP address range of 224.1.0.0 to 239.255.255.255. (224.0.x.x are in the valid range for multicast addresses but are reserved.) Therefore, valid values for this configuration file parameter are 224.1.0.0 to 239.255.255.255. The GIGExD sends out data to multicast addresses because multicast addresses have the unique ability to be "listened to" by more than 1 Host machine. Therefore, a practical application would be one or more GIGExDs with unique multicast addresses plugged into a network fabric (routers) and tens or hundreds of Host computers selectively listening to a GIGExD by choosing the appropriate multicast address

C. VLAN Address

The SDDS specification includes 802.1Q VLAN extensions to "group" multicast data into VLANs for higher throughput through router networks. As of version 2.0 setting the VLAN configuration parameter to a non-zero value will add the 802.1Q VLAN extensions to the multicast packets that go out. The VLAN field in each data output packet will be set to the value specified for the VLAN configuration parameter.

Setting the VLAN configuration parameter to 0 will disable the 802.1Q extensions in each data output packet.

Finally, the GIGEXD will now receive configuration or programming packets with 802.1Q VLAN extensions or without-either type is valid. In summary, the GIGEXD can be connected to a network that has 802.1Q extension or does not. It can be programmed or configured with either type of network packets. This feature allows the user to move the GIGEXD between VLAN or non-VLAN networks and have the same functionality-regardless of the network type. The VLAN configuration parameter controls only whether the outgoing Multicast data is 802.1Q enabled or disabled. The GIGEXD command and control may be done over either VLAN or NON-VLAN networks.

D. Module Type

The current supported module types are A2DR7 (A/D Conversion), A2DR9, DR2D, and Sonet. The DR2D is a Differential Receiver module-it will acquire any differential signal. Examples are Differential ECL, Differential LVECL, Differential Positive ECL (PECL), Differential Positive LVECL, LVDS, Differential TLL, etc. Therefore, this configuration parameter should be set to either A2DR7, A2DR9, DR2D, or SNTXD depending on the type of module attached.

E. Data Clock Frequency

This is the frequency of the Input clock to the I/O module attached to the GIGExD. For SONET this value should be set to 155.52e6 for OC3/STM1 or 622.08e6 for

OC12/STM4-which are the line rates of corresponding signals. For the DR2D module, the Data Clock Frequency parameter should be set to the clock frequency of the digital clock line.

For more information on the I/O modules and their options see the section titled **I/O modules**.

F. Input Data Clock Decimation Value

As briefly discussed for configuration parameter E. (Data Clock Frequency) configuration parameter F. is the Input Data Clock Decimation Value. Its valid ranges are the values between 1 and 8. This value is used to effectively divide down the sample rate of the data by its value. (Effective Sample Rate = Clock Rate E/Input Decimation F). For example, if the Data Clock Frequency is set to 100MHz then setting this configuration value (Input Data Clock Decimation) to 1 will produce a data rate of 100MHz/1 = 100MHz- effectively turning off any decimation. Setting this value to 2 will cause every other sample to be acquired or give an effective sample rate of 100MHz/2 = 50MHz. The value set to 3 would cause 1 sample to be acquired and 2 samples discarded for an effective rate of 100MHz/3 = 33.3MHz.--and so on for other values of the Input Clock Decimation up to the value of 8. The configuration parameter is highly useful when using the A2DR7 I/O module and high speed PLL for clocking. It allows the user to set the frequency at 90-100MHz and then divide down to rate that may be more favorable for signal analysis. Finally, this configuration parameter works to divide down the input clock rate for any of the I/O modules that attach to the GIGExD.

G. Number Of Data Bits

This configuration parameter describes the number of bits for packetization into gigabit Ethernet packets. The GIGExD has a maximum throughput of 100MBytes/Sec. With this maximum sustainable throughput of 100MBytes/Sec, a user must choose this parameter (Number Of Data Bits) other signal parameters so as not to violate the maximum throughput.

For wide band operation (no tuner use) this value needs to be set to either 8 or 16. Setting the value for 8 bit samples will allow for a maximum Data Clock Frequency (E) of 100MHz. Setting this value to 16 will allow for a maximum Data Clock Frequency (E) of 50MHz.

4.5. Real Mode Tuner

As of version 1.4 (and higher) of the GIGExD firmware and software the GIGExD supports real mode tuner values. Therefore, for Digital Tuner Use this configuration parameter must be set to 8, 16 -8 or -16. -8 represents Complex 8 bit samples. -16 represents Complex 16 bit samples. Choosing positive 8 or 16 will place the tuner in real mode. Data will be output as only real values as 8 or 16 bit. Based on Data Clock Frequency(E), Input Clock Decimation Value (F), programmed tuner base decimation (2,4, or 8) and the Tuner Decimation (L) a user can choose Complex 8 bit (effective 2

bytes per sample) or Complex 16 bits (effective 4 bytes per sample) so as not to violate the 100MBytes/Sec maximum throughput of the GIGExD.

Finally, when the *rtcfg* or *flashcfg* commands are run, the *gigexd* program computes the effective throughput based on user values in the configuration text file. If the calculated effective throughput is above the limit of 100MBytes/Sec, an error message is posted and the configuration is halted before being sent to the GIGExD.

H. Bit Shift

The appropriate values for this configuration parameter are 0 to 7. As data enters the GIGEXD from the I/O module, value of this parameter will cause of bit shift of the value toward the MSBit. Each bit shift toward the MSBit equals a multiply by 2 of the value. Note that the data entering is considered to be 2's complement data. Too high a number of bit shifts could change the data value from positive to negative or visa versa. This configuration parameter is useful for when the A/D converter I/O module that is attached to the GIGEXD is not fully loaded. All bits in the LSB portion of the shifted data are zero filled.

I. GIGExD Input/Output Mode

The appropriate value for this configuration parameter is **TX** only. TX instructs the GIGExD to take in data from the I/O module and then transmit (TX) the data as multicast UDP/SDDS packets. Future revisions of the GIGExD firmware will allow for receive (RX) operations where the GIGExD receives multicast UDP/SDDS packets and then sends the data out via an output module. In summary, this value should ALWAYS be set to TX.

J. Insert Time Code (ON/OFF/SYN)

The appropriate value for this configuration parameter are either **TC=ON, TC=OFF, or TC=SYN**.

Setting **TC=ON**

When a valid IRIG and PPS signal are present, setting this parameter to TC=ON will cause the GIGExD to insert valid time code into the UDP/SDDS gigabit Ethernet output. The GIGExD inserts time code into the UDP/SDDS stream at every occurrence of the PPS signal and when this configuration parameter is set to TC=ON.

Some Key Points On Time Code Insertion:

--If a valid PPS signal is NOT present and TC=ON is set, time code will NOT be inserted.

--If a valid PPS signal is present, but a valid IRIG signal is not present and TC=ON is set, time code will not be inserted.

In short, a valid PPS and IRIG signal must be present for time code to be inserted with the parameter TC=ON. The value of the time code will be constantly updated by the time code acquired from the IRIG signal input.

Setting **TC=OFF**

Setting TC=OFF will cause the GIGExD to disregard the PPS and IRIG signal and not insert any time code.

Setting **TC=SYN**

Setting TC=SYN will cause the input to the IRIG port to be ignored. The PPS signal is still needed but time is inserted base upon a synthetic start value. The synthetic start value is specified by using the flag **TCSYNSEC=<value>** Where <value> represents the number of seconds from the start of year (see section P. Flags for valid values). If the flag TCSYNSEC=<value> is omitted from the flag string a warning message will be printed and an assumed start value of 0 seconds from the start of year will be used. The use of TC=SYN and flag TCSYNSEC=<value> is for users who have available an accurate PPS signal for input to the GIGEXD but do NOT have the corresponding IRIG signal from a GPS source. This time code insertion mode allows the user sample time stamping accuracy without the need for the IRIG input that may not be present with some GPS receivers.

K. Tuner Operation/Coefficients Download

This configuration parameter has 4 options for values:

1. **TUNER=OFF**
2. **TUNER=DEFAULT**
3. **TUNER=COMPUTE**
4. **TUNER=FILE**

Setting this configuration parameter to Option 1, **TUNER=OFF**, turns off use of the embedded Digital Tuner and sets the GIGExD to send only wide band data that does not pass through the tuner.

Setting this configuration parameter to Option 2, **TUNER=DEFAULT**, turns on use of the embedded Digital Tuner and causes coefficients to be downloaded to the tuner's FIR Filter which are calculated for the **widest bandwidth** value for the 1st Nyquist zone.

For example, if:

(Configuration Parameter E) Input Clock Frequency = 100e6

(Configuration Parameter F) Input Data Clock Decimation Value = 2

(Configuration Parameter L) Tuner Decimation Scalar = 4

The current firmware programmed into the GIGExD**R2** is gigexdr2_255d8.rpd

Then the FIR Filter width will be. 781.25kHz

The equation used to calculate the widest bandwidth for the FIR Filter is:

Filter Width = clock frequency / (Input clock decim * Firmware Filter Base Decimation * Decimation Scalar * 2)

Setting this configuration parameter to Option 3, **TUNER=COMPUTE** turns on use of the embedded Digital Tuner and causes the *gigexd* program to compute the coefficients using the configuration parameter L (Coefficient Location/Value). For TUNER=COMPUTE, the *gigexd* program reads the next parameter L in the file as the value to compute for the FIR Filter width. For example, setting configuration parameter L. (Coefficient Location/Value) to 1.0e6 will produce a low pass FIR Filter with a width of 0 to 1MHz (1e6). An error is reported if the value for L is greater than Nyquist for the signal rate.

Setting this configuration parameter to Option 4, **TUNER=FILE** turns on use of the embedded Digital Tuner and causes the *gigexd* program to retrieve the coefficients from a text file that is listed for configuration parameter L-Coefficient Location/Value. For example, if configuration parameter L was set to */data11/home/coeffs.txt* then the coefficients for the FIR Filter would be read from the file */data11/home/coeffs.txt*, checked for symmetry, scaled and downloaded to the FIR Filter.

The file *gigexd79coef.txt* is provided in the zip software package as a reference for user file based coefficients.

Here are the rules for coefficient syntax as they appear in the text file.

1.) The coefficients MUST be symmetrical.

2.) The coefficients MUST appear 1 per line in the file.

3.) The coefficients MUST be in floating point/exponential format (Example: See gigexd79coef.txt in zip package)

4.) The coefficients must NOT be scaled. The *gigexd* program will scale them for use in the FIR Filter.

5.) The number of coefficients in the file must match the number that are needed for the particular firmware download currently programmed into the GIGExD.

For firmware gigexdr2_79d2.prm the number is 79.

For firmware gigexdr2_127d4.prm the number is 127.

For firmware gigexdr2_255d8.prm the number is 255.

For firmware gigexdr2_1023d32.prm the number is 1023

L. Coefficient Location/Value

This configuration parameter is used in conjunction with configuration parameter K. If K is set to TUNER=OFF or TUNER=DEFAULT then this configuration parameter is a "don't care" as to it's value. (A string of some characters must be present but when read the value is discarded.)

If configuration parameter K is TUNER=COMPUTE then the configuration parameter L is a value in exponential format that represents the width of the FIR Filter in Hz. Ex. 1.0e6, 500e3, 0.5e6, etc.

If configuration parameter K is TUNER=FILE then the configuration parameter L is a string that is the path and a filename for a text file from which the coefficients for the FIR Filter will be read. See the explanation for configuration parameter K for file content syntax when TUNER=FILE is set for configuration parameter K.

M. Tuner Decimation Scalar

From the ***program* and *prgclrcfg* Commands** section there are currently 3 firmware downloads that can be programmed into the GIGExD. These downloads and their FIR Filter tap count and base/minimum decimation values are listed again below for reference.

Firmware 0 - gigexdr2_79d2.rpd provides a Tuner with a 79 Tap output FIR Filter and base/minimum decimation of 2.

Firmware 1 - gigexdr2_127d4.rpd provides a Tuner with a 127 Tap output FIR Filter and base/minimum decimation of 4.

Firmware 2 - gigexdr2_255d8.rpd provides a Tuner with a 255 Tap output FIR Filter and base/minimum decimation of 8.

Firmware 3 - gigexdr2_1023d32.rpd provides a Tuner with a 1023 Tap output FIR Filter and base/minimum decimation of 32.

Depending on the firmware that is currently programmed into the GIGExD the base/minimum decimation can be 2, 4,8, or 32.

This configuration parameter M (Tuner Decimation Scalar) is a scalar or multiplier for the base decimation that is programmed into the GIGExD. This configuration value is to be listed as an integer with a valid range of 1 to 255. This value is multiplied by the base/minimum decimation of the current download to produce a total decimation value after the FIR Filter.

As an example, if the GIGExD is programmed with gigexd_79d2.rpd, and an overall decimation of 32 is desired after the FIR Filter, then this configuration parameter M should be set to 16.

(Total Tuner Decimation = Base/Minimum Decimation * Tuner Decimation Scalar, Total Tuner Decimation = 2 * 16).

As a 2nd example, if the GIGExD is programmed with gigexd_255d8.rpd, and an overall decimation of 32 is desired after the FIR Filter then this configuration parameter M should be set to 4. (Total Tuner Decimation = Base/Minimum Decimation * Tuner Decimation Scalar, Total Tuner Decimation = 8 * 4)

N. Center Frequency For Tuner

When the configuration parameter K is set to any value EXCEPT for TUNER=OFF, this configuration parameter N (Center Frequency For Tuner) represents the frequency that the embedded Digital Tuner will be centered. This configuration parameter should be listed in exponential notation. EX. 11e6 12.2e6 13.003e6, etc.

O. Tuner Gain (in dB)

This configuration parameter controls the Digital Gain stage after the Tuner. Valid values for this configuration parameter are 0 (no gain), 6, 12, 18, 24 or 30. These values represent the amount of gain (in dB) added to the data after the tuner. Depending on the coefficients used, there is a unique situation of scaling coefficients that may lead to the user not being able to increase the gain by the amount set for this configuration parameter when this parameter is set near or at its maximum value of 30dB. If this case occurs, the *gigexd* program will display a message to the user as to how much gain was available and add the most gain possible. This situation is not reported as an error but simply as a warning that what the user requested was unable to be implemented.

P. Flags

The Flags parameter is used to enable special features of the GIGExD. Some of these features pertain to only certain I/O modules and are ignored for others. When multiple flags are used, they should be listed with no spaces and comma separated (Ex. Flags=LEND,CLKI). Below are a list of flags that are currently supported and their meanings.

LEND-Little-Endian, 16 bit data is sent via big-endian format as specified in the SDDS specification. This is problematic for users that don't want to acquire data then have to swap bytes to convert to little-endian. By using the LEND flag, the GIGExD will send the data in little-endian format. (The SDDS header will still be big-endian but the data portion will be little-endian) This flag is ignored for any data width that is not 16 bit or complex 16 bit format.

CLKI-Invert Input Clock, As of version 1.7 this flag is available. CLKI is used with the DR2D module only. As the default, the GIGExD acquires data on the rising edge of a

clock input. Using this flag will cause the GIGExD to acquire data on the falling edge. This flag should never be used with the A2DR7 or A2DR9 modules.

MUXCLK=P,MUXCLK=N,MUXCLK=C-For use with the GIGExD**R2** and the A2DR7 module only. These flags allow a user to choose the clock source of the new A2DR7. Previously for the GIGExD**R1** and A2DR7 combination, the clock source was manually chosen with jumper settings on the A2DR7. With the GIGExD**R2**, the A2DR7 or A2DR9 clock source can be chosen in software by using these flags. See the **I/O Modules** section for a description of how to use the MUXCLK flags to control the GIGExD**R2** A2DR7 or A2DR9 clock selection.

XTCDEL=*value*-External Time Code delay. The value assigned to XTCDEL represents a correction for external signal path delays in 250-picosecond increments. (value=400=100nSec of delay) This number may be positive or negative and in the range of -4096<=value<=4096. Positive values cause an offset to be added to the time stamp of packets-in essence making the samples appear as though they occurred later in time. Negative values cause an offset to be subtracted from the time stamp of the packets-making the samples appear as though they happened earlier in time.

A positive value of XTCDEL is useful when the IRIG signals relative to the Data source are externally delayed before entering the GIGExD. In lab situations, this will happen if there are delays in the IRIG signal routing equipment. A negative value of XTCDEL is useful when the Data source is delayed relative to the IRIG timing signals. For instance when using an A2DR7, a negative value can correct for external Analog delays before the analog data signal is digitized.

A2DGAIN=*value* For Use With GIGEXDR2 & A2DR9/10/13 Modules ONLY
Syntax: A2DGAIN=<value> where <value> represents the A2D GAIN in dB.
The acceptable range for <value> is: -4 <= <value> <= 20. The A2DR9 I/O Module has an on-board variable gain front end. Setting the amount of gain (or attenuation) is done via the use of this flag. Without use of this flag, the front end gain is set to 0 dB.

TEST=<Test Value> For Use With GIGEXDR2 Only

Syntax: TEST=<Test Value> where <test value> is **SINE** or **RAMP**

The TEST flag with values of SINE or RAMP causes the GIGEXDR2 to ignore all incoming data and instead output either a sine wave or ramp.

The flag TEST=RAMP will output a 16 bit ramp at the sample rate of the incoming clock from the I/O module attached to the GIGEXDR2. Therefore, if an A/D module is attached, a user can configure the clock rate using the A/D module and output a ramp instead of the analog converted data. Note that when the flag TEST=RAMP is present the user must configure the output as 16 bit data in raw mode. Raw (or wideband) data at 8 bits is not permitted. Tuner and timecode usage is permitted.

TEST=SINE will output a sine wave at a frequency of 1/32 of the sample rate. As with the flag TEST=RAMP, the clock source is from received from the I/O module that is connected to the GIGEXDR2. Raw (wideband) 8 bit and 16 bit data are both permitted. Also, the use of the tuner and timecode are permitted.

TCSYNSEC=<value> For Use With GIGEXDR2 Only

Syntax: TCSYNSEC=<value> where <value> is 0 <= value <= 31535999

The flag TCSYNSEC=<value> is used in conjunction with setting J. of TC=SYN. This flag represents the synthetic start time of time code from the start of year in seconds. The maximum value is the maximum number of seconds in a year (31536000) minus 1. At 31536000 the time code count rolls over to zero to emulate the start of a new year. A PPS input is needed to increment the synthetic count by one until the value of 31535999 is reached. Upon the next subsequent PPS signal time code rolls over to 0-start of new year.

*****New Feature*****

TRUNC For Use With GIGEXDR2 with accompanying DR2D Module Only

Syntax: TRUNC

The flag TRUNC is to be used with the a GIGEXD/DR2D combination only. This flag only has meaning when a user wants to input 8 bits data from the DR2D's 16 bit data interface. The default operation (TRUNC flag not present) is to round the upper most 8 bits of data based upon a whole 16 bit word. The rounded 8 bits are then acquired. The lower 8 bits are discarded. The TRUNC flag prevents the rounding and simply truncates the 16 bits of input data to 8 bits. Once the Least Significant 8 bits are truncated, the upper 8 bits are acquired.

TCSECOFF=<value> For Use With GIGEXDR2 Only *****New Feature*****

Syntax: TCSECOFF=<value> where <value> is -128 <= value <= 128

The flag TCSECOFF=<value> is used to add <value> seconds to the incoming IRIG timecode (Setting <value> to a negative number will subtract seconds.) This flag's intended application is for “leap second” applications that may occur during a year.

Typing:

>>./gigexd flags help <return>

at the command line will give a list of available flags, a description and usage.

4.6. *query* and qryw*cfg* Commands

At the command line, help on the *query* or *qrywcfg* commands is available by typing:

>> **./gigexd query help <return> OR ./gigexd qrywcfg help <return>**

Will display the following:

```
gigexdv1_8> ./gigexd query help

Syntax: gigexd query <Host IP Addr> <GIGEXD IP Addr>

               query OR qrywcfg    = Flash Configuration Query Utility
               <Host IP Addr>      = Dot Notation Of Host Machine IP Addr, Ex. 192.9.200.110
               <GIGEXD IP Addr>    = Dot Notation Of GIGEXD IP Addr, Ex. 192.168.200.1


The 'query' operation is used retrieve the Firmware version number and the
FLASH configuration settings for review by a user. The information displayed gives
the last configuration that was written to the flash memory using the 'flashcfg'
operation for the CURRENT download of firmware.


The 'qrywcfg' operation gives the same query info as listed above for the 'query'
command.  In addition, once the query information has been reported to the user the
'qrywcfg' command will reload the configuration settings that were reported.
gigexdv1_8>
```

The *query* command displays the current firmware download, the firmware version number, tuner capabilities of the current firmware download and the contents of the current configuration region back to the user. In other words, it is a way of reading back what was written to the flash configuration area using the *flashcfg* command. If the embedded Digital Tuner was configured to be in use, the *query* command will write coefficients of the FIR Filter to the file gigexdquerycoef.txt.

The *qrywcfg* (query with configuration) command will cause the same information to be reported back to the user, but once reported, the GIGExD will be instructed to re-load configuration information. The use for this command will be made apparent in the "Catch-22" section of this document.

Below is an example of the information that is returned to a user when the *query* or *qrywcfg* command is executed. The example below is for a GIGExD that was flash configured with the *flashcfg* command using the gigexdusr.cfg file in the zip package.

>> **./gigexd query 192.9.200.110 192.9.200.104 <return>**

```
Start Communications With GigE...Done
Starting Operation: QUERY
*********************
* GIGExD Query Info *
*********************
GIGExD Hardware Type:
      GIGExDR2
Firmware Download Info:
      Current Firmware Download - gigexdr2_255d8.rpd
      Current Firmware Download Number = 2
      Firmware  Version = 2.13
      Vlan Disabled In Firmware
Tuner Capabilities:
      Base Decimation Value = 8
      Number Of Filter Taps = 255
User Flash Config Values:
      GIGE IP Address = 192.9.200.104
      GIGE MC Address = 224.1.2.104
      VLAN = 0
      Sample Rate = 100000000
      Input Decimation = 1
      Number Of Bits = -16
      Number Of Bits To Shift = 0
      Flags = None
      TimeCode Insertion = ON
      Tuner Enable = ON
      Tuner Coefficients = DEFAULT
      Tuner Coefficients Written To File gigexdquerycoef.txt
      Decimation Multiplier = 2
      Requested Center Frequency  = 12000000
      Actual    Center Frequency  = 12000000
      Tuner Gain (dB)  = 0
*********************
```

Note that the information returned applies to ONLY the information stored in the current configuration region. In this case, as specified by the *query* command results above, the firmware download is #2 (gigexdr2_255d8.rpd).

4.7. *a2dgain* Command

At the command line, help on the a2dgain command is available by typing:

>> ./gigexd a2dgain help <return>

Will display the following:

```
  Syntax: gigexd a2dgain <Host IP Addr> <GIGEXD IP Addr> <gain>


             a2dgain             = A2DR9/10/13 Realtime Gain Manipulation Utility

             <Host IP Addr>      = Dot Notation Of Host Machine IP Addr, Ex. 192.9.200.110

             <GIGEXD IP Addr>    = Dot Notation Of GIGEXD IP Addr, Ex. 192.168.200.1

             <gain>              = A2DR9/10/13 Gain (In dB) Value-Integer Format

The 'a2dgain' operation is used to set, in realtime, the front end gain of an A2DR9/10/13
module connected to the GIGEXD.  The operation is a 'set and forget' command with no
response back to the host of command completion. All other commands, query, flashcfg, etc.
operate with a response back to the host-thereby stopping data. from the GIGEXD to send
the command response. The 'a2dgain' operation allows the user to set the A2DR9/10/13 front
end gain without stopping data output from the GIGEXD.
```

The *a2dgain* command allows a user to set the front end gain of an A2DR9/10/13 module without stopping the GIGEXDR2 from outputting data. The *a2dgain* command is a “set and forget” operation. In other words, there is no response back to the host machine whether the gain setting was received by the GIGEXD. ALL other commands, flashcfg, query, prgclrcfg, etc. receive a response back to the host that the commands have been received and implemented. The process of the response back to the host, halts an ongoing output of data in which the GIGEXD may have actively been involved. The *a2dgain* command requires no response back and therefore does not halt the flow of data from the GIGEXD. This allows a user to actively manipulate the front end gain of the A2DR9/10/13 module attached to the GIGEXDR2 without interfering with any active data flow.

a2dgain Command (IMPORTANT) NOTE:

The above statements are true if the host computer running the GIGEXD software (gigexd.c executable) has an updated routing table with the MAC (hardware) address of the GIGEXDR2. In short, if the computer has not “talked/communicated” with the GIGEXDR2 since power up, it will not know the MAC address of the GIGEXDR2. To obtain this address, the host computer, without user knowledge, will issue a “ping” of the GIGEXDR2. The ping of the GIGEXDR2 will momentarily stop the GIGEXDR2 data transfer until the host computer is sent a reply to the previously issued ping. The situation arises that if a new host computer is brought on line to issue the *a2dgain* command the resulting ping of GIGEXDR2, before the issuance of the *a2dgain* command will halt data until the host computer is sent a response to the ping. Once

the host IP table is updated with the MAC address of the GIGEXDR2 all subsequent *a2dgain* commands will execute without the need for the host to issue a ping. But, it is important to note that the first *a2dgain* command will halt data due to the ping response. If the host computer has already contacted the GIGEXDR2 with any other commands, query, flashcfg, etc, the IP tables will already be updated and the first *a2dgain* command will execute without the need for a ping and the subsequent momentary stoppage of data. In short, this situation of a ping to the GIGEXDR2 will only occur upon the first communications from a host computer to the GIGEXDR2. And, it only affects the *a2dgain* command if the very first communication from the host to the GIGEXDR2 is using the *a2dgain* command.

4.8.*acquire, acqall,* and *acqwtc* Commands

At the command line, help on the *acquire*, *acqwtc*, or *acqall* commands is available by typing:

>> **./gigexd acquire help <return> OR ./gigexd acqwtc help <return> OR ./gigexd acqall help <return>**

Will display the following:

```
gigexdv1_8> ./gigexd acquire help

Syntax: gigexd acquire <Host IP Addr> <GIGEXD IP Addr>  <Path & Filename> <MC Address> <file size>

               acquire OR acqwtc OR acqall  = acquire data from the GIGEXD and store to file
               <Host IP Addr>               = Dot Notation Of Host Machine IP Addr, Ex. 192.9.200.110
               <GIGEXD IP Addr>             = Dot Notation Of GIGEXD  IP Addr, Ex. 192.168.200.1
               <path & filename>            = Path & Filename Data File Where Data is to be Stored
               <MC Address>                 = Multicast Address Of GiGExD To Acquire Data
               <file size>                  = Size Of File To Acquire Data (In KBytes)

The 'acquire' operation opens a multicast receive connection (sends out a Join Request)
Data acquisition is to file and for a duration as specified by the user. Only the SDDS Data
portion is acquired. The SDDS header, including timecode information, is discarded.

The 'acqall' operation opens a multicast receive connection (sends out a Join Request)
Data acquisition is to file and for a duration as specified by the user. All of the SDDS/UDP
packet is acquired (56 Bytes Header + 1024 Bytes Data = 1080 Bytes Per Packet)
Unlike the 'acquire' or 'acqwtc' operations the 'acqall' operation does NOT Endian convert
the SDDS header (timecode) or SDDS data. Endian conversion must be performed by a user
application on the data written to file

The 'acqwtc' operation adds 16 bytes of Timecode info before each 1KByte of data.
to the acquired file.  Data block sizes are in 1040 bytes with the first 8 bytes
being the actual timecode. Bytes 9-15 are not used (TBD) and can be ignored. Byte 16
of the Timecode info specifies whether the Timecode in bytes 1-8 is valid.  If byte 16
is 0xC0 then the timecode is valid and applies to the 1st sample of data in the 1K block
after the current 16 byte Timecode info field.  If the 16th byte is 0x00 then the Timecode
is not valid for the current block of 1KBytes and can be ignored.
*****************************************************************
*Bytes 1-8 TC**Bytes 9-15 TBD**Byte 16 Valid TC Marker**Bytes 17-1040 Data**REPEAT PATTERN*
*****************************************************************
gigexdv1_8>
```

The *acquire* command allows a user to bring into a Host system GIGExD generated data and store it to a file. The parameters for the acquire command are:

<Host IP Addr> **IP address of the Host System**

<GigE IP Addr> **IP address of the GIGExD (set with *flashcfg* command)**

<Path & Filename> **Path & Filename Of Where Data is to be stored on the Host machine**

<MC Address> **The multicast Address from which to acquire data. This is the multicast Address of the GIGExD**

<file size> **The amount of data to acquire in 1k chunks (setting this value to 2048 will acquire 2 MBytes of data)**

All of these parameters must be present for data to be acquired from the GIGExD.

It is important to note the multicast packets that the GIGExD produces are UDP protocol in SDDS format. SDDS format is simply a UDP protocol packet of 1080 bytes. The first

56 bytes of the 1080 bytes is the SDDS header and remaining 1024 bytes are data. The 56 bytes of SDDS contains a number of fields one of which is the time code. Using the *acquire* command, the 56 bytes of SDDS header is discarded and the 1024 bytes of data is written to the path and filename listed as the third command argument. Therefore, using the *acquire* command ONLY writes to file the data. It does NOT write the associated time code to the file.

The *acqall* command writes the entire 1080 bytes of SDDS header and data to file. In other words, the *acqall* command writes the entire UDP packet payload to the file. This command has the fastest disk write operations because the SDDS data is not manipulated by the software. Instead, it is read from the gigabit Ethernet port and immediately written to disk. It is the responsibility of a user's application to remove time code for the SDDS header and to do endian conversion on both the time code and data-if needed. The 2nd byte of the SDDS header, byte #1, is the Data Format ID Byte. If this byte is 0x08 then the SDDS data portion is BYTE represented and the user application does not need to do endian conversion. If the Data Format ID Byte is 0x10 then the SDDS data portion is 16 BIT represented and the user application MUST perform big-endian to little-endian conversion by swapping every other byte. When the GIGExD is producing 8 bit or complex 8 bit samples the SDDS data will be BYTE represented. When the GIGExD is producing 16 bit or complex 16 bit samples the SDDS data will be 16 bit represented-and endian conversion will need to be done.

In the SDDS header, time code occupies byte #4 through #15. These 16 bytes must always be endian converted regardless of the SDDS data format. The endian conversion should be applied to all 16 bytes. In other words, byte #0 of the 16 bytes should be swapped with byte #15. Byte #1 should be swapped with byte #14, and so on. Once this is done, the 16 bytes of time code take the form of time code when using the *acqwtc* command below. (The first 8 bytes are the time code value, next 7 bytes are RFU and should be ignored, finally the 16th byte will be the time code Validation Marker).

The *acqwtc* (acquire with time code) command removes time code from the SDDS header and then writes it, with the SDDS data portion of packet, to the file indicated by the user. The time code portion of the SDDS header is 16 bytes. Therefore, for each packet that is received from the GIGExD 16 bytes of time code will be written to file followed by 1024 bytes of data-for a total of 1040 bytes per packet.

For the 16 bytes of time code written, only the first 8 bytes are actually the time code value. The next 7 bytes are Reserved for Future Use and should be ignored. Finally, the 16th byte written to file will be the time code Validation Marker. The time code Validation Marker will be one of two values. If the value is 0x00 then the time code present in bytes 1-8 is not valid for the current 1024 bytes of data that follow the 16 bytes of time code information. If the time code Validation Marker is 0xC0 then the time code in bytes 1-8 is valid and applies to the FIRST sample of data in the 1024 bytes of data that follow the current time code information.

Sample Rate

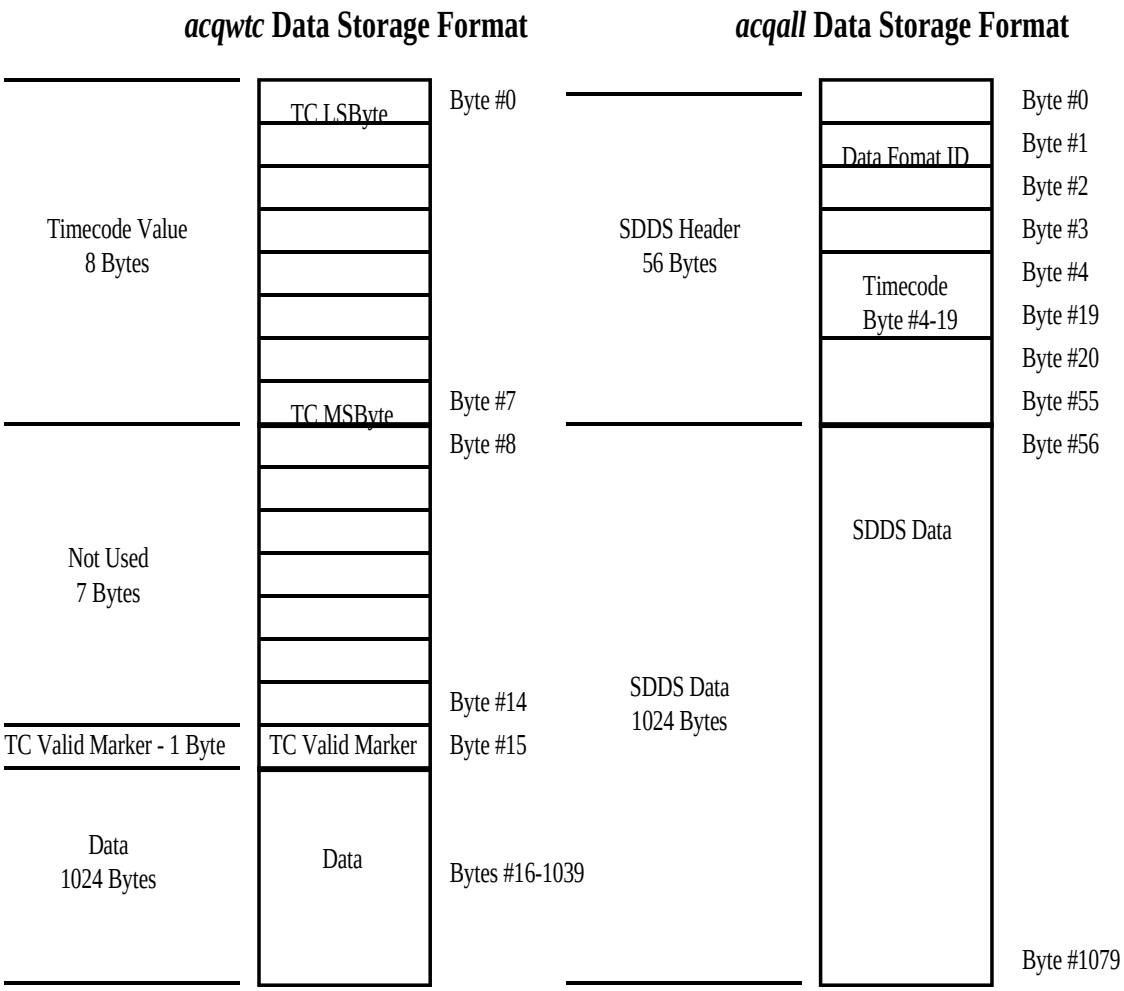
As of version 2.0 the sample rate is added to the SDDS header portion of the packet. The sample rate is offset 24 bytes from the start of the SDDS header in each packet and is 8 bytes long in Big Endian format. This eight byte value is a 2's complement number with the LSBit representing (125MHz/2^63). In other words:

From the SDDS Specification, Value = (sample rate/125Mhz) * 2^63

Packet Count

As of version 1.4 (and higher) of the GIGExD firmware, a modulo 31 counter has been placed in each data packet. The counter value conforms to the SDDS specification, is represented in big-endian format, and occupies the 3rd and 4th bytes of each packet. The first 5 bits of the counter count from 0 to 30 and then "roll over". The next 11 bits count the number of roll overs of the 5 bit modulo 31 bit counter. Having a modulo 31 counter provides for easy detection of lost packets due to host buffer limitations.

All the acquire commands, *acquire*, *acqwtc*, and *acqall* check the packet count of arriving packets to determine if data has been lost. At the end of and acquisition, the amount of packets lost during acquisition will be output to the screen for the user.


acqwtc Data Storage Format
acqall Data Storage Format
Timecode Value
8 Bytes
TC LSByte
Byte #0
TC MSByte
Byte #7
Not Used
7 Bytes
Byte #8
Byte #14
TC Valid Marker - 1 Byte
TC Valid Marker
Byte #15
Data
1024 Bytes
Data
Bytes #16-1039
SDDS Header
56 Bytes
SDDS Data
1024 Bytes
Data Fomat ID
Timecode
Byte #4-19
SDDS Data
Byte #0
Byte #1
Byte #2
Byte #3
Byte #4
Byte #19
Byte #20
Byte #55
Byte #56
Byte #1079

The 8 byte time code number represents the number of 250picoSeconds from the start of the CURRENT year. (SDDS time code does not include calculations for more than 1 year) When the time code Valid Marker is set to 0xC0 then the time code in byte numbers 0 to 7 represent the time of the FIRST sample of data in the associated data portion written to file. The sample size of 8, 16, complex 8, or complex 16 bits is set via the user.

It's important to note that packets can be "dropped" by the host computer if the storage device (RAID) is unable to write data to media at the rate that the GIGExD is outputting data. With this release of firmware/software there is a way of detecting "dropped data". This release has a change in the GIGExD firmware that outputs a prime number roll over sample counter in the SDDS header. This counter marks each SDDS packet with the counter value such that if any packets are dropped because of Host system latency it can be detected in software.

When using either the *acqwtc* or *acquire* commands, the *gigexd* program needs to do endian conversion on time code, 16 bit, and complex 16 bit data as the data is received on the Host machine. This is because the SDDS specification requires that these numbers be transmitted big-endian. This (and higher) releases of firmware have the ability to send data Little Endian so the conversion does not need to be done in software on the Host machine. Currently, the need to do Endian conversion prior to data storage impacts the Host machine's ability to quickly store data. Since the *acqall* command does not do data manipulation (Endian Conversion), it is the fastest method to store data. When used, it is the responsibility of the user application to do the Endian conversion. Application of the LEND flag can greatly enhance data throughput of 16 bit data because the host no longer needs to do endian manipulation.

5. Communications

From the parameters of the *gigexd* program above, the user must know two facts before they can use the *gigexd* program:

1) The IP address of the Host Machine

2) The IP address of the GIGExD.

The Host machine IP address can be found/modified by using the *ifconfig* command under Linux. A user must be root or super-user to run the *ifconfig* command under Linux. Example:

>> **/sbin/ifconfig <return>**

The above command will report back the Ethernet devices and their IP addresses.

The GIGExD has it's IP address set at the time of delivery to a factory default value of 192.9.200.105. (This address can be changed by the user using the *gigexd* program, and the *flashcfg* command to any IP address the user wishes-see the *flashcfg* command under the "Commands" section). Also the GIGExD has a unique way of setting the IP address to the factory default in case the GIGExD IP address is lost or forgotten. (See the "Catch-22" section)

As an example, assume that the IP address of the Host machine is 138.9.40.110 and the GIGExD is newly received and is programmed with it's default IP address of the 192.9.200.105. In order for the Host machine to "talk" to the GIGExD a "route" must be added for the Ethernet port on the Host machine that has the IP address of 138.9.40.110. To do this, a user must add the domain of 192.9.200.aaa to the route for the Ethernet port of the Host computer. Under Linux this is done with the *route* command. Please note, a user must be root or super-user to run the *route* command under Linux. Below is an example of how to add the route:

From the *ifconfig* command above it was earlier determined that device **eth0** has an IP address of 138.9.40.110.

>> /sbin/route add -net 192.9.200.0 netmask 255.255.255.0 dev eth0 <return>

(Note that the address is 192.9.200.0 and the netmask is 255.255.255.0. The ".0" at the end of the address and netmask are used instead of the ".105" that is the actual address. This is correct syntax and not a typographical error.)

The above command now identifies to the Host machine that all addresses of the form 192.9.200.aaa should go out the device eth0.

Thereby, all communication with the GIGExD with address **192.9.200.105** will go out on Host machine IP address 138.9.40.110 (eth0).

In summary, to "talk" to (configure, program, acquire data) the GIGExD requires the IP address of the Host machine and the GIGExD. Also, no more than 2 commands (*ifconfig* and *route*) on the Host machine to establish the communications.

Once the route is set on the Host machine, the user can issue a "ping" command on the Host machine to confirm that communications have been established. Example:

>> **ping 192.9.200.105 <return>**

5.1. "Catch-22"

The GIGExD is programmed, configured, and data accessed using its configured IP and multicast addresses. The user without initial knowledge of GIGExD IP address cannot change any parameters including the IP address. This presents a problem when the GIGExD IP address is lost or forgotten. In this case, a *flashcfg* command cannot be sent to reset the IP address to a known value. In summary, this presents a "Catch-22" situation when the IP address is lost or forgotten-the IP address is needed to change the IP address to a known value.

When the GIGExD is delivered its IP address is set to 192.9.200.105. The user may (and is encouraged) to change the IP address to match those in the domain that is current for their location. Once done, there is a hardware mechanism that will allow you to "make" the GIGExD assume the address of 192.9.200.105 again if the new IP address is lost or forgotten. The hardware mechanism is the "Default" button. On the prototype GIGExD boxes, this is the white button that is visible from the front of the unit. On the production GIGExD units, a small screw on the top of the units must be removed then the button is accessible by using a "paper clip" to press the button (similar to a CDROM drive). Once the button is pressed, it must be held for 2 seconds. When held for 2 seconds the GIGExD will default back to its factory default IP address of 192.9.200.105 and any data flow will be turned off.

After the GIGExD is returned to its factory default IP address of 192.9.200.105 a *query* command should be run. The *query* command will report back the IP address .(along with other information about the firmware download, data rate, etc..) configured by the user using the *flashcfg* command.

After the *query* command, the user programmed IP address is known. At this point the *qrywcfg* command can be run to restart the module with its current configuration parameters including the user programmed IP address that was reported with the *query* command. Or, a *flashcfg* command can be run using 192.9.200.105 as the GIGExD IP address and the GIGExD can be reconfigured to a new IP address.

In general, in this situation a Host machine "ping" operation is an extremely useful tool. Once the default button is pressed, a user can execute:

>> ping 192.9.200.105 <return>

The GIGExD will reply to the 192.9.200.105 address. Once the *query,* and then subsequent *qrywcfg* commands are run, the user can then confirm the switch back to their configured IP address by typing:

>> ping <user IP Address> <return>

5.2. Maximizing Host Computer GIGExD Port Data Transfer Rates

Host machines that configure and receive data from the GIGExD typically do not have their GIGE ports optimized for fast data transfer rates. In other words, under Linux, using the standard installed configuration for GIGE ports will provide for slow data transfer rates and dropped packets.

Below is a command that increases the GIGE driver memory buffer size so that maximum data transfer rates can be realized. Users have typically gotten the maximum 100MBytes/Sec throughput using this command and version 1.4 (or higher) of the GIGExD software to take advantage of the increased memory buffer size. The command MUST be run as root or Superuser. Once executed, the command only needs to be run once until the next power/reset cycle of the host machine.

The command is:

>>/sbin/sysctl -w net.core.rmem_max=10485760

Note that the number at the end is 10Megabytes-this is the new size of the allocation buffer. With this value users have been able to obtain maximum data rates. Larger sizes may be tried but a corresponding change to the gigexd.c software must be made to take advantage of the new size. At the top of the GIGExD C program file gigexd.c is the declaration:

#define GIGEMEMBUFSIZE 10485760

This value should be changed to the value that is set by the user using the sysctl command above. Then the program gigexd.c should be recompiled -see the section **Build Script & gigexd Executable** for compilation instructions.

6.I/O Modules

6.1. A2DR7

There are four input cable connections for the A/D module. From right to left, in the pictures below, the connections are Analog Signal Input, Clock Input, 1PPS signal input, and IRIG time code input. (Note that the leftmost connection is obscured from view by the gigabit Ethernet SFP transceiver.)



Analog Signal:	1.2Vp-p maximum into 50 Ohms
Clock Input:	0.5Vp-p maximum into 50 Ohms
1PPS Input:	TTL Level not to exceed 3.5V (No Amplitude Modulation)

IRIG Input: TTL Level not to exceed 3.5V (No Amplitude Modulation)

Most equipment (GPS Receivers) that outputs time code in IRIG format have jumper settings or configuration software that allows for either TTL or AM output for the IRIG and 1PPS signals. These signals must be set to TTL output for the GIGExD to correctly interpret the signal levels.

6.2. A2DR7 Clocking For GIGExD**R2**

Option 1, No Flags Specified: High Speed PLL using External 10MHz reference

The A2DR7 has an on-board, high precision, low jitter, PLL that is capable of taking an input 10MHz and producing a clock that is between 90-100MHz with a programming granularity of 12.5kHz. (Using the configuration parameter F, Input Decimation, the 90-100MHz rate can be divided down to a frequency that is divisible by a number between 1 and 8-yielding additional sampling frequencies other than 90-100MHz.) Typically this 10MHz reference comes from a GPS receiver that also produces IRIG time code and a 1PPS pulse but can be any 10MHz reference 1/2Vp-p square wave or sine wave. Using the 10MHz reference that is produced from a GPS receiver allows for coherent clocking and time code insertion into the Ethernet packets. Used in this way, the "time stamping" of data is EXTREMELY accurate.

Option 2, MUXCLK=P: High Speed PLL using on-board 10MHz reference

Used in this manner, time code coming from an external source would NOT be coherent with the A/D sampling. This mode of clocking is typically used when a user is not concerned with time code and does not have an external 10MHz available. In this case the 10MHz reference (socketed 14 pin DIP package-supplied by the user) will allow the user to tune the PLL to sample rates between 90-100MHzwith a programming granularity of 12.5kHz, for various sampling values and not have to depend on external hardware.

Option 3, MUXCLK=C: Direct On-Board Oscillator clocking (No PLL use)

As in option 2, used in this manner time code coming from an external source would not be coherent with the A/D sampling. Also, the user would not be able to selectively program the rate between 90-100MHz or an integer between 1 and 8 to divide down of 90-100MHz. This option's benefit is that a user could select a rate that is not obtainable by a divide down of 90-100MHz by an integer between 1 and 8.

Option 4, MUXCLK=N: Direct External Clocking (No PLL use)

This option provides the most freedom for clocking the on-board A/D chip. The user can supply any rate between 10-100MHz as the clock source. Option 4 allows the A/D chip to be driven directly from an external source (1/2Vp-p). If the

external clock source is locked to an IRIG time code source then time code will be accurate and locked to the clocking of the A/D chip.A2DR7 I/O Cables

The A2DR7 for the GIGExD**R2** has that ability to be either AC or DC coupled. When shipped the A2DR7 is AC coupled. Jumpers J7, J8, and J9 select DC or AC coupling. When shipped these jumper are ALL set to positions 2 &3-this enables AC coupled mode. To enable DC coupled mode, mode jumpers J7,J8 and J9 to positions 1 & 2.

6.3. A2DR9 Clocking For GIGExD**R2**

Clocking

There are 4 clocking options for the A2DR9 with the GIGExD**R2**. The clocking options are chosen via the Flags section of the configuration file. The flags associated with the 4 clocking options are: none (no flags specified), MUXCLK=P, MUXCLK=C, or MUXCLK=N

Option 1, No Flags Specified: High Speed PLL using External 10MHz reference

The A2DR9 has an on-board, high precision, low jitter, PLL that is capable of taking an input 10MHz and producing a clock that is between 180-200MHz with a programming granularity of 12.5kHz. (Using the configuration parameter F, Input Decimation, the 180-200MHz rate can be divided down to a frequency that is divisible by a number between 1 and 8-yielding additional sampling frequencies other than 180-200MHz.) Typically this 10MHz reference comes from a GPS receiver that also produces IRIG time code and a 1PPS pulse but can be any 10MHz reference 1/2Vp-p square wave or sine wave. Using the 10MHz reference that is produced from a GPS receiver allows for coherent clocking and time code insertion into the Ethernet packets. Used in this way, the "time stamping" of data is EXTREMELY accurate.

Option 2, MUXCLK=P: High Speed PLL using on-board 10MHz reference

Used in this manner, time code coming from an external source would NOT be coherent with the A/D sampling. This mode of clocking is typically used when a user is not concerned with time code and does not have an external 10MHz available. In this case the 10MHz on-board reference will allow the user to tune the PLL to sample rates between 180-200MHz with a programming granularity of 12.5kHz, for various sampling values and not have to depend on external hardware.

Option 3, MUXCLK=C: Direct On-Board Oscillator clocking (No PLL use)

As in option 2, used in this manner time code coming from an external source would not be coherent with the A/D sampling. Also, the user would not be able to selectively program the rate between 180-200MHz or an integer between 1 and 8 to divide down of 180-200MHz. This option's benefit is that a user could select a rate that is not obtainable by a divide down of 180-200MHz by an integer between 1 and 8.

Option 4, MUXCLK=N: Direct External Clocking (No PLL use)

This option provides the most freedom for clocking the on-board A/D chip. The user can supply any rate between 40-200MHz as the clock source. Option 4 allows the A/D chip to be driven directly from an external source (1/2Vp-p). If the external clock source is locked to an IRIG time code source then time code will be accurate and locked to the clocking of the A/D chip.A2DR7 I/O Cables

6.4. A2DR10 Clocking for GIGEXD**R2**

Clocking

There are 4 clocking options for the A2DR10 with the GIGExD**R2**. The clocking options are chosen via the Flags section of the configuration file. The flags associated with the 4 clocking options are: none (no flags specified), MUXCLK=P, MUXCLK=C, or MUXCLK=N

Option 1, No Flags Specified: High Speed PLL using External 10MHz reference

The A2DR10 has an on-board, high precision, low jitter, PLL that is capable of taking an input 10MHz and producing a clock that is within the full range sample rate of the A/D device, 40-200MHz. Frequency synthesis granularity depends upon multiple factors and typically results in the sub Khz to 2Khz range. Typically this 10MHz reference comes from a GPS receiver that also produces IRIG time code and a 1PPS pulse but can be any 10MHz reference 1/2Vp-p square wave or sine wave. Using the 10MHz reference that is produced from a GPS receiver allows for coherent clocking and time code insertion into the Ethernet packets. Used in this way, the "time stamping" of data is EXTREMELY accurate.

Option 2, MUXCLK=P: High Speed PLL using on-board 10MHz reference

Used in this manner, time code coming from an external source would NOT be coherent with the A/D sampling. This mode of clocking is typically used when a user is not concerned with time code and does not have an external 10MHz available. In this case the 10MHz on-board reference will allow the user to tune the PLL to sample rates between 40-200MHz with a programming granularity of 250kHz, for various sampling values and not have to depend on external hardware.

Option 3, MUXCLK=C: Direct On-Board Oscillator clocking (No PLL use)

As in option 2, used in this manner time code coming from an external source would not be coherent with the A/D sampling. Also, the user would not be able to selectively

program the rate between 40-200MHz This option's benefit is that a user could select a rate that is not obtainable by a divide down of 180-200MHz by an integer between 1 and 8.

Option 4, MUXCLK=N: Direct External Clocking (No PLL use)

This option provides the most freedom for clocking the on-board A/D chip. The user can supply any rate between 40-200MHz as the clock source. Option 4 allows the A/D chip to be driven directly from an external source (1/2Vp-p). If the external clock source is locked to an IRIG time code source then time code will be accurate and locked to the clocking of the A/D chip.

6.5. A2DR9/A2DR10/A2DR13 GAIN

The A2DR9 and A2DR10 have a front end variable gain stage before the A/D converter. Using the flag **A2DGAIN=<value>** will enable the front end variable gain stage. <value> is an integer in the range of -4 to 20. These integers represent in dB the amount of gain to add to the analog signal. Negative values attenuate the signal. User care should be taken when assigning positive gain values. Too much can damage the A/D converter. When assigned a gain value, the A2DR9/A2DR10 logic increments the gain to the desired value. While incrementing, if the A/D device asserts its (Out of Range) indicator then gain incrementing is stopped. In summary, there is some logic protection for gain insertion but care should be taken not to “overdrive” the A/D converter. Knowledge of the incoming signal power level is important BEFORE setting the gain.

6.6. A2DR13 I/O Module

The A2DR10 module has been renamed to the A2DR13 module. This is a name change only!! Under version 2.10 (or higher) of the software specifying either A2DR10 or A2DR13 will configure the module accordingly. GIGEXDR2 units labeled as containing either an A2DR10 or A2DR13 module are functionally equivalent.

7. Technical Specifications

GIGExDR0 Specifications

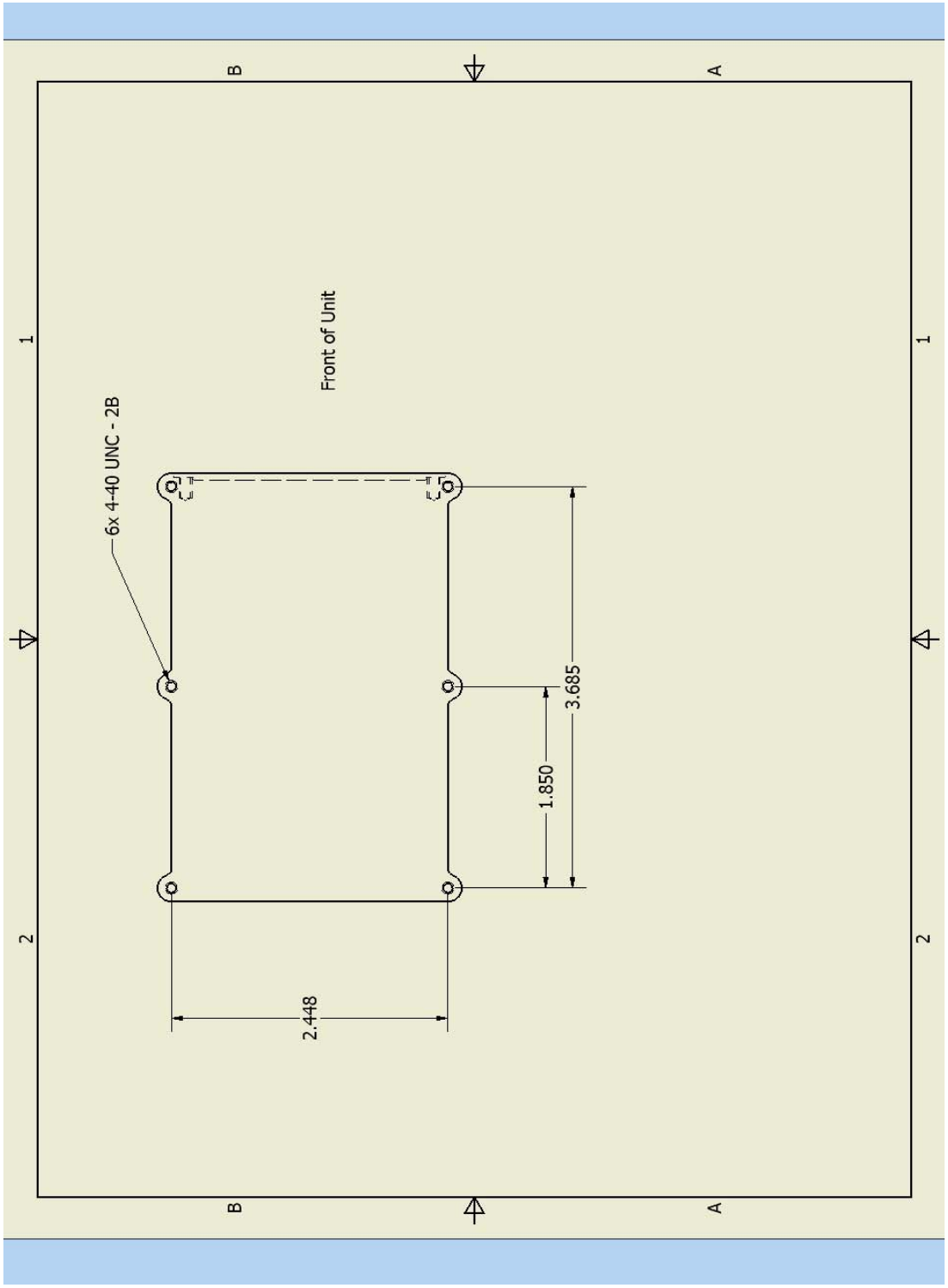
Input Voltage	7-18 Volts, 12 Volts nominal
Input Power	60 Watts maximum, 40 Watts nominal
Power Connector	CUI, Inc. PJ-015A-SMT
Mating Connector	CUI, Inc. PP-002A
Dimensions	101.6 x 80.01 x 41.91 mm(4 x 3.15 x 1.65 in.)
Weight	0.205 kg (0.450 lb)
Operating Temp	0°C-40°C (32°F-102°F) ambient
Operating Humidity	90% maximum relative humidity, non-condensing

GIGExDR1 Specifications

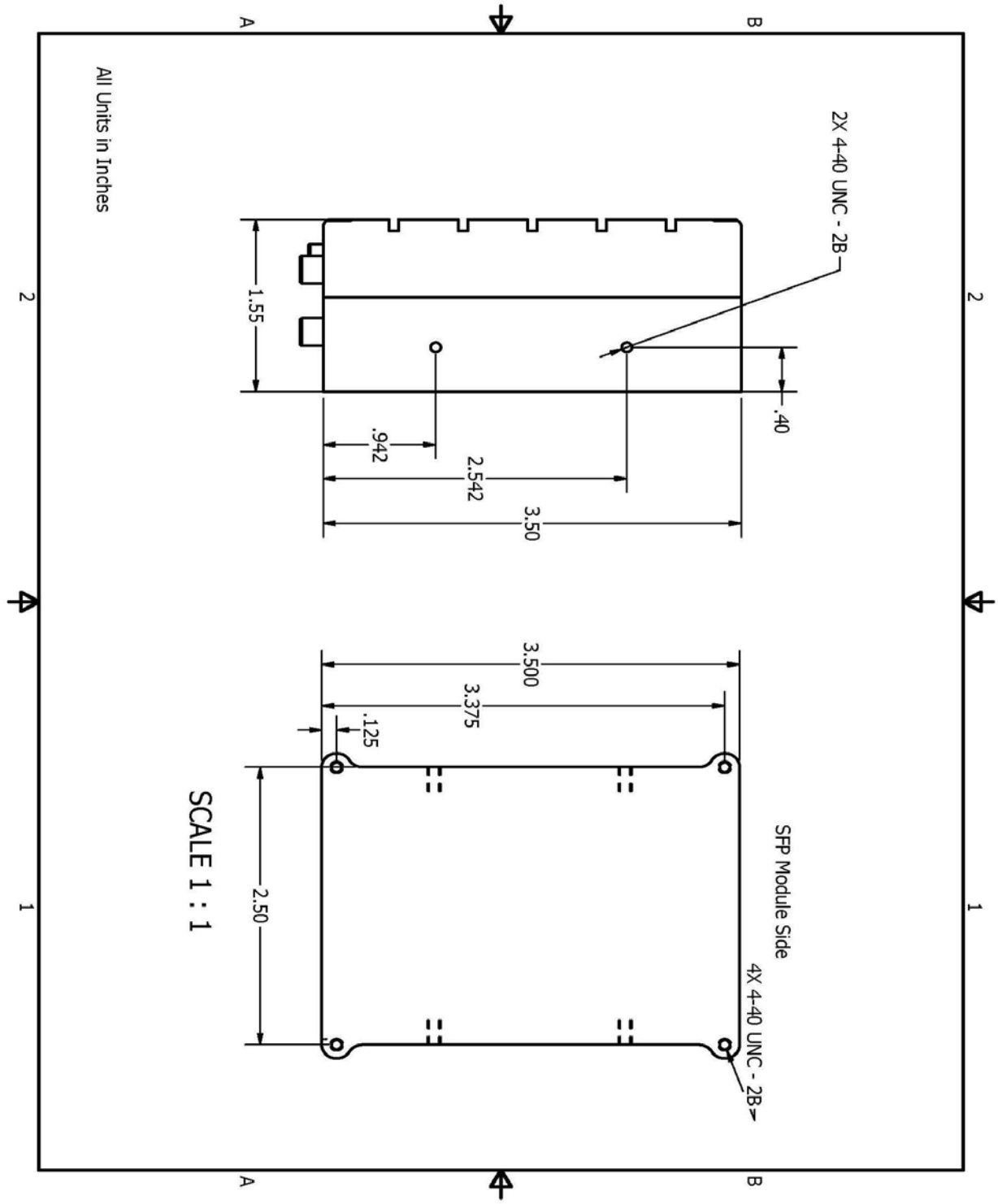
Input Voltage	7-18 Volts, 12 Volts nominal
Input Power	40 Watts maximum, 30 Watts nominal
Power Connector	CUI, Inc. PJ-015A-SMT
Mating Connector	CUI, Inc. PP-002A
Dimensions w/o SFP	101.6 x 66.04 x 38.1 mm(4 x 2.6 x 1.5 in.)
Dimensions w/ SFP	128.27 x 66.04 x 38.1 mm(5.05 x 2.6 x 1.5 in.)
Weight	0.443 kg (0.975 lb)
Operating Temp	0°C-40°C (32°F-102°F) ambient
Operating Humidity	90% maximum relative humidity, non-condensing

GIGExDR2 Specifications

Input Voltage	7-18 Volts, 12 Volts nominal
Input Power	12 Watts maximum, 10 Watts nominal
Power Connector	Switchcraft 712RA
Mating Connector	Switchcraft 760K
Dimensions w/o SFP	88.9 x 66.04 x 39.37 mm (3.5 x 2.6 x 1.55 in.)
Dimensions w/ SFP	107.95 x 66.04 x 39.37 mm (4.25 x 2.6 x 1.55 in.)
Weight	0.389 kg (0.855 lb)
Operating Temp	0°C-40°C (32°F-102°F) ambient
Operating Humidity	90% maximum relative humidity, non-condensing



Mechanical Drawings GIGExD**R1**



Mechanical Drawings GIGExD**R2**

Artisan Technology Group is an independent supplier of quality pre-owned equipment

Gold-standard solutions

Extend the life of your critical industrial, commercial, and military systems with our superior service and support.

We buy equipment

Planning to upgrade your current equipment? Have surplus equipment taking up shelf space? We'll give it a new home.

Learn more!

Visit us at **artisantg.com** for more info on price quotes, drivers, technical specifications, manuals, and documentation.

Artisan Scientific Corporation dba Artisan Technology Group is not an affiliate, representative, or authorized distributor for any manufacturer listed herein.

We're here to make your life easier. How can we help you today?

(217) 352-9330 | sales@artisantg.com | artisantg.com

ARTISAN® TECHNOLOGY GROUP