8.2X Card User Guide

EDM01-19

Protection Against Harmful Interference

When present on equipment this manual pertains to, the statement "This device complies with part 15 of the FCC rules" specifies the equipment has been tested and found to comply with the limits for a Class A digital device, pursuant to Part 15 of the Federal Communications Commission [FCC] Rules.

These limits are designed to provide reasonable protection against harmful interference when the equipment is operated in a commercial environment.

This equipment generates, uses, and can radiate radio frequency energy and, if not installed and used in accordance with the instruction manual, may cause harmful interference to radio communications.

Operation of this equipment in a residential area is likely to cause harmful interference in which case the user will be required to correct the interference at their own expense.

Extra Components and Materials

The product that this manual pertains to may include extra components and materials that are not essential to its basic operation, but are necessary to ensure compliance to the product standards required by the United States Federal Communications Commission, and the European EMC Directive. Modification or removal of these components and/or materials, is liable to cause non compliance to these standards, and in doing so invalidate the user's right to operate this equipment in a Class A industrial environment.

Disclaimer

Website

www.endace.com

Copyright 2006-2007 Endace Technology Ltd. All rights reserved.

Contents

Introduction 1
Overview 1
Card Features 1
Purpose of this User Guide 2
System Requirements 2
General 2
Operating System 2
Other Systems 2
Card Description 3
Battery removal – don't do it! 3
Card Architecture 4
Line Types 5
Supported Line Types 5
Extended Functions 5
Data Stream Management 5
Inline Forwarding 6
Timed Release TERF (TR-TERF) 6
Installation 7
Introduction 7
DAG Software package 7
Inserting the DAG Card 7
Port Connectors 8
Pluggable Optical Transceivers 8
Overview 8
Optical modules 9
Power Input 9
Splitter Losses 9
Configuring the DAG card 11
Introduction 11
Before configuring the DAG card 11
Firmware images 11
Setting up the FPGA 12
Programming the FPGA 12
dagrom 13
Loading new firmware images onto a DAG Card 14
Preparing the DAG card for use 14
Configuring the DAG card 15
Display Current Configuration 15
Configuring Memory Allocation 16
dagconfig tokens explained 17
dagconfig options 23
Viewing the DAG card status 24
Interface Status 24
Extended statistics 24
Universal counters 25
Using your DAG card to capture data 27
Introduction 27
Basic data capture 27
Starting a capture session 27
dagsnap 28
Capturing data at high speed 29
Viewing captured data 30

dagbits 30
Converting captured data 32
Dagconvert 33
Using third party applications 34
Configuring DSM 34
Transmitting captured data 34
Configuration 34
Explicit Packet Transmission 35
Trace Files 35
TR TERF 35
Selectable CRC Length 35
Retransmitting Errored Packets 36
Usage Notes 36
TR TERF Known Issues 36

Synchronizing Clock Time 37

Overview 37
DUCK Configuration 37
Common Synchronization 37
Network Time Protocol 38
Timestamps 39
Example 39
Dagclock 40
Dagclock Statistics reset 41
Dagclock output explained 42
Card with Reference 44
Overview 44
Pulse Signal from External Source 44
Connecting the Time Distribution Server 44
Testing the Signal 44
Single Card No Reference 45
Two Cards No Reference 46
Overview 46
Synchronizing with Each Other 46
Synchronizing with Host 47
Connector Pin-outs 48
Overview 48
Pin Assignments 48

Data Formats 49

Overview 49
Generic ERF Header 50
ERF 2. TYPE_ETH 52
ERF 16. TYPE_DSM_COLOR_ETH 53
Extension Headers (EH) 54
Introduction 54

Troubleshooting 55

Reporting Problems 55

Version History 57

Introduction

Overview

The Endace 8.2X card provides the means to transfer data at the full speed of the network into the memory of the host computer, with zero packet loss in even worst-case conditions. Further, unlike a Network Interface Card (NIC), Endace products actively manage the movement of network data into memory while only consuming a minimal amount of the host computer's resources. The full attention of the CPU remains focused on the analysis of incoming data without a constant stream of interruptions as new packets arrive from the network. For a busy network link, this feature has a turbo-charging effect similar to that of adding a second CPU to the system.

The 8.2X card is a single port, PCIe card that allows capture and transmission of data. It is designed for use with 10 Gigabit optical Ethernet networks. It provides full-rate (10 000Mbps) capture and transmission of Ethernet traffic with precision timestamping capability.

Card Features

The following features are available on this DAG card. **Note:** Different firmware images may be required. Not all features are available on each firmware image. For further information on which feature is available in what firmware image, see Firmware images (page 11).

- 10 Gigabit Ethernet
- Timed Release TERF (TR-TERF)
- DSM

Purpose of this User Guide

The purpose of this User Guide is to provide you with an understanding of the 8.2X card architecture, functionality and to guide you through the following:

- Installing the card and associated software and firmware
- Configuring the card for your specific network requirements
- Running a data capture session
- Synchronizing clock time
- Data formats

You can also find additional information relating to functions and features of the 8.2X card in the following documents which are available from the Support section of the Endace website at www.endace.com:

- *EDM04-01 DAG Software Installation Guide*
- *EDM04-03 dagflood User Manual*
- *EDM04-04 dagfwddemo User Guide*
- *EDM04-06 Daggen User Guide*
- *EDM04-07 dsm-loader User Guide*
- *EDM04-08 Configuration and Status API Programming Guide*
- *EDM04-10 Data Stream Management API Programming Guide*
- *EDM04-19 DAG Programming Guide*
- *EDM05-01 Time Distribution Server User Guide*
- *EDM11-01 ERF types*
- *PN01-13 DAG Card Quick Start Guide*

This User Guide and the *EDM04-01 DAG Software Installation Guide* are also available in PDF format on the installation CD shipped with your 8.2X card.

System Requirements

General

The minimum system requirements for the 8.2X card are:

- A computer, with at least a Intel Xeon 1.8GHz or faster and a minimum of 1GB RAM.
- At least one free PCIe slot supporting 8-lane operation..
- Software distribution requires 60MB free space.
- For details of the supported operating systems, see one of the following documents:
  - *EDM04-01 DAG Software Installation Guide*
  - Current release notes - See the Documentation CD or the Endace support website at https://www.endace.com/support.

Operating System

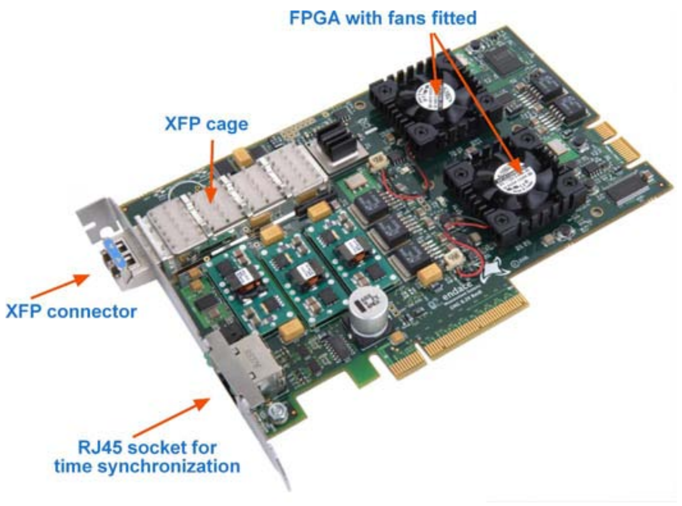
This document assumes you are installing the 8.2X card in a computer which already has an operating system installed. To install refer to *EDM04-01 DAG Software Installation Guide*. All related documentation is included on the CD shipped with the 8.2X card.

Other Systems

For advice on using an operating system that is substantially different from any of those specified above, please contact Endace Customer Support at support@endace.com.

Card Description

The 8.2X provides 1 full-duplex optical 10 Gigabit interface. It is capable of full-line rate (10 000Mbps) capture of Ethernet traffic. The 8-lane PCIe bridge enables a maximum 16 Gbit/s throughput which makes the 8.2X the first full rate 10 Gigabit Ethernet capture product.



The key features of the card are:

- User configurable single channel network monitoring for 10G Base-LAN networks.
- 100% capture into host memory at full line rate using DSM load balancing.
- Header only or variable length packet capture.
- 10 Gigabit Small Form Factor (XFP) pluggable full duplex network interface.
- Integrated Data Stream Management functions.
- Conditioned clock with one PPS input and local synchronization capability.
- 8 Lane PCIe bus interface.

Battery removal – don’t do it!

Removing the battery from a DAG card voids your warranty.

Removing the battery from a DAG card will cause the loss of encryption key used to decode the DAG card's firmware. Once the encryption key is lost the DAG card must be returned to Endace for reprogramming.

The battery in this product is expected to last a minimum of 10 years.

Caution

Risk of explosion if the battery is replaced by an incorrect type.
Dispose of used batteries carefully.

Card Architecture

The 8.2X uses a dual PCI-X to PCIe bridge which provides two internal PCI-X channels.

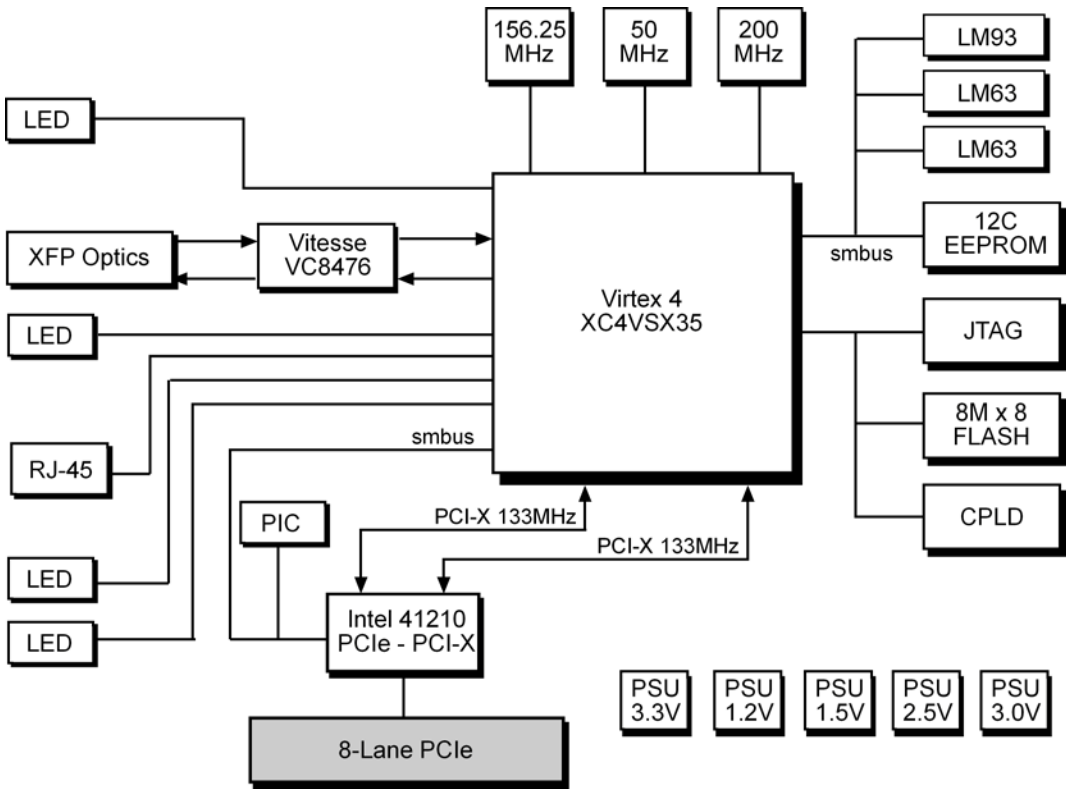
Serial Ethernet network data received by the XFP optical interface flow directly in to the Field Programmable Gate Array (FPGA).

The FPGA contains the packet record processor, the two PCI-X interfaces and the DAG Universal Clock Kit (DUCK) timestamp engine. The DUCK provides high resolution per-packet timestamps which can be accurately synchronized.

Note: For further information on the DUCK and time synchronization, see Synchronizing Clock Time (page 37) later in this user guide.

Because of component close association, packets are time-stamped accurately. Time stamped records are stored in an internal FIFO memory before transmission to the host.

The following diagram shows the card's major components and the flow of data:



Line Types

It is important that you understand the physical characteristics of the network to which you want to connect. If your configuration settings do not match your network, the 8.2X card will not function as expected.

There are various Ethernet line speeds and corresponding protocols which are identified using the IEEE naming convention. Each line speed has a set of requirements associated with it relating to the type of cable, maximum allowable distance, etc.

Note: If you are unsure about which of the options listed below to apply to your network, please contact your Network Administrator for further information.

Supported Line Types

The line types supported by the DAG 8.2Z card are described below.

Type	Description
10GBase-SR	10Gbps over multi mode fiber optic cable with short wavelength laser driver (850nm)
10GBase-LR	10Gbps over single mode fiber optic cable with long wavelength laser driver (1310nm)

Note: For more detailed information regarding Ethernet line types and speeds, please refer to IEEE Standard 802.3 available from the IEEE website at www.ieee.org.

Extended Functions

Data Stream Management

The 8.2X card supports the Data Stream Manager (DSM) feature. DSM allows you to drop packets or route them to a particular receive stream based on the packet contents, physical port and the output of two load balancing algorithms.

The DSM logic is implemented in firmware on the 8.2X card and does not require intervention from the host CPU once it is configured.

Note: For detailed information on using the Data Stream Manager please refer to *EDM04-10 Data Stream Management API Programming Guide* and *EDM04-07 dsm-loader User Guide* available from the support section of the Endace website at www.endace.com.

Filter / Load Balancing Block

Packets are received from the line and stamped with an ERF header, then passed to the filter and load balancing block. The filter block applies eight bit-mask filters simultaneously from the start of the packet, producing a single true/false value for each filter (64 bytes).

The load balancing block applies two algorithms to the packet data, also producing one true/false Boolean output per algorithm.

Lookup Table Block

The lookup table accepts the filter and load balancing outputs. It also receives the physical port the packet arrived on and calculates a classification (also known as color) for the packet.

Colorizer and Drop Block

The color is then passed to the Colorizer And Drop (CAD) block to check if the packet should be dropped. If not the color is inserted into the packet ERF record header and the packet record is passed to the packet steering module.

Packet Steering Module

The Packet steering module looks at the color information contained in the record and determines which receive stream the record should be routed to.

Inline Forwarding

The 8.2X card supports inline forwarding which enables the card to receive and transmit packets directly from a single memory. This allows you to forward packets from the DAG card receive interface(s) to the DAG cards transmit interface(s) without the requirement to copy them. Using inline forwarding you can receive, inspect, filter and forward packets between ports.

`dagfwddemo` which is a tool supplied with your DAG card demonstrates how you can apply a user-defined BSD Packet Filter (BPF) to the traffic forwarded by the DAG card. Packets which match the filter are forwarded, while packets that do not match are dropped.

For more detailed information on inline forwarding and using `dagfwddemo` please refer to the *EDM04-04 dagfwddemo User Guide* available from the support section of the Endace website at www.endace.com.

Timed Release TERF (TR-TERF)

The Timed Release TERF (TR-TERF) module is a option that enables you to transmit an ERF stream while reproducing the timestamps of the packets within that stream. It is able to transmit on all channels.

TR-TERF has two modes of operation. They are:

- No Delay Mode, and
- Relative Timed Release Mode.

Installation

Introduction

The 8.2X card operates on an 8 lane PCIe bridge and can be installed in any free PCIe slot.

DAG Software package

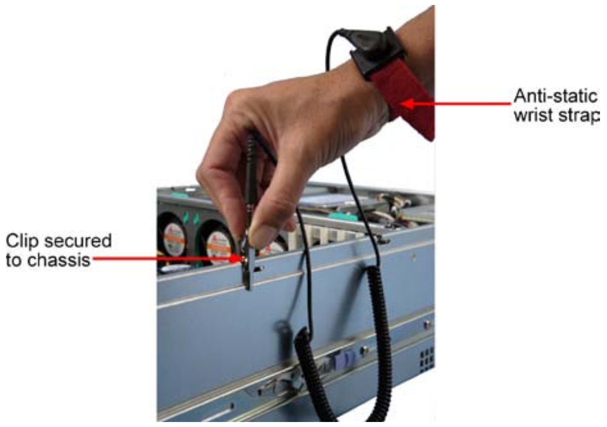
The latest DAG Software package must be installed before you install the 8.2X card itself. See *EDM04-01 DAG Software Installation Guide*, which is included on the CD shipped with the 8.2X card.

Inserting the DAG Card

Caution:

It is very important to protect both the computer and the 8.2X card from damage by electro-static discharge (ESD). Failure to do so could cause damage to components and subsequently cause the card to partially or completely fail.

1. Turn power to the computer OFF.
2. Remove the PCIe bus slot screw and cover.
3. Using an approved ESD protection device attach the end with the strap to your wrist and pull or clip firmly so there is firm contact with your wrist.
4. Securely attach the clip on the other end of the strap to a solid metal area on the computer chassis as shown below.



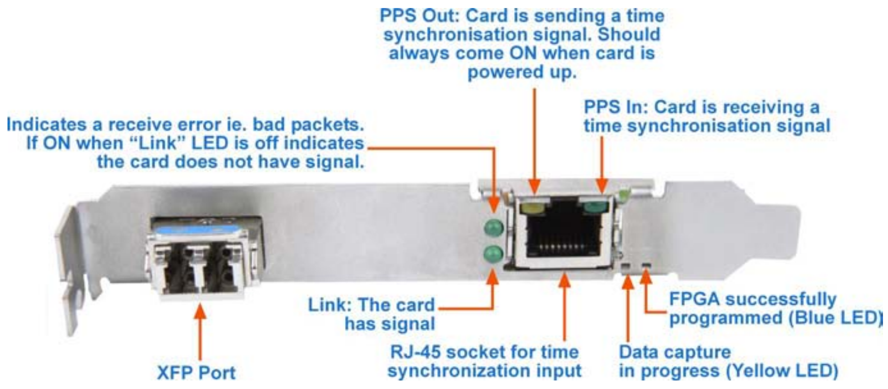
5. Insert the 8.2X card into PCIe bus slot ensuring it is firmly seated.
6. If this DAG card requires an external power supply, complete the following steps:
   a. Connect the supplied (or equivalent) power cable to the external power connector on the DAG card.
   b. Connect the cable to the appropriate power connector on your server's power supply unit.
7. Check the free end of the card fits securely into the card-end bracket that supports the weight of the card.
8. Secure the card with the bus slot cover screw.
9. Turn power to the computer ON.
10. Ensure the blue (FPGA successfully programmed) LED on the DAG card illuminates.

Port Connectors

The DAG 8.2Z has 1 XFP socket connector consisting of an optical fiber transmitter and receiver. The upper connector of the pair is used for the physical atransmit signal. The bottom connector used for the received signal.

There is an 8-pin RJ-45 socket located below the XFP connector on the PCIe bracket. This is available for connection to an external time synchronization source.

Caution: Never connect an Ethernet network or telephone line to the RJ-45 sockets.


PPS Out: Card is sending a time synchronisation signal. Should always come ON when card is powered up.
PPS In: Card is receiving a time synchronisation signal
Indicates a receive error ie. bad packets. If ON when "Link" LED is off indicates the card does not have signal.
Link: The card has signal
FPGA successfully programmed (Blue LED)
XFP Port
RJ-45 socket for time synchronization input
Data capture in progress (Yellow LED)

Pluggable Optical Transceivers

Overview

The DAG 8.2Z card uses an industry standard 10 Gigabit Small Form Factor Pluggable (XFP) optical transceiver.

The transceiver consists of two parts:

- Mechanical chassis attached to the circuit board
- Transceiver unit which may be inserted into the chassis

Note: You must select the correct transceiver type to match the optical parameters of the network to which you want to connect. Configuring the card with the wrong transceiver type may damage the card.

You can connect the transceiver to the network via LC-style optical connectors.

For further information on pluggable optical transceivers please refer to the Endace website at www.endace.com.

Optical modules

The optical power range depends on the particular XFP module that is fitted to the DAG card. Optics modules are supplied in either Single or Multi mode. See the following table for details.

Optical power is measured in dBm. This is decibels relative to 1 mW where 10 dB is equivalent to a factor of 10 in power. A negative optical power value indicates power that is less than 1 mW. The most sensitive devices can work at power levels down as low as –30dBm or 1µW.

The 8.2X card optics power module specification is shown below:

		Receive Characteristics				Transmit Characteristics		
		Wavelength (nm)		Sensitivity (dBm)		Wavelength	Tx Power (dBm)	
Product Number	Network Support	Min	Max	Min	Max	(nm)	Min	Max
TXR-10G 850 MM	10GBASE-LR Multimode	840	860	-11.1	-1	850	-7.3	*
TXR-10G 1310 SM	OC-192 (STM-64) & 10GBASE-LR/LW Single mode	1290	1330	-14.4	0.5	1310	-6	-1

Power Input

Note: The optical power input to the DAG card must be within the receiver's dynamic range. See the previous table for details. If it is slightly outside of this range it will cause an increased bit error rate. If it is significantly outside of this range the system will not be able to lock onto the signal.

When power is above the upper limit the optical receiver saturates and fails to function. When power is below the lower limit the bit error rate increases until the device is unable to obtain lock and fails. In extreme cases, excess power can damage the receiver.

When you set up the DAG card you should measure the optical power at the receiver and ensure that it is within the specified power range. If it is not, adjust the input power as follows:

- Insert an optical attenuator if power is too high, or
- Change the splitter ratio if power is too high or too low.

Splitter Losses

Splitters have the insertion losses either marked on their packaging or described in their accompanying documentation. General guidelines are:

- A 50:50 splitter will have an insertion loss of between 3 dB and 4 dB on each output
- 90:10 splitter will have losses of about 10 dB in the high loss output, and <2 dB in the low loss output

Note: Endace recommends that you do not use a combination of single mode and multi mode fibers and optics modules on the same link, as the quality of the received signal cannot be guaranteed.

If you have no choice but to mix single mode and multi mode you should be aware that a single mode input connected to a multi mode fiber will have some attenuation but may still be acceptable. However a multi mode input connected to a single mode fiber will likely have large and unpredictable losses.

Configuring the DAG card

Introduction

Configuring the 8.2X card ready for capturing data requires the following steps:

- Setting up the FPGA (page 12)
- Preparing the DAG card for use (page 14)
- Configuring the DAG Card (page 15)
- Viewing the DAG card statistics (page 24)

Once the 8.2X is configured you can start capturing data, see Using your DAG card to capture data (page 27) for details on capturing data.

Before configuring the DAG card

Before configuring the FPGA, you should ensure that:

- `dagmem` has been run and memory allocated to each installed DAG card.
- `dagload` has been run so that all DAG drivers have been installed.

Refer to the *Installing the drivers* section for the required Operating system in *EDM04-01 DAG Software Installation Guide* for the further details.

Firmware images

The following lists the features available on each firmware image available on this DAG card.

FPGA image (Software version string)	10 Gigabit Ethernet	DSM	TR-TERF
`dag82xpci_terf-tr-hmm.bit` (`dag82xpci_terf-tr-hmm_pci...`)	✓	✓	✓

The software version strings are displayed in the `dagconfig` output and when using the `dagrom -x` command. They include a version number and creation date.

Setting up the FPGA

All DAG cards have at least one Field-Programmable Gate Array (FPGA). The FPGA contains the firmware for the DAG card. The firmware defines how the DAG card operates when capturing data and contains the specific configuration.

Note: Some DAG cards have multiple FPGA's.

For each FPGA there are two firmware images:

- a factory image - contains fixed basic functionality for operating the DAG card.
- a user image - contains an upgradable version of the DAG card firmware. Additional functionality for the DAG card is available via the user image. Different user images may be available with different functionality, i.e. TERF, DSM etc.

Firmware images are loaded into DAG card flash ROM in the factory. The image is programmed into the FPGA each time the DAG card is powered up. The user image can then be programmed into the FPGA either manually or via a script.

Programming the FPGA

Before configuring the DAG card for capture, you must load and program the DAG card with the appropriate FPGA image.

Note: For information about the `dagrom` options, see dagrom (page 13).

Because the 8.2X uses a dual PCI-X to PCIe bridge which provides two PCI-X internal interfaces your system will recognize each interface as a separate DAG card. However you only need to load the FPGA image onto the first card.

- Load the FPGA image using:
  ```
  dagrom –rvp d0 –f dag82xpci_terf-tr-hmm.bit
  ```
  (where "0" is the device number of the DAG card you wish to capture data from)

Note: `dagrom` automatically resets the second card when the FPGA image is loaded onto the first card. This means you do not need to manually reset the second card.

dagrom

`dagrom` is a software utility that enables you to configure the FPGA on Endace DAG cards. The following is a list of options available in `dagrom`.

Option	Description
`-a,--alternate-half`	Use alternate (stable) half. [Default is current half.] Factory / User.
`-A,--entire-rom`	Entire ROM. [Default is current half only.]
`-b,--swid-rom-check`	Check if there is a SWID on the ROM.
`-c,--cpu-region <region>`	Access CPU region: c=copro, b=boot, k=kernel, f=filesystem.
`--continue`	Continue on erase error.
`-d,--device <device>`	DAG device to use.
`-e,--erase`	Erase ROM. [Default is read.]
`-F,--disable-cfi-fast`	Disable fast program option for CFI mode.
`-f,--file <filename>`	File to be read when programming ROM. There are multiple FGPA images per DAG card, covering the different versions, ERF, TERF DSM etc.
`--force`	Force loading firmware. Dangerous.
`-g,--rom-number <rom>`	Access specified ROM controller. [Default is 0.]

`-h,--help`
`-?,--usage`
	This page.

`-i,--halt-ixp`	Halt the embedded IXP Processor (DAG 7.1S only).
`--image-table-fpga <image table fpga>`	Specify the Power On image selection table FPGA number
`--image-table-image <image table image>`	Specify the Power On image selection table Image number
`-j,--swid-rom-check-key <key>`	Check the ROM SWID key with the one supplied.
`-l,--hold-bus`	Hold PBI bus from XScale (DAG 3.7T only).
`-m,--swid-key <key>`	Hexadecimal key for writing the Software ID (aka SWID).
`-o,--swid-rom-read`	Read SWID from ROM.
`-p,--program-current`	Program current User 1 Xilinx image into FPGA.
`-q,--image-number <image number>`	Specify the image number to write or to program the card.[0 - 3]. 0 factory image, 1 user image 1, 2 user image 2, 3 user image 3. (7.5G2/G4 only)
`--swid-write <swid>`	Write given SWID. The key must be supplied with the -m option, requires a valid running XScale ROM Image. (3.7T, 3.7D, 3.8S and 7.1S only)
`-r,--reprogram`	Reprogram ROM (may imply erase and write).
`--reset-method <reprogram method>`	Specify the method to reprogram the card.[1.Ringo 2.George 3.Dave]
`-s,--swid-rom-write <swid>`	Write given SWID to ROM. The key must be supplied with the -m option.
`-t,--swid-read-bytes <bytes>`	Read <bytes> of SWID, requires a valid running XScale ROM image (3.7T only)
`-u,--swid-erase`	Erase SWID from ROM.
`--unknown`	Force loading firmware. Dangerous.
`-v,--verbose`	Increase verbosity.
`-V,--version`	Display version information.
`-w,--write`	Write ROM (implies erase). [Default is read.]
`--write-out <filename>`	Write the contents of the ROM to a file.
`-x,--list-revisions`	Display Xilinx revision strings (the default if no arguments are given).
`-y,--verify`	Verify write to ROM.
`-z,--zero`	Zero ROM. [Default is read.]

All commands apply to the current image portion of the ROM, unless one of the options `-a`, `-A`, `-c` is specified.

Note: Not all commands are supported by all DAG cards.

To view the FPGA image revision strings, type the following:

```
dagrom -d0 -x
```

where "0" is the device number of the DAG card you wish to capture data from

```
user:       dag82xpci_terf-tr-hmm_pci_v2_7 4vsx35ff668 2008/02/11 10:38:14
(active)
factory:    dag82xpci_terf-tr-hmm_pci_v2_4 4vsx35ff668 2006/08/29 11:58:56
```

Loading new firmware images onto a DAG Card

New DAG card FPGA images are released regularly by Endace as part of software packages. They can be downloaded from the Endace website at https://www.endace.com/support.

Endace recommends you use the `dagrom -r` command when loading images from the computer to the ROM on the DAG card.

The `-r` option invokes a comparison of images on the computer and in the DAG card. Newer versions are automatically loaded onto the DAG card and programmed into the FPGA. See dagrom (page 13). This eliminates unnecessary reprogramming of the ROM and extends its life.

Preparing the DAG card for use

Before configuring the 8.2X card you must run the following `dagconfig` command to set the default parameters in the DAG card. This ensures the 8.2X card functions correctly once you begin capturing data.

Note: Ensure you run this command each time the FPGA is reprogrammed.

```
dagconfig -d0 default
```

where "0" is the device number of the DAG card you wish to capture data from.
The current 8.2X configuration displays and the firmware is verified as correctly loaded. See dagconfig (page 23) for more information.

Configuring the DAG card

Display Current Configuration

Once you have loaded the FPGA image you should run the `dagconfig` tool without arguments to display the current card configuration and verify the firmware has been loaded correctly.

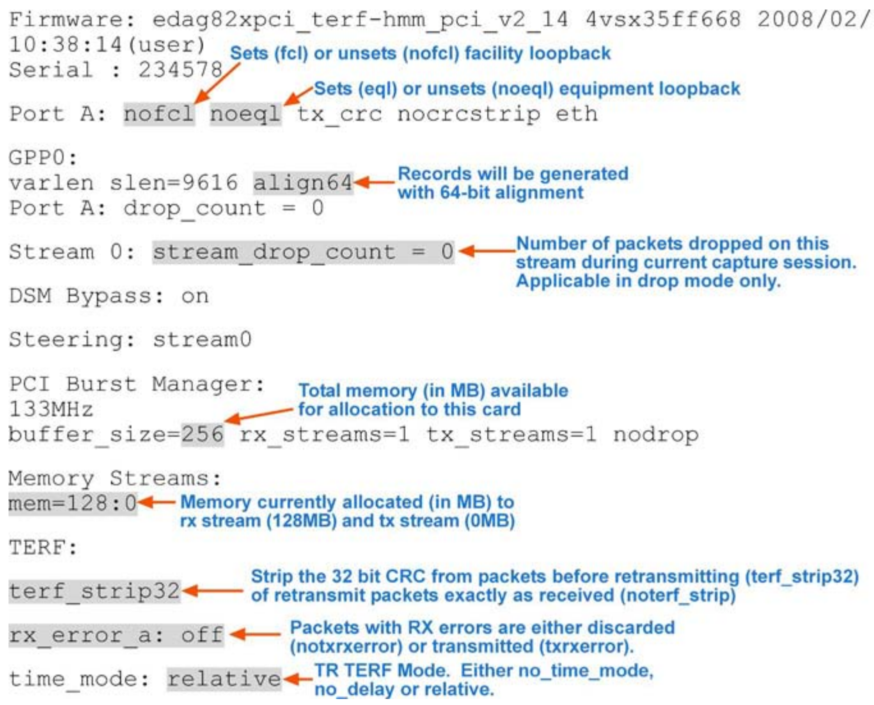
To display the default configuration for the first card, use:

```
dagconfig -d0 default
```

where "0" is the device number of the DAG card you wish to capture data from
A description of available tokens follows.

Note: Not all tokens displayed in the following diagram.

Display Current Configuration



```
Firmware: edag82xpci_terf-hmm_pci_v2_14 4vsx35ff668 2008/02/11
10:38:14(user)
Serial : 234578

Port A: nofcl noeql tx_crc nocrcstrip eth

GPP0:
varlen slen=9616 align64
Port A: drop_count = 0

Stream 0: stream_drop_count = 0

DSM Bypass: on

Steering: stream0

PCI Burst Manager:
133MHz
buffer_size=256 rx_streams=1 tx_streams=1 nodrop

Memory Streams:
mem=128:0

TERF:

terf_strip32

rx_error_a: off

time_mode: relative
```

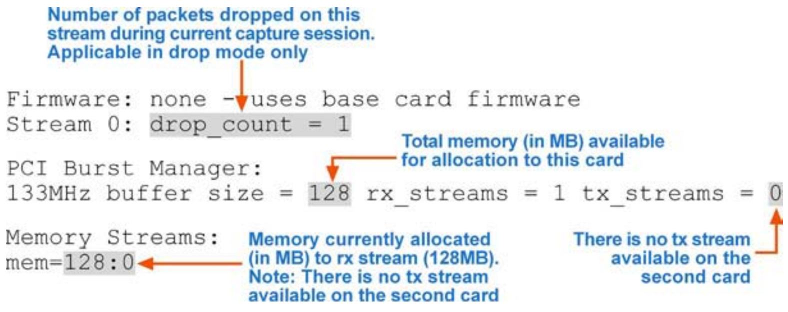
Sets (fcl) or unsets (nofcl) facility loopback
Sets (eql) or unsets (noeql) equipment loopback
Records will be generated with 64-bit alignment
Number of packets dropped on this stream during current capture session. Applicable in drop mode only.
Total memory (in MB) available for allocation to this card
Memory currently allocated (in MB) to rx stream (128MB) and tx stream (0MB)
Strip the 32 bit CRC from packets before retransmitting (terf_strip32) of retransmit packets exactly as received (noterf_strip)
Packets with RX errors are either discarded (notxrxerror) or transmitted (txrxerror).
TR TERF Mode. Either no_time_mode, no_delay or relative.

Note: Although your system recognizes each PCI-X interface as a separate card, all configuration is performed via the first card. The exception to this is the memory allocation, which can be configured separately on each card.
For further information on memory allocation please refer to Configuring Memory Allocation later in this chapter.

To display the configuration for the second card use:

```
dagconfig -d1
```

where "0" is the device number of the DAG card you wish to capture data from



```
Firmware: none -uses base card firmware
Stream 0: drop_count = 1

PCI Burst Manager:
133MHz buffer size = 128 rx_streams = 1 tx_streams = 0

Memory Streams:
mem=128:0
```

Number of packets dropped on this stream during current capture session. Applicable in drop mode only
Total memory (in MB) available for allocation to this card
Memory currently allocated (in MB) to rx stream (128MB). Note: There is no tx stream available on the second card
There is no tx stream available on the second card

Configuring Memory Allocation

You can configure the memory separately for each card. On the first card you can divide the memory allocation between the receive and transmit streams. However on the second card you can only allocate memory to a receive stream.

The amount of memory available to allocate to each card is displayed in the `buffer size` in the `dagconfig` output for each card as shown earlier.

Note: If you want to capture data at 100% of the line rate it is recommended that you allocate **all** available memory on **both** cards to the receive streams.
Allocating part of the memory on the first card to the transmit stream may affect your ability to capture at 100% of line rate. Allocating all of the memory on the first card to the transmit stream will allow you to transmit at approximately 50% of line rate.

dagconfig tokens explained

10gbase

Sets the Ethernet mode to one of the following modes. The wave length of the laser is changed.

- sr - short range
- lr - long range
- er - extended range

Example

```
dagconfig 10gbase_sr
dagconfig 10gbase_lr
dagconfig 10gbase_er
```

align64

Sets the packets to be all generated as multiples of 8 bytes, 64-bit aligned, (`align64`) before being received by the host. Not a configurable option.

buffer_size

The buffer size=*n*MB indicates that a total of *n* MB of memory have been allocated to the DAG card in total. Memory allocation occurs when the `dagmem` driver is loaded at boot time. See *EDM04-01 DAG Software Installation Guide* for details on how to allocate memory.

crcstrip/nocrcstrip

Don't include CRC in ERF record and wlen count.

Example

```
dagconfig crcstrip
dagconfig nocrcstrip
```

default

The `default` command initializes the DAG card configuration and sets all settings to default values. The command also resets the DAG card configuration back to its default state.

Note: When you run `dagconfig -d0 default` the `dagclock` inputs and outputs are also reset to defaults.

Example

```
dagconfig -d0 default
```

drop/nodrop

Determines if the DAG card's memory holes are de-coupled.

In `drop` mode, the memory holes are de-coupled. If the data rate on one memory hole slows, the data rate on any other memory holes is not affected. The dropping of packets occurs at the individual stream's burst manager.

In `nodrop` mode, the memory holes are coupled. The speed of the slowest memory hole determines the overall speed of the data rate. The dropping of packets occurs at the port.

Example

```
dagconfig drop
dagconfig nodrop
```

drop_count

Note: The `drop/nodrop` setting for the DAG card determines where the packets are being dropped.

Port/GPP (no drop)

The number of packets dropped during current capture session on this port. Resets to 0 when `dagconfig reset` is run.

Stream/Burst manager (drop)

The number of packets dropped during current capture session on the individual stream. Resets to 0 when a capture is started.

enable/disable

Sets whether this DAG card captures data on the defined port (a or b).

Note: DAG ports are enabled by default. You do not need to use `dagconfig` to enable the card in order to begin capture unless you have previously disabled it.

Example

```
dagconfig -d0 enablea
dagconfig -d0 disablea
dagconfig -d0 enableb
dagconfig -d0 disableb
```

where "0" is the device number of the DAG card you wish to capture data from

Note: On some firmware images changes to this option may not take effect.

eql/noeql

Sets or unsets equipment loopback. For testing set to `eql` mode and normal operation set to `noeql` mode.

Note: `eql` mode loops transmit data from the host back to the PCIe bus.

Example

```
dagconfig eql
dagconfig noeql
```

eth

Sets framer into ETH receive mode. Image dependent - only operates in Ethernet images

Example

`dagconfig` eth

fcl/nofcl

Note: Sets or unsets Facility loop back. For testing set to `fcl` mode and normal operation set to `nofcl` mode.

FCL retransmits the data received and also send it to the host.

Example

```
dagconfig fcl
dagconfig nofcl
```

mem

You can split the DAG card's allocated memory between the receive and transmit stream buffers to suit your own requirements. The split is displayed as a ratio as shown below:

```
mem=X:Y
```

where:
`X` is the memory allocated in MB to the `rx` stream
`Y` is the memory allocated in MB to the `tx` stream.

If there are multiple `rx` or `tx` streams memory can be allocated to each stream:

```
mem=X:Y:X:Y:X:Y
```

Buffer_size (page 17) and rx and tx Streams are related to `mem`.

Example
You can split 128MB of memory evenly between the `tx` and `rx` streams using:

```
dagconfig -d0 mem=64:64
```

Note: You can not change the stream memory allocations while packet capture or transmission is in progress.

overlap/nooverlap

Configures the `rx` and `tx` memory hole to be overlapped. This enables in-line forwarding without copying the data across the memory holes.

Example
```
dagconfig overlap
dagconfig nooverlap
```

Note: This option is only applicable on firmware images containing TX.

PCI

Describes the following information about the DAG card:

- The type of PCI used by the DAG card (PCI, PCIx or PCIe)
- Bus speed
- Bus width

Example
```
pcix  133MHz  64-bit
pci  33MHz  32-bit
pcie 8 Gbs 4Lane
```

reset/noreset

Resets the images to the default state. Clears the Port drop counters. All settings done before are kept in place.

Example
```
dagconfig reset
dagconfig noreset
```

rxonly

Configures the memory hole to only receive.

Example
```
dagconfig rxonly
```

rxtx

Enables both transmit and receive, and splits the memory hole for `rx` and tx.

This allocates 16MB of memory to each transmit stream, and divides the remaining memory between the receive streams.

Example

```
dagconfig rxtx
```

Note: This option is only applicable on firmware images containing TX.

slen

Before you begin to capture data you can set the size that you want the captured packets to be. You can do this using the `dagconfig` tool to define the packet snaplength (`slen`).

Note: The snaplength value must be a multiple of 8 and in the range 48 to 9600 per card inclusive.

By default, `slen` which is the portion of the packet that you want to capture is set to 48 per card. This means that only the first 48 bytes of each packet will be captured.

If for example you want to capture only the IP header of each packet you may want to set the length to a different value. Alternatively if you want to ensure you capture the whole packet you can set the length to a larger value.

Example

Setting up a 8.2X card with a snap length of 200 bytes:

```
dagconfig -d0 slen=200
```

Note: The ERF header is not included in the `slen` value. Therefore a `slen` of 48 will produce a 64-byte capture record made up of 48 bytes plus the number of bytes in the ERF header.

steer

The algorithm to use to steer the incoming packet.

Option	Description
`stream0`	All traffic sent to stream 0 (3.8 balance f/w only)
`parity`	Load balance traffic by parity (3.8 balance f/w only)
`crc`	Load balance traffic by crc (3.8 balance f/w only)
`iface`	Port A to stream 0, Port B to stream 2 (3.8 balance f/w only)
`colour/dsm`	Load balance traffic by colour/dsm. Colour is applicable to IPF traffic.

terf_strip16/terf_strip32/noterf_strip

Strips the CRC value (32 bits) from the packet or sends packet “as is” (`noterf_strip`). The TERF line in the current configuration indicates the current Terf option.

Note: Only displayed if the DAG card supports transmit (i.e. has a terf image).

Example

```
dagconfig terf_strip32
dagconfig noterf_strip
```

time_mode

Sets the time release mode. `nodelay` sends packets out with no delay. Shows as `no_time_mode`. `relative` mode sends out packets in the same time spacing as they where received. Applicable to TR-TERF.

Example

```
dagconfig nodelay
dagconfig relative
```

relative

Sets TR-TERF to Relative Timed Release mode.

In Relative Timed Release Mode, each packet is transmitted according to its timestamp, relative to the first packet in the transmit stream. This is shown in the following example where Packet A will be transmitted immediately, Packet B will be transmitted one second later and Packet C will be transmitted one second after that:

```
Packet A: Timestamp = 2006-07-24 03:45:55.9720326 UTC
Packet B: Timestamp = 2006-07-24 03:45:56.9720326 UTC
Packet C: Timestamp = 2006-07-24 03:45:57.9720326 UTC
```

If for any reason the transmission of Packet B is delayed, the module will still attempt to transmit Packet C two seconds after Packet A. It will do this even if it means reducing the gap between Packet B and Packet C.

When using multiple output ports in this mode the timing for each port is independent. This is shown in the following example where Packet A and Packet B are transmitted at the same time even though their timestamps are one second apart. **Note:** The relative placement of the packets is accurate to ± 100ns.

```
Packet A (Port A): Timestamp = 2006-07-24 03:45:55.9720326 UTC
Packet B (Port B): Timestamp = 2006-07-24 03:45:56.9720326 UTC
```

Example
```
dagconfig relative
```

no delay

Disables Relative Timed Release mode (No Delay Mode).

In No Delay Mode all packets are transmitted at the maximum rate permitted by the line interface, using only minimum gaps between packets.

Example
```
dagconfig nodelay
```

tx_crc

Enables the crc(fcs) in the transmitted packets CRC(FCS). Ethernet only.

Example
```
dagconfig tx_crc
dagconfig notx_crc
```

txonly

Configures the memory hole to only transmit.

Note: Only displayed if the DAG card supports transmit (i.e. has a terf image).

Example
```
dagconfig txonly
```

Note: This option is only applicable on firmware images containing TX.

txrxerror

Indicates how to process packets that are transmitted with an **RX error** flag. Packets are either discarded `(notxrxerror) on` or transmitted `(txrxerror).`

Note: Only displayed if the DAG card supports transmit (i.e. has a terf image).

Example
```
dagconfig txrxerror
dagconfig notxrxerror
```

varlen/novarlen

The 8.2X card is able to capture packets in two ways. They are:

- Variable length capture (`varlen`)
- Fixed length capture (`novarlen`) - (not support on some firmware images)

In **variable length** (`varlen`) mode, the DAG card will capture the whole packet, providing its size is less than the `slen` value. Therefore to use this capture mode effectively you should set the `slen` value to the largest number of bytes that a captured packet is likely to contain. For more information on snaplength, see slen (page 20).

Any packet that is larger than the `slen` value will be truncated to that size. Any packet that is smaller than the `slen` value will be captured at its actual size therefore producing a shorter record which saves bandwidth and storage space.

Example

The example below shows a configuration for variable length full packet capture:

```
dagconfig -d0 varlen
```

In **fixed length** (`novarlen`) mode the card will capture all packets at the same length. Any packet that is longer than the `slen` value will be truncated to that size, in the same way as for `varlen` capture. However any packet that is shorter than the `slen` value will be captured at its full size and then padded out to the size of the `slen` value.

This means that in `novarlen` mode you should avoid large `slen` values because short packets arriving will produce records with a large amount of padding which wastes bandwidth and storage space.

Note: Using the `novarlen` option on DAG cards with an on-board Co-Processor (accelerated cards) is not recommended. It may cause excessive loss of packets.

Version information

Details the following information about the connected DAG card:

- Firmware image programmed in the FPGA
- The DAG card serial number
- The MAC address(s) of the DAG card's ports (ethernet cards only).

dagconfig options

`dagconfig` is a software utility used to configure and display statistics.

By default all commands, unless otherwise defined, run on device 0 (`-d0`). Commands only apply to one DAG card.

The following is a list options available in `dagconfig`. Not all options listed are applicable to all cards.

Options:	Description
`-1,--porta`	Port A only (default all). Multi-port cards only.
`-2,--portb`	Port B only (default all). Multi-port cards only.
`-3,--portc`	Port C only (default all). Four-port cards only.
`-4,--portd`	Port D only (default all). Four-port cards only.
`--porte to --portp`	As above, for extra ports on the 3.7T DAG card.
`-c,--concfg <conncfg>`	Connection configuration. Used by the DAG 7.1S only.
`-C,--counters`	Outputs the counters. Verbosity levels from 0=(basic / default) to 3=(full).
`-d,--device <device>`	DAG device to use. Default is d0.
`-e,--extended`	Displays the current extended statistics (non boolean and image dependant). Verbosity levels from 0=(basic / default) to 3=(full). Note: Some images may not contain extended statistics.
`-G,--getattribute <getattribute>`	Gets individual attributes by attribute name. Use in conjunction with the --porta or --portb options to get individual only multi-port cards.
`-h,--help`	Displays the MAN pages. The information displayed is dynamically based on the DAG card and does not work correctly when there is no DAG card in the system. **Note:** There are a few commands that display even though they are not applicable.
`-i,--interval <seconds>`	Interval to repeat in seconds.
`-m,--hmon`	Outputs the hardware monitor information.
`-n,--voltages`	Outputs the DAG card voltage monitor information.
`-S,--setattribute <setattribute>`	Sets individual attributes by attribute name. Use in conjunction with the --porta or --portb options to get individual only multi-port cards.
`-s,--statistics`	Outputs the statistics for the DAG card. Verbosity levels from 0=(basic / default) to 3=(full).
`-T,--tree`	Outputs the supported Configuration and Status attributes and components with the description and name. Using the -v2 verbosity level also outputs all components and attribute codes. Verbosity levels from 0=(basic / default) to 3=(full).
`-t,--txstats`	Outputs the transmit statistics for the DAG card. Where applicable.
`-u, --ucounters`	Outputs the universal counters for the DAG card. Where applicable.
`-v,--verbose <level>`	Sets the verbosity level, from 0 (basic) to 3 (full).
`-V,--version`	Display the DAG card version information.

Note: For cards with more than 2 ports you can select the required port using: -(portnumber) or --(portletter).

Viewing the DAG card status

Interface Status

When you have configured the card according to your specific requirements you can view the interface statistics to check the status of each of the links using:

```
dagconfig -d0 -s
```

(where "0" is the device number of the DAG card you wish to capture data from)
Example outputs are shown below:

Note: "1" indicates the condition is present on the link "0" indicates the condition is not present on the link.

```
Port   link  fault  tx_fault  rx_fault  lock
   A     1      0         0         0     1
```

Status Conditions

A definition of each of the status conditions is described below:

Condition	Definition
Link	The link is fully validated
Fault	
TXFault	There is a transmission fault
RxFault	There is a receive fault
Lock	The link has signal

Extended statistics

Extended statistics are also available: The following example shows the extended statistics.

```
dagconfig -e
```

For more information see `dagconfig` (page 23).

There are multiple printout levels. The statistics displayed below are printout level 0.

Note: "1" indicates the condition is present on the link "0" indicates the condition is not present on the link.

Ethernet

```
Port   rx_bytes   fcs_errors   short_packet_errors   long_packet_errors   tx_bytes
A      0          0            0                     0                    0
```

A definition of each of the status conditions is described below:

Condition	Definition
Port	Indicates the port to which the status conditions are applicable.
rx_bytes	Number of RX bytes received.
fcs_errors	Number of FCS errors received.
short_packet_error	The number of packets shorter than the Short packet size setting. By default these packets are dropped.
long_packet_errors	The number of packets longer than the Long packet size settings. By default these packets are dropped.
tx_bytes	Number of TX bytes received.

Universal counters

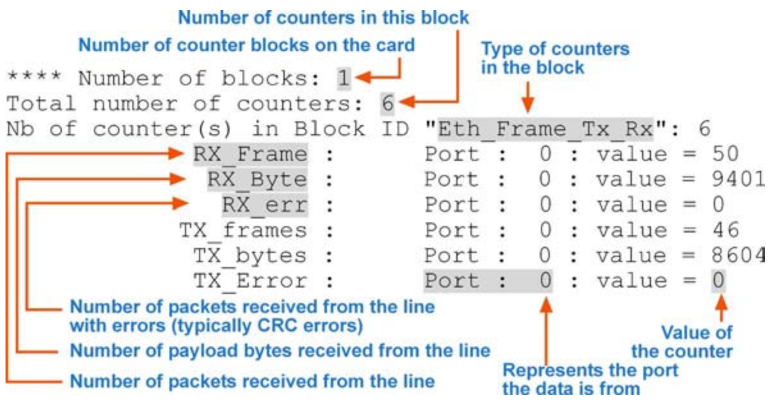
The counters contain details of the number of frames and any errors. The counters are latch and clear so values indicate the amount of data since the last time the counters were read.

```
dagconfig –d0 –u
```

For more information see `dagconfig (page 23)`.

Example outputs are shown below:

Ethernet



```
**** Number of blocks: 1
Total number of counters: 6
Nb of counter(s) in Block ID "Eth_Frame_Tx_Rx": 6
             RX_Frame :      Port :  0 : value = 50
              RX_Byte :      Port :  0 : value = 9401
               RX_err :      Port :  0 : value = 0
            TX_frames :      Port :  0 : value = 46
             TX_bytes :      Port :  0 : value = 8604
             TX_Error :      Port :  0 : value = 0
```

Number of counters in this block
Number of counter blocks on the card
Type of counters in the block
Number of packets received from the line with errors (typically CRC errors)
Number of payload bytes received from the line
Number of packets received from the line
Value of the counter
Represents the port the data is from

Using your DAG card to capture data

Introduction

This chapter describes how to complete the following operations for a DAG card:

- Basic data capture (page 27)
- Viewing captured data (page 30)
- Converting captured data (page 32)
- Using third party applications (page 34)
- Transmitting captured data (page 34)

Basic data capture

`dagsnap` is a software utility used to write to disk the raw data captured from a DAG card.

Data is collected in Extensible Record Format (ERF) which can then be viewed using `dagbits`, or converted to other formats using `dagconvert`.

When capturing high speed data Endace recommends you use `dagsnap`, see Capturing data at high speed (page 29).

For further information on the software utilities see:

- dagsnap (page 28)
- dagbits (page 30)
- dagconvert (page 33)

Starting a capture session

To start the capture session type the following at the prompt:

```
dagsnap -d0 –v -o tracefile
```

(where "0" is the device number of the DAG card you wish to capture data from)

Note: You can use the `-v` option to provide user information during a capture session although you may want to omit it for automated trace runs.

By default, `dagsnap` runs indefinitely. To stop, use *CTRL+C*. You can also configure `dagsnap` to run for a fixed time period then exit.

dagsnap

`dagsnap` is a data capture software utility.

The following is a list of the options available in `dagsnap`.

Option	Description

`-d DEVICE`
`--device DEVICE`
	Use the DAG device referred to by DEVICE. Supported syntax includes 0, dag1, and /dev/dag3 to refer to DAG cards, and 0:2, dag1:0, and /dev/dag2:0 to refer to specific streams on cards.

`-h, -?`
`--help, --usage`
	Display usage (help) information.

`-j`	Maximize disk writing performance by only writing data to disk in chunks. This option may not be available on all operating systems.
`-m NUM`	Write at most NUM megabytes of data per call to the DAG API (default is 4 MiB).

`-o FILE`
`--fname FILE`
	Write the captured packets to FILE, in ERF format (default is standard output).

`-s NUM`
`--runtime NUM`
	Run for NUM seconds, then exit.

`-v` `--verbose`	Increase output verbosity. When `dagsnap` is run with this command three columns of data are reported every second. These columns contain 1. The cumulative total of data written out from the DAG card. 2. The buffer occupancy. Small values indicate no packet loss. 3. The rate at which data is currently being written.
`-V, --version`	Display version information.
`-w,--wait SEC`	Delay(wait) in seconds before capture and after.

Capturing data at high speed

As the 8.2X card captures packets from the network link, it writes a record for each packet into a large buffer in the host computer's main memory.

To avoid packet loss, the user application reading the record, such as `dagsnap`, must be able to read records out of the buffer as fast or faster than they arrive. If not the buffer will eventually fill and packet records will be lost.

If the user process is writing records to hard disk, it may be necessary to use a faster disk or disk array. If records are being processed in real-time, a faster host CPU may be required.

In Linux and Free BSD, when the computer buffer fills, the following message displays on the computer screen:

```
kernel: dagN: pbm safety net reached 0xNNNNNNNN
```

The same message is also printed to log `/var/log/messages` file. In addition, when the computer buffer fills the "Data Capture" LED on the card will flash or flicker, or may go OFF completely.

In Windows no screen message displays to indicate when the buffer is full. Please contact Endace Customer Support at support@endace.com for further information on detecting buffer overflow and packet loss in Windows.

Detecting Packet Losses

Once the buffer fills, any new packets arriving will be discarded by the 8.2X card until some data is read out of the buffer to create free space.

You can detect any such losses by observing the Loss Counter (`lctr` field) of the Extensible Record Format (ERF). See Data Formats (page 49) later in this User Guide for more information on the Endace ERF record format.

Increasing Buffer Size

You can increase the size of the host computer buffer to enable it to cope with bursts of high traffic load on the network link.

For information on increasing the buffer size, see buffer_size (page 17).

Viewing captured data

Data captured in ERF format can be viewed with `dagbits`. For further details on how to use `dagbits`, see dagbits (page 30).

Note: `dagbits` decodes and displays ERF header fields and packet contents are displayed as a Hex dump only. To decode higher level protocols, Endace recommends using a third party application, see Using third party applications (page 34).

Examples

Test live traffic on dag0, stream 0 for 60 seconds running the lctr, flags and fcs tests:

```
dagbits -vvc -d0:0 -s60 lctr flags fcs
```

To read a trace log file using dagbits:

```
dagbits -vvc print -f logname.log | less
```

To check for errors in the trace:

```
dagbits -vvc ltcr flags fcs -f logname.log
```

If `dagbits` reaches the end of the traffic and prints its report then the ERF records were valid.

dagbits

`dagbits` is a software utility used to view and test ERF records. `dagbits` can receive data from either:

- directly from the DAG card (using the `-d` option), or
- a ERF data file created by `dagsnap`.

The following is a list options available in `dagbits`:

Options	Description

`-0`
`-16`
`-32`
	ERF records contain no Link-Layer CRCs.
	ERF records contain 16 bit Link-Layer CRCs (PoS).
	ERF records contain 32 bit Link-Layer CRCs (PoS and Ethernet).

`-a`	Set legacy format to ATM (this is the default).
`-b`	Treat ERF timestamps as big-endian.
`-c`	Print real-time progress reports as `dagbits` captures traffic. This is a useful indicator that a test is running correctly.
`-C NUM`	Sets CRC correction factor for BTX test (0, 16 or 32 bits).

`-d DEVICE`
`--device DEVICE`
	Use the DAG device referred to by DEVICE. Supported syntax includes 0, dag1, and /dev/dag3 to refer to DAG cards, and 0:2, dag1:1, and /dev/dag2:0 to refer to specific streams on cards.

`-D NUM`	Introduces a NUM nanosecond delay between processing each record.
`-e`	Set legacy format to Ethernet (default: ATM).
`-E NUM`	Halt operation after a maximum of NUM errors. This option prevents `dagbits` from creating extremely large output files when being redirected to a file.
`-f FILE`	Read captured data from FILE.

`-h, -?`
`--help, --usage`
	Display usage (help) information.

`-i API`	Use "API" interface for live DAG API capture. Possible options are: 0 DAG 2.4 legacy API interface [dag_offset(3)]. 1 DAG 2.5 API interface [dag_advance_stream(3)]. 2 DAG 2.5 API interface [dag_rx_stream_next_record(3)].
`-I`	Assume the ERF contains color information in the pad and offset bytes (for Ethernet ERFs) or HDLC header bytes (for PoS ERFs) and display this information as a packet classification and destination memory buffer.
`-j NUM`	Set the threshold for the jitter test to NUM microseconds.
`-m NUM`	Print the first NUM errored records only, and then continue to count errors silently for the duration of the session.

`-n NUM`	Expected number of packets to receive. Returns an error if the actual number is different.
`-p`	Set legacy format to PoS (default: ATM).
`-P PARAMS`	DAG 3.5S capture parameters.
`-q`	Quiet. This instructs `dagbits` to suppress summary information when terminating. Error messages are not affected by this option.
`-r NUM`	Set legacy format record lengths to NUM.
`-R NUM`	When used in conjunction with the rlen test, indicates the RLEN of ERF records to match against. NUM.
`-s`	Check for strictly monotonic (increasing) timestamps, rather than monotonic (non-decreasing). Affects the behavior of the mono test. With strict checking it is an error for consecutive timestamps to be equal; they must always increase.
`-S NUM`	Terminate `dagbits` after NUM seconds of capture. This option only makes sense when capturing packets from a DAG card (i.e. when used in conjunction with the -d flag).
`-t NUM`	Terminate `dagbits` if any ERF record type does not match NUM.
`-U NUM`	Process at most NUM records in one pass. This option enables the user to reduce the performance of `dagbits` for various purposes. See also -D.

`-v`
`-vv`
`--verbose`
	Increase verbosity of `dagbits`. This option increase the amount of data displayed when printing an ERF record due to the print test or errors in other tests. -v will print payload contents, -vv will print payload contents and an accompanying ASCII dump of the packet payload.

`-V, --version`	Display version information.
`-w`	Instruct dagbits to treat all warnings as errors.
`-W NUM`	When used in conjunction with the wlen test, the wire length of ERF records must be exactly NUM bytes.
`-z`	Stop when no traffic is received for one second.

`dagbits` takes several options that serve as parameters to particular tests. Available tests include monotonic time-stamp increment and frame checksum (FCS, aka CRC) validation. See the `dagbits` help for further details.

Converting captured data

`dagconvert` is the software utility that converts captured data from ERF format to Pcap (and other formats). Once in non ERF format the data can be read using third party applications (page 34).

`dagconvert` can also be used to capture data directly into pcap format.

Examples

To read from DAG card 0 and save to a file in ERF format:

```
dagconvert -d0 -o outfile.erf
```

To read from DAG card 0 and save to a file in pcap format:

```
dagconvert -d0 -T dag:pcap -o outfile.pcap
```

To convert a file from ERF format to pcap format:

```
dagconvert -T erf:pcap -i infile.erf -o outfile.pcap
```

To convert a file from pcap format to ERF format, ensuring the ERF records are 64-bit aligned (and therefore suitable for transmission using dagflood):

```
dagconvert -T pcap:erf -A 8 -i infile.pcap -o outfile64.erf
```

To capture from DAG Card 0 using a BPF filter:

```
dagconvert -d0 -o outfile.erf -b "host 192.168.0.1 and tcp port 80"
```

To capture from DAG card 0 using ERF filtering:

```
dagconvert -d0 -o outfile.erf -f "rx,a"
```

To capture from DAG card 0 to a series of files of size 128 MB:

```
dagconvert -d0 -o outfile.erf -r 128m
```

The first file created is labeled `outfile0000.erf`, once the file size reaches 128MB, a second file is created. The second is labeled `outfile0001.erf` etc.

Dagconvert

`dagconvert` is a software utility for converting data to various file formats. Supported formats are:

File format	Description
dag	Read ERF records directly from DAG card (input only).
erf	ERF (Extensible Record Format) file (input and output).
atm	Legacy ATM files (input only).
eth	Legacy Ethernet files (input only).
pos	Legacy PoS files (input only).
null	Produces no input or output.
pcap	pcap(3) format file (input or output).
prt	ASCII text packet dump (output only).

Data can be input from a file or captured from a DAG card. `dagconvert` can be used for converting data captured from a DAG card to pcap format. This allows the trace file to be used with tools that support the pcap file format. Also the reverse is possible, where data can be converted to ERF format for use in other dag utilities. The following is a list of options available in `dagconvert`.

Options	Description
`-A NUM`	Set the record alignment of the ERF to NUM bytes (ERF only).
`-b EXPRESSION`	Specify a tcpdump(1) style BPF expression to be applied to the packets.
`-c 0|16|32`	Specify the size (in bits) of the frame checksum (FCS) (pcap(3) only).

`-d DEVICE`
`--device DEVICE`
	Use the DAG device referred to by DEVICE. Supported syntax includes 0, dag1, and /dev/dag3 to refer to DAG cards, and 0:2, dag1:1, and /dev/dag2:0 to refer to specific streams on cards.

`-f FILTERS`	A comma-delimited list of filters to be applied to the data. Supported filters are: • `rx` Filter out rx errors (link layer). • `ds` Filter out ds errors (framing). • `trunc` Filter out truncated packets. • `a,b,c,d` Filter on indicated port/interface(s).
`-F`	Select fixed length output (ERF only).
`-G NUM`	Set the GMT offset to NUM seconds (pcap(3) only).
`-h, -?, --help, --usage`	Display usage (help) information.
`-i FILES`	Name(s) of the input file(s). If more than one filename is given, the ERF records from the files will be merged in timestamp order to the output.
`-o FILE`	Name of the output file.
`-r NUM`	Rotate the output file after NUM bytes. Add `k` (kilobytes), `m` (megabytes), `g` (gigabytes) and `t` (terabytes) suffixes.
`-s NUM`	Set the snap length to NUM bytes.
`-t NUM`	Capture from the DAG card for NUM seconds.
`-T atm|dag|erf|eth |pcap|pos : erf|pcap|prt`	Input and output types. See the DESCRIPTION section above for more information about the input and output types.
`-v, --verbose`	Increase output verbosity.
`-V, --version`	Select variable length output (ERF only). Display version information.
`-y <DLT>`	This sets the pcap data link type to be used for BPF filtering (-b) and for pcap output. Previously only one DLT was mapped to each ERF type. Supported DLT types (case insensitive): EN10MB: Ethernet; DOCSIS : Ethernet CHDLC : HDLC; PPP_SERIAL : HDLC MTP2 : HDLC; ATM_RFC1483 : ATM, AAL5 SUNATM : ATM, AAL5

Note: Not all options are applicable to all DAG cards.

Using third party applications

Once the captured data is in Pcap format you can use third party applications to examine and process the data. The third party applications include:

- Wireshark /Tshark (formerly Ethereal /Tethereal)
- TCPDump
- Libpcap
- SNORT
- Winpcap, etc.

Note: Wireshark can also read ERF formatted data.

Configuring DSM

If you wish to perform packet filtering on the data you are capturing you can do so using the Data Stream Manager (DSM) function described in Extended Functions (page 5) earlier in this user guide.

By default the DSM Lookup Table, is empty and does not contain any entries. This means it will have no effect on packets as they pass through the DSM Coloriser and Drop (CAD) block.

To configure DSM filtering and populate the DSM Lookup Table you must pass an XML file containing filtering instructions to the DSM filter table using:

```
dsm_loader -d0 -f <filter_filename>
```

If you have configured DSM filtering but want to revert to normal capture mode, you will need to bypass DSM filtering to the original null configuration using:

```
dsm_loader -d0 -b
```

For detailed information on configuring the DSM including the format of the XML file, please refer to *EDM04-07 dsm-loader User Guide* and *EDM04-10 Data Stream Management API Programming Guide*. These documents are available from the support section of the Endace website at www.endace.com.

Transmitting captured data

Configuration

The 8.2X card is able to transmit as well as receive packets and can capture received traffic **while** transmitting. This allows you to use capture tools such as `dagsnap`, `dagconvert`, and `dagbits` while `dagflood` is sending packets.

To configure the 8.2X card for transmission, you must allocate some memory to a transmit stream. By default, 16 MB of memory is allocated to the tx stream and the remainder is allocated to the rx stream. For information on setting the Memory allocation see mem (page 19).

Note: You can not change the stream memory allocations while packet capture or transmission is in progress.

Explicit Packet Transmission

The operating system does not recognize the 8.2X card as a network interface and will not respond to ARP, ping, or router discovery protocols.

The 8.2X card will only transmit packets that are explicitly provided by the user. This allows you to use the 8.2X card as a simple traffic load generator.

You can also use it to retransmit previously recorded packet traces. The packet trace is transmitted as fast as possible. The packet timing of the original trace file is not reproduced.

Trace Files

You can use `dagflood` to transmit ERF format trace files, providing the files contain **only** ERF records of the type matching the current link configuration.

When you use DAG cards with multiple ports, ensure all ports referred to by the Trace file are active. This ensures the `dagflood` traffic is not blocked when trying to delivering data to an inactive port. Check the interface status output for the DAG card and ensure the link status for all required destination ports is active. See Viewing the DAG card statistics. (page 24)

For further information on using `dagflood` please refer to the *EDM04-03 dagflood User Manual* available from Endace Customer Support at https://www.endace.com/support.

In addition the length of the ERF records to be transmitted must be a multiple of 64-bits. You can configure this when capturing packets for later transmission by setting 64-bit alignment using the `dagconfig align64` command.

If packets have been captured without using the `align64` option you can convert the trace files so that they can be transmitted by using dagconvert (page 33) as shown below:

```
dagconvert -v -i tracefile.erf -o tracefile.erf -A8
```

Alternatively if you are unsure if a trace file is 64-bit aligned you can test the file using dagbits (page 30) as shown below:

```
dagbits -v align64 -f tracefile.erf
```

If you do not have any ERF trace files available, you can use `daggen` to generate trace files containing simple traffic patterns. This allows the 8.2X card to be used as a test traffic generator.

For further information on using `daggen` please refer to the *EDM04-06 Daggen User Guide* available from Endace Customer Support at https://www.endace.com/support.

TR TERF

Selectable CRC Length

TR-TERF can be optionally configured to strip the CRC of an incoming ERF stream, to allow the retransmitted stream to calculate and add a CRC. There are three basic situations where this option may be useful.

Situation 1

The ERF stream was generated without a CRC.

In this case, you should configure TR-TERF to **not** strip the 32-bit CRC and configure the MAC to add a hardware-calculated CRC. This has the effect of reducing the burden of calculating a CRC in the software.

Example

```
    dagconfig noterf_strip tx_crc
```

Situation 2

The ERF stream was generated with a CRC but the CRC must be correct on the retransmitted packet.

In this case you should configure TR-TERF to strip the 32-bit CRC and configure the MAC to add a hardware calculated CRC as in Configuration 1 above. This means even if the CRC is incorrect in the received ERF stream, a correct CRC will be generated when the packet is retransmitted.

Example

```
dagconfig terf_strip32 tx_crc
```

Situation 3

The ERF stream was generated with a CRC, and the CRC must be retransmitted exactly even if it is incorrect.

In this case you should configure TR-TERF **not** to strip the CRC, and configure the MAC **not** to add a hardware-calculated CRC. This allows received packets with bad CRCs to be retransmitted which may be a useful tool for testing or diagnostic purposes.

Example

```
dagconfig noterf_strip notx_crc
```

Retransmitting Errored Packets

In some circumstances it may not be desirable to retransmit packets which have been incorrectly received for any reason. To allow for this, TR-TERF can be optionally configured not to retransmit any packets marked with the `rxerror` (receive error) flag.

Usage Notes

The following points should be noted when using TR-TERF:

- When using `dagflood` to transmit an ERF stream to the card you should set the "`-l`" flag (maximum data burst length) to a value greater than the default of 1MB. During testing Endace found a value of 16MB (16777216) to be effective. This reduces the possibility of a buffer under-run occurring if insufficient data is committed in a burst and the `dagflood` process is not scheduled by the OS to run in a timely manner.
- For best accuracy when testing, you should ensure both the sending and receiving cards are synchronized to the same time source.

TR TERF Known Issues

TR-TERF is less accurate when using multiple interfaces. If a large packet is transmitted to the card on a particular interface, the transmit buffers for the other interfaces may experience a buffer under-run. This is because data is only transferred to one interface at anyone time, record by record as the input ERF stream is interleaved between interfaces.

This may be especially noticeable where very small packets are transmitted on one interface and very large packets transmitted on another interface.

Synchronizing Clock Time

Overview

DAG cards have sophisticated time synchronization capabilities. This allows for high quality timestamps and optional synchronization to an external time standard.

The core of the DAG synchronization capability is known as the DAG Universal Clock Kit (DUCK).

A clock in each DAG card runs independently from the computer clock. The DAG card's clock is initialized using the computer clock, and then free-runs using a crystal oscillator.

Each DAG card's clock can vary relative to a computer clock, or other DAG cards.

DUCK Configuration

The DUCK (DAG Universal Clock Kit) is designed to reduce time variance between sets of DAG cards or between DAG cards and coordinated universal time [UTC].

You can obtain an accurate time reference by connecting an external clock to the DAG card using the time synchronization connector. Alternatively you can use the host computer's clock in software as a reference source without any additional hardware.

Each DAG card can also output a clock signal for use by other DAG cards.

Common Synchronization

The DAG card time synchronization connector supports a Pulse-Per-Second (PPS) input signal, using RS-422 signaling levels.

Common synchronization sources include GPS or CDMA (cellular telephone) time receivers.

Endace also provides the TDS 2 Time Distribution Server modules and TDS 6 expansion units that enable you to connect multiple DAG cards to a single GPS or CDMA unit.

For more information, please refer to the Endace website at https://www.endace.com/support, or the *EDM05-01 Time Distribution Server User Guide*.

Network Time Protocol

NTP (Network Time Protocol) can be used to synchronize a computer clock to a network based reference. When the NTP daemon starts, it exchanges packets with network time servers to establish the correct time. If the computer clock is significantly different, the NTP can adjust the computer clock in a single large 'step'. Over time, NTP adjusts the rate of computer clock to minimize the offset from its reference. It can take several days for NTP to fully synchronize the computer clock.

The DAG card clock is initialized from the computer's clock rather than from the NTP. Using NTP to synchronize the computer's clock ensures the DAG card clock remains accurate.

DAG cards can also be synchronized to external references such as GPS or to the computer clock directly. In both cases the computer clock time is loaded onto the DAG clock when the DAG card is started (`dagload, dagreset, dagrom -p`).

When clock synchronization is enabled, the DAG card time is compared to the computer time once per second, regardless of the synchronization source. If the times differ by more than 1 second, the DAG card clock is reloaded from the computer clock and synchronization is restarted. For this reason, the computer clock should be maintained with better than 1 second accuracy.

If the DAG card clock is synchronized to the computer clock, then small 'step' adjustments of the computer clock by the NTP daemon can cause the DAG driver to emit warning messages to the console and system log files if the adjustment exceeds the warning threshold. These messages are intended to allow the user to monitor the quality of the clock synchronization over time.

The best synchronization is achieved when the DAG card is synchronized to an external GPS reference clock, and the computer clock is synchronized to a local NTP server.

Timestamps

ERF files contain a hardware generated timestamp of each packet's arrival.

The format of this timestamp is a single little-endian 64-bit fixed point number, representing the number of seconds since midnight on the 1st January 1970.

The high 32-bits contain the integer number of seconds, while the lower 32-bits contain the binary fraction of the second. This allows an ultimate resolution of 2^{-32} seconds, or approximately 233 picoseconds.

Different DAG cards have different actual resolutions. This is accommodated by the lower most bits that are not active being set to zero. In this way the interpretation of the timestamp does not need to change when higher resolution clock hardware is available. The 8.2X implements the 27 most significant bits which provides a time resolution of 7.5 nanoseconds.

The ERF timestamp allows you to find the difference between two timestamps using a single 64-bit subtraction. You do not need to check for overflows between the two halves of the structure as you would need to do when comparing Unix time structures.

Example

Below is example code showing how a 64-bit ERF timestamp (erfts) can be converted into a `struct timeval` representation (tv):

```
unsigned long long lts;
struct timeval tv;


  lts = erfts;
  tv.tv_sec = lts >> 32;
  lts = ((lts & 0xffffffffULL) * 1000 * 1000);
  lts += (lts & 0x80000000ULL) << 1;      /* rounding */
  tv.tv_usec = lts >> 32;
  if(tv.tv_usec >= 1000000) {
    tv.tv_usec -= 1000000;
    tv.tv_sec += 1;
      }
```

Dagclock

The DUCK is very flexible and can be used with or without an external time reference. It can accept synchronization from one of several input sources and also be made to drive its synchronization output from one of several sources.

Synchronization settings are controlled by the `dagclock` utility.

Note: You should only run `dagclock` after you have loaded the appropriate FPGA images. If at any stage you reload the FPGA images you must rerun `dagclock` to restore the configuration.

Note: when you run `dagconfig -d0 default` the `dagclock` inputs and outputs are also reset to defaults.

A description of each argument is shown below:

Option	Description

`-d DEVICE`
`--device DEVICE`
	Use the DAG device referred to by DEVICE. Supported syntax includes 0, dag1, and /dev/dag3 to refer to DAG cards.

`-h, -?`
`--help, --usage`
	Display the information on this page

`-k` `--sync`	Wait for DUCK synchronization before exiting
`-K NUM`	Set the synchronization timeout in seconds (default is 60 seconds)
`-l NUM`	Set the Health threshold in nanoseconds. (default is 596ns)
`-v`	Increase output verbosity
`-V`	Display version information
`-x` `--clearstats`	Clear clock statistics

Command	Description
`default`	Set the `dagclock` input and output to RS422 in and none out.
`none`	Clears the input and output settings.
`rs422in`	Sets the `dagclock` input to RS422.
`hostin`	Sets the `dagclock` input to Host (unused)
`overin`	Sets the `dagclock` input to Internal input
`auxin`	Sets the `dagclock` input to Auxiliary input (unused)
`rs422out`	Sets the `dagclock` output to repeat the RS422 input signal
`loop`	Output the selected input
`hostout`	Sets the `dagclock` output to host (unused)
`overout`	Internal output (master card)
`set`	Sets the DAG card's clock to computer clock time and clears clock statistics. The DAG card takes approximately 20 to 30 seconds to re-synchronize.
`reset`	Full clock reset. Load time from computer, set RS422in, none out. Clears clock statistics. The DAG card takes approximately 20 to 30 seconds to re-synchronize.

Note: By default, all DAG cards listen for synchronization signals on their RS-422 port, and do not output any signal to that port.

Dagclock Statistics reset

Statistics are reset to zero when the following occur:

- Loading a DAG driver
- Loading firmware
- `dagclock` with a `-x` option
- `dagclock` with a `set` or `reset` command.

Example

To view the default `dagclock` configuration:

```
dagclock -d0
```

The following is the output from DAG card that has its clock reference connected. The clock statistics have been reset since the card was last synchronized. **Note:** Values will differ for each DAG card type.

```
muxin   rs422
muxout  none
status  Synchronised Threshold 596ns Failures 0 Resyncs 0
error   Freq -30ppb Phase -60ns Worst Freq 75ppb Worst Phase 104ns
crystal Actual 100000028Hz Synthesized 67108864Hz
input   Total 3765 Bad 0 Singles Missed 5 Longest Sequence Missed 1
start   Thu Apr 28 13:32:45 2007
host    Thu Apr 28 14:35:35 2007
dag     Thu Apr 28 14:35:35 2007
```

Note: For a description of the `dagclock` output see Dagclock output explained (page 42).

Dagclock output explained

Muxin

Lists the `dagclock` time input source for this DAG card. The options are `RS422in, Hostin, Overin or Auxin.`

Example
```
   muxin   rs422
```

Muxout

Lists the `dagclock` time output source for this DAG card. The options are `RS422out, Hostout, Overout or Loop.`

Example
```
   muxout  none
```

Status

This line reports on the status of the DAG card.

Output	Description
Synchronised / Not synchronised	This indicates whether this DAG card is synchronized to the time source listed (`Muxin`) . The DAG card becomes **Not Synchronized** when the absolute *Phase error (page 42)* is above the *Threshold* value for 10 consecutive seconds.
Threshold	This is the value above which the DAG card port is considered **Not Synchronized**. The *Threshold* value changes depending on the type of input time synchronization. The defaults are: • 596 for RS422 synchronization • 12000 for host synchronization (Unix) • 50000 for host synchronization (Windows) This value can be adjusted using the `dagclock -l` option.
Failures	This is a count of the number of times the DAG card has become **Not Synchronized.**
Resyncs	This is a count of the number of times the DAG card Phase error has exceeded 1 second. See Error (Dagclock) (page 42). If the DAG card is **Not Synchronized** for more than 10 seconds the DAG card automatically runs the following command to update the time on the DAG card: `dagclock -d0 set` Where "0" is the device number.

Example
```
   status  Synchronised Threshold 596ns Failures 0 Resyncs 0
```

Error

This line reports on the synthesized frequency of the DAG card.

Output	Description
Freq	An estimate of the synthesized frequency error over the last second in parts per billion.
Phase	The difference between the DAG card's clock and the reference clock at the last time pulse.
Worst Freq	Highest absolute value of the *Frequency error* since statistic collection began. Reset to zero when statistics are reset, see Dagclock Statistics reset (page 41).
Worst Phase	Highest absolute value of the *Phase error* since statistic collection began. Reset to zero when statistics are reset, see Dagclock Statistics reset. (page 41)

Example
```
   error   Freq -30ppb Phase -60ns Worst Freq 75ppb Worst Phase 104ns
```

Crystal

This line reports on the DAG card crystal oscillator.

Output	Description
Actual	The DAG card's crystal frequency calculated based on the reference clock.
Synthesized	The target time stamping frequency. Different for each DAG card type.

Example

```
crystal Actual 100000028Hz Synthesized 67108864Hz
```

Input

This line reports on the time pulses received by the DAG card.

Output	Description
Total	The total number of time pulses received. Reset to zero when statistics are reset, see Dagclock Statistics reset (page 41).
Bad	The number of time pulses that were rejected (considered Bad) by the DAG card. Reset to zero when statistics are reset, see Dagclock Statistics reset (page 41). Time pulses are considered *Bad* if they were not received 1 second (approximately) after the last time pulse. These may be caused by noise.
Singles missed	The number of times a single time pulse failed to be received by the DAG card (i.e. a two second gap). Reset to zero when statistics are reset, see Dagclock Statistics reset (page 41).
Longest Sequence Missed	This displays the longest time gap (in seconds) between a pair of time pulses. Reset to zero when statistics are reset, see Dagclock Statistics reset (page 41).

Example

```
input   Total 3765 Bad 0 Singles Missed 5 Longest Sequence Missed 1
```

Start / Host / DAG

Output	Description
Start	This is the time statistics collection started. See Dagclock Statistics reset. (page 41)
Host	Current Host (computer) time.
DAG	The DAG card time at the last time pulse. If the DAG card has never been synchronized, the following displays: `No active input - free running.`

Example

```
start   Thu Apr 28 13:32:45 2007
host    Thu Apr 28 14:35:35 2007
dag     Thu Apr 28 14:35:35 2007
```

Card with Reference

Overview

To obtain the best timestamp accuracy you should connect the DAG card to an external clock reference, such as a GPS or CDMA time receiver.

To use an external clock reference source, the host computer's clock must be accurate to UTC to within one second. This is used to initialize the DUCK.

When the external time reference source is connected to the DAG card time synchronization connector, the DAG card automatically synchronizes to a valid signal.

Pulse Signal from External Source

The DAG time synchronization connector supports an RS-422 (PPS) signal from an external source. This is derived directly from an external reference source or distributed through the Endace TDS 2 (Time Distribution Server) module which allows two DAG cards to use a single receiver. It is also possible for more than two DAG cards to use a single receiver by "daisy-chaining" TDS-6 expansion modules to the TDS-2 module. Each TDS-6, module provides outputs for an additional 6 DAG cards.

Synchronize to an external source as follows:

```
dagclock -d0
```

Output:

```
muxin   rs422
muxout  none
status  Synchronised Threshold 596ns Failures 0 Resyncs 0
error   Freq 30ppb Phase -15ns Worst Freq 238ppb Worst Phase 326ns
crystal Actual 100000023Hz Synthesized 67108864Hz
input   Total 225 Bad 0 Singles Missed 1 Longest Sequence Missed 1
start   Thu Apr 28 14:55:20 2007
host    Thu Apr 28 14:59:06 2007
dag     Thu Apr 28 14:59:06 2007
```

Connecting the Time Distribution Server

You can connect the TDS 2 module to the DAG card using DUCK crossover cable (page 48) (**Note:** A 4-pin to RJ45 adapter may be required). The TDS may be located up to 600m (2000ft) from the DAG card depending upon the quality of the cable used, possible interference sources and other environmental factors. Please refer to the *EDM05-01 Time Distribution Server User Guide* for more in formation.

Caution:

Never connect a DAG card and/or the TDS 2 module to active Ethernet equipment or telephone equipment.

Testing the Signal

For Linux and FreeBSD, when a synchronization source is connected the driver outputs messages to the console log file `/var/log/messages`.

To test the signal is being received correctly and has the correct polarity use the `dagpps` tool as follows:

```
dagpps -d0
```

`dagpps` measures the input state many times over several seconds, displaying the polarity and length of input pulse.

Single Card No Reference

When a single DAG card is used with no external reference, the DAG card can be synchronized to the host computer clock. Most computer clocks are not very accurate by themselves, but the DUCK drifts smoothly at the same rate as the computer clock.

The synchronization achieved with this method is not as accurate as using an external reference source such as GPS.

The DUCK clock is synchronized to a computer clock by setting input synchronization selector to overflow as follows:

```
dagclock –d0 none overin
```

Output

```
muxin   overin
muxout  none
status  Synchronised Threshold 11921ns Failures 0 Resyncs 0
error   Freq 1836ppb Phase 605ns Worst Freq 147ppb Worst Phase 324ns
crystal Actual 49999347Hz Synthesized 16777216Hz
input   Total 87039 Bad 0 Singles Missed 0 Longest Sequence Missed 0
start   Wed Apr 27 14:27:41 2007
host    Thu Apr 28 14:38:20 2007
dag     Thu Apr 28 14:38:20 2007
```

Two Cards No Reference

Overview

If you are using two DAG cards in a single host computer with no reference clock, you must synchronize the DAG cards using the same method if you wish to compare the timestamps between the two DAG cards. You may wish to do this for example if the two DAG cards monitor different directions of a single full-duplex link. You can synchronize the DAG cards in two ways:

- One DAG card can be a clock master for the second. This is useful if you want both DAG cards to be accurately synchronized with each other, but not so for absolute time of packet time-stamps, or
- One DAG card can synchronize to the host and also act as a master for the second DAG card.

Synchronizing with Each Other

Although the master DAG card's clock drifts against UTC, the DAG cards will be locked together. This is achieved by connecting the time synchronization connectors of both DAG cards using a DUCK crossover cable (page 48) (**Note:** A 4-pin to RJ45 Adapter may be required).

Configure one of the DAG cards as the master so that the other defaults to being a slave as follows:

```
dagclock –d0 none overout
```

Output:

```
muxin   none
muxout  over
status  Not Synchronised Threshold 596ns Failures 0 Resyncs 0
error   Freq 0ppb Phase 0ns Worst Freq 213ppb Worst Phase 251ns
crystal Actual 100000000Hz Synthesized 67108864Hz
input   Total 0 Bad 0 Singles Missed 0 Longest Sequence Missed 0
start   Thu Apr 28 14:48:34 2007
host    Thu Apr 28 14:48:34 2007
dag     No active input - Free running
```

Note: The slave DAG card configuration is not shown as the default configuration will work.

Synchronizing with Host

To prevent the DAG card clock time-stamps drifting against UTC, the master DAG card can be synchronized to the host computer's clock which in turn utilizes NTP. This provides a master signal to the slave DAG card.

Configure one DAG card to synchronize to the computer clock and output a RS-422 synchronization signal to the second DAG card as follows:

```
dagclock –d0 none overin overout
```

Output:

```
muxin   over
muxout  over
status  Synchronised Threshold 11921ns Failures 0 Resyncs 0
error   Freq -691ppb Phase -394ns Worst Freq 147ppb Worst Phase 424ns
crystal Actual 49999354Hz Synthesized 16777216Hz
input   Total 87464 Bad 0 Singles Missed 0 Longest Sequence Missed 0
start   Wed Apr 27 14:27:41 2007
host    Thu Apr 28 14:59:14 2007
dag     Thu Apr 28 14:59:14 2007
```

Note: The slave DAG card configuration is not shown, the default configuration is sufficient.

Connector Pin-outs

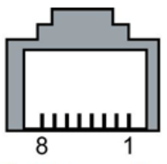
Overview

DAG cards have an 8-pin RJ45 connector for time synchronization. The connector has two bi-directional RS422 differential circuits, A and B. The Pulse Per Second (PPS) signal is carried on circuit A, and the SERIAL packet is connected to the B circuit.

Pin Assignments

The 8-pin RJ45 connector pin assignments and plugs and sockets are shown below:

1.	Out PPS+
2.	Out PPS-
3.	In PPS+
4.	In SERIAL+
5.	In SERIAL-
6.	In PPS-
7.	Out SERIAL+
8.	Out SERIAL-



RJ45 socket

Normally, you connect the GPS input to the PPS (A) channel input (pins 3 and 6).

The DAG card can also output a synchronization pulse for use when synchronizing two DAG cards (i.e. without a GPS input). The synchronization pulse is output on the Out PPS channel (pins 1 and 2).

To connect two DAG cards, use a DUCK crossover cable (page 48) to connect the two time synchronization sockets.

DUCK Crossover cable

To synchronize two DAG cards together use a cable with RJ45 plugs and the following wiring.

TX PPS+	**1**	**3**	RX PPS+
TX PPS-	**2**	**6**	RX PPS-
RX PPS+	**3**	**1**	TX PPS+
RX SERIAL+	**4**	**7**	TX SERIAL+
RX SERIAL-	**5**	**8**	TX SERIAL-
RX PPS-	**6**	**2**	TX PPS-
TX SERIAL+	**7**	**4**	RX SERIAL+
TX SERIAL-	**8**	**5**	RX SERIAL-

Note: This wiring is the same as an Ethernet crossover cable (Gigabit crossover, All four pairs crossed).

Data Formats

Overview

The DAG Card produces trace files in its own native format called ERF (Extensible Record Format). The ERF type depends upon the type of connection you are using to capture data.

The 8.2X supports the following ERF Types:

ERF Type	Description
2	TYPE_ETH Ethernet Variable Length Record
16	TYPE_DSM_COLOR_ETH Ethernet Variable Length Record

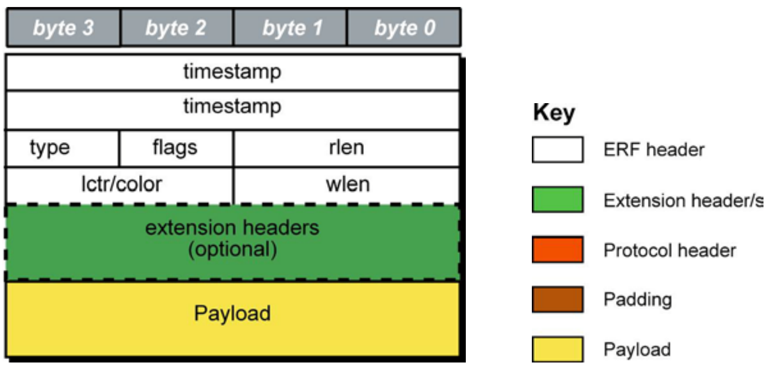
The ERF file contains a series of ERF records with each record describing one packet. ERF files consists only of ERF records, there is no file header or trailer. This allows for simple concatenation and splitting of files to be performed on ERF record boundaries.

For information on other ERF types, please refer to *EDM11-01 ERF types*.

Generic ERF Header

All ERF records share some common fields. Timestamps are in little-endian (Pentium® native) byte order. All other fields are in big-endian (network) byte order. All payload data is captured as a byte stream in network order, no byte or re-ordering is applied.

The generic ERF header is shown below:



The fields are described below:

timestamp			The time of arrival of the cell, an ERF 64-bit timestamp.
type	Bit 7		Extension header present.
	Bit 6:0		Extension header type. See table below:
flags			This byte is divided into several fields as follows:
		Bits	**Description**
		1-0:	Binary enumeration of capture interface: 11 Interface 3 or D 10 Interface 2 or C 01 Interface 1 or B 00 Interface 0 or A Cards with more than four interfaces typically use Multichannel ERF types (type 5 to 9, 12 and 17) which provide a separate larger interface field.
		2:	Varying length record. When set, packets shorter than the snap length are not padded and rlen resembles wlen. When clear, longer packets are snapped off at snap length and shorter packets are padded up to the snap length. rlen resembles snap length. Setting novarlen and slen greater than 256 bytes is wasteful of bandwidth
		3:	Truncated record - insufficient buffer space. • `wlen` is still correct for the packet on the wire. • `rlen` is still correct for the resulting record. But, rlen is shorter than expected from snap length or wlen values. **Note:** truncation is depreciated and this bit is unlikely to be set in an ERF record.
		4:	RX error. An error in the received data. Present on the wire
		5:	DS error. An internal error generated inside the card annotator. Not present on the wire.
		6:	Reserved
		7:	Reserved
rlen			Record length in bytes. Total length of the record transferred over the PCIe bus to storage. The timestamp of the next ERF record starts exactly rlen bytes after the start of the timestamp of the current ERF record.
lctr			Depending upon the ERF type this 16 bit field is either a loss counter of color field. The loss counter records the number of packets lost between the DAG card and the stream buffer due to overloading on the PCIe bus. The loss is recorded between the current record and the previous record captured on the same stream/interface. The color field is explained under the appropriate type details.

wlen		Wire length. Packet length "on the wire" including some protocol overhead. The exact interpretation of this quantity depends on physical medium. This may contain padding.
extension headers		Extension headers in an ERF record allow extra data relating to each packet to be transported to the host. Extension header/s are present if bit 7 of the type field is '1'. If bit 7 is '0', no extension headers are present (ensures backwards compatibility). **Note:** There can be more than one Extension header attached to a ERF record.
Payload		Payload is the actual data in the record. It can be calculated by either : • Payload = rlen - ERF header - Extension headers (optional) - Protocol header - Padding

Extension header types

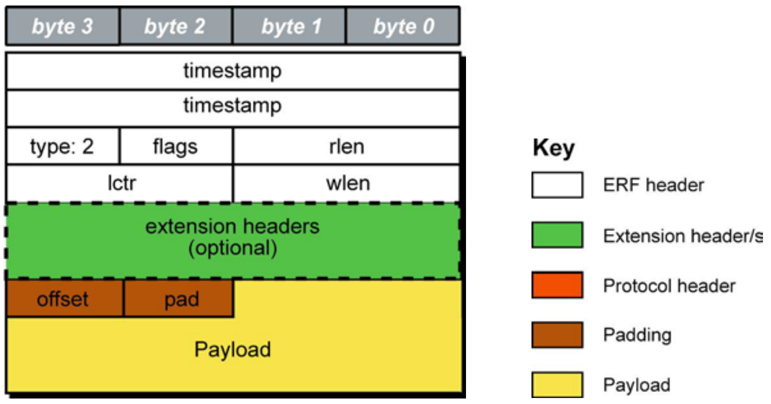
Number	Type	Description
0:	TYPE_LEGACY	Old style record
1:	TYPE_HDLC_POS	Packet over SONET / SDH frames, using either PPP or CISCO HDLC framing.
2:	TYPE_ETH	Ethernet
3:	TYPE_ATM	ATM cell
4:	TYPE_AAL5	reassembled AAL5 frame
5:	TYPE_MC_HDLC	Multi-channel HDLC frame
6:	TYPE_MC_RAW	Multi-channel Raw time slot link data
7:	TYPE_MC_ATM	Multi-channel ATM Cell
8:	TYPE_MC_RAW_ CHANNEL	Multi-channel Raw link data
9:	TYPE_MC_AAL5	Multi-channel AAL5 frame
10:	TYPE_COLOR_HDLC_ POS	HDLC format like TYPE_HDLC_POS, but with the LCNTR field reassigned as COLOR
11:	TYPE_COLOR_ETH	Ethernet format like TYPE_ETH, but with the LCNTR field reassigned as COLOR
12:	TYPE_MC_AAL2	Multi-channel AAL2 frame
13:	TYPE_IP_COUNTER	IP Counter ERF Record
14:	TYPE_TCP_FLOW_ COUNTER	TCP Flow Counter ERF Record
15:	TYPE_DSM_COLOR_ HDLC_POS	HDLC format like TYPE_HDLC_POS, but with the LCNTR field reassigned as DSM COLOR
16:	TYPE_DSM_COLOR_ ETH	Ethernet format like TYPE_ETH, but with the LCNTR field reassigned as DSM COLOR
17:	TYPE_COLOR_MC_ HDLC_POS	Multi-channel HDLC like TYPE_MC_HDLC, but with the LCNTR field reassigned as COLOUR
18:	TYPE_AAL2	Reassembled AAL2 Frame Record
19:	TYPE_COLOR_HASH_ POS	Colored PoS HDLC record with Hash load balancing
20:	TYPE_COLOR_HASH_ ETH	Colored Ethernet variable length record with Hash load balancing
21:	TYPE_INFINIBAND	Infiniband Variable Length Record
22:	TYPE_IPV4	IPV4 Variable Length Record
23:	TYPE_IPV6	IPV6 Variable Length Record
24	TYPE_RAW_LINK	Raw link data, typically SONET or SDH Frame
32-47:	-	Reserved for CoProcess Development Kit (CDK) Users and Internal use
48:	TYPE_PAD	Pad Record type

Note: The Ethernet frame begins immediately after the pad byte so that the layer 3 (IP) header is 32-bit aligned.

ERF 2. TYPE_ETH

Type	Bit 7	1 = Extension header present. See Extension Headers (page 54).
	Bits 6:0	Type 2
Short description	TYPE_ETH	
Long description	Type 2 Ethernet Record	
Use	This record format is for Ethernet [802.3] data links.	

The *TYPE_ETH* record is shown below:



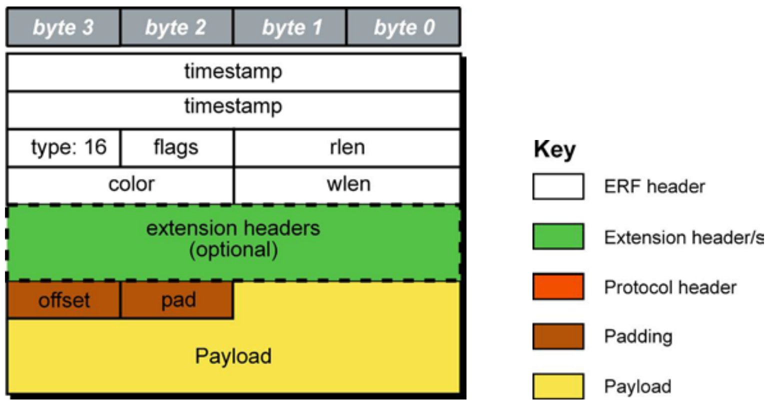
The following is a description of the *TYPE_ETH* record format:

Field	Description
Offset (1 byte)	Number of bytes **not** captured from start of frame. Typically used to skip link layer headers when not required in order to save bandwidth and space. **Note: This field is currently not implemented, contents should be disregarded.**
Pad (1 byte)	The Ethernet frame begins immediately after the pad byte so that the layer 3 [IP] header is 32-bit aligned. This is typically used to skip link layer headers when they are not required in order to save bandwidth and space.
Payload (bytes of record)	Payload = rlen - ERF header (16 bytes) - Extension headers (optional) - Padding (2 bytes)

ERF 16. TYPE_DSM_COLOR_ETH

Type	Bit 7	1 = Extension header present. See Extension Headers (page 54).
	Bits 6:0	Type 16
Short description	TYPE_DSM_COLOR_ETH	
Long description	Type 16 DSM Color Ethernet Record	
Use	This record format is for Ethernet [802.3] data links, incorporating filter results. The record format is the same type as the Type 2 TYPE_ETH (page 52) record, with the exception that the *lctr* field reassigned as *DSM* type *color.*	

The *TYPE_DSM_COLOR_ETH* record is shown below:



The following is a description of the *TYPE_DSM_COLOR_ETH* record format:

Field	Description
Color (2 bytes)	The color field is a hardware generated tag indicating the result of a filtering or classification operation. This field is divided into the following:
Offset (1 byte)	Number of bytes **not** captured from the start of the frame. This is typically used to skip link layer headers when they are not required in order to save bandwidth and space. **Note: This field is currently not implemented; contents should be disregarded.**
Pad (1 byte)	The Ethernet frame begins immediately after the pad byte so that the layer 3 [IP] header is 32-bit aligned. This is typically used to skip link layer headers when they are not required in order to save bandwidth and space.
Payload (bytes of record)	Payload = rlen - ERF header (16 bytes) - Extension headers (optional) - Padding (2 bytes)

Color field bits:

Bit	Description
0-5	Receive stream number (0-63)
6-13	Filter match bits (bit6 = filter0, bit7 = filter1 and so on).
14	hlb0 (CRC calculation) output bit.
15	hlb1 (parity calculation) output bit.

Note: Because both the Type-2 and Type-16 Ethernet record header occupies 18 bytes instead of the standard 16 bytes, any payload captured will always be 2 bytes less than the `slen` value i.e. a `slen` of 48 which will produce a 64 byte record made up of 18 bytes of header and 46 bytes of payload.
Because the `slen` value must always be a multiple of 8, this means that if you wish to capture 48 bytes of payload for instance, the `slen` must to set to 56.

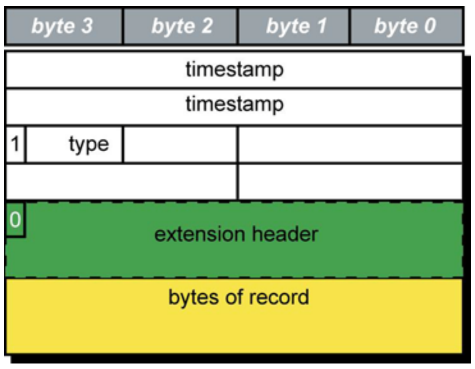
Extension Headers (EH)

Introduction

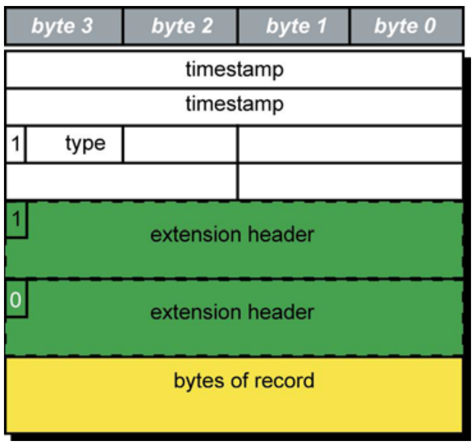
The addition of an Extension Header into the ERF record allows extra data relating to the packet to be transported to the host. The extension header allows certain features to be added independently of ERF types, for example, features shared by different ERF records do not have to be implemented separately. This results in automatic support across ERF types.

Bit 7 of the ERF type field is used to indicate that Extension Header's are present. If set to '1' Extension Headers are present. The Extension Header type field indicates the type and format of the Extension Header. It also indicates whether further Extension Headers are present. If bit 7 of the Extension Header is set to '1' further Extension Headers exist in the record. The Extension Headers are 8 bytes in length.

The following diagram shows presence of an Extension Header in addition to the ERF record.



The following diagram shows presence of two Extension Headers with Bit 7 of the first Extension Header set to '1'.



Reporting Problems

If you have problems with a 8.2X card or Endace supplied software which you are unable to resolve, please contact Endace Customer Support at support@endace.com.

Supplying as much information as possible enables Endace Customer Support to be more effective in their response to you. The exact information available to you for troubleshooting and analysis may be limited by nature of the problem.

The following items may assist in a quick resolution:

- 8.2X card[s] model and serial number.
- Host computer type and configuration.
- Host computer operating system version.
- DAG software version package in use.
- Any compiler errors or warnings when building DAG driver or tools.
- For Linux and FreeBSD, messages generated when DAG device driver is loaded. These can be collected from command `dmesg`, or from log file `/var/log/syslog`.
- Output of `daginf`.
- Firmware versions from `dagrom -x`.
- Physical layer status reported by: `dagconfig`
- Link statistics reported by: `dagconfig -si`
- Statistics either (depending on the DAG card):
  - Extended statistics reported by: `dagconfig -ei`
  - Universal statistics reported by: `dagconfig -ui`
- Network link configuration from the router where available.
- Contents of any scripts in use.
- Complete output of session where error occurred including any error messages from DAG tools. The `typescript` Unix utility may be useful for recording this information.
- A small section of captured packet trace illustrating the problem.
- If you have just rebooted and the system can not see any DAG cards, you need to load the DAG drivers. Run `dagload`.

Version History

Version	Date	Reason
1	July 2006	First beta release
2	August 2006	First production release. Changes to process for loading FPGA image. Minor typographical and diagrammatic changes.
3	November 2007	Converted to AuthorIT. Added new sections and information.
4	June 2008	Added card features and firmware image list. Updated tokens list. Added Universal counters. Added ERF Extension header information.
5	August 2008	Released. Removed terf_strip16 dagconfig option. This option is not supported by this DAG card. Corrected firmware image naming.
6	November 2008	Updated Buffer_size and mem dagconfig tokens and associated cross references. Updated front matter. Update dagconfig options table. Added new dagrom options. Supported OS information now in release notes. Added card description to the overview. Updated firmware images.

Status	Description
Preliminary	The products described in this technical document are in development and have yet to complete final production quality assurance.
Released	The products described in this technical document have completed development and final production quality assurance.