DAG 8.4I Card User Guide

EDM01-25

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

Whilst every effort has been made to ensure accuracy, neither Endace Technology Limited nor any employee of the company, shall be liable on any ground whatsoever to any party in respect of decisions or actions they may make as a result of using this information.

Endace Technology Limited has taken great effort to verify the accuracy of this manual, but nothing herein should be construed as a warranty and Endace shall not be liable for technical or editorial errors or omissions contained herein.

In accordance with the Endace Technology Limited policy of continuing development, the information contained herein is subject to change without notice.

Website

http://www.endace.com

Copyright 2008 Endace Technology Ltd. All rights reserved.

Contents

Introduction

Overview

The Endace DAG 8.4I card provides the means to transfer data into the memory of the host computer. Substantial buffering is provided to avoid packet loss in cases where data capture is limited by the PCIe bus bandwidth. 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 DAG 8.4I card is a two port, PCIe card that allows capture of data. It is designed for use with the InfiniBand interface. It provides full-rate capture of SDRx4 2.5 Gigabits per second (Single Data Rate) InfiniBand traffic with precision time stamping capability. The PCIe interface enables a capture rate of up to 13 Gigabits per second.

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 7).

- SDRx4 InfiniBand
- Classification

Purpose of this User Guide

The purpose of this User Guide is to provide you with an understanding of the DAG 8.4I 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 DAG 8.4I card in the following documents which are available from the Support section of the Endace website at http://www.endace.com:

- *EDM04-01 DAG Software Installation Guide*
- *EDM04-08 Configuration and Status API Programming Guide*
- *EDM04-19 DAG Programming Guide*
- *EDM05-01 Time Distribution Server User Guide*
- *EDM11-01 ERF types*

This User Guide and the *EDM04-01 DAG Software Installation Guide* are also available in PDF format on the installation CD shipped with your DAG 8.4I card.

System Requirements

General

The minimum system requirements for the DAG 8.4I card are:

- A computer, with at least a Intel Xeon 1.8GHz or faster and a minimum of 1GB RAM.
- At least one free PCI-e 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.
- Requires 12V external power via 4.2mm Pitch Mini-Fit Jr , 6 way, (PCI Express Card Power Connector).

Operating System

This document assumes you are installing the DAG 8.4I 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 DAG 8.4I 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

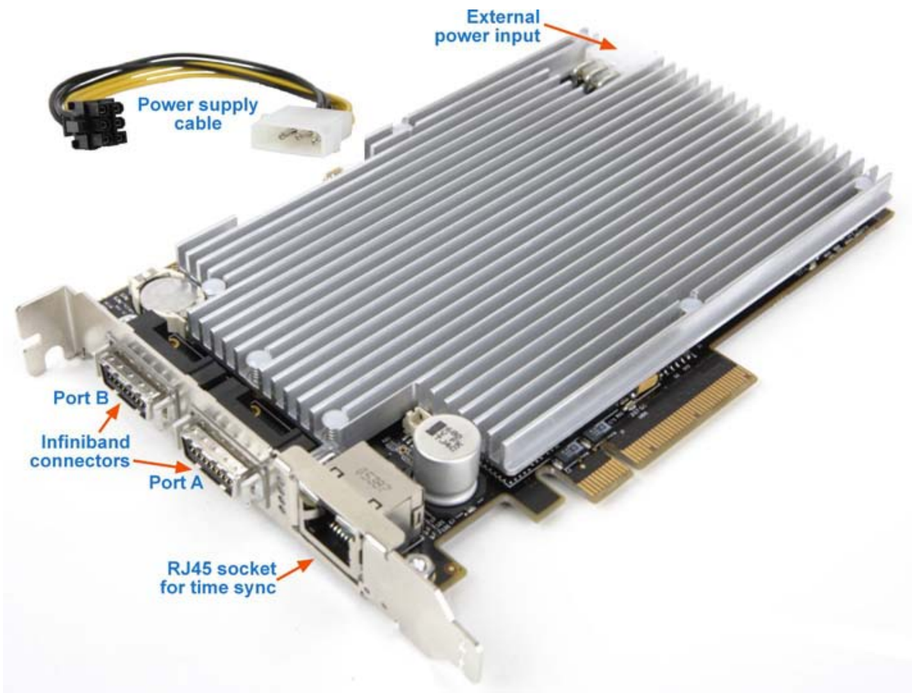
The Endace InfiniBand DAG 8.4I card is for use in InfiniBand networks running at 10 Gigabits per second over 4x copper cabling using single data rate (SDRx4, 2.5 Gigabits per second). The interface to the computer is an 8-lane PCIe, which has a throughput (receive-only) of 13 Gigabits per second.

The card has two inputs with InfiniBand 4x connectors. The DAG card can be used in:

- PORT mode, where each input is terminated at the DAG card, or
- TAP mode, where a single InfiniBand link passes through the card, and data is sampled from both directions of the link.

The DAG 8.4I Card has an additional power supply socket. This must be connected to a 12 V power supply using the supplied cable.

Note: For passive fail safe you required a NinjaITM. Please refer to *EMD09-20 NinjaITM User Guide*.



The key features of the card are:

- User configurable dual channel network monitoring for InfiniBand networks.
- 100% capture into host memory at up to 13 Gigabits per second
- Header only or variable length packet capture.
- SDRx4 InfiniBand network interface.
- Conditioned clock with one PPS input and local synchronization capability.
- 8 Lane PCIe bus interface.
- A TCAM external to the FPGA for packet classification.

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

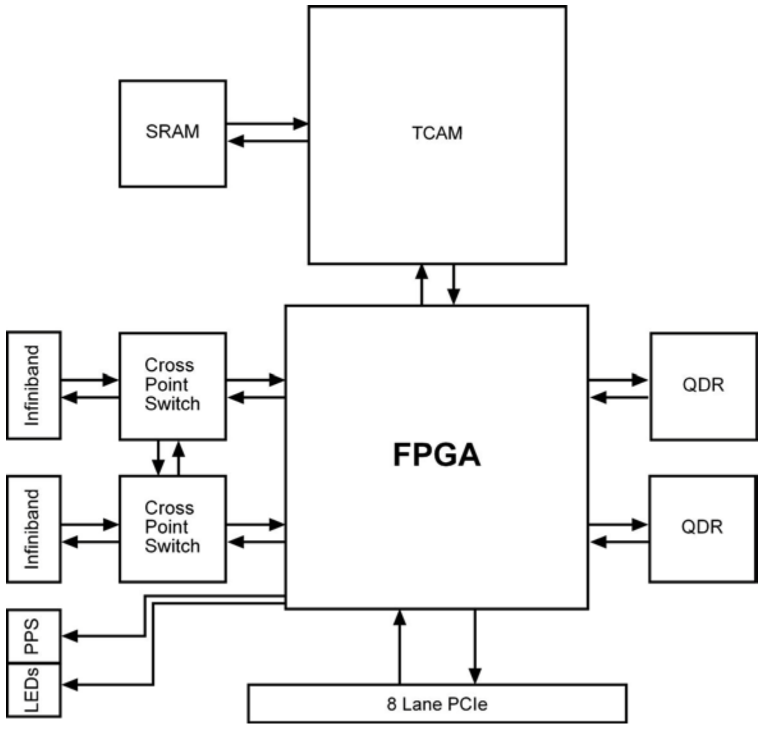
SDRx4 Infiniband data, received by the two Infiniband interfaces, flows directly in to the Field Programmable Gate Array (FPGA).

The FPGA contains the packet record processor 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 to Synchronizing Clock Time *(page 31)* later in this user guide.

Time stamped records are classified and can be optionally discarded without further processing depending on classification results. Packets passing the classification process 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:



InfiniBand network input and output

InfiniBand I/O to and from the DAG card is through two InfiniBand 4x connectors mounted on the bezel edge of the card. The two connectors are labeled A and B. Each input and output consists of four signal pairs.

Pin assignment on the InfiniBand connectors follows the InfiniBand 4x standard. For further details on the InfiniBand standard refer to the *InfiniBand™ Architecture Release 1.2* (http://www.infinibandta.org).

Installation

Introduction

The DAG 8.4I card operates on an 8 lane PCIe bus and can be installed in any free 8 lane PCIe slot. The PCIe bus allows multiple DAG cards to be installed without affecting the bandwidth used by each DAG 8.4I card.

Best performance (13 Gigabits per second) can only be achieved using an 8-lane PCIe slot, however the DAG card will function in a 4-lane PCIe slot.

DAG Software package

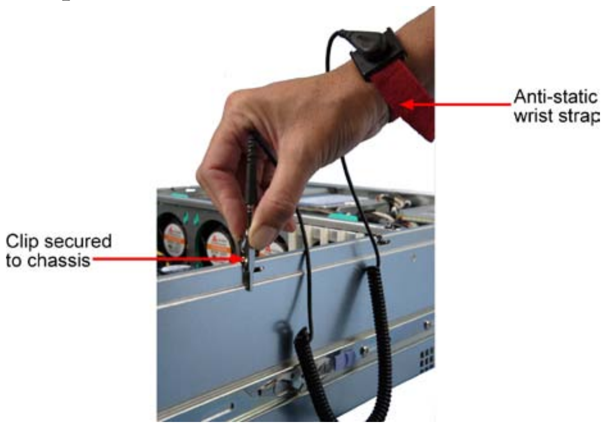
The latest DAG Software package must be installed before you install the DAG 8.4I card itself. See *EDM04-01 DAG Software Installation Guide*, which is included on the CD shipped with the DAG 8.4I card.

Inserting the DAG Card

Caution:

It is very important to protect both the computer and the DAG 8.4I 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.

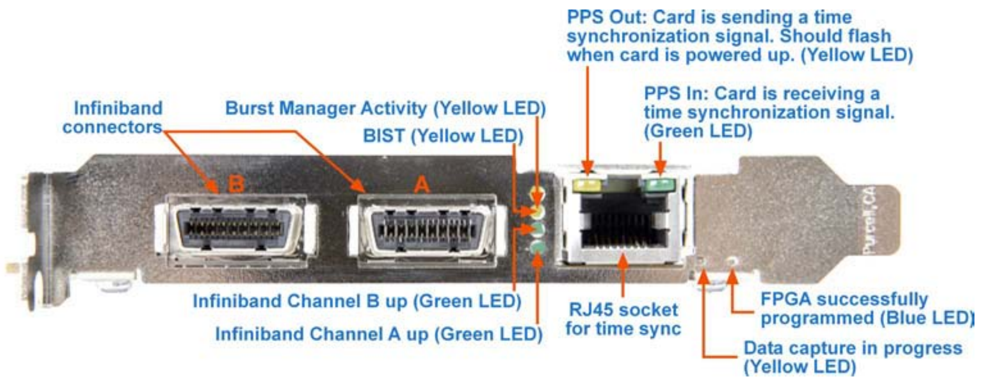


5. Insert the DAG 8.4I card into PCIe bus slot ensuring it is firmly seated.
6. If this DAG card requires an external power supply, complete the following steps:
   a. Connect the supplied (or equivalent) power cable to the external power connector on the DAG card.
   b. Connect the cable to the appropriate 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 8.4I card is for use in InfiniBand networks running at 10 Gigabits per second over 4x copper cabling using single data rate (SDRx4, 2.5 Gigabits per second).

The card has two inputs with InfiniBand 4x connectors.


PPS Out: Card is sending a time synchronization signal. Should flash when card is powered up. (Yellow LED)
Infiniband connectors
Burst Manager Activity (Yellow LED)
BIST (Yellow LED)
PPS In: Card is receiving a time synchronization signal. (Green LED)
B
A
Infiniband Channel B up (Green LED)
Infiniband Channel A up (Green LED)
RJ45 socket for time sync
FPGA successfully programmed (Blue LED)
Data capture in progress (Yellow LED)

There is an 8-pin RJ-45 socket located near the InfiniBand connectors which is available for connection to an external time synchronization source.

Caution:

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

External power supply

The DAG 8.4I PCB has an additional power supply socket. This must be connected to a 12 V power supply using the supplied cable.

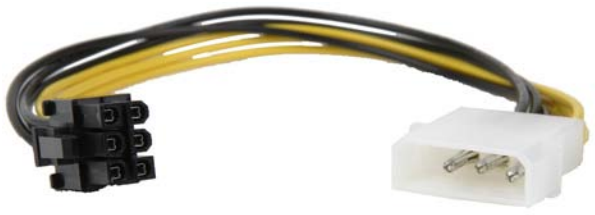
Caution:

Prolonged operation of the DAG 8.4I card without this extra power supply connected could damage parts of the card.

The current draw through this extra power connector is up to .

The external power supply connector is a 4.2mm Pitch Mini-Fit Jr , 6 way, (PCI Express Card Power Connector).

Some modern PSU units have connectors on the power supply designed to fit directly to this. Otherwise use the Endace supplied adapter cable to connects this input to a standard 5.25" Drive power connector on the system power supply unit.



Configuring the DAG card

Introduction

Configuring the DAG 8.4I card ready for capturing data requires the following steps:

- Setting up the FPGA (page 8)
- Preparing the DAG card for use (page 10)
- Configuring the DAG Card (page 11)
- Viewing the DAG card statistics (page 16)

Once the DAG 8.4I is configured you can start capturing data, see Using your DAG card to capture data (page 19) 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	SDRx4 InfiniBand
dag84ipci_erf.bit	✓

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 9).

Load the appropriate FPGA image using `dagrom` as follows:

```
dagrom -d0 -rvp -f dag84ipci_erf.bit
```

(where "0" is the device number of the DAG card you wish to capture data from). The filename of the FPGA image may differ from the above depending on the version required.

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:    dag84ipci_erf_v2_1 5vlx110tff1738 2008/06/24 20:03:27 (active)
factory:    dag84ipci_erf_pci_v2_7 5vlx110tff1738 2008/03/03 16:53:05
```

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 9). This eliminates unnecessary reprogramming of the ROM and extends its life.

Preparing the DAG card for use

Before configuring the DAG 8.4I card you must run the following `dagconfig` command to set the default parameters in the DAG card. This ensures the DAG 8.4I 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 DAG 8.4I configuration displays and the firmware is verified as correctly loaded. See dagconfig (page 15) for more information.

Configuring the DAG card

Display Current Configuration

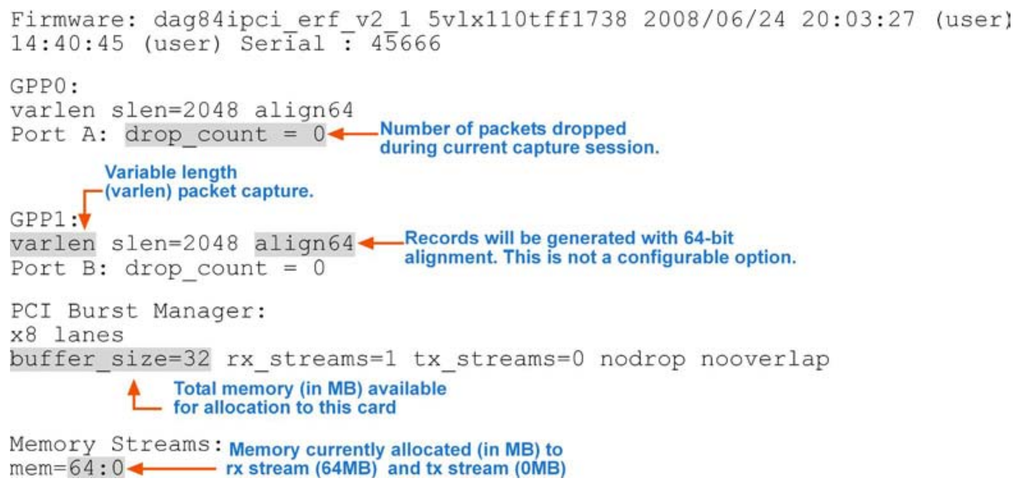
Once you have loaded the FPGA image you should run the `dagconfig` tool without arguments to display the current card configuration and verify the firmware has been loaded correctly.

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

```
dagconfig -d0
```

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

Note: Not all commands displayed in the following diagram.



```
Firmware: dag84ipci_erf_v2_1 5vlx110tff1738 2008/06/24 20:03:27 (user)
14:40:45 (user) Serial : 45666

GPP0:
varlen slen=2048 align64
Port A: drop_count = 0

GPP1:
varlen slen=2048 align64
Port B: drop_count = 0

PCI Burst Manager:
x8 lanes
buffer_size=32 rx_streams=1 tx_streams=0 nodrop nooverlap

Memory Streams:
mem=64:0
```

Number of packets dropped during current capture session.
Variable length (varlen) packet capture.
Records will be generated with 64-bit alignment. This is not a configurable option.
Total memory (in MB) available for allocation to this card
Memory currently allocated (in MB) to rx stream (64MB) and tx stream (0MB)

dagconfig tokens explained

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.

default

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

Example

```
dagconfig default
```

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.

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.

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 12) 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.

reset

Resets the ethernet framers, set auto mode.

Example

```
dagconfig -d0 reset
```

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` 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 you want to capture the whole packet you need to set the length to a larger value.

Example

Setting up a DAG 8.4I 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.

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.

varlen

The DAG 8.4I uses variable length capture. This allows you to capture the whole packet, providing its size is less than the `slen` value. Therefore to be effective you should set the `slen` value to the largest number of bytes that a captured packet is likely to contain.

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 save bandwidth and storage space.

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

```
dagconfig –d0 slen=9600
```

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
```

For more information see `dagconfig` (page 15).

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

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  rx_byte_align  rx_channel_align     rx_link_up
A     0              0                    0
B     0              0                    0


rx_packet_error_latch   pll detect    GTP Reset Status
0                       1             0
0                       1             0
```

Status Conditions

A definition of each status condition up to printout level 2 is described below.

Condition	Definition
`GTP Reset Status`	The GTP Reset for all the four lanes.
`lane_zero_rx_byte_align`	Rx Byte Align for lane zero.
`lane_one_rx_byte_align`	Rx Byte Align for lane one.
`lane_two_rx_byte_align`	Rx Byte Align for lane two.
`lane_three_rx_byte_align`	Rx Byte Align for lane three.
`lane_zero_rx_channel_align`	Rx Channel Align for lane zero.
`lane_one_rx_channel_align`	Rx Channel Align for lane one.
`lane_two_rx_channel_align`	Rx Channel Align for lane two.
`lane_three_rx_channel_align`	Rx Channel Align for lane three.
`pll detect`	The Phase Locked Loop detected for tile 0 and tile 1.
`Port`	Indicates the port the status conditions apply to.
`rx_byte_align`	Rx Byte Align for all four lanes.
`rx_channel_align`	Receive Channel Bonding for all four lanes.
`rx_link_up`	Indicates the receive link status. 0 = down, 1 = up.
`rx_packet_error_latch`	Rx Packet Error Latch.

Extended Statistics

Extended statistics are also available:

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

The following example shows the extended statistics.

```
dagconfig -e

Port      rx_byte_count
A         0
B         0
```

Status Conditions

A definition of each status condition up to printout level 2 is described below:

Condition	Definition
`Port`	Indicates the port the status conditions apply to.
`rx_byte_count`	Received byte counter .(Wraps Around).
`rx_packet_count`	Received packet counter .(Wraps Around).

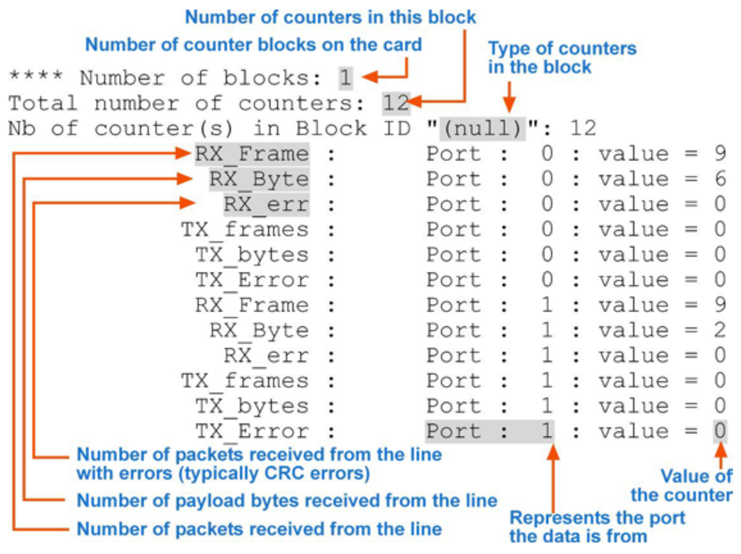
Universal counters

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 15).`

Example outputs are shown below:



```
**** Number of blocks: 1
Total number of counters: 12
Nb of counter(s) in Block ID "(null)": 12
              RX_Frame :      Port :  0 : value = 9
               RX_Byte :      Port :  0 : value = 6
                RX_err :      Port :  0 : value = 0
             TX_frames :      Port :  0 : value = 0
              TX_bytes :      Port :  0 : value = 0
              TX_Error :      Port :  0 : value = 0
              RX_Frame :      Port :  1 : value = 9
               RX_Byte :      Port :  1 : value = 2
                RX_err :      Port :  1 : value = 0
             TX_frames :      Port :  1 : value = 0
              TX_bytes :      Port :  1 : value = 0
              TX_Error :      Port :  1 : 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:

- Capturing data (page 19)
- Viewing captured data (page 22)
- Using third party applications (page 25)

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 21).

For further information on the software utilities see:

- dagsnap (page 20)
- dagbits (page 22)
- dagconvert (page 24)

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 DAG 8.4I 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 DAG 8.4I 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 43) 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 12).

Viewing captured data

Data captured in ERF format can be viewed with `dagbits`. For further details on how to use `dagbits`, see dagbits (page 22).

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 25).

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.

Additional capture feature

`dagconvert` is a software utility that enables you to capture and filter InfiniBand data. `dagconvert` can not convert InfiniBand ERF into pcap as Libpcap does not support InfiniBand.

Examples

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

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

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 labelled `outfile0000.erf`, once the file size reaches 128MB, a second file is created. The second is labelled `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)
  Wireshark (version 0.99.8 onwards) can read native InfiniBand packets.

Note: Wireshark can also read ERF formatted data.

Classification

Overview

The DAG 8.4I classification feature, allows you to accept or reject InfiniBand packets based upon the following:

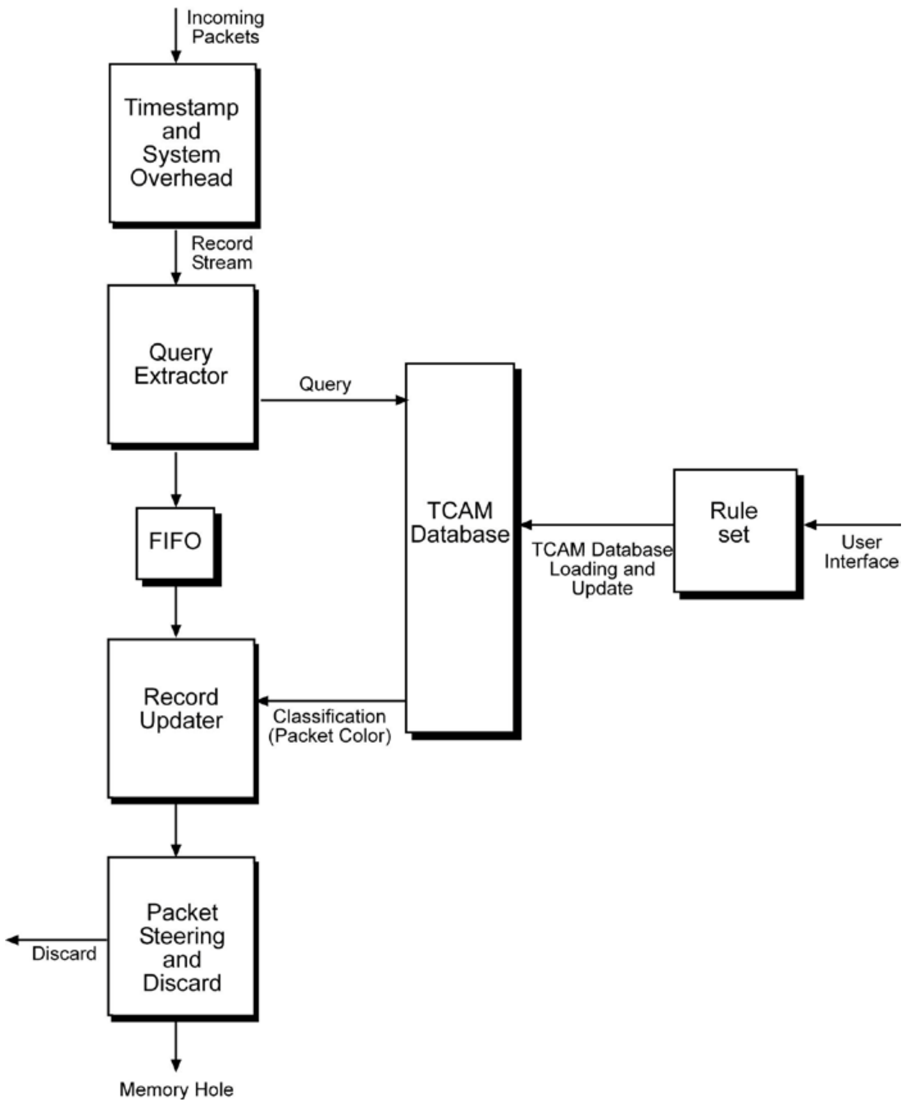
- SLID - Source Identifier
- DLID - Destination Identifier
- SL - Service Level
- LNH - Local Routing Header field Link Next Header
- OPCODE - Base Transport Header Operation Code
- DQP - Destination Queue Pair
- SQP - Source Queue Pair

The output of the classification is then rejected or accepted. The output tag is then packaged into a 16-bit user defined packet tag.

For further information about InfiniBand classification see *EDM04-22 InfiniBand Filter Software Guide*.

filter_loader application

The `filter_loader` application loads the filters into the TCAM and associated memory and activates the filtering operation. The flow of data is shown below:



filter_loader operation

The `filter_loader` loads filter rule sets (from a text file) into the database on a TCAM equipped DAG card running classification firmware. Each filter rule set defines what packets are dropped or colorized. Packets are captured in either ERF or PCAP format depending on the application used to capture the data.

Filter rule set are manually written in the Endace filter format.

A DAG card running classification firmware supports up to eight databases, one active and the other inactive. Each filter rule set can contain up to 16k of filter entries (rules).

Each time you load a new filter rule set into the TCAM database you need to define where it will be stored, database - 0 to 7.

Once the first filter rule set is loaded, you can:

- Make the new filter rule set active using `dagconfig` and changing the active database.
- Load the new filter rule set using `filter_loader`.
- Hot swap between different filter rule sets using `dagconfig`. This allows the filter rule set to be dynamically modified with zero packet loss.
- Set the default color and classification value in case no rule has been matched (miss) using `dagconfig`.

The following shows the size of the filter sets for different card configurations.

Interfaces	Hot-Swap Ability	DAG 8.4I Filters Per Set	DAG 8.4I Sets (databases)
2	Yes	16k	8(0-7)

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 DAG 8.4I implements the 28 most significant bits which provides a time resolution of 3.7 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 36).

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 36)* 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 36). 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 35).
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 35)

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 35).
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 35). 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 35).
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 35).

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 35)
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 42) (**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 42) (**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

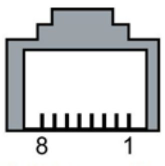
Overview

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 42) 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 DAG 8.4I supports the following ERF Types:

ERF Type	Description
21	TYPE_INFINIBAND Infiniband 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.

The addition of an Extension Header into the ERF record allows extra data relating to the packet to be transported to the host.

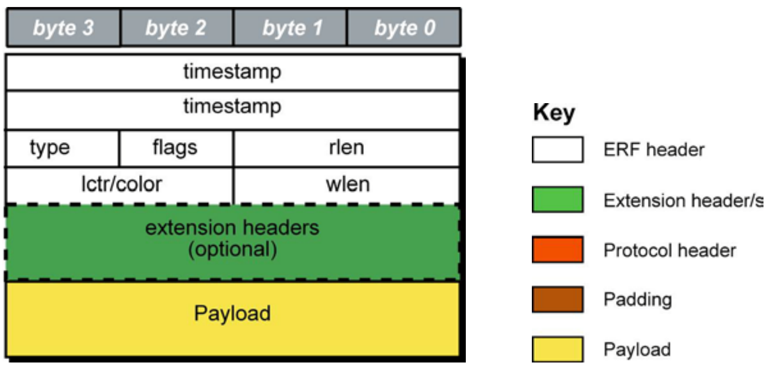
Extension header	Description
3	Classification Used to report filter and steering results. Used in conjunction with ERF 21. TYPE_INFINIBAND (page 46)

For information on other ERF types and Extension Headers, 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

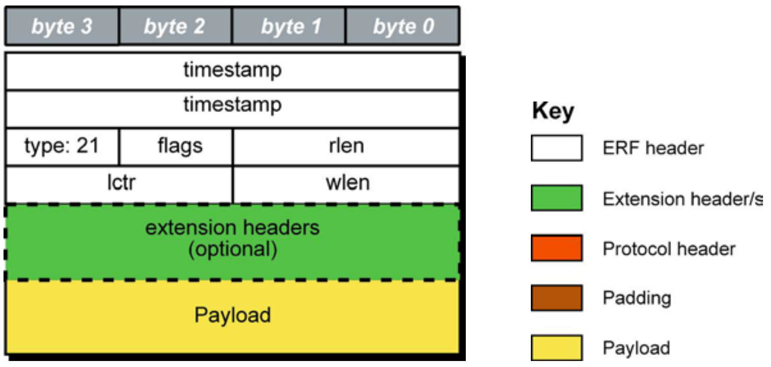
Extension header types

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

ERF 21. TYPE_INFINIBAND

Type	Bit 7	1 = Extension header present. See Extension Headers (page 47).
	Bits 6:0	Type 21
Short description	TYPE_INFINIBAND	
Long description	Type 21 Infiniband Variable Length Record.	
Use	This record format captures Infiniband data. Used in conjunction with EH 3. Classification (page 48).	

The *TYPE_INFINIBAND* record is shown below:



Field	Description
Payload (bytes of record)	Payload = rlen - ERF header (16 bytes) - Extension headers (optional)

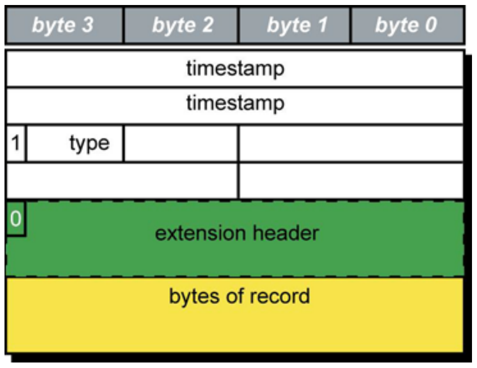
Extension Headers (EH)

Introduction

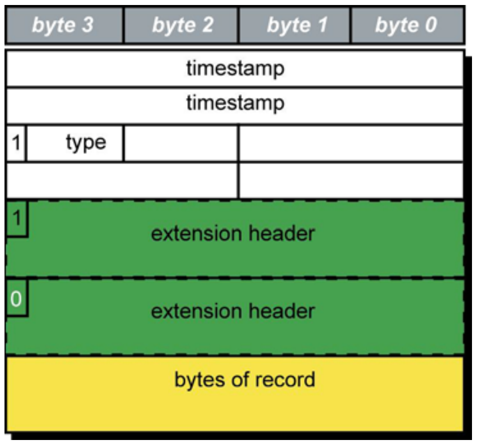
The addition of an Extension Header into the ERF record allows extra data relating to the packet to be transported to the host. The extension header allows certain features to be added independently of ERF types, for example, features shared by different ERF records do not have to be implemented separately.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



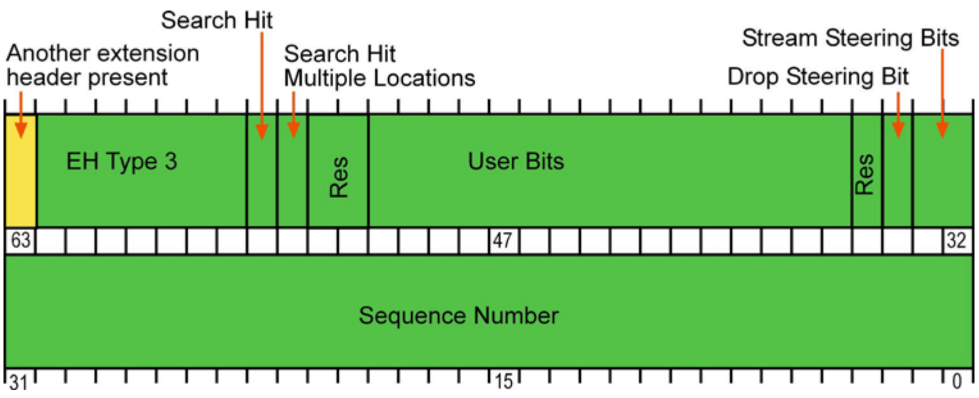
The following diagram shows presence of two Extension Headers with Bit 7 of the first Extension Header set to '1'.



EH 3. Classification

Type	Bit 7	Extension header present
	Bits 6:0	Type 3
Short description	Classification	
Long description	-	
Use	Used with ERF 21. TYPE_INFINIBAND (page 46). Entries marked Metadata are derived by firmware. Entries marked SRAM are stored in the TCAM Associated SRAM.	

Note: The following is provisional and subject to change.



The following details the make up of the *Classification* Extension Header:

Bit	Length	Meaning
63	1	More Extension Headers present (1 = more)
62:56	7	0x03 - Assigned type code.
55	1	Search Hit, rest of bits are meaningful.
54	1	Search Hit Multiple Locations, lowest-numbered shown.
53:52	2	Reserved.
51:36	16	User Bits.
35	1	Reserved.
34	1	Drop Steering Bit. May have Stream Steering bits set too.
33:32	2	Stream Steering Bits. Binary encoded.
31:0	32	Sequence Number from Blackbird framer chip.

Note: For NinjaProbe 40G1 this is the format at the output of the RXOne chip, and therefore the input to the Steering logic and also the Software.

Bits 31:0 are optional for InfiniBand and can be sequence number or set to 0. TCAM Associated SRAM Data (colour) for InfiniBand and NinjaProbe 40G1 used for classification.

Bit	Length	Meaning
31:20	12	Reserved set to 0
19:4	16	Tag (user classification(data))
3	1	Reserved
2	1	Drop Steering Bit. May have Stream Steering bits set too.
1:0	2	Stream Steering Bits. Binary encoded.

Reporting Problems

If you have problems with a DAG 8.4I 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:

- DAG 8.4I 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	March 2008	First release
2	May 2008	Added Classification information
3	June 2008	Added ERF Extension header information. Added card features and firmware image list.
4	November 2008	Updated Buffer_size and mem dagconfig tokens and associated cross references. Updated front matter. Update dagconfig options table. Supported OS information now in release notes. Added external power supply information. Updated classification section. Updated dagrom section. Updated data capture section.

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.