User's Guide: RI-Network Diode One-Way Transfer Device

rumeL, Inc.
www.rumel-online.com
540-338-3252

May 2008
For Software Version 1.00

Prepared by: rumeL, Inc
10302 Eaton Place Suite 240
Fairfax, Virginia 22030
www.rumel-online.com

Bill Owen
President
540.338.3252
bill@rumel-online.com

Table of Contents

1.0 Summary and Scope

The application of one way transfer devices are typically in the transfer of data from “low” side (unclassified) networks to “high” side (classified) networks. In subsequent sections, the operations for rumeL, Inc's RI-Network Diode, a network one way transfer device will be discussed. **The design for the RI-Network Diode provides for full Gigabit Ethernet data transfer rates in a one way network direction. In other words, the Network Diode internally, operates at much faster rates than Gigabit Ethernet standards and therefore create a data transfer “bottlenecks”.**

2.0 rumeL, Inc. Network Diode Quickstart Guide

1.) The most up-to-date software can be obtained at:
http://www.rumel-online.com/software.htm
(Note extension is .htm NOT .html)

2). Attach power supplies and network cables for Rx and Tx hardware communications. (The Network Diode may be connected directly to the host machine or to a router network where the host machine resides.)

3) Copy the Rx Hardware software *rxdiodevA_BC.zip* and the Tx hardware software *txdiodevA_BC.zip* to there respective hosts.

4).The Rx hardware IP address when delivered is 192.9.200.115. The Tx hardware IP address when delivered is 192.9.200.116. See the section titled **Communications** for information on how to set the host computer route to identify the address of 192.9.200.115 and 192.9.200.116

5) On the Rx software Host computer execute: ping 192.9.200.115 <return> to be sure communications are established with the Rx hardware. On the Tx software Host computer execute: ping 192.9.200.116 <return> to be sure communication are established with the Tx hardware.

6) Refer to the section: **Build Scripts and the txdiode and rxdiode Executables** in the User Guide on how unzip/build the *rxdiode* and *txdiode* executables.

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

8) Refer to the section: **Rx Hardware *flashcfg* Command**, and **Rx Hardware Configuration File Parameter Description** on how to configure the Rx hardware for your applications.

9) Refer to the section: **Tx Hardware *flashcfg* Command**, and **Tx Hardware Configuration File Parameter Description** on how to configure the Tx hardware for your applications.

10) Execute the *flashcfg* command to configure the Rx hardware with a new IP address and configuration parameters.
11) Execute the *flashcfg* command to configure the Tx hardware with a new IP address and configuration parameters.

12) Refer to the section: ***query* And *qrywcfg* Commands** on how to verify configuration settings for steps 10 &11.

13) Refer to the section **Network Diode File Transfer** on how to transfer files across the Network Diode.

14) Refer to the section: ***acquire, acqall,* and *acqwtc* Commands** on how to acquire data to the Host computer that is in SDDS format (generated by a rumeL, Inc. GIGEXD product).

15) Refer to section: **"Catch-22"** for information on setting the Rx or Tx hardware back to its factory default settings.(Important)

Other points:
A. The Rx and Tx hardware of the Network Diode should only be connected directly to Gigabit Ethernet interfaces. They will 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 Network Diode User Guide.
C. Finally, any questions can be directed to:

Bill Owen
rumeL, Inc.
bill@rumel-online.com
540-338-3252

Applicable Documents

1. V. Anantharam, and S.Verdu, “Bits through queues,” Information Theory, IEEE Transactions on , Volume: 42 , Issue: 1 , Jan. 1996.

2. V. Anantharam and S. Verdú, ``Reflections on the 1998 Information Theory Society Paper Award: Bits through Queues," IEEE Information Theory Society Newsletter vol. 49, no. 4, Dec. 1999.

3. J. S. Holmgren and R. P. Rich, Metric Methodology for the Creation of Environments and Processes to Certif Component: The NRL Pump, Naval Postgraduate School Monterey CA, March 2003.

3.0 One Way Transfer Diode Introduction

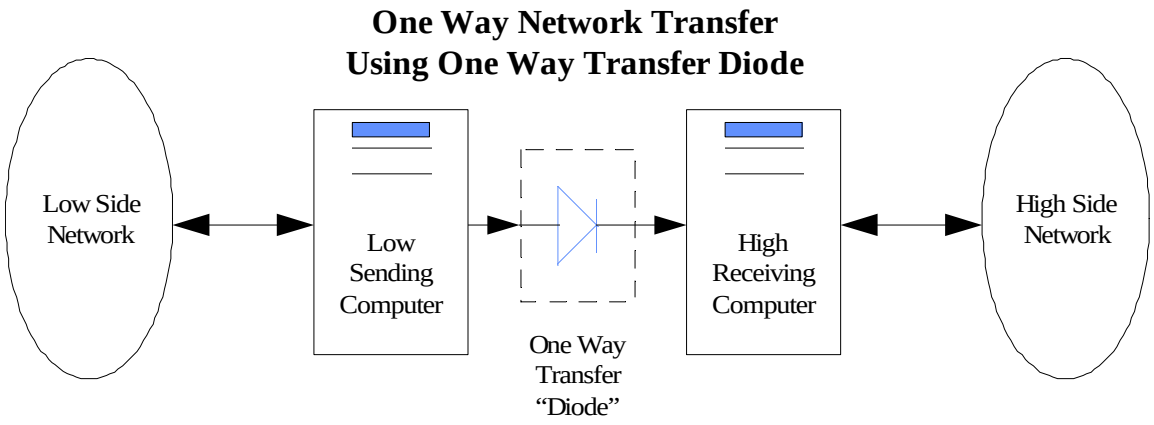


Figure 1 – High Level One Way Transfer Diode

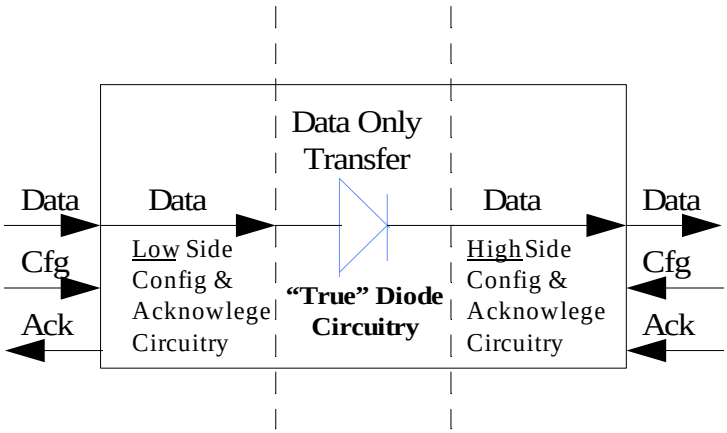


Figure 2 – Overview Of Diode Circuitry

The diagrams above show conceptually the use of the One Way Transfer Diode. In implementation, the One Way transfer Diode has 2 sections of hardware. The receive (Rx), or input, hardware section is shown on left side of Figure 2. The Rx hardware can be programmed, configured, and accepts data for transfer to the fiber diode interface. Once at the fiber diode interface data is transported to the transmit (TX) interface (right side of Figure 2). From the Tx interface data is sent out to host IP addresses previously configured in the Tx interface hardware.

3.1 Operation

As mentioned above the Network Diode is comprised of two pieces of hardware connected by a single fiber optic link. The Rx piece of hardware contains a Primary IP address and up to 4 Target IP addresses. The Primary IP address is used to configure or

program the Rx side of the Diode over its Gigabit Ethernet Interface. The 4 Target IP addresses are set by the user via the same Gigabit Ethernet interface during configuration commands to the Primary IP address. These Target IP addresses may be any valid IP address, including the Multicast group of addresses. These 4 Target IP addresses are listening addresses. Any UDP protocol packet that corresponds as being sent to these addresses will be forward across that fiber link to the Tx hardware interface.(See Fig. 3)

Like the Rx circuitry, the Tx circuitry has a Primary IP address that is used for programming and configuration. Along with the Primary IP address, up to 4 Target IP addresses can be configured with the Tx section of hardware. These 4 Target IP addresses serve as the target IP addresses where data is to be sent once it is received across the fiber interface from the Rx section of hardware. All valid IP addresses are allowed including those in the Multicast address range. (See Figure 3)

Data transfers are shown and discussed in this document from low side to high side but use of the Network Diode in the opposite direction, high side to low side, is also possible. rumeL, Inc. 1U rack mount units can be configured with 2 Network Diodes to provide for the ability to go both directions.

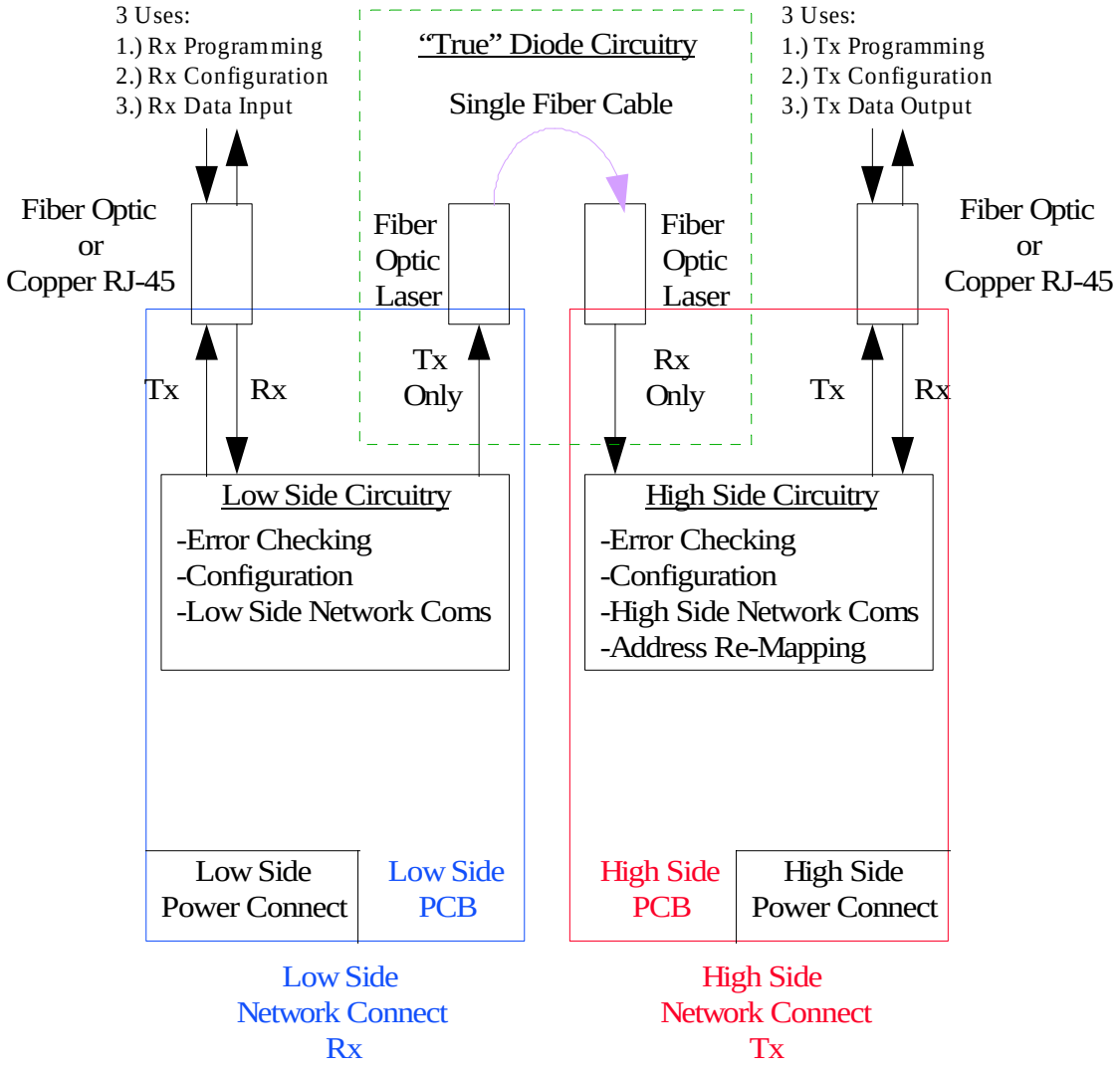


Figure 3 – Detail Of Diode Circuitry

3.2 Configuration

As briefly stated in section 3.1 the Diode configuration is a two step process that is accomplished over the Rx and Tx hardware Gigabit Ethernet ports. The Rx section of hardware, is configured with 4 IP addresses that it listens to for receipt of UDP packets. The listening IP addresses are numbered 1 through 4 in the configuration section of the Rx hardware. These 4 listening addresses correspond to 4 target addresses configured in the Tx section of hardware. The mapping is simply a 1:1 correspondence. Data received for listening IP address 1 will be forwarded to the Tx section of hardware and then sent out to target IP address 1. Data received for listening IP address 2 will be forwarded to the Tx section of hardware and then sent out to target IP address 2—and so on.

For both the Rx (Low Side) and Tx (High Side) IP addresses configuration is done over the Rx or Tx Gigabit Ethernet port, respectively, via a C program. The C program takes input from a text configuration file that is setup by the user. Once read, the data from the configuration file is sent to the Rx or Tx section's flash memory for non-volatile storage. The non-volatile storage means that all configuration is retained through power cycles-whether the power cycles are intentional or not. Along with the Target IP addresses in both the Rx and Tx hardware, the Primary IP addresses of both the Rx and Tx hardware are configurable. Upon delivery the Rx hardware has the Primary IP address of 192.9.200.115. The Tx hardware has the default IP address of 192.9.200.116.

Once configured the Rx or Tx hardware will run in its programmed mode until re-configured again. This allows for lower configuration cycles and maintenance. See the description of the rxdiode.c and txdiode.c programs and the rxdiodeusr.cfg and txdiodeusr.cfg configuration files below for more details on hardware configuration.

Finally, on both the Rx and Tx hardware is a "default button". This button when pressed and held for 2 seconds will return the Rx and/or Tx hardware to its factory default IP addresses. This allows the user to return a Rx or Tx hardware to a default IP address for updating configuration or programming.

3.3 Programming

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

4.0 Software

Two C programs are used to program the Rx and Tx hardware via a host computer's network port. A Linux build script is included with the software. Currently, the Rx and Tx hardware software zip packages rxdiodev*A_BC*.zip and txdiodev*A_BC*.zip, (Where *A* and *BC* represent the major and minor version numbers of the associated Rx and Tx software packages) reside on the rumeL, Inc. web site at:
http://www.rumel-online.com/software.htm
(Note that the above extension is .htm NOT .html)

It is VERY important that versions of Rx and Tx hardware are the same. As an example, users should NOT use Rx software version 1.01 on the low side and at the same time be using Tx software version 1.02 on the high side. Instead, both Rx and Tx software versions should either both be 1.01 or 1.02. Rx software and Tx software versions should be the same for compatibility during data transfer across the Network Diode. Problems WILL arise if this rule is not followed.

4.1 Build Scripts and the *txdiode* and *rxdiode* Executables

The software and firmware for the Rx and Tx hardware is bundled into two zip packages of the form rxdiodev*A_BC*.zip and txdiodev*A_BC*.zip. Where *A* and *BC*, as detailed above, represent the major and minor version numbers of the associated software packages. When unzipped to a host computer, a directory of rxdiodev*A_BC* and txdiodev*A_BC* are created and all source/firmware files are unzipped into the directory of rxdiodev*A_BC* and txdiodev*A_BC*-this prevents the mixing of old and new software when packages are unzipped. As an example, version 1.00 of the Rx and Tx software would be bundled as the zip packages rxdiodev1_00.zip and txdiodev1_00.zip. Extraction of these 2 files would yield the directories of rxdiodev1_00 and txdiodev1_00 with the associated software files in each of the directories.

Once the rxdiodev*A_BC*.zip package is unzipped, the directory rxdiodev*A_BC* will have a number of files:

File Name	Description
rxdiode.c	C program source code
rxdiodebuild.lnx	Linux Build Script
rxdiodeusr.cfg	Rx Diode Configuration File
rxdiode.rpd	Rx Diode Firmware
rxinitdiode	Production Initialization Script (NOT FOR END USER USE)
rxdiode.jic	Production Testing/Initialization Firmware (NOT FOR END USER USE)

Once the txdiodev*A_BC*.zip package is unzipped, the directory txdiodev*A_BC* will have a number of files:

File Name	Description
txdiode.c	C program source code
txdiodebuild.lnx	Linux Build Script
txdiodeusr.cfg	Tx Diode Configuration File
txdiode.rpd	Tx Diode Firmware
txinitdiode	Production Initialization Script (NOT FOR END USER USE)
txdiode.jic	Production Testing/Initialization Firmware (NOT FOR END USER USE)

Tables 1 & 2 – Software Contents

The C programs **rxdiode.c** and **txdiode.c** are the command/control and data transfer source code for the Rx and Tx hardware. These programs use standard C programming to configure and transfer data to/from the Rx and Tx hardware using the Gigabit Ethernet ports of host computers. The host computer connections to the Rx and Tx hardware may be a direct connection, or may be separated by several levels of routers.
The files **rxdiodebuild.lnx** and **txdiodebuild.lnx** are build scripts for Linux that will compile the files **rxdiode.c** and **txdiode.c** into the executables *rxdiode* and *txdiode,* respectively.
The configuration files **rxdiodeusr.cfg** and **txdiodeusr.cfg** are text files that can be edited to reset the Primary IP address and Target IP addresses of the Rx and Tx hardware respectively.
The program download/firmware files of **rxdiode.rpd** and **txdiode.rpd** are binary files that should NOT be edited. These files are used to reprogram the FPGAs of the Rx and Tx hardware as new features are added and all end in the extension of ***.rpd***.

rxinitdiode, **txinitdiode**, **rxdiode.jic** and **txdiode.jic** – Production testing and programming files. DO NOT ATTEMPT to use these files.
The C programs **rxdiode.c** and **txdiode.c** provides the user with three tools.
1) Ability to reprogram the Rx and Tx firmware with new updates.
2) Ability to reconfigure the Rx and Tx hardware with new configuration settings for different IP addresses.
3) The ability to send and receive data to/from host computers.
The **rxdiode.c** and **txdiode.c** programs can be built under Linux by using the build script in **rxdiodebuild.lnx** and **txdiodebuild.lnx,** respectively.
ALL terminal line commands listed in this document are performed in the directory rxdiodev*A_BC* or txdiode*vA_BC* unless otherwise specified.
To build the rxdiode.c source into the executable *rxdiode* type:
>>**source rxdiodebuild.lnx <return>**
The source file rxdiode.c is compiled and the executable r*xdiode* is produced.
To build the txdiode.c source into the executable *txdiode* type:
>>**source txdiodebuild.lnx <return>**
The source file txdiode.c is compiled and the executable *txdiode* is produced.

4.2 The *rxdiode* and *txdiode* Executables

Once built the executables *rxdiode* and *txdiode* provide several commands for the user. At the command line type:
>> **./rxdiode <return>**
The program returns a listing of available commands and their syntax:

```
rxdiodev1_00> ./rxdiode
Syntax:rxdiode <Operation Type> <Host IP Addr> <DIODERx Primary IP> <Path & Filename> <Rx Target Ip> <Redundant>

        <Operation Type>       = program, prgclrcfg, flashcfg, query, qrywcfg, flags, or transmitfile
        <Host IP Addr>         = Dot Notation Of Host Machine IP Addr, Ex. 192.9.200.110
        <DIODERx Primary IP>   = Dot Notation Of DIODERx IP Addr, Ex. 192.168.200.1
        <path & filename>      = Path & Filename Of Program or Config File(OPTIONAL) or Data File(MANDATORY)
        <DIODERx Target IP>    = Target IP Address Of DIODERx To Send Data-OPTIONAL For Transmitfile Operation
        <Redundant Tx>         = Number Of Times To Resend File Data-OPTIONAL For Transmitfile
        <Block Wait>           = Amount Of uSec To Wait Between Block Transfer-OPTIONAL For Transmitfile


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

>> **./txdiode <return>**

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

```
txdiodev1_00> ./txdiode
Syntax: txdiode <Operation Type> <Host IP Addr> <DIODETX IP Addr>  <Path & Filename> <MC Address> <file size>

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

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

Depending on the command used, the parameters to the executables *rxdiode* or *txdiode* can number between 3 and 7. **All** commands require the <command> , <Host IP Addr> and <Primary IP Addr> parameters. The parameters <command> , <Host IP Addr> and <Primary IP Addr> are the command to be executed, the host IP address and the Rx or Tx Primary 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>, <file size>, or <Rx Address>, <Rx Port> are used only for the *acquire* and *receivefile* commands of the Tx hardware, respectively. For acquiring data from the Tx hardware, <MC Address> is the multicast address from which the Host computer should receive data. <file size> is used for acquiring data from the Tx hardware and represents the size of the file to be acquired in 1k blocks. For the *receivefile* command <Rx Address> is the address to which the data is being SENT. In some cases this may be the host machine IP address that is listed as the parameter <Host IP Addr> in other cases this may be a multicast address-depending of how the Target IP address was specified during configuration. Finally, <Rx Port> is the UDP port number that the data is being sent to.

5.0 Commands

From the *./rxdiode <return>* listing, the available commands are *program, prgclrcfg, flashcfg, query, qrywcfg, flags, or transmitfile*.
To get syntax specific help on a command type:
>> **./rxdiode <Command> help <return>**

From the *./txdiode <return>* listing, the available commands are *program, prgclrcfg, flashcfg, query, qrywcfg, flags, acquire, acqwtc, acqall, receivefile.*
To get syntax specific help on a command type:
>> **./txdiode <Command> help <return>**

5.1 Rx & Tx Hardware *program* and *prgclrcfg* Commands

For the Rx hardware, at the command line, help on the *program* or *prgclrcfg* commands is available by typing:

>> ./rxdiode program help <return> OR >>./rxdiode prgclrcfg help <return>

Will display the following:

```
Syntax: rxdiode program <Host IP Addr> <DIODERX IP Addr>  <Path & Filename>

       program OR prgclrcfg   = firmware update utility (re-programs DIODERX with new firmware)
       <Host IP Addr>         = Dot Notation Of Host Machine IP Addr, Ex. 192.9.200.110
       <DIODERX Primary IP>   = Dot Notation Of DIODERX 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 DIODERX.
For the DIODERx, once programming is finished, the unit must be power cycled for the
new firmware to be loaded into the FPGA.
       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-along with the new firmware. Firmware
files for the DIODERX 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 DIODERx,
 the DIODERx will default to IP address 192.9.200.115
```

For the Rx hardware, upon completion of the 'prgclrcfg' operation the new firmware will be reloaded after a power cycle of the unit. Once reloaded, the Rx hardware will default to IP address 192.9.200.115

Example usage:

./rxdiode program 192.9.200.110 192.9.200.115 ./rxdiode.rpd <return>

For the Tx hardware, at the command line, help on the *program* or *prgclrcfg* commands is available by typing:

>> ./txdiode program help <return> OR >>./txdiode prgclrcfg help <return>

Will display the following:

```
Syntax: txdiode program <Host IP Addr> <DIODETX IP Addr>  <Path & Filename>

        program OR prgclrcfg   = firmware update utility (re-programs DIODETX with new firmware)
        <Host IP Addr>         = Dot Notation Of Host Machine IP Addr, Ex. 192.9.200.110
        <DIODETX Primary IP>   = Dot Notation Of DIODETX 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 DIODETX.
For the DIODETX, once programming is finished, the unit must be power cycled for the
new firmware to be loaded into the FPGA.
       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-along with the new firmware. Firmware
files for the DIODETX 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 DIODETX,
 the DIODETx will default to IP address 192.9.200.116
```

For the Tx hardware, upon completion of the 'prgclrcfg' operation the new firmware will be reloaded after a power cycle of the unit. Once reloaded, the Tx hardware will default to IP address 192.9.200.116

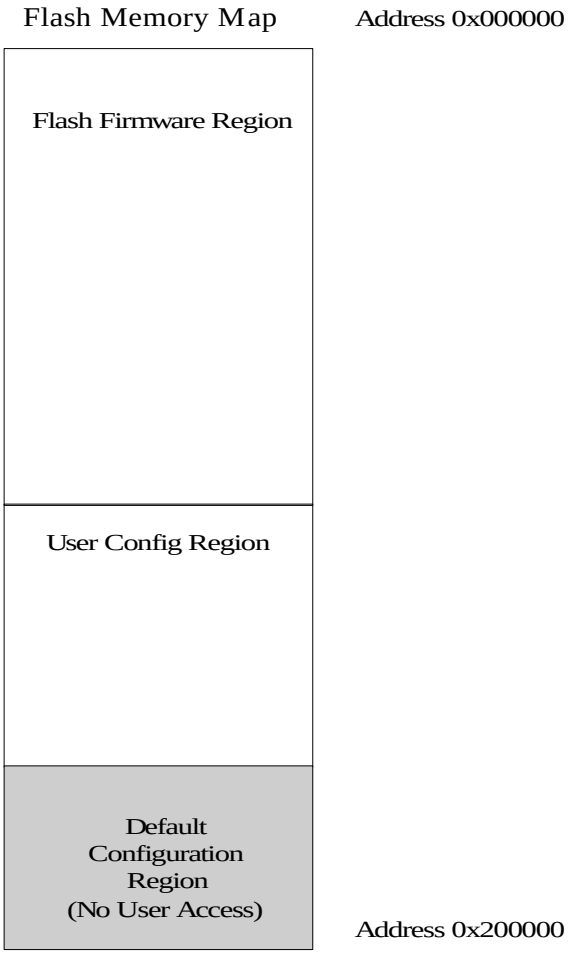
Example usage:

./txdiode program 192.9.200.110 192.9.200.116 ./txdiode.rpd <return>

As the previous software description explains, the *program* command downloads new firmware to the Rx or Tx hardware. In essence, as new features are developed, this is the way in which this new firmware with new features can be put into the Rx and Tx hardware. The *program* command simply overwrites the the old firmware in the flash memory of the Rx or Tx hardware. For the both the Rx and Tx hardware, the new firmware takes effect upon power cycle after completion of programming. 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 Primary IP and Target IP addresses. In other words, after using the *program* command the Rx or Tx hardware 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 Rx or Tx hardware to 192.9.200.115 or 192.9.200.116, respectively. Upon completion of the *prgclrcfg* command and a subsequent power cycle, the new firmware is loaded and any previous IP addresses that were set using the *flashcfg* command are gone and the Rx or Tx hardware now respond to the IP addresses of 192.9.200.115 or 192.9.200.116, respectively.



There is currently 1 download available to the user for each of the hardware sections. For the Rx hardware, rxdiode.rpd should be used as the firmware download file. For the Tx hardware, txdiode.rpd should be used as the firmware download file.

5.2 Rx Hardware *flashcfg* Command

For the Rx hardware, at the command line, help on the *flashcfg* command is available by typing:

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

```
Syntax: rxdiode flashcfg <Host IP Addr> <DiodeRx Primary IP>  <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
  <DiodeRx Primary IP> = Dot Notation Of DiodeRx 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 DIODERx.
This config info is used to configure DIODERx parameters such as Primary and Target IP
addresses.
Writing config info to flash allows the user to remove power from the DIODERx 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 DIODERx units
are programmed with default config before shipment) the DIODERx will default to IP address
192.9.200.115
```

Example usage:
./rxdiode flashcfg 192.9.200.110 192.9.200.115 ./rxdiodeusr.cfg <return>

The commands *program* and *prgclrcfg* are used to download firmware to the Rx hardware. The command *flashcfg* (flash config) is used to configure the Primary and Target IP addresses.

The *flashcfg* command performs configuration on the internall flash memory. The non-volatile nature of the internal flash means configuration is retained between power cycles.

As previously mentioned, new Rx hardware when delivered will arrive programmed with the firmware download rxdiode.rpd and have the Primary IP addresses 192.9.200.115 . A user may change this Primary IP address (and Target/Listening IP Addresses) using the *flashcfg* command. If, later, the user then changes firmware downloads by using the *prgclrcfg* command, the flash configuration area is erased. In erasing the configuration area, the Rx hardware will again default back to IP address 192.9.200.115

Example Listing of Configuration Text FIle rxdiodeusr.cfg For Rx Hardware (Included With Rx zip Software Package)

```
192.9.200.114                                   -- A. DIODERx Primary IP Address
0                                               -- B. DIODERx Primary VLAN Address
179.168.1.2                                     -- C. DIODERx Listening IP Address 1
224.1.2.3                                       -- D. DIODERx Listening IP Address 2
0.0.0.0                                         -- E. DIODERx Listening IP Address 3
0.0.0.0                                         -- F. DIODERx Listening IP Address 4
Flags=NONE                                      -- G. User Flags For Special Ops

****Nothing Read From File Below This Line (Documentation)****

--A.   DIODERx Primary IP Addr, Sets IP Addr For Communications, Programming, Config, Etc.
--B.   DIODERx VLAN Support, Set = 0, MUST BE SET EQUAL TO 0, NO VLAN SUPPORT--YET!
--C-F. DIODERx Listening IP Addresses That Receive Data From Outside Sources
--G.   User Flags For Special Ops, Comma Separated (NO Spaces), Currently NONE Supported
```

In the command line examples above, the ./rxdiode *flashcfg* command retrieves user values from a configuration file (rxdiodeusr.cfg), "packetizes" the configuration data contained in the file and then sends the packetized data to the Rx hardware via the Host machine Ethernet interface. When the configuration file (rxdiodeusr.cfg) is read by the *rxdiode* program, error checking is done on user values as it pertains to violations of the Rx 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 Rx hardware internal flash memory. The *rxdiode* 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 Rx hardware. 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 Rx hardware and the Rx hardware is not configured with the faulty values.

Below is the information that is returned to a user when the ./rxdiode *flashcfg* command is executed using the rxdiodeusr.cfg file that is shown above as the example configuration file.

./rxdiode flashcfg 192.9.200.110 192.9.200.115 ./rxdiodeusr.cfg <return>

```
Start Communications With DIODERx Interface...Done

Starting Operation: FLASHCFG
Config File rxdiodeusr.cfg Info :
        DiodeRx IP Addr = 192.9.200.114
        VLAN = 0
        DiodeRx Listening IP Adr #1 = 179.168.1.2
        DiodeRx Listening IP Adr #2 = 224.1.2.3
        DiodeRx Listening IP Adr #3 = 0.0.0.0
        DiodeRx Listening IP Adr #4 = 0.0.0.0
        Flags = NONE
        Firmware  Version = 1.00
Operating On User Config Space

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

Turning Off Incoming Data Stream ...Done
Reading Configuration Section (30 Secs)
Clearing Flash 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 Rx hardware with the IP address of **192.9.200.115** but after the *flashcfg* command the Rx hardware has a new IP address and can only be accessed via the NEW IP address of **192.9.200.114**

From the listing above, there are 7 parameters (A-G) that are used to control/configure the Rx hardware for user specific operation.

The syntax for the configuration file rxdiodeusr.cfg is fairly straight forward. All 7 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 C through F are Network addresses and should be written as such-"dot notation". Items B is currently unused and should be set to zero. Finally the flags field, item G, is also unused and should be set to Flags=NONE. Note once again that there are no spaces around the equals sign.

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

5.3 Rx Hardware Configuration File Parameter Description

In the configuration file rxdiodeusr.cfg which comes with the software zip package there are 7 values. These values are listed as items A through H At the bottom of the configuration file rxdiodeusr.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. Primary IP Address.

This is the IP address that the Rx hardware responds to for "pings" configuration and programming, only. Upon shipment of the Rx hardware portion of the Network Diode, rumeL, Inc defaults this address to 192.9.200.115. Using the *flashcfg* command, a user can change this address to any valid IP address they desire. In the example configuration file rxdiodeusr.cfg, included with the zip package, this address (as in the example above) has been changed to 192.9.200.114. After editing and saving the file, run the *rxdiode* program with the *flashcfg* command to reset the IP address (and all other parameters) to those listed in the configuration file.

B. VLAN Address

Some specifications, including the SDDS specification, includes 802.1Q VLAN extensions to "group" multicast and unicast data into VLANs for higher throughput through router networks. Currently the 802.1Q extensions for the Network Diode device (both Rx or Tx hardware) are not supported. At a later date, new software and firmware downloads will support this option. For now, this field MUST always be set equal to 0.

Finally, when the 802.1Q extensions are supported both the Rx and Tx hardware will be able to receive configuration or programming packets with 802.1Q VLAN extensions or without-either type will be valid.

C, D, E, F. Target/Listening IP Address #1, #2, #3, #4

These configuration items are the IP addresses that are listened to by the Rx hardware. Any UDP packet, with IP address C, D, E, F as its destination address will be forwarded across the fiber diode interface to the Tx hardware. Size of the packets do not matter, as long as they conform to gigabit ethernet standards-64 to 1536 bytes. Setting parameters C, D, E, or F to the IP address of 0.0.0.0 will disable the listening function for that particular configuration entry. IP addresses that are not 0.0.0.0 will enable listening for that configuration entry. Multicast addresses, addresses in the range of 224.1.0.0 to 239.255.255.255 (224.0.x.x addresses are reserved) will cause the Rx hardware to issue an IGMP join request and respond to IGMP queries. In other words, the Rx hardware handles the IGMP protocol messages to constantly allow for data reception from any Multicast address configured.

H. Flags

Currently there are no user flag settings. This entry should appear as:
Flags=NONE
No spaces before or after the '=' sign.

5.4 Tx Hardware *flashcfg* Command

For the Tx hardware, at the command line, help on the *flashcfg* command is available by typing:
>> **./txdiode flashcfg help <return>**

```
Syntax: txdiode flashcfg <Host IP Addr> <DiodeTx Primary IP>  <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
  <DiodeTx Primary IP> = Dot Notation Of DiodeTx 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 DIODETx.
This config info is used to configure DIODETx parameters such as Primary and Target IP
addresses.
Writing config info to flash allows the user to remove power from the DIODETx 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 DIODETx units
are programmed with default config before shipment) the DIODETx will default to IP address
192.9.200.116
```

Example usage:
./txdiode flashcfg 192.9.200.110 192.9.200.116 ./txdiodeusr.cfg <return>

The commands *program* and *prgclrcfg* are used to download firmware to the Tx hardware. The command *flashcfg* (flash config) is used to configure the Primary and Target IP addresses.

The *flashcfg* command performs configuration on the internal flash memory. The non-volatile nature of the internal flash means configuration is retained between power cycles.

As previously mentioned, new Tx hardware when delivered will arrive programmed with the firmware download txdiode.rpd and have the Primary IP addresses 192.9.200.116 . A user may change this Primary IP address (and Target/Listening IP Addresses) using the *flashcfg* command. If, later, the user then changes firmware downloads by using the *prgclrcfg* command, the flash configuration area is erased. In erasing the configuration area, the Rx hardware will again default back to IP address 192.9.200.116

Example Listing of Configuration Text FIle txdiodeusr.cfg For Tx Hardware (Included With Tx zip Software Package)

```
192.9.200.117                   -- A.  TxDiode Module IP Address
0                               -- B.  TxDiode Module VLAN Address
192.9.200.110                   -- C1. TxDiode Destination IP Address 1
0                               -- D1. TxDiode Destination VLAN 1
00:30:48:75:79:FC               -- E1. TxDiode Destination HW Address 1
0                               -- F1. TxDiode Source Port 1
29495                           -- G1. TxDiode Destination Port 1
224.1.2.3                       -- C2. TxDIODE Destination IP Address 2
0                               -- D2. TxDiode Destination VLAN 2
00:00:00:00:00:00               -- E2. TxDiode Destination HW Address 2
0                               -- F2. TxDiode Source Port 2
29495                           -- G2. TxDiode Destination Port 2
0.0.0.0                         -- C3. TxDIODE Destination IP Address 3
0                               -- D3. TxDiode Destination VLAN 3
00:00:00:00:00:00               -- E3. TxDiode Destination HW Address 3
0                               -- F3. TxDiode Source Port 3
29495                           -- G3. TxDiode Destination Port 3
0.0.0.0                         -- C3. TxDIODE Destination IP Address 4
0                               -- D3. TxDiode Destination VLAN 4
00:00:00:00:00:00               -- E3. TxDiode Destination HW Address 4
0                               -- F3. TxDiode Source Port 4
29495                           -- G3. TxDiode Destination Port 4
Flags=NONE                      -- H. User Flags For Special Ops

****Nothing Read From File Below This Line (Documentation)****

--A.  DIODETX Primary IP Addr, Sets IP Addr For Communications, Programming, Config, Etc.
--B.   DIODETx VLAN Support, Set = 0, MUST BE SET EQUAL TO 0, NO VLAN SUPPORT--YET!
--Cx. Target IP Address, IP Address To Send Data (Multicast Addresses Allowed)
--Dx. Target VLAN Value, VLAN (802.1Q) Not Support--Yet!! Set = 0
--Ex. Target HW Address, MAC/Ethernet Address That Corresponds To IP Address In Cx
--Fx. Sending Port Number, Port Number Tx Diode Sends Data From
--Gx. Target Port Number, Destination Port Number Where Data is To be Sent
--H.  Flags, None In Use-Yet Always Set Flags=NONE, No Spaces Around '='
```

In the command line examples above, the ./txdiode *flashcfg* command retrieves user values from a configuration file (txdiodeusr.cfg), "packetizes" the configuration data contained in the file and then sends the packetized data to the Tx hardware via the Host machine Ethernet interface. When the configuration file (txdiodeusr.cfg) is read by the *txdiode* program, error checking is done on user values as it pertains to violations of the Tx 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 Tx hardware internal flash memory. The t*xdiode* 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 Tx hardware. 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 Tx hardware and the Tx hardware is not configured with the faulty values.

Below is the information that is returned to a user when the ./txdiode *flashcfg* command is executed using the txdiodeusr.cfg file that is shown above as the example configuration file.

./txdiode flashcfg 192.9.200.110 192.9.200.116 ./txdiodeusr.cfg <return>

```
Start Communications With DiodeTx...Done

Starting Operation: FLASHCFG
Config File txdiodeusr.cfg Info :

        Tx Diode IP Addr = 192.9.200.117
        VLAN = 0

        Destination Info #1
        IP Adr = 192.9.200.110 HW Adr = 00:30:48:75:79:FC    VLAN = 0
        Source Port = 0               Destination Port = 29495

        Destination Info #2
        IP Adr = 224.1.2.3     HW Adr = 00:00:00:00:00:00    VLAN = 0
        Source Port = 0               Destination Port = 29495

        Destination Info #3
        IP Adr = 0.0.0.0       HW Adr = 00:00:00:00:00:00    VLAN = 0
        Source Port = 0               Destination Port = 29495

        Destination Info #4
        IP Adr = 0.0.0.0       HW Adr = 00:00:00:00:00:00    VLAN = 0
        Source Port = 0               Destination Port = 29495

        Flags = NONE
        Firmware  Version = 1.00
Operating On User Config Space

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

Turning Off Incoming Data Stream ...Done
Reading Configuration Section (30 Secs)
Clearing Flash 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 Tx hardware with the IP address of **192.9.200.116** but after the *flashcfg* command the Tx hardware has a new IP address and can only be accessed via the NEW IP address of **192.9.200.117**

From the listing above, there are 23 parameters (A,B,C1-G1,C2-G2,C3-G3,C4-G4,H) that are used to control/configure the Tx hardware for user specific operation.

The syntax for the configuration file txdiodeusr.cfg is fairly straight forward. All 23 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, C1, C2, C3, and C4 are Network IP addresses and should be written as such-"dot notation". Items B, D1, D2, D3, and D4 are currently unused and should be set to zero.

Items E1, E2, E3, and E4 are Hardware or MAC addresses for the corresponding Cx IP address. These values should be written as 6 hexadecimal numbers that are colon separated. Finally the flags field, item H, is also unused and should be set to Flags=NONE. Note once again that there are no spaces around the equals sign.

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

5.5 Tx Hardware Configuration File Parameter Description

In the configuration file txdiodeusr.cfg which comes with the software zip package there are 23 values. These values are listed as items A through H At the bottom of the configuration file txdiodeusr.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. Primary IP Address.

This is the IP address that the Tx hardware responds to for "pings" configuration and programming, only. Upon shipment of the Tx hardware portion of the Network Diode, rumeL, Inc defaults this address to 192.9.200.116. Using the *flashcfg* command, a user can change this address to any valid IP address they desire. In the example configuration file txdiodeusr.cfg, included with the zip package, this address (as in the example above) has been changed to 192.9.200.117. After editing and saving the file, run the *txdiode* program with the *flashcfg* command to reset the IP address (and all other parameters) to those listed in the configuration file.

B. VLAN Address

Some specifications, including the SDDS specification, includes 802.1Q VLAN extensions to "group" multicast and unicast data into VLANs for higher throughput through router networks. Currently the 802.1Q extensions for the Network Diode device (both Rx or Tx hardware) are not supported. At a later date, new software and firmware downloads will support this option. For now, this field MUST always be set equal to 0.

Finally, when the 802.1Q extensions are supported both the Rx and Tx hardware will be able to receive configuration or programming packets with 802.1Q VLAN extensions or without-either type will be valid.

Cx, Dx, Ex, Fx, Gx Target/Destination Network Config (where x is 1 to 4) Configuration parameters Cx, Dx, Ex, Fx, Gx represent the information needed to redirect UDP packets that were received via the fiber diode interface and will be resent via the Tx hardware gigabit ethernet. The Rx hardware accepts a UDP packet, strips the packet headers of data and fills them in with the values of 1, 2, 3, or 4 corresponding to which listening address it was received on. The stripped packet is then sent across the fiber diode to the Tx hardware. The Tx hardware receives the stripped packet and decodes the value of 1, 2, 3, or 4 that is placed in the packet header by the Rx hardware. The Tx hardware then

inserts the values that are listed as Cx, Dx, Ex, Fx, and Gx into the packet header, where x corresponds to 1, 2, 3, or 4 that is in the stripped packet header. Finally, once all of the packet header values are filled in, the Tx hardware sends the newly created packet header and data out the Tx hardware Gigabit Ethernet interface with the new packet header of:

IP address of Cx,
VLAN number Dx,
Hardware/Mac address Ex,
sending Port Fx,
and destination Port Gx.

Where x represents a value of 1, 2, 3, or 4. Note that at this time all VLAN values Dx should be set to 0. VLAN (802.1Q) support is not present—at this time.

Cx is the destination IP address. This value should be entered via network "Dot" notation. It corresponds to the Target/Destination IP address where data will be forwarded.

VLAN value Dx is currently not supported and should be set equal to 0

Hardware/MAC address Ex corresponds to the Hardware/MAC address of the IP address in Cx. This value should be listed as a 6 byte hexadecimal value that is colon separated. See the "Communications" section for more details on how to determine this value for a given IP address. If the destination IP address for Cx happens to be a Multicast address then the Hardware/MAC address Ex should be set to 00:00:00:00:00:00. Multicast IP addresses have unique Hardware/MAC addresses and the value Ex will be computed by the Tx hardware program *txdiode*. In other words, for Multicast Destination IP addresses Cx the user simply sets the Ex parameter to 00:00:00:00:00:00. Refer to the section of **Determining Hardware/MAC Addresses** for more information on how to determine this value.

The sending Port parameter Fx is added to the packet header as the sending port of the data. For SDDS specifications this value should be 0. But, users may want to set this value to known good sending ports that are common in there network. Or if running a single computer to interface to the Tx hardware, set the Fx parameter such that it does not interfere with other ports that are being used. Note, the Tx hardware program, *txdiode*, uses sending port 1300 for configuration communications. This value of 1300 should be avoided so as not to conflict with the *txdiode* program. Parameter Fx, the sending port value should be entered as a decimal value.

The destination port parameter Gx, is the port that data will be received on by a host machine connected to the Tx hardware Gigabit Ethernet connection. This port number is important and will be needed when transferring a file, or acquiring data via the Tx hardware software *txdiode*. The txdiode uses destination port value of 2300 for configuration communications. Therefore, use of this number should be avoided to prevent conflicts with the *txdiode* program. Parameter Gx, the destination port value should be entered as a decimal value.

H. Flags

Currently there are no user flag settings. This entry should appear as:
Flags=NONE
No spaces before or after the '=' sign.

5.6 Rx & Tx Hardware *query* and *qrywcfg* Commands

For the Rx hardware, at the command line, help on the *query* or *qrywcfg* commands is available by typing:
>> **./rxdiode query help <return> OR ./rxdiode qrywcfg help <return>**
Will display the following:

```
Syntax: rxdiode query <Host IP Addr> <DIODERX IP Addr>

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

The 'query' operation is used to 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.
```

The *query* command displays the current firmware download, the firmware version number, 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.

The *qrywcfg* (query with configuration) command will cause the same information to be reported back to the user, but once reported, the Rx hardware 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 the Rx hardware that was flash configured with the *flashcfg* command using the rxdiodeusr.cfg file in the zip package.

>> **./rxdiode query 192.9.200.110 192.9.200.114 <return>**

```
Start Communications With DIODERx Interface...Done
Starting Operation: QUERY
Turning Off Incoming Data Stream ...Done

*********************
* DIODERX Query Info *
*********************
DIODERx Hardware Type
      DIODERxR1
Firmware Download Info:
      Current Firmware Download - rxdiode.rpd
      Firmware  Version = 1.00
User Flash Config Values:
      DIODERx IP Address = 192.9.200.114
      VLAN Value = 0
      DIODERx Listening IP Addr #0 = 179.168.1.2
      DIODERx Listening Vlan #0 = 0
      DIODERx Listening IP Addr #1 = 224.1.2.3
      DIODERx Listening Vlan #1 = 0
      DIODERx Listening IP Addr #2 = 0.0.0.0
      DIODERx Listening Vlan #2 = 0
      DIODERx Listening IP Addr #3 = 0.0.0.0
      DIODERx Listening Vlan #3 = 0
      Flags = NONE
*********************

Turning ON Incoming Data Stream ...Done
```

For the Tx hardware, at the command line, help on the *query* or *qrywcfg* commands is available by typing:

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

Will display the following:

```
Syntax: txdiode query <Host IP Addr> <DIODETX IP Addr>

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

The 'query' operation is used to 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.
```

The *query* command displays the current firmware download, the firmware version number, 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.

The *qrywcfg* (query with configuration) command will cause the same information to be reported back to the user, but once reported, the Tx hardware 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 the Tx hardware that was flash configured with the *flashcfg* command using the rxdiodeusr.cfg file in the zip package.

```
Start Communications With DiodeTx...Done
Starting Operation: QUERY
Turning Off Incoming Data Stream ...Done

*********************
* DIODETx Query Info *
*********************
DIODETX Hardware Type
      DIODETxR1
Firmware Download Info:
      Current Firmware Download - txdiode.rpd
      Firmware  Version = 1.00
User Flash Config Values:
      DIODETx IP Address = 192.9.200.117
      VLAN = 0
      Flags = NONE

Tx Destination Address #1
      IP Address  = 192.9.200.110
      VLAN        = 0
      HW Address  = 30:00:75:48:fc:79
      Source Port = 0
      Dest Port   = 29495

Tx Destination Address #2
      IP Address  = 224.1.2.3
      VLAN        = 0
      HW Address  = 00:00:00:00:00:00
      Source Port = 0
      Dest Port   = 29495

Tx Destination Address #3
      IP Address  = 0.0.0.0
      VLAN        = 0
      HW Address  = 00:00:00:00:00:00
      Source Port = 0
      Dest Port   = 29495

Tx Destination Address #4
      IP Address  = 0.0.0.0
      VLAN        = 0
      HW Address  = 00:00:00:00:00:00
      Source Port = 0
      Dest Port   = 29495
*********************
Turning ON Incoming Data Stream ...Done
```

5.7 Rx Hardware *transmitfile,* and Tx Hardware *receivefile* Commands

The Rx hardware *transmitfile* and Tx hardware *receivefile* were designed to be used together to allow a user to transfer a file from the Rx (low side) to the Tx (high side) controlling host. These two commands work with a defined protocol that uses block data transfers, delays and duplicate transfers to provide for a high degree of success in transferring the entire file.

At the command line, help on the Tx hardware executable *txdiode* command *receivefile* is available by typing:
>> **./txdiode receivefile help <return>**
Will display the following:

```
Syntax: txdiode receivefile <Host IP Addr> <DIODETX IP Addr>  <Path Name> <Rx Address> <Rx Port>

       receivefile         = acquire data, sent by DIODERx code, from the DIODETX and store to file
       <Host IP Addr>      = Dot Notation Of Host Machine IP Addr, Ex. 192.9.200.110
       <DIODETX IP Addr>   = Dot Notation Of DIODETX  IP Addr, Ex. 192.168.200.1
       <path name>         = Path (SubDirectory) Where Data is to be Stored
       <Rx Address>        = IP Address Where DiodeTx is Sending Data
       <Rx Port>           = Receive (Destination) Port Number

The 'receivefile' operation opens a receive connection from the IP address of <Rx Address>
and the port number is specified as <Rx Port>.
Data acquisition is to file of filename contained in the transferred data. The file
is stored in the directory listed as <Path Name>.
```

For file transfers across the rumeL, Inc. Network Diode the *txdiode receivefile* command should be executed. The *receivefile* command takes as its parameters:

```
<Host IP Addr> <DIODETX IP Addr>  <Path Name> <Rx Address> <Rx Port>
```

<Host IP Addr> is the IP address (in network “dot” notation) of the host machine where the *txdiode* program is being executed.

<DIODETX IP ADDR> is the Primary IP address of the Tx hardware, in dot notation

<Path Name> is the subdirectory on the host machine where the user would like the data file stored after the transfer. There should be NO trailing “slash” after this parameter. Valid examples are /data11/folder or /home/data etc. An INVALID example would be /data11/folder/ .

<Rx Address> is the destination/target address of where the Tx hardware is sending the data. In a majority of the situations, this parameter will be the same as the <Host IP Addr> parameter-which is simply the IP address of the host system where the *txdiode* program resides and files are being stored to the

directory of <Path Name>. This parameter <Rx Address> is listed here and needs to entered for the case that this parameter may be a Multicast address that was configured as a target/destination IP address using the *flashcfg* command. This parameter should be entered in dot notation.

<Rx Port> is the destination port number to which the Tx hardware is sending data. It corresponds to the configuration settings that where entered, in combination with <Rx Address> using the *flashcfg* command.

The Tx hardware *txdiode receivefile* command should be started first before using the Rx hardware *rxdiode transmitfile* command. When started, the *txdiode receivefile* command runs continuously. It listens for connections to the IP address and port specified. When a connection is made, it acquires the data and stores it to a file. Then it moves back to its "listening" state waiting for the next connection. A count is maintained to show the number of files transferred and any losses of data for each particular transfer. See the sections titled **Address Re-Mapping** and **Network Diode File Transfer** for more information.

At the command line, help on the Rx hardware executable *rxdiode* command *transmitfile* is available by typing:
>> **./rxdiode transmitfile help <return>**
Will display the following:

```
Syntax:
rxdiode transmitfile <Host IP > <DIODERX CFG IP>  <Path & File> <DIODERX Targ IP> <Redundant Tx> <Blk
Wait>

       transmitfile          = transmits file to DIODERX then to DIODETX and store to file
       <Host IP Addr>        = Dot Notation Of Host Machine IP Addr, Ex. 192.9.200.110
       <DIODERX CFG IP Addr> = Dot Notation Of DIODERX Config IP Addr, Ex. 192.168.200.1
       <path & filename>     = Path & Filename For Data File To Transmit
       <DIODERX RX  IP Addr> = Dot Notation Of DIODERX Listening IP Addr, Ex. 192.168.200.2
       <Redundant Tx>        = OPTIONAL Number of Times To Resend Data Packet, Default = 2
       <Block Wait>          = OPTIONAL Amount of Wait Time (uSec) Between Block Send, Default = 500000

The 'transmitfile' operation opens a transmit connection to one of the four receive IP Addresses
of the DiodeRx. Once successfully opened the filetrans operation sends a file via UDP packets
to corresponiding open socket on the DiodeRx. Using the parameter <Redundant Tx> a user can
optionally specify the number or redundant transmits of data within the file to guarantee
the high probability of success for complete file transfer.  If not specified the default
number of redundant transmits is 2.  Which means the total number of times the file will be
transmitted is 3.
Also, to 'tune' the file transfer and to again provide for high reliability the parameter
<Block Wait> is provided as the number of uSec to wait between block transfers.
This block wait time allows the receiving disks time to store data.  The default
value for this parameter is 50000 uSec.
```

For file transfers across the rumeL, Inc. Network Diode the *rxdiode transmitfile* command should be executed **AFTER** the *txdiode receivefile* command has been executed on the Tx hardware software host.

The *transmitfile* command takes as its parameters:

```
<Host IP > <DIODERX CFG IP>  <Path & File> <DIODERX Targ IP> <Redundant Tx> <Blk Wait>
```

<Host IP Addr> is the IP address (in network “dot” notation) of the host machine where the *rxdiode* program is being executed.

<DIODETX Cfg IP> is the Primary IP address of the Rx hardware, in dot notation

<Path & File> is the subdirectory and filename on the host machine of the file to be transmitted. NOTE: the filename may NOT be more than 19 characters in length. The path name may be any length. Examples are: /data11/folder/test.dat or /home/data/archivedata.tmp , etc.

<DiodeRx Targ IP> is the destination/target address of the Rx hardware where the host is sending the data. This IP address is one of the 4 target/destination IP addresses that configured the Rx hardware using the *flashcfg* command. This parameter should be entered in dot notation.

<Redundant Tx> is the amount of times that blocks of data should be re-transmitted from the host to the Rx hardware. Blocks of data are typically 1 Megabyte in size. This parameter valid values are 0 to 10. The default value is 2 . The default value has been tested and shown to be highly successful when used.

<Block Wait> is the amount of time to wait between block data transfers from the host to the Rx hardware. This parameters value is measured in uSeconds. Valid ranges are 0 to 1000000. The default value is 50000 or 50000 uSeconds (50 mSeconds) between block transfers. This value can be used to “tune” the data transfer speed to accommodate slow disk write speeds at the Tx hardware host. The default value has been tested to show high reliability when data is being stored to a RAID capable of 320MBytes per second storage.

The *rxdiode transmitfile* command is used to transmit a single file across the Network Diode to a “waiting” host that is currently running the *txdiode receivefile* command. The *rxdiode transmitfile* command is a “one shot” command. It transfer the user file and then exits the program. For multiple transfers, the user must execute the command for each individual file transfer. Or, subsequently, write a “batch” program to execute multiple times for multiple file transfers.

5.8 Tx Hardware *acquire, acqwtc,* and *acqall* Commands

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

>> **./txdiode acquire help <return> OR ./txdiode acqwtc help <return> OR ./txdiode acqall help <return>**
Will display the following:

```
Syntax:
txdiode acquire <Host IP Addr> <DIODETX IP Addr>  <Path & Filename> <MC Address> <Rx Port> <file size>

       acquire OR acqwtc OR acqall  = acquire data from the DIODETX and store to file
       <Host IP Addr>               = Dot Notation Of Host Machine IP Addr, Ex. 192.9.200.110
       <DIODETX IP Addr>            = Dot Notation Of DIODETX  IP Addr, Ex. 192.168.200.1
       <path & filename>            = Path & Filename Data File Where Data is to be Stored
       <MC Address>                 = Multicast Address Of DiodeTx To Acquire Data
       <Rx Port>                    = Receive (Destination) Port Number
       <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*
**************************************************************
```

The *acquire*, *acqwtc*, and *acqall* commands are implemented in the *txdiode* program to acquire SDDS data-UDP packets with specific port numbers, size, and data headers. The use of these 3 commands have been implemented to work with any piece of hardware that generates the SDDS protocol. Special features have been added to acquire data from rumeL, Inc's GIGEXD (www.rumel-online.com/RI-GIGEXD.htm Note extension is .htm not .html) which generates SDDS network data from various analog and digital inputs. These commands are provided in the *txdiode* program to collect SDDS data and store it to a user specified file.

The *acquire* command allows a user to bring into a Host system SDDS 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 DIODETX (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 Target IP address of the DIODETX**
<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 Tx hardware.

It is important to note the multicast packets that is SDDS are UDP protocol. 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 wich 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 SDDS data is 8 bit or complex 8 bit samples the SDDS data will be BYTE represented. When the SDDS data is 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 SDDS source (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

The Sample rate of the data is a field in 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

A modulo 31 counter has been placed in each SDDS header portion of the 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.

6.0 Communications

From the parameters of the *rxdiode* and *txdiode* programs above, the user must know two facts before they can use the these programs:
1) The IP address of the Host Machine
2) The IP address of the Rx and Tx hardware.
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 Rx and Tx hardware has there IP addresses set at the time of delivery to a factory default value of 192.9.200.115 and 192.9.200.116, respectively. (These addresses can be changed by the user using the *rxdiode* and *txdiode* programs, and the *flashcfg* command to any IP address the user wishes-see the *flashcfg* command under the "Commands" section). Also the Rx and Tx hardware have a unique way of setting their IP addresses to the factory defaults in case the Rx and/or Tx hardware IP addresses are 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 Rx hardware is newly received and is programmed with it's default IP address of the 192.9.200.115. In order for the Host machine to "talk" to the Rx hardware 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 ".115" 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 Rx hardware with address **192.9.200.115** will go out on Host machine IP address 138.9.40.110 (eth0).
In summary, to "talk" to (configure, program, acquire data) the Rx and Tx software requires the IP address of the Host machine and the Rx or Tx hardware. Also, no more than 2 commands (*ifconfig* and *route*) on the Host machine are needed 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 192.9.200.115 <return>**

6.1 "Catch-22"

The Rx and Tx hardware are programmed, configured, and data accessed using their configured Primary IP addresses. The user without initial knowledge of Rx and Tx Primary IP addresses cannot change any parameters including the IP address. This presents a problem when the Rx and Tx Primary IP addresses are lost or forgotten. In this case, a *flashcfg* command cannot be sent to reset the IP addresses 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 Rx and Tx hardware are delivered their IP addresses are set to 192.9.200.115 and 192.9.200.116, respectively. The user may (and is encouraged) to change the IP addresses 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 Rx and Tx hardware assume the default addresses of 192.9.200.115 and 192.9.200.116 again if the new IP address is lost or forgotten. The hardware mechanism is the "Default" button. On the Rx and Tx hardware units, a small hole on the front of the units provides access to a button that can be pressed by using a "paper clip" inserted into the hole. (similar to a CDROM drive). Once the button is pressed, it must be held for 2 seconds. When held for 2 seconds the Rx or Tx hardware will default back to its factory default IP

addresses of 192.9.200.115 or 192.9.200.116, respectively, and any data flow will be turned off.

After the Rx or Tx hardware is returned to its factory default IP address of 192.9.200.115 or 192.9.200.116 a *query* command should be run. The *query* command will report back the Primary IP address .(along with other information about the firmware download, Target IP addresses, 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 the default IP addresses of 192.9.200.115 or 192.9.200.116 as the Rx or Tx Primary IP addressees and the Rx or Tx hardware 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 192.9.200.115 <return> OR ping 192.9.200.116 <return>
The Rx hardware will reply to the 192.9.200.115 address. The Tx hardware will reply to 192.9.200.116. Once the *query,* and then subsequent *qrywcfg* commands are run, the user can then confirm the switch back to their configured Primary IP addresses by typing:
>> ping <user IP Address> <return>

6.2 Maximizing Tx Hardware to Host Computer Data Transfer Rates

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

Below is a command that increases the Gigabit Ethernet 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. 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 10 Megabytes-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 txdiode.c software must be made to take advantage of the new size. At the top of the txdiode.c program file 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 txdiode.c should be recompiled -see the section **Build Scripts and the txdiode and rxdiode Executables** for compilation instructions.

6.3 Determining Hardware/Mac Addresses

When using the Tx hardware executable *txdiode* to configure the Tx hardware, the user for non-Multicast IP addresses must include the Hardware/MAC addresses in order for the Tx hardware to forward packets to the Target IP address. There are 2 methods to determine the Hardware/MAC.

The 1st method is to access the host that has the target IP address. Once logged in as root or super-user execute the command:

>> /sbin/ifconfig <return>

A listing similar to the following will appear:

```
eth0      Link encap:Ethernet  HWaddr 00:30:48:75:79:FC
          inet addr:192.9.200.110  Bcast:192.9.200.255  Mask:255.255.255.0
          inet6 addr: fe80::230:48ff:fe75:79fc/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:1 errors:0 dropped:0 overruns:0 frame:0
          TX packets:28 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:64 (64.0 b)  TX bytes:2894 (2.8 KiB)
          Base address:0x3000 Memory:dd300000-dd320000

eth1      Link encap:Ethernet  HWaddr 00:30:48:75:79:FD
          inet addr:192.9.200.210  Bcast:192.9.200.255  Mask:255.255.255.0
          inet6 addr: fe80::230:48ff:fe75:79fd/64 Scope:Link
          UP BROADCAST MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 b)  TX bytes:0 (0.0 b)
          Base address:0x3040 Memory:dd320000-dd340000

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:8 errors:0 dropped:0 overruns:0 frame:0
          TX packets:8 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:560 (560.0 b)  TX bytes:560 (560.0 b)
```

From the information above, the host computer has 2 Gigabit Ethernet ports: eth0 and eth1. If the user wants the Tx hardware to forward data to eth0 then, the user, would specify as one of the Target IP addresses the address 192.9.200.110. The Hardware/MAC address is listed under the heading of Hwaddr which is the value 00:30:48:75:79:FC These 6 bytes, colon separated would be entered as the configuration parameter Ex to correspond with Cx as the

Target IP address 192.9.200.110. The source port and destination ports are now arbitrary and can be set to any that are not currently in use-the user could even leave these set to the SDDS default of 0 for sending and 29495 as the destination.

The 2nd method of determining the Hardware/MAC address can be used if the user does not have login or root permissions on the host of interest to which to send data. This is the case where 1 host computer is being used for "configuration" and a 2nd host is being used to collect data. In this case using "ping" to obtain the Hardware/MAC address of the 2nd data host computer can be used.

In this example the configuration host computer is going to ping the data host computer. Assuming that the data host computer's IP address is 192.9.200.103 and the configuration and data hosts are connected via a network issue the ping command to the data host computer:

>> ping 192.9.200.103 <return>

Once the ping is successful, issue a 2nd command from the configuration host computer:

>> /sbin/arp <return>

The listing returned from the above command will look similar to the listing below:

```
Address                  HWtype  HWaddress           Flags Mask            Iface
192.9.200.1              ether   00:0E:84:BB:35:C0   C                     eth0
192.9.200.103            ether   00:16:41:E3:82:64   C                     eth0
```

From the listing of the ARP cache above, the data computer at IP address 192.9.200.103 has the Hardware/MAC address of 00:16:41:E3:82:64 . With the information of the IP address of 192.9.200.103 and the Hardware/MAC address of 00:16:41:E3:82:64 the *txdiode* configuration file (txdiodeusr.cfg) entries Cx and Ex can be filled in. Once configured by the configuration host computer, the Tx hardware of the Network Diode will forward data to the data host computer.

7.0 Address Removal and Re-Mapping

As mentioned in other sections of this document the Network Diode performs Address re-mapping of ALL UDP packets that are passed through the device. The Rx hardware is configured with 1 Primary IP address and up to 4 Target IP addresses. The Primary IP address is ONLY for configuration and programming access of the Rx hardware. No data sent to the Primary IP address will be forwarded across the fiber diode interface to the Tx hardware.

On the other hand, UDP packets, of any valid Gigabit Ethernet size (64 to 1536 bytes) will be forwarded if the packet's IP address matches 1 of the 4 Target IP addresses. When a match is made between packet IP address and one of the Rx hardware Target IP address, the packet's Ethernet, IP,and UDP header are cleared and replaced simply with the value of 1, 2, 3, or 4 for each of the fields in the Ethernet, IP, and UDP headers. The value of 1, 2, 3, or 4, that is inserted, corresponds to which Target IP address the received packet has matched when it was received by the Rx hardware. Once these values are replaced (Hardware/MAC address, IP destination and source addresses, UDP destination and source ports) the information about where the data originated is lost and the data can be moved from the low side to the high side circuitry via the fiber diode interface. (Refer to Figure 3)

Once across the fiber diode interface, the packet header that contains the value of 1, 2, 3, or 4 is read by the Tx hardware. The Tx hardware uses a look up table to access the NEW Hardware/MAC address, IP destination address, and UDP destination and source ports at the address of 1, 2, 3, or 4 and then inserts the new header information. Once the new addresses and port numbers are inserted into the header, the Tx hardware inserts its Primary IP address as the source IP address. Once all of the addresses and ports are inserted, the Tx hardware re-calculates checksums and inserts these into the packet. The packet is now complete and it is sent out the Tx hardware Gigabit Ethernet port to a network or host computer.

From the information above, the mapping between Rx hardware Target IP addresses and Tx hardware Target/Destination IP addresses is simply 1 to 1. As an example, if a packet is received and its destination IP address matches Target IP address number 1 of the Rx hardware, it's headers will be stripped and a 1 inserted for the fields of the header. The packet will then be sent over the fiber Diode interface where the Tx hardware will look up the Target/Destination addresses in location 1, insert these values, and send the packet out the Gigabit Ethernet port. A user simply has to maintain the 1 to 1 correspondence in their configuration. No information about the Rx hardware IP addresses must be known by the Tx hardware. The only information that has to be known is whether to configure the Tx hardware's Target/Destination of 1, 2, 3, or 4 with the information discussed above for the *flashcfg* command.

If the received packet had matched Target IP address 2 of the Rx hardware, then a stripped packet with a value of 2 in the header fields would have been passed through the fiber Diode interface. The Tx hardware would then have retrieved its stored Target/Destination values from location 2, inserted

these values, and then sent the packet out the Gigabit Ethernet port. Once again maintaining the 1 to 1 correspondence when Address Re-Mapping the packets.

As additional information the user should note, that what enters the Rx hardware as a unicast packet can be re-mapped on the Tx hardware to a new IP address that is in the multicast range. This type of re-mapping would then allow several computers to access what was previously a single stream of data. The reverse is also true. If a packet enters the Rx hardware as a multicast packet it can be re-mapped by the Tx hardware as a unicast packet. For some applications this change may be beneficial for data processing.

In summary, the packet address removal provides for information hiding in moving data from a non-secure to secure environment or visa-versa. Neither side has knowledge, or is presented information, that could compromise systems on the other side of the Network Diode.

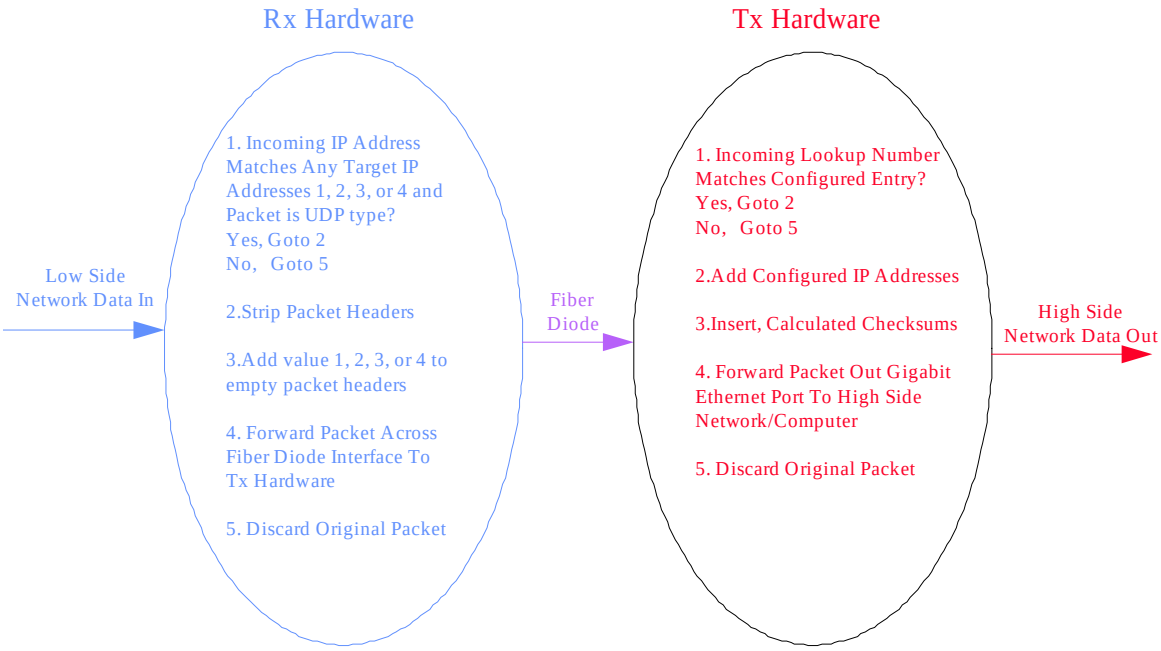


Figure 4 – Detail Of Address Re-Mapping

8.0 Network Diode File Transfer

As stated earlier, the Rx and Tx software programs rxdiode and txdiode provide the user with a file transfer mechanism. The *txdiode receivefile* command the the *rxdiode transmitfile* command work in cooperation to accomplish the sending and receiving of files through the Network Diode hardware.

To accomplish file transfer the Tx hardware *txdiode receivefile* command must be executed PRIOR to executing the Rx hardware *rxdiode transmitfile* command. The *txdiode receivefile* command takes as its parameters:

```
<Host IP Addr> <DIODETX IP Addr>  <Path Name> <Rx Address> <Rx Port>
```

The parameters as described earlier in the section **Rx Hardware *transmitfile* and Tx Hardware *receivefile* Commands** supply the subdirectory (<Path Name>) where the data is to be stored. The <Rx Address> and <Rx Port> parameters are 1 of the 4 target/destination IP addresses and destination ports that were configured using the *txdiode flashcfg* command. These represent the IP address and port number of the host that the Tx hardware will be sending data. Typically, the Target/destination IP address is the IP address of the host computer that holds the Tx hardware executable *txdiode*.

When the *txdiode receivefile* command executes it waits for a start of transfer from the Rx hardware *rxdiode transmitfile* command. The output from the *txdiode receivefile* appears as listed below while waiting for the start:

```
Start Communications With DiodeTx...Done
Starting Operation: RECEIVEFILE
```

When the start is received, the *txdiode receivefile* command prints to the screen the statistics about the file-filename, size, etc. The output appears as listed below:

```
************************************************************
File Transfer Number = 1
Received Valid Start Command, File Transfer Random Number = f9fe981a
DiodeRx Firmware  Version = 1.00
DiodeTx Firmware  Version = 1.00
Filename To Transfer = testramp.tmp
Number Of Full Blocks For Transfer          = 200
Packets Per Full Block                      = 1024
Data Size Of Packets                        = 1024 Bytes
Additional Bytes After Full Block Transfers = 512
File Transfer Size, Number Of 1K Blocks   = 204800, Plus 512 Bytes
File Will Be Stored As /data11/xmidas/testramp.tmp
```

Note that in the communications between the *rxdiode transmitfile* command the the *txdiode receivefile* command that a random number is passed that uniquely identifies this file transfer from all others. Also, the size of packets,

number of packets, number of packets per block, the number of blocks , and any left over bytes, that do not fit into the packet transfer size are listed. From the information listed the total file size will be 200Megabyte plus an additional 512 bytes. Total file transfer size will be 209715712 bytes. Finally, where the file will be stored and filename are listed. When the transfer is complete more information will be printed that appears as the following example:

```
Ending File Transfer, Closing File.
Total Packets Lost During File Transfer = 0
**********************************************************
```

Here the information that is reported back is the end of the transfer and the number of packets that were lost during transfer. **Any number but 0 reported as the number of packets lost represents a loss of data and the file transfer not completing without loss of data from within the file.**

The loss of packets is typically due 1 of 2 reasons, or both.

1) The Tx hardware host computer's Gigabit Ethernet interface buffer is too small and data is being lost at the OS level when buffered.
2) The Tx hardware host computer's disk I/O speeds when writing the file is too slow.

To solve for case 1 of a Gigabit Ethernet buffer overflow, be sure that the Communications section has been followed concerning setting the Gigabit Ethernet driver memory buffer to a larger size.

To solve for case 2 the Rx hardware *rxdiode transmitfile* command has to be “tuned” to transfer data at a slower rate to accommodate the slow I/O speed of the Tx hardware host system disks. Subsequent information on the *rxdiode transmitfile* command will solve case 2.

Finally here is an listing from 2 successful file transfers as printed by the *txdiode receivefile* command:

```
Start Communications With DiodeTx...Done

Starting Operation: RECEIVEFILE

************************************************************
File Transfer Number = 1
Received Valid Start Command, File Transfer Random Number = f9fe981a
DiodeRx Firmware  Version = 1.00
DiodeTx Firmware  Version = 1.00
Filename To Transfer = testramp.tmp
Number Of Full Blocks For Transfer          = 200
Packets Per Full Block                      = 1024
Data Size Of Packets                        = 1024 Bytes
Additional Bytes After Full Block Transfers = 512
File Transfer Size, Number Of 1K Blocks   = 204800, Plus 512 Bytes
File Will Be Stored As /data11/xmidas/testramp.tmp
Ending File Transfer, Closing File.
Total Packets Lost During File Transfer = 0
************************************************************
************************************************************
File Transfer Number = 2
Received Valid Start Command, File Transfer Random Number = fe3dab50
DiodeRx Firmware  Version = 1.00
DiodeTx Firmware  Version = 1.00
Filename To Transfer = testramp.tmp
Number Of Full Blocks For Transfer          = 200
Packets Per Full Block                      = 1024
Data Size Of Packets                        = 1024 Bytes
Additional Bytes After Full Block Transfers = 512
File Transfer Size, Number Of 1K Blocks   = 204800, Plus 512 Bytes
File Will Be Stored As /data11/xmidas/testsine.tmp
Ending File Transfer, Closing File.
Total Packets Lost During File Transfer = 0
************************************************************
```

At this point the *txdiode receivefile* command is still executing, waiting for another file transfer. If the user wishes to exit, simply use <Ctrl-C> to stop the program.

The *rxdiode transmitfile* command takes as input the following parameters:

```
<Host IP > <DIODERX Cfg IP>  <Path & File> <DIODERX Targ IP> <Redundant Tx> <Blk Wait>
```

The parameters as described earlier in the section **Rx Hardware *transmitfile* and Tx Hardware *receivefile* Commands** supply the *rxdiode transmitfile* command with the path and filename <Path & File> with the file to be transmitted through the Rx hardware. The Target IP address of the Rx hardware is given by <DIODERX Targ IP>. The last 2 parameters <Redundant Tx> and <Blk Wait> are OPTIONAL and are used for “tuning” the rate of the file transfer through the Network Diode to host computer connected to the Tx hardware. <Redundant

Tx> is the number of times that a block of data is retransmitted. A block of data consists of 1024 packets, where each packet carries 1024 bytes. Therefore, a block size is equal to 1 Megabyte of data (1048576 bytes). Once transferred once, the block of data will be re-transmitted <Redundant Tx> number of times. The valid values for <Redundant Tx> are 0 <= <Redundant Tx> <= 10. The Tx hardware *txdiode receivefile* command, recognizes redundant transfers and only stores the redundant data packets if initial transfer packet(s) did not get stored to disk. Therefore, the only data that gets stored to file is the original data. Redundant data is only used to fill in the “holes” of packets that were lost. In short, lower values of <Redundant Tx> will increase the rate of the file transfer but limit the *txdiode receivefile* command's ability to fill in lost packets on the other side of the Network Diode, thereby reducing the probability of a complete file transfer.

Once again, the parameter <Redundant Tx> is optional. The default value for <Redundant Tx> is 2. Through extensive testing the value of 2 has been shown to create very reliable file transfers.

The final parameter is <Blk Wait> (Block Wait). This parameter represents the amount of time, in microseconds (uSec), the *rxdiode transmitfile* command should wait between block transfers. <Blk Wait> is useful in “throttling” data bursts through the Network Diode to a waiting host computer that has slow disk I/O rates. The valid values for <Blk Wait> are 0 to 250000. Lower values of block transfer wait times will increase the rate of file transfer but will require the disk I/O speed of the host computer connected to the Tx hardware to be greater. Higher values of wait time will allow the host computer connected to the Tx hardware to perform disk I/O slower, thereby potentially eliminating the need for fast RAIDs.

And again ,the parameter <Blk Wait> is optional. The default value for <Blk Wait> is 50000. Through extensive testing the value of 50000 has been shown to create very reliable transfers.

Finally, the parameters <Redundant Tx> and <Blk Wait>, as mentioned above, are optional. For the *rxdiode transmitfile* command to function correctly either BOTH of the parameters must be specified by the user on the command line OR neither of the parameters are present. In the case where neither are specified, the defaults will be used.

Below is a listing of the rxdiode transmitfile command after sending the file testramp.txt:

```
Start Communications With DIODERx Interface...Done

Starting Operation: TRANSMITFILE
Total Packets Per Block           = 1024
Total Blocks For Transmission     = 201
Left Over Packets For Transfer    = 1
Packet Data Size                  = 1024 Bytes
Data Left Over For Single Transfer = 512 Bytes
Total Packets For Transmission    = 204801
Redundant Packet Transmissions    = 2
```

```
Wait 8 Seconds For Receive End File Deletion/Creation
1 2 3 4 5 6 7 8

Transfer Status: 1K Blocks Transmitted = 1024, Percent Complete = 0
Transfer Status: 1K Blocks Transmitted = 2048, Percent Complete = 0
Transfer Status: 1K Blocks Transmitted = 3072, Percent Complete = 1
Transfer Status: 1K Blocks Transmitted = 4096, Percent Complete = 1
Transfer Status: 1K Blocks Transmitted = 5120, Percent Complete = 2
Transfer Status: 1K Blocks Transmitted = 6144, Percent Complete = 2
....
....
....
Transfer Status: 1K Blocks Transmitted = 204801, Percent Complete = 100
File Transfer Complete
Total 1K Blocks Sent = 204801
```

Note that the default is for an 8 second wait after the *rxdiode transmitfile* command has informed the *txdiode receivefile* command that a file is to be sent. This delay is to allow time for the host that is executing the *txdiode receivefile* command to open and create the file. Depending on the size of the files, and if the user is **re-transmitting files to overwrite previous transmitted files** on the Tx hardware host computer, 8 seconds of time may not be enough for the Tx hardware host computer to delete, re-create, and open the file. In this case, packets will be lost. The user has the option of changing this wait time by modifying the rxdiode.c source code. If the user searches for the string, **STARTWAITTIME**, they will find the first occurrence to be the line below:

#define STARTWAITTIME 8

The STARTWAITTIME value is the number of seconds to wait after notifying the Tx hardware that a file is to be sent BEFORE sending the first data. Increasing this value will allow for more wait time so that the Tx hardware host computer will have enough time to delete a large file, re-create, and open a new file. After editing the rxdiode.c file, a user must remember to re-compile the source file rxdiode.c into the executable *rxdiode* by typing:

>> source rxdiodebuild.lnx <return>

Finally, remember that the need to modify the #define above will only occur if:

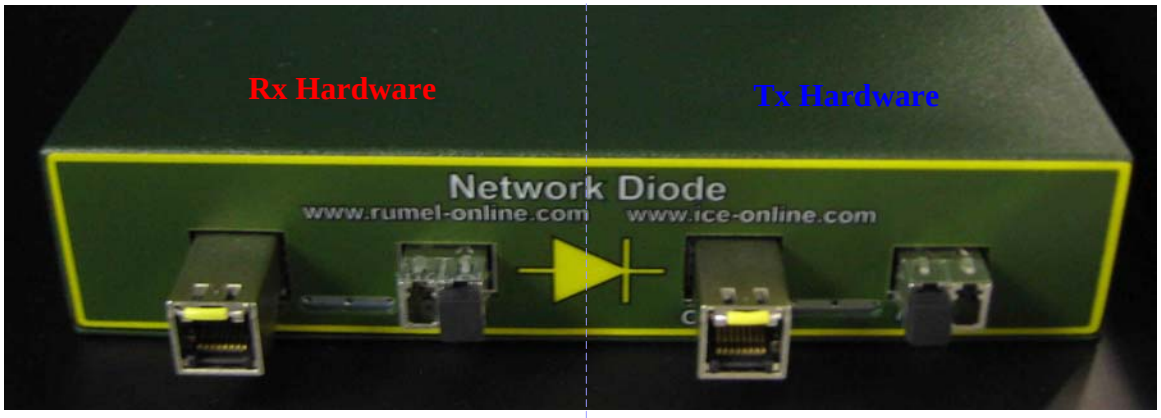
1) The file is large.
2) The file name already exists on the Tx hardware host computer and therefore an overwrite is occurring.

9.0 Security

From the detailed description of the operation of the Network Diode, a user can see that the Hardware/MAC addresses, IP addresses and port numbers are removed before transfer from the Rx hardware, across the fiber diode, to the Tx hardware. Other security features include separate power and ground connections. As shown in Figure 3, the Rx hardware and Tx hardware have separate power connections but what is not shown is that ALL ground connections are independent. Therefore, all current returns from the Rx hardware flows back through the ground of the Rx hardware power supply. Also, all current returns from the Tx hardware flows back through the ground of the Tx hardware power supply. Both pieces of hardware, Rx and Tx hardware, have independent power and ground connections. These separate connections allow for the Tx hardware to be connected to dedicated “high” side power and ground. Also, Rx hardware can be connected to dedicated “low” side power and ground.

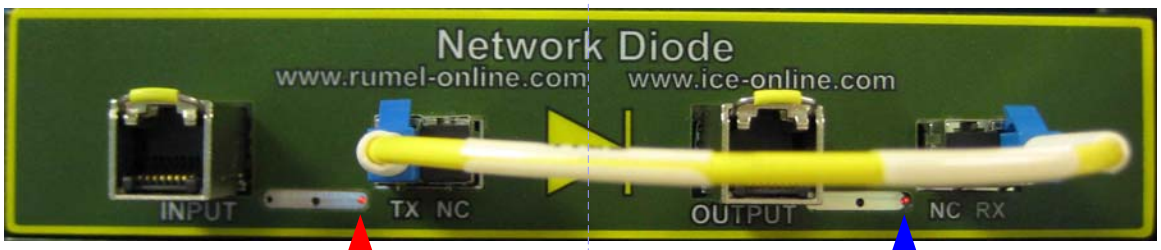
10.0 rumeL, Inc. Network Diode Pictures

Network Diode-Rx & Tx Hardware Separation (Fiber Diode Cable Not Installed)



Rx Hardware Gigabit Ethernet Interface | **Rx Fiber Diode (Tx Only)** | **Tx Hardware Gigabit Ethernet Interface** | **Tx Fiber Diode (Rx Only)**

Network Diode Fiber Diode Link LED (Fiber Cable Installed)



Rx Fiber Diode
Tx Link LED
OFF-No Link
RED-Link
GREEN-Active Tx

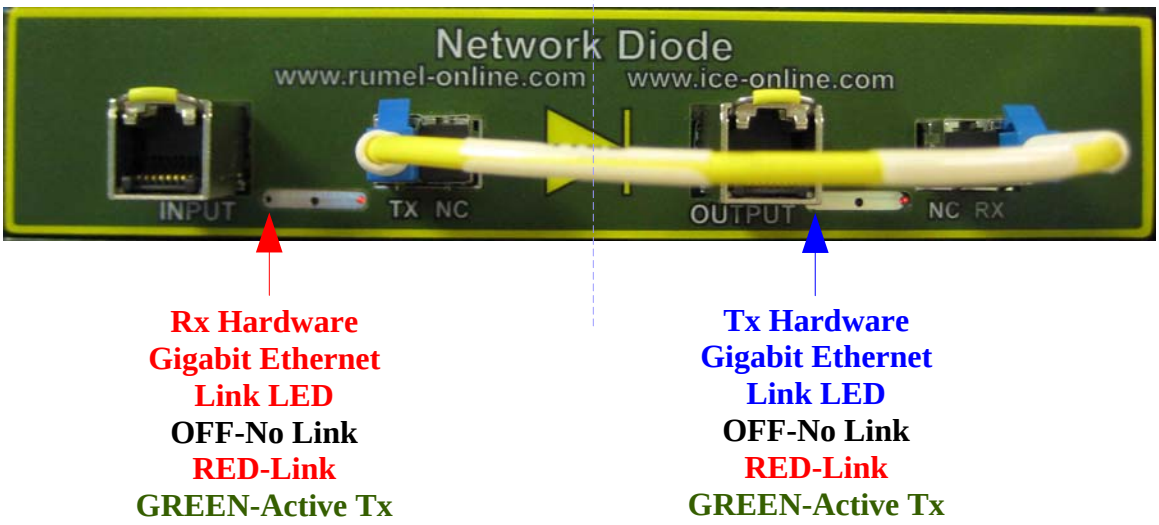
Tx Fiber Diode
Rx Link LED
OFF-No Link
RED-Link
GREEN-Active Rx

Rx & Tx Hardware Default Holes/Buttons

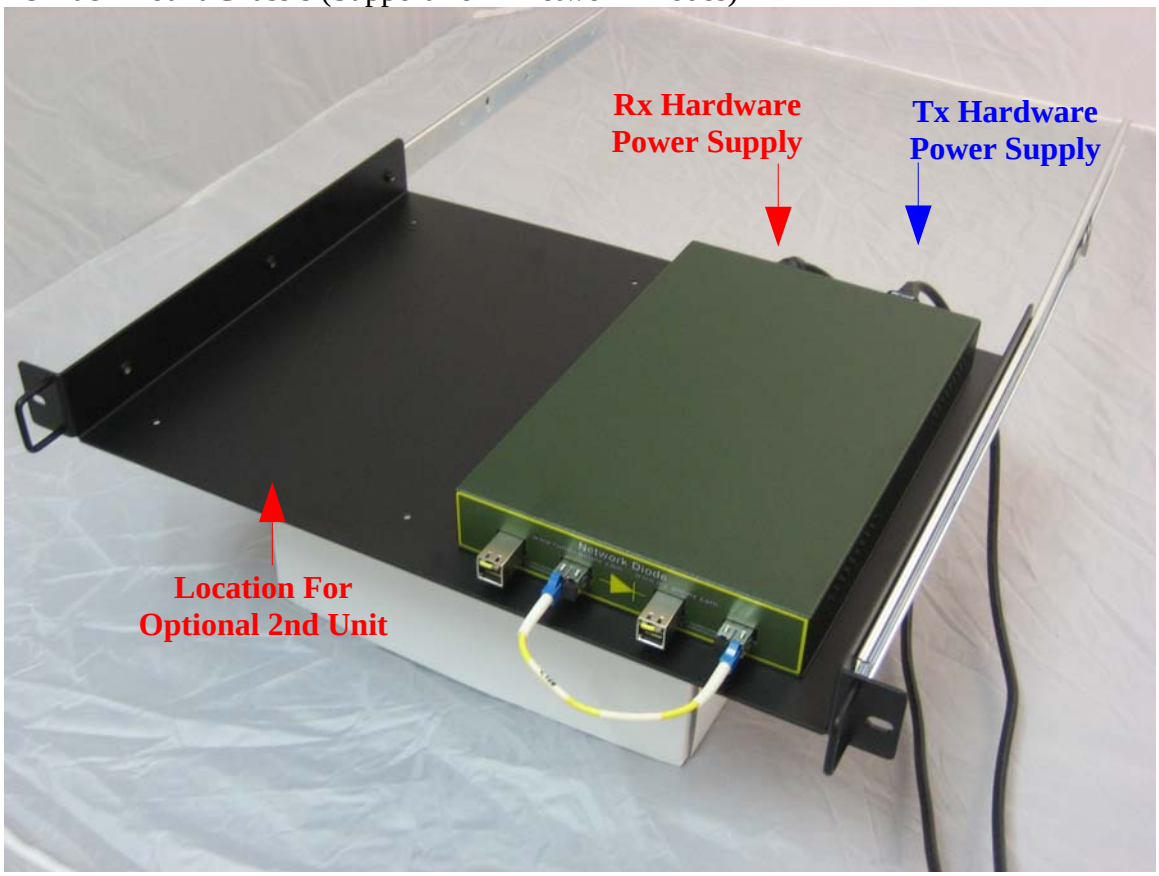
Rx Hardware Default Hole/Button

Tx Hardware Default Hole/Button

Rx & Tx Hardware Gigabit Ethernet Link LEDs



1U Rack mount Chassis (Support For 2 Network Diodes)



Internal Hardware, Power, Ground Isolation