



endace
accelerated

SAR API

EDM04-13



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 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 Limited has taken great effort to verify the accuracy of this manual, but assumes no responsibility for any technical inaccuracies or typographical errors.

In accordance with the Endace Limited policy of continuing development, design and specifications are subject to change without notice.

Published by:

Endace Technology® Ltd

Level 9
85 Alexandra Street
PO Box 19246
Hamilton 2001
New Zealand

Phone: +64 7 839 0540
Fax: +64 839 0543
support@endace.com
www.endace.com

International Locations

New Zealand

Endace Technology Limited
Building 7
Lambie Drive
PO Box 76802
Manukau City 1702
New Zealand
Phone: +64 9 262 7260
Fax: +64 9 262 7261

Americas

Endace USA® Ltd
Suite 220
11495 Sunset Hill Road
Reston
Virginia 20190
United States of America
Phone: +1 703 382 0155
Fax: +1 703 382 0155

Europe, Middle East & Africa

Endace Europe® Ltd
Sheraton House
Castle Park
Cambridge CB3 0AX
United Kingdom
Phone: +44 1223 370 176
Fax: +44 1223 370 040

Copyright 2007 Endace Technology Ltd. All rights reserved.

No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means electronic, mechanical, photocopying, recording, or otherwise, without the prior written permission of the publisher.

Table of Content

Purpose	1
Revision History.....	1
Introduction	2
Overview of the AAL reassembler.....	2
Notes on DAG3.7T specific functions.....	2
Functional Overview of the AAL Reassembler	3
Default behavior of the Embedded Software.....	3
AAL2 specific default behaviour.....	4
AAL5 specific default behaviour.....	4
Activating and Deactivating Virtual Connections.....	4
Activating and Deactivating Channels (CID).....	5
Configuration Options.....	5
Statistics.....	5
Scanning.....	6
Filtering.....	6
General Filtering.....	6
Extended Filtering (only DAG 3.7T).....	7
Extended Filtering Examples.....	7
DAG Sar Function Definitions	9
dagsar_vci_activate.....	9
Description.....	9
Arguments.....	9
Return Codes.....	10
dagsar_vci_deactivate.....	11
Description.....	11
Arguments.....	11
Return Values.....	11
dagsar_cid_activate.....	12
Description.....	12
Arguments.....	12
Return Codes.....	12
DAG3.7T Specific Return Codes.....	12
dagsar_cid_deactivate.....	13
Description.....	13
Arguments.....	13
Return Values.....	13
DAG3.7T Specific Return Codes.....	13
dagsar_vci_set_sar_mode.....	14
Description.....	14
Arguments.....	14
Return Codes.....	14
dagsar_vci_get_sar_mode.....	15
Description.....	15
Arguments.....	15
Return Codes.....	15
dagsar_channel_set_net_mode.....	16
Description.....	16
Arguments.....	16
Return Codes.....	16
dagsar_set_buffer_size (only DAG3.7T).....	17
Description.....	17

Return Codes	17
dagsar_get_stats.....	18
Description.....	18
Arguments	18
Return Codes	19
dagsar_get_interface_stats (deprecated)	20
Description.....	20
Return Codes	20
DAG3.7T Specific Return Codes	20
dagsar_reset_stats.....	21
Description.....	21
Arguments	21
Return Codes	22
dagsar_reset_stats_all	23
Description.....	23
Arguments	23
Return Codes	23
dagsar_set_filter_bitmask.....	24
Description.....	24
Arguments	24
Examples	25
Return Codes	25
dagsar_reset_filter_bitmask	26
Description.....	26
Arguments	26
Return Codes	26
dagsar_init_scanning_mode	27
Description.....	27
Arguments	27
Return Codes	27
dagsar_set_scanning_mode	28
Description.....	28
Arguments	28
Return Codes	28
dagsar_get_scanning_mode.....	29
Description.....	29
Arguments	29
Return Codes	29
dagsar_get_scanned_connections_number.....	30
Description.....	30
Arguments	30
Return Codes	30
dagsar_get_scanned_connection	31
Description.....	31
Arguments	31
Return Codes	31
dagsar_set_trailer_strip_mode (only DAG7.1S).....	32
Description.....	32
Arguments	32
Return Codes	32
dagsar_get_trailer_strip_mode (only DAG7.1S)	33
Description.....	33
Arguments	33
Return Codes	33

Data Structures, Attributes, Defines and Enums	34
sar_mode_t.....	34
net_mode_t.....	34
stats_t.....	34
filter_action_t.....	35
filter_operations_t (only DAG3.7T).....	35
dag_filter_level_t (only DAG3.7T).....	35
list_operations_t (only DAG3.7T).....	35
scanning_mode_t.....	36
connection_info_t.....	36
trailer_strip_mode_t.....	36
Data Formats	37
Introduction.....	37

Purpose

This document describes the software interface of the “Segmentation and Reassembly API” (SAR API). The SAR API runs on the host computer and manages the AAL segmentation and reassembly software running on an embedded processor in a DAG card.

User programs wanting to use the SAR features on a DAG card need other libraries in order to manage the embedded software. The “DAG Embedded Message API” (from now on DAGEMA) it is needed to communicate with the embedded processor. Also the “DAG Configuration API” may be needed to configure the DAG card. This document doesn’t cover other libraries than the SARAPI, although some references can be done to the other APIs. Each library has its own documentation, further information can be obtained reading them.

Not all DAG cards support SAR. At the moment this document was created only DAG 3.7T and DAG 7.1S support it. There are important architecture differences between those cards. Differences sometimes are reflected on the software interface. When the behavior is not the same on all the cards, a separate note is added for each of them.

Please be aware that this document is subject to change as additional functionality becomes available. It is recommended to always consult the latest revision of this document.

Revision History

Rev.	Date of Change	Description of Change	Revision Originator
0.1	13/March/2006	Initial version. This is a major modification from document “DAG 3.7T Host API”, Cassandra, 20-Dec-2005.	Abel
0.2	17/July/2007	Added the <code>dagsar_set_trailer_strip_mode</code> and <code>dagsar_get_trailer_strip_mode</code> functions.	Ben Gray

Overview of the AAL reassembler

The AAL reassembler is designed to allow reassembling of AAL5 or AAL2-SSSAR frames on the card without involving the host in processing. The reassembler will receive ATM traffic from the lines, this traffic is then either sent to the host unchanged, dropped or reassembled into AAL5 or AAL2-SSSAR frames, depending on the configuration used. Different configurations can be utilized on different virtual connections, allowing only what data is required to be reassembled, other data can be preserved or rejected.

When reassembly is used, the frame is constructed from the received data. The frame will end when:

The reassembly type is AAL5 and the Payload Type Indication (PTI) indicates the end of the frame has been received or, the addition of another ATM cell will cause the buffer allocated to the virtual connection to overflow.

The reassembly type is AAL2-SSSAR and the User to User Indication (UUI) indicates that the end of the frame has been received or, the addition of another CPS packet will cause the buffer allocated to the virtual connection and channel (cid) to overflow.

If the frame is AAL5 and has been completed with a PTI end of message indication the length and CRC contained in the trailer is checked. On the DAG 3.7T the AAL5 trailer is removed, on the DAG 7.1S the trailer is kept on the reassembled frame. Errors are indicated in the ERF or Multi-channel header.

If the frame is completed due to a buffer overflow it is assumed the length is incorrect. The error is indicated and no attempt is made to check or remove the AAL5 trailer even if the connection is in the AAL5 reassembly mode.

Notes on DAG3.7T specific functions

Function definitions are described in later chapters of this document. Functions which begin with the prefix *d37t* are available from the DAG3.7T specific library. These functions will be similar, where possible, across different embedded software for the 3.7T, for example, functions for receiving the Software ID will be similar for AAL reassembly, and for IMA.

Functional Overview of the AAL Reassembler

Default behavior of the Embedded Software

After the embedded processor is running, the card will start with all possible Virtual Connections idle. As data is detected on a connection, the virtual connection will be activated, as an ATM cell connection (AAL0), with no reassembly occurring. The virtual connection can be deactivated by using the function **dagsar_vci_deactivate()**. This default behavior can be overridden by changing the SAR mode before data is seen on a connection, this will stop the activation of a ATM Virtual Connection.

All virtual connections default to using the Network to Network Interface (NNI). To use User to Network Interface (UNI), the mode can be changed using the **dagsar_set_net_mode()** function, described later in this document.

DAG 3.7T

Data is returned to the card in bursts. By default the burst size is one megabyte, this can be changed by using the **aal_msg_set_burst_size()** function. Altering this size can make significant differences to the throughput of the AAL reassembler, depending on the traffic, increasing the size introduces higher latency. When the burst size is changed, during reassembly, the current block of data will be filled and sent at the previous burst size. All further bursts will be sent at the new size. If the traffic source is to be stopped (for example if the card is to be disconnected) then the **aal_msg_flush_burst_buffer()** function can be used to force any remaining part of the block to be sent to the host. There is also a timeout mechanism when AAL frames are being reassembled. This will cause an incomplete AAL frame to be sent to the burst buffer, if no further cells are seen on the connection for 5 seconds after the previous cell. This is to prevent incomplete AAL frames from being stalled in the card.

DAG 7.1S

Data is returned to the card as soon as it is available. The functions **aal_msg_set_burst_size()** and **aal_msg_flush_burst_buffer()** don't apply to this card. There is no timeout mechanism for reassembled frames. If an AAL2 or AAL5 frame is incomplete, the buffer allocated for that connection will remain used until the frame is completed or the embedded processor is reset.

AAL2 specific default behaviour

DAG 7.1S

still to be defined

When a virtual connection has been set to reassemble AAL2-SSSAR frames the virtual connection can be activated in two different ways. It can be activated by either, **dagsar_vci_activate()** or **dagsar_cid_activate()**. When a virtual connection is activated, in AAL2 reassembly mode, all detected channels (CID) will attempt to reassemble any data that is detected, unless the channel (CID) has been previously deactivated using the **dagsar_cid_deactivate()** function.

The AAL2-SSSAR frame returned will be in an Extensible Record Format (ERF) of TYPE_MC_AAL2. This ERF type is described at the end of this document.

When reassembly is occurring, if the reassembler detects an error in the length of the reassembled frame the error bit will be set in the Multi-channel header in the ERF record. The reassembler will then attempt to continue reassembling on the affected channel (CID).

The AAL reassembler will also identify and report MAAL errors as described in the International Telecommunication Unions specification document ITU I.363.2. Currently, the MAAL errors with errnum from 0 to 8 are reported. MAAL error 9 "The CID value in the received CPS-Packet header is not associated with a SAP" is not currently implemented, as any unconfigured CIDs will be reassembled and returned to the host, unless deactivated.

MAAL errors will be reported by sending a MAAL ERF packet. This packet will be of type TYPE_MC_AAL2 (12), and will have the MAAL error indication bit set. The ATM Header field will be completed, and the CID will also be set where appropriate. The first data byte will contain the MAAL errnum, as specified in the specification.

AAL5 specific default behaviour

If a connection has completed reassembling an AAL5 frame, the length of the packet will be checked against the value in the AAL5 trailer. If the length of the frame is incorrect the length error bit will be set, and the frame will be returned without any data removed. If the length is correct the CRC is also checked. On the DAG 3.7T the trailer and padding is removed, on the DAG 7.1S trailer and padding are made available to the user. If the CRC is incorrect this will be indicated in the CRC error indication bit in the Multi channel header, in this case the trailer and padding will still be removed. When working in concatenated modes only the rx_error bit is set on the ERF header flags field, whether it is a length or a crc error.

The AAL5 frame returned will be in an Extensible Record Format (ERF) of TYPE_AAL5 or TYPE_MC_AAL5 (for concatenated / channelized modes). These ERF types are described at the end of this document.

Activating and Deactivating Virtual Connections

Activating and deactivating Virtual Connections can be done by using the function **dagsar_vci_activate()** and **dagsar_vci_deactivate()**. By default all connections are activated.

When a virtual connection is activated memory will be allocated for the connection. This can be used until the connection is deactivated. It is recommended to deactivate all those connections not being inspected to save memory on the embedded processor. If the embedded processor runs out of memory, attempting to activate more connections will fail.

Any change to a virtual connection, like changing the sar mode or the buffer size (3.7T), first requires the connection to be deactivated. After the change is done the connection can be activated again.

Activating and Deactivating Channels (CID)

Activating and deactivating Channels to reassemble AAL2-SSSAR frames can be done in a similar way to Virtual Connections, by using the functions **dagsar_cid_activate()** and **dagsar_cid_deactivate()**. These functions allow individual cids to be turned on and off, within a virtual connection. All cid operations require the Interface, Connection Number, VPI and VCI to be identified, and will only occur on the specified virtual connection.

For these functions to succeed the virtual connection must be put into the AAL2 SAR mode, as described below, prior to an activate or deactivate function call. The individual channels will then be set as requested.

DAG 3.7T

When a CID is activated, memory will be allocated for the connection. This can be used until the connection is deactivated or the virtual connection is deactivated. If the amount of memory allocated needs to be changed, for example, if a connection has been assembling smaller AAL2 frames and will now be assembling AAL2 frames which are larger than the currently allocated buffer, then the CID must be deactivated first. Then the configuration options altered with **dagsar_set_buffer_size()** and finally the connection reactivated. If multiple CIDs are to be changed to a particular size only, one call to **dagsar_set_buffer_size()** is required.

Because memory is a limited resource on the DAG3.7T reassembler, it is recommended that only the channels that will be in use should be activated. A total of 128MiB of buffer space is available for Virtual Connections and CID's. Attempting to activate more channels when all memory space is used will fail.

Configuration Options

The AAL Reassembler allows some configuration options to be altered.

dagsar_vci_set_sar_mode() allows a virtual connection to be altered between AAL2 and AAL5 reassembly or AAL0 (unchanged).

dagsar_vci_get_sar_mode() allows the current reassembly mode of a Virtual Connection to be retrieved from the card.

dagsar_channel_set_net_mode() sets the UNI or NNI setting for the specified channel.

DAG 3.7T

dagsar_set_buffer_size() allows the amount of memory allocated for new Virtual Connections or channels(CID) to be set. The buffer size should be provided in bytes and will only affect new AAL5 or AAL2 CID connections, as they are activated. When **dagsar_set_buffer_size()** is changed, the value is retained until the XScale is reset, or another call to **dagsar_set_buffer_size()** overwrites the value.

Statistics

The AAL reassembler provides statistic counters. Values are available via the **dagsar_get_stats()** function, by giving different statistic requested values.

The values will be reset via the **dagsar_reset_stats_all()** function, and by the **dagsar_reset_stats()** function, when the particular statistic is specified. The **dagsar_reset_stats_all()** function should be called at the beginning of a user program to reset all counters. For further information on which statistics are currently available consult the function definitions.

Scanning

Scanning is used to be automatically aware of the connections present on the physical links. When the scanning mode is set to on, the host software will gather all the observed connections which are currently unconfigured or deactivated and receiving data. Virtual Connections which are currently configured to return data to the host will not be recorded.

The process of scanning begins with initialization. This allows the system to prepare for recording data and also removes any previous scanning information which may not be correct at the later time. The scanning mode can then be set which will mean the software starts recording the observed connections. During the scanning mode recorded data is not available until scanning mode is turned off again.

During scanning there is an extra load on the system. Although data on configured Virtual Connections may be returned to the host, the AAL reassembler will not necessarily perform correctly or efficiently during scanning. It is therefore not recommended to leave the card in this state during normal operation.

After a period of time, in scanning mode, with data available on some unconfigured or deactivated lines, the scanning mode can then be set to off. When the scanning mode is off the recorded data becomes available. To access the data, first find the total number of entries recorded by calling `dagsar_get_scanned_connections_number()`. The entries can then be indexed from 0-(Total number of entries-1). Entries can be received using the `dagsar_get_scanned_connection()` function. All entries will persist until the application is terminated, or the scanning system is initialized again.

Filtering

All filtering is performed on ATM cells, prior to any other processing. Filtering is not available on reassembled AAL2 or AAL5 frames.

General Filtering

As well as the activation/deactivation of Virtual Connections, there is another possibility to filter cells from the AAL reassembler. Bit masks can be applied to the ATM header and depending if there is a match, the cell can then be discarded. The user has to provide two values to set the filter: a *bit mask* and a *match* value. To each ATM header (4 bytes, HEC is removed) the system calculates the logical AND with the *bit mask*. The result is compared with the *match* value. If they are equal, the result of the match is decided by the value of the *action* argument, which can take two values: *sar_accept* or *sar_reject*. If the values do not match, then the opposite result is taken.

DAg 7.15

The general filtering, also known as bitmask filtering, is highly recommended to be used. This first line filtering removes extra processing on the embedded processor, making the reassembly process faster and more reliable against packet losses. The bitmask filtering can operate with no problem at 4xOC12 speeds.

There is one bitmask filter per interface port, four in total.

Extended Filtering (only DAG 3.7T)

The extended filter module is able to perform simple comparison operations on any contiguous four bytes of data in the atm cell.

There are two types of filters available. The first is an operational filter which applies a mask to 4 bytes of data, and compares the result to an expected value using a defined filter comparison operator. This is similar to the general SAR filtering described above with the extensions of being able to use other logical operators, and being able to apply the filter operator

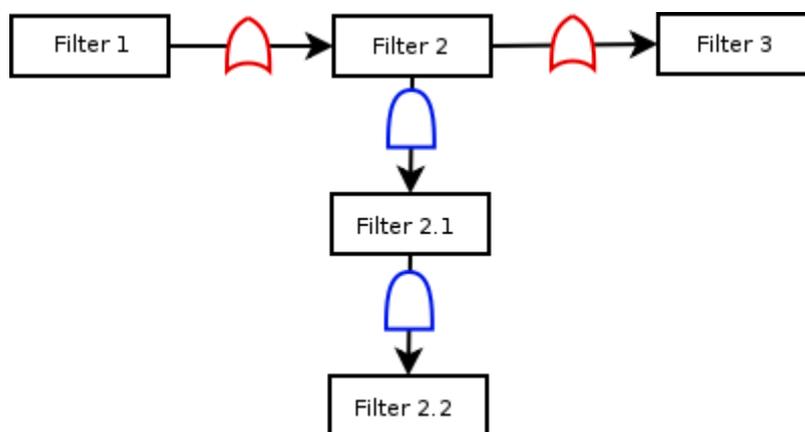
The logical operator is determined after the data and *bit mask* values have been logically AND'd together. In general filtering, this is a simple equal operator, where the result must be equal to the *match* value for the filter to be true. In extended filtering, this can also be: not equal, greater than, less than etc. For a full list of the available operators refer to the function definition.

The second type is a history filter, which applies a mask to 4 bytes of data (using an AND operator) and compares it to the masked value from the last time the filter run. If the result is a match, the filter is true and *action* is taken. If they do not match then the opposite *action* is taken.

It is possible to have up to 32 different filters configured at any one time. This limit includes the one possible general filter entry.

Extended Filtering Examples

Extended Filtering also allows multiple filters to be linked together in a tree structure. This allows multiple levels of filters to be constructed by adding sub-filters to filters. If for example, a filter structure is created with filters such as this:



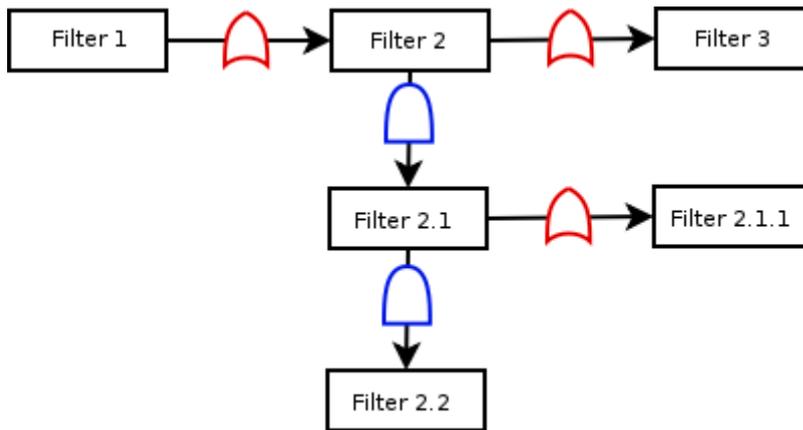
In the above example,

If Filter 1 is true, then *action* is taken.

If Filter 1 and 2 are false, there is no need to evaluate Sub-filters 2.1 and 2.2 so Filter 3 is evaluated. If it is true *action* is taken, otherwise the opposite of *action* is taken.

If Filter 1 is false, and Filter 2 is true then Sub-filters 2.1 and 2.2 are evaluated. If they are both true *action* is taken. If either or both of Filters 2.1 and 2.2 are false the opposite of *action* is taken.

To take this concept further, sub-filters can be added to sub-filters, for example:



With this example, in order for Sub-filter 2.1 to be true, either of Sub-filters 2.1 or 2.1.1 needs to be true and if both of them are false then the entire Filter 2 Sub-filter tree is false and Filter 3 would then be applied.

Although this system allows a large amount of flexibility, it should be noted that adding many filters places additional load on the system. Therefore, it is not efficient to add many filters, unless doing so causes the majority of ATM cells to be dropped.

DAG Sar Function Definitions

dagsar_vci_activate

```
uint32_t dagsar_vci_activate
    uint32_t    dagfd
    uint32_t    iface
    uint32_t    channel
    uint32_t    vpi
    uint32_t    vci
```

Description

The **dagsar_vci_activate()** function sets a virtual connection to return data. This will include allocating memory for use in the reassembly of AAL frames.

DAG 3.7T

The amount of memory allocated will be either the buffer size and an ERF overhead as previously set by **dagsar_set_buffer_size()** or 64kB. The total amount of memory available for reassembling is 128MiB. Attempting to open more connections when the total memory is used will not succeed.

DAG 7.1S

The maximum amount of simultaneously activated connections is 8192. Attempting to open more connections will not succeed.

Note that virtual connections are activated by default. The first packet received from that connection will activate it unless it has been filtered or deactivated before.

Arguments

- dagfd** The file descriptor for the DAG card as returned from `dag_open()`. This card also should have been initialized via `dagama_open_conn()`.
- iface** Specifies the interface of the virtual connection to be activated. In the case of the DAG3.7T card, this should always be zero.
- channel** Specifies the channel of the virtual connection to be activated. This is the same as the connection number given in the Multi-channel ERF header. For concatenated network links this should always be zero.
- vpi** Specifies the virtual path identifier of the virtual connection to be activated.
- vci** Specifies the virtual channel identifier of the virtual connection to be activated.

Return Codes

Code	Description
0	Function was successful.
!0	Function was unsuccessful

Code	Description
-1	Memory resources not available (only DAG3.7T)
-2	Virtual Connection is currently activated (only DAG3.7T)
-3	Timeout communicating with the XScale.
-4	Configuration could not be altered (only DAG3.7T)
-5	Unsupported sar_mode type (only DAG3.7T)
-6	Message not transmitted
-7	Message not responded to correctly
-8	Connection Number out of range
-9	VPI out of range
-10	VCI out of range
-11	Interface out of range

dagsar_vci_deactivate

uint32_t dagsar_vci_deactivate

```
uint32_t    dagfd
uint32_t    iface
uint32_t    channel
uint32_t    vpi
uint32_t    vci
```

Description

The **dagsar_vci_deactivate()** function sets a virtual connection to not return any data (all ATM cells will be discarded). This function will also deallocate any memory associated with the virtual connection. If this connection has previously been in the AAL2 reassembly mode then all channels (CID) will return to the unconfigured state.

Arguments

- dagfd** The file descriptor for the DAG card as returned from `dag_open()`. This card also should have been initialized via `dagama_open_conn()`.
- iface** Specifies the interface of the virtual connection to be deactivated. In the case of the DAG3.7T card, this should always be zero.
- channel** Specifies the channel of the virtual connection to be activated. This is the same as the connection number given in the Multi-channel ERF header. For concatenated network links this should always be zero.
- vpi** Specifies the virtual path identifier of the virtual connection to be deactivated.
- vci** Specifies the virtual channel identifier of the virtual connection to be deactivated.

Return Values

Value	Description
0	Function was successful.
!0	Function was unsuccessful

Value	Description
-1	Memory resources not available (only DAG3.7T)
-2	Virtual Connection is currently deactivated (only DAG3.7T)
-3	Timeout communicating with the XScale.
-4	Configuration could not be altered (only DAG3.7T)
-6	Message not transmitted
-7	Message not responded to correctly
-8	Connection Number out of range
-9	VPI out of range
-10	VCI out of range
-11	Interface out of range

dagsar_cid_activate

```
uint32_t dagsar_cid_activate
    uint32_t    dagfd
    uint32_t    iface
    uint32_t    channel
    uint32_t    vpi
    uint32_t    vci
    uint32_t    cid
```

Description

The **dagsar_cid_activate()** function sets an individual AAL2 channel identification (CID) to return data. Before this function will succeed the virtual connection that is described by the arguments channel, vpi and vci must be set to reassemble AAL2 frames. An individual CID may not be activated or deactivated for ATM or AAL5 connections. If the virtual connection described by the arguments channel, vpi and vci is not currently active it will be set to active by use of this function. This could result in other, currently unconfigured CIDs to also return data if there is data present on the line and CID and they have not been previously deactivated. Activating a CID will include allocating memory for use in the reassembly of AAL2-SSAR frames. The amount of memory allocated will be either the buffer size and an ERF overhead as previously set by **dagsar_set_buffer_size()** or 64kB. The total amount of memory available for reassembling is 128MiB. Attempting to open more connections when the total memory is used will not succeed.

Arguments

- dagfd** The file descriptor for the DAG card as returned from dag_open(). This card also should have been initialized via dagema_open_conn().
- iface** Specifies the interface of the CID to be activated. In the case of the DAG3.7T card, this should always be zero.
- channel** Specifies the channel of the CID to be activated. This is the same as the connection number given in the Multi-channel ERF header. For concatenated network links this should always be zero.
- vpi** Specifies the virtual path identifier of the CID to be activated.
- vci** Specifies the virtual channel identifier of the CID to be activated.
- cid** Specifies the channel identifier (CID) to be activated.

Return Codes

Code	Description
0	Function was successful.
!0	Function was unsuccessful

DAG3.7T Specific Return Codes

Code	Description
-1	Memory resources not available
-2	CID is currently activated
-3	Timeout communicating with the XScale.
-5	Unsupported sar_mode type (not AAL2)
-6	Message not transmitted
-7	Message not responded to correctly
-8	Connection Number out of range
-9	VPI out of range
-10	VCI out of range
-11	Interface out of range

dagsar_cid_deactivate

uint32_t dagsar_cid_deactivate

```

uint32_t    dagfd
uint32_t    iface
uint32_t    channel
uint32_t    vpi
uint32_t    vci
uint32_t    cid

```

Description

The **dagsar_cid_deactivate()** function sets an individual AAL2 channel identifier (CID) to not return any data. All data on the CID will be discarded. This will affect only the CID on the virtual connection specified by the arguments channel, vpi and vci. This will not cause any other CIDs or virtual connections to discard data.

Arguments

dagfd	The file descriptor for the DAG card as returned from dag_open(). This card also should have been initialized via dagema_open_conn().
iface	Specifies the interface of the virtual connection to be deactivated. In the case of the DAG3.7T card, this should always be zero.
channel	Specifies the channel of the virtual connection to be deactivated. This is the same as the connection number given in the Multi-channel ERF header. For concatenated network links this should always be zero.
vpi	Specifies the virtual path identifier of the virtual connection to be deactivated.
vci	Specifies the virtual channel identifier of the virtual connection to be deactivated.
cid	Specifies the channel identifier (CID) to be deactivated.

Return Values

Value	Description
0	Function was successful.
!0	Function was unsuccessful

DAG3.7T Specific Return Codes

Value	Description
-1	Memory resources not available
-2	Virtual Connection is currently deactivated
-3	Timeout communicating with the XScale.
-5	Unsupported sar_mode type (not AAL2)
-6	Message not transmitted
-7	Message not responded to correctly
-8	Connection Number out of range
-9	VPI out of range
-10	VCI out of range
-11	Interface out of range

dagsar_vci_set_sar_mode

```
uint32_t dagsar_vci_set_sar_mode
    uint32_t    dagfd
    uint32_t    iface
    uint32_t    channel
    uint32_t    vpi
    uint32_t    vci
    sar_mode_t  sar_mode
```

Description

The **dagsar_vci_set_sar_mode()** function allows the mode of operation to be changed on a virtual connection. The available modes are ATM, AAL2-SSSAR and AAL5 reassembly. To change the mode a virtual connection must be deactivated before the **dagsar_vci_set_sar_mode()** function is called.

Arguments

- dagfd** The file descriptor for the DAG card as returned from `dag_open()`. This card also should have been initialized via `dagemma_open_conn()`.
- iface** Specifies the interface of the virtual connection. In the case of the DAG3.7T card, this should always be zero.
- channel** Specifies the channel of the virtual connection. This is the same as the connection number given in the Multi-channel ERF header. For concatenated network links this should always be zero.
- vpi** Specifies the virtual path identifier of the virtual connection.
- vci** Specifies the virtual channel identifier of the virtual connection.
- sar_mode** Specifies the mode the virtual connection should be changed to. Possible values: {sar_aal0, sar_aal2, sar_aal5}

Of the modes **sar_aal0**, **sar_aal2** and **sar_aal5** are currently valid modes. **sar_error** is used to indicate a virtual connection is in an error mode.

Return Codes

Code	Description
0	Function was successful.
!0	Function was unsuccessful

Code	Description
-1	Memory resources not available (only DAG3.7T)
-2	Virtual Connection is currently activated (only DAG3.7T)
-3	Timeout communicating with the XScale.
-4	Configuration could not be altered (only DAG3.7T)
-6	Message not transmitted
-7	Message not responded to correctly
-8	Connection Number out of range
-9	VPI out of range
-10	VCI out of range
-11	Interface out of range

dagsar_vci_get_sar_mode

```

sar_mode_t dagsar_vci_get_sar_mode
    uint32_t  dagfd
    uint32_t  iface
    uint32_t  channel
    uint32_t  vpi
    uint32_t  vci

```

Description

The **dagsar_vci_get_sar_mode()** function returns the current mode of operation on a virtual connection.

Arguments

- dagfd** The file descriptor for the DAG card as returned from `dag_open()`. This card also should have been initialized via `dagama_open_conn()`.
- iface** Specifies the interface of the virtual connection. In the case of the DAG3.7T card, this should always be zero.
- channel** Specifies the channel of the virtual connection. This is the same as the connection number given in the Multi-channel ERF header. For concatenated network links this should always be zero.
- vpi** Specifies the virtual path identifier of the virtual connection.
- vci** Specifies the virtual channel identifier of the virtual connection.

Return Codes

Code	Description
Valid sar_mode	Function was successful. Possible values: {sar_aal0, sar_aal2, sar_aal5}
Invalid sar_mode	Function was unsuccessful

Code	Description
-3	Timeout communicating with the XScale.
sar_aal0	Virtual Connection is configured to return ATM cells as they are received.
sar_aal2	Virtual Connection will return AAL2 - SSSAR frames as they are reassembled.
sar_aal5	Virtual Connection will return AAL5 frames as they are reassembled.
sar_error	The Virtual Connection has returned an error state.
-6	Message not transmitted
-7	Message not responded to correctly
-8	Connection Number out of range
-9	VPI out of range
-10	VCI out of range
-11	Interface out of range

dagsar_channel_set_net_mode

```
uint32_t dagsar_vci_set_net_mode
    uint32_t    dagfd
    uint32_t    iface
    uint32_t    channel
    net_mode_t  net_mode
```

Description

The **dagsar_vci_set_net_mode()** sets the net mode of a virtual connection to user to network interface (UNI) or network to network interface (NNI).

Arguments

- dagfd** The file descriptor for the DAG card as returned from dag_open(). This card also should have been initialized via dagema_open_conn().
- iface** Specifies the interface of the virtual connection. In the case of the DAG3.7T card, this should always be zero.
- channel** Specifies the channel of the virtual connection. This is the same as the connection number given in the Multi-channel ERF header. For concatenated network links this should always be zero.
- net_mode** Specifies the net mode the virtual connection should be changed to. Possible values: {uni, nni}.

Return Codes

Code	Description
0	Function was successful
!0	Function was unsuccessful

Code	Description
-1	Memory resources not available. (only DAG3.7T)
-3	Timeout communicating with the XScale.
-4	Configuration could not be altered. (only DAG3.7T)
-6	Message not transmitted
-7	Message not responded to correctly
-8	Connection Number out of range
-9	VPI out of range
-10	VCI out of range
-11	Interface out of range

dagsar_set_buffer_size (only DAG3.7T)

```

sar_mode_t dagsar_set_buffer_size
    uint32_t    dagfd
    uint32_t    size

```

Description

The **dagsar_set_buffer_size()** function sets the largest expected size for AAL frames on connections or channels to be activated. This size does not include the extra size required for the ERF header, ATM header, AAL5 Trailer or Padding, when required. By changing this size, virtual connections that have been activated previously will not be altered to have the new buffer size. The change will only effect virtual connections or channels that are activated after the buffer size change. To update the buffer size of a previously activated virtual connection or channel, it is necessary to deactivate and re-activate the connection or channel after the buffer size has been altered. If a larger AAL frame is received than the size of the buffer, when the buffer is full it will return the first portion of the AAL frame with the length error bit set. The virtual connection or channel will then keep collecting the AAL frame until the current frame is finished, and return the remaining portion, also with the length error bit set.

- dagfd** The file descriptor for the DAG card as returned from dag_open(). This card also should have been initialized via dagema_open_conn().
- size** The size of the largest AAL frame expected on virtual connections or channels to be activated.

Return Codes

Code	Description
0	Function was successful
!0	Function was unsuccessful

Code	Description
-1	Buffer size is larger than allowable (64 kilobytes).
-2	Buffer size is less than one ATM cell.
-3	Timeout communicating with the XScale.
-6	Message not transmitted
-7	Message not responded to correctly

dagsar_get_stats

```
uint32_t dagsar_get_stats
    uint32_t    dagfd
    stats_t     statistic
```

Description

The **dagsar_get_stats()** function retrieves the internally held value that corresponds to the requested statistic. The possible statistics currently available for the DAG3.7T are defined by the enumeration `stats_t`.

Arguments

- dagfd** The file descriptor for the DAG card as returned from `dag_open()`. This card also should have been initialized via `dagama_open_conn()`.
- statistic** The statistic which should be returned
The currently available options for the statistic are:

DAG3.7T	
dropped_cells	This is the number of cells not returned to the host, either due to the cells arriving on a Virtual Connection that is deactivated; or arriving on unconfigured Virtual Connections while in the Scanning mode
filtered_cells	This is the number of cells not returned to the host due to a filter that has determined the cell should be rejected.

DAG7.1S	
dag71s_stats_rx_pkt_0	Number of cells received by the embedded processor before any filtering. Each one of the statistic counters belongs to an interface port.
dag71s_stats_rx_pkt_1	
dag71s_stats_rx_pkt_2	
dag71s_stats_rx_pkt_3	
dag71s_stats_filter_drop_0	Number of cells dropped by the bitmask filters applied to the ATM headers. Each one of the statistic counters belongs to an interface port.
dag71s_stats_filter_drop_1	
dag71s_stats_filter_drop_2	
dag71s_stats_filter_drop_3	
dag71s_stats_to_host_0	Number of cells or frames actually sent to the host. The counter doesn't differentiate between AAL0, AAL2 or AAL5 frames. Each one of the statistic counters belongs to an interface port.
dag71s_stats_to_host_1	
dag71s_stats_to_host_2	
dag71s_stats_to_host_3	
dag71s_stats_hash_collisions	Number of collisions in the hash table. Only for debugging purposes.
dag71s_stats_datapath_resets	Number of datapath resets. Only for debugging purposes.
dag71s_stats_cmds_exec	Number of user commands executed by the embedded processor. Only for debugging purposes.
dag71s_stats_crc_error_0	Number of CRC errors on AAL5 frames. Each one of the statistic counters belongs to an interface port.
dag71s_stats_crc_error_1	
dag71s_stats_crc_error_2	

DAG7.1S	
dag71s_stats_crc_error_3	
dag71s_stats_loss_counter	Number of packet losses reported by the firmware. This drop counter will increase when the embedded processor cannot process all the incoming packets. The number of packet losses is not accurate, but gives an idea of the performance of the embedded processor. If this statistic counter is greater than zero, it is recommended to decrease the input bandwidth to the embedded processor. This can be achieved through bitmask filters applied to the ATM headers.

Note for DAG7.1S statistic counters: Statistic counters are unsigned 32-bit integers. When the counter reaches the maximum value (0xFFFFFFFF) it will continue counting from the minimum value (0x00000000).

Return Codes

Code	Description
Any	Value of the statistic

Code	Description
-3	Timeout communicating with the XScale.
-6	Message not transmitted
-7	Message not responded to correctly

dagsar_get_interface_stats (deprecated)

```
uint32_t dagsar_get_interface_stats
    uint32_t    dagfd
    uint32_t    iface
    uint32_t    statistic
```

Description

The **dagsar_get_interface_stats()** function, when used on the DAG3.7T is functionally the same as the **dagsar_get_stats()** function, due to the DAG3.7T not having specified interfaces. This function retrieves the internally held value that corresponds to the requested statistic. This is the total number of cells which fit the criteria for the statistic since the last restart or reset. The possible statistics currently available for the DAG3.7T are defined by the enumeration `stats_t`.

The arguments to the function are described below:

- dagfd** The file descriptor for the DAG card as returned from `dag_open()`. This card also should have been initialized via `dagemma_open_conn()`.
- iface** Specifies the interface of the virtual connection to be activated. In the case of the DAG3.7T card, this should always be zero.
- statistic** The statistic which should be returned
The currently available options for the statistic are:
- dropped_cells** This is the number of cells not returned to the host, either due to the cells arriving on a Virtual Connection that is deactivated; or arriving on unconfigured Virtual Connections while in the Scanning mode.
- filtered_cells** This is the number of cells not returned to the host due to a filter that has determined the cell should be rejected.

Return Codes

Code	Description
Any	Value of the statistic

DAG3.7T Specific Return Codes

Code	Description
-3	Timeout communicating with the XScale.
-6	Message not transmitted
-7	Message not responded to correctly

dagsar_reset_stats

```
uint32_t dagsar_reset_stats
    uint32_t    dagfd
    stats_t     statistic
```

Description

The **dagsar_reset_stats()** function allows a single statistic to be reset to zero without affecting any other statistics, which will continue counting from their current position.

Arguments

- dagfd** The file descriptor for the DAG card as returned from dag_open(). This card also should have been initialized via dagema_open_conn().
- statistic** The statistic which should be returned

The currently available options for the statistic are:

DAG3.7T	
dropped_cells	This is the number of cells not returned to the host, either due to the cells arriving on a Virtual Connection that is deactivated; or arriving on unconfigured Virtual Connections while in the Scanning mode
filtered_cells	This is the number of cells not returned to the host due to a filter that has determined the cell should be rejected.

DAG7.1S	
dag71s_stats_rx_pkt_0	Number of cells received by the embedded processor before any filtering. Each one of the statistic counters belongs to an interface port.
dag71s_stats_rx_pkt_1	
dag71s_stats_rx_pkt_2	
dag71s_stats_rx_pkt_3	
dag71s_stats_filter_drop_0	Number of cells dropped by the bitmask filters applied to the ATM headers. Each one of the statistic counters belongs to an interface port.
dag71s_stats_filter_drop_1	
dag71s_stats_filter_drop_2	
dag71s_stats_filter_drop_3	
dag71s_stats_to_host_0	Number of cells or frames actually sent to the host. The counter doesn't differentiate between AAL0, AAL2 or AAL5 frames. Each one of the statistic counters belongs to an interface port.
dag71s_stats_to_host_1	
dag71s_stats_to_host_2	
dag71s_stats_to_host_3	
dag71s_stats_hash_collisions	Number of collisions in the hash table. Only for debugging purposes.
dag71s_stats_datapath_resets	Number of datapath resets. Only for debugging purposes.
dag71s_stats_cmds_exec	Number of user commands executed by the embedded processor. Only for debugging purposes.
dag71s_stats_crc_error_0	Number of CRC errors on AAL5 frames. Each one of the statistic counters belongs to an interface port.
dag71s_stats_crc_error_1	
dag71s_stats_crc_error_2	

DAG7.1S	
dag71s_stats_crc_error_3	
dag71s_stats_loss_counter	Number of packet losses reported by the firmware. This drop counter will increase when the embedded processor cannot process all the incoming packets. The number of packet losses is not accurate, but gives an idea of the performance of the embedded processor. If this statistic counter is greater than zero, it is recommended to decrease the input bandwidth to the embedded processor. This can be achieved through bitmask filters applied to the ATM headers.

Note for DAG7.1S statistic counters: Statistic counters are unsigned 32-bit integers. When the counter reaches the maximum value (0xFFFFFFFF) it will continue counting from the minimum value (0x00000000).

Return Codes

Code	Description
0	Function was successful
!0	Function was unsuccessful

Code	Description
-2	Statistic is not recognised. (only DAG3.7T)
-3	Timeout communicating with the XScale.
-6	Message not transmitted
-7	Message not responded to correctly

dagsar_reset_stats_all

```
uint32_t dagsar_reset_stats
        uint32_t    dagfd
```

Description

The **dagsar_reset_stats_all()** function will reset all statistics to zero. It is recommended to call this function at the start of a program that will be using the statistics to put all statistics into a known state.

Arguments

dagfd The file descriptor for the DAG card as returned from `dag_open()`. This card also should have been initialized via `dagemma_open_conn()`.

Return Codes

Code	Description
0	Function was successful
!0	Function was unsuccessful

Code	Description
-2	A Statistic was unable to be identified. (only DAG3.7T)
-3	Timeout communicating with the XScale.
-6	Message not transmitted
-7	Message not responded to correctly

dagsar_set_filter_bitmask

```
uint32_t dagsar_set_filter_bitmask
    uint32_t      dagfd
    uint32_t      iface
    uint32_t      bitmask
    uint32_t      match
    filter_action_t filter_action
```

Description

The **dagsar_set_filter_bitmask()** function sets the values of the bitmask, match value and action to be taken for a filter on the card. These values define the filter on the card. Any ATM cells received will have the 32 bits of the ATM header logically AND'd with the bitmask value supplied. The result of this calculation is then compared with the match value, if they are identical, the action defined by filter_action is then taken. The possible filter actions are:

- sar_accept** Any ATM cells which are the same as the match value after processing the bitmask should be accepted. This will involve passing them onto the Virtual Connections list to determine what reassembly action should be taken. Any ATM cells which are not the same as the match value after filter processing will be discarded immediately
- sar_reject** Any ATM cells which are the same as the match value after processing the bitmask should be rejected regardless of the Virtual Connection status of the ATM cell. Any ATM cells which are not the same as the match value after filter processing will be processed by the reassembler normally.

Arguments

- dagfd** The file descriptor for the DAG card as returned from dag_open(). This card also should have been initialized via dagemma_open_conn().
- iface** Specifies the interface where the bitmask filter is going to be applied. In the case of the DAG3.7T card, this should always be zero.
- bitmask** This is the 32 bit value used by the filter in the mask calculation. This involves performing a logical AND between this value and the ATM header (32 bits not including the HEC).
- match** The value which the calculation will need to match for the action specified by filter_action to occur. This argument is unused for the DAG3.7T card and should be left at the default.
- filter_action** The action to be taken when an ATM cell matches the filter value. The contrary to this action is then performed on those ATM cells which do not match the filter value after the bit mask has been applied.

Examples

```

Action: ACCEPT
ATM header: 11001011 11001000 11100111 00100010 0xCBC8E722
Bitmask:    11111111 00000000 00001111 00000000 0xFF000F00
-----
Logical AND: 11001011 00000000 00000111 00000000 0xCB000700
Match value: 11110111 00000000 00000111 00000000 0xF7000700
(does not match -> packet rejected)
Action: REJECT
ATM header: 11001011 11001000 11100111 00100010 0xCBC8E722
Bitmask:    11111111 00000000 00001111 00000000 0xFF000F00
-----
Logical AND: 11001011 00000000 00000111 00000000 0xCB000700
Match value: 11110111 00000000 00000111 00000000 0xF7000700
(does not match -> packet accepted)
Action: ACCEPT
ATM header: 11001011 11001000 11100111 00100010 0xCBC8E722
Bitmask:    11111111 00000000 00001111 00000000 0xFF000F00
-----
Logical AND: 11001011 00000000 00000111 00000000 0xCB000700
Match value: 11001011 00000000 00000111 00000000 0xCB000700
(match -> packet accepted)
Action: REJECT
ATM header: 11001011 11001000 11100111 00100010 0xCBC8E722
Bitmask:    11111111 00000000 00001111 00000000 0xFF000F00
-----
Logical AND: 11001011 00000000 00000111 00000000 0xCB000700
Match value: 11001011 00000000 00000111 00000000 0xCB000700
(match -> packet rejected)

```

Return Codes

Code	Description
0	Function was successful
!0	Function was unsuccessful

Code	Description
-1	Filter could not be set (only DAG3.7T)
-2	The filter was unable to be allocated. This can be due to being out of memory or attempting to add more than 32 filters. (only DAG3.7T)
-3	Timeout communicating with the XScale.
-6	Message not transmitted
-7	Message not responded to correctly

dagsar_reset_filter_bitmask

```
uint32_t dagsar_reset_filter_bitmask
    uint32_t    dagfd
    uint32_t    iface
```

Description

The **dagsar_reset_filter_bitmask()** function resets the filter on the board so that no further cells are rejected based on the previous filter values.

Arguments

- dagfd** The file descriptor for the DAG card as returned from dag_open(). This card also should have been initialized via dagema_open_conn().
- iface** Specifies the interface where the filter is going to be reset. In the case of the DAG3.7T card, this should always be zero.

Return Codes

Code	Description
0	Function was successful
!0	Function was unsuccessful

Code	Description
-1	There is no identifiable filter on the card. (only DAG3.7T)
-2	Filter could not be deleted. (only DAG3.7T)
-3	Timeout communicating with the XScale.
-6	Message not transmitted
-7	Message not responded to correctly

dagsar_init_scanning_mode

```
uint32_t dagsar_init_scanning_mode
        uint32_t    dagfd
```

Description

The **dagsar_init_scanning_mode()** function initializes the internal set of scanned entries, so that scanning can be performed correctly. This function will remove any current entries and set the number of scanned connections to zero.

This function must be called prior to setting the scanning mode to on. Also the scanning mode must be off for the structure to be initialized. On the DAG3.7T an error will result if this function is called while scanning mode is set to on. On the DAG7.1S no action will be performed on that case.

This function should also be called after scanning has been completed and scanned connections have been processed to free internal memory.

Arguments

dagfd The file descriptor for the DAG card as returned from `dag_open()`. This card also should have been initialized via `dagama_open_conn()`.

Return Codes

Code	Description
0	Function was successful
!0	Function was unsuccessful

Code	Description
-1	Scanning mode is currently on (only DAG3.7T)
-3	Timeout communicating with the XScale.
-6	Message not transmitted
-7	Message not responded to correctly

dagsar_set_scanning_mode

```
uint32_t dagsar_set_scanning_mode
uint32_t      dagfd
scanning_mode_t  scan_mode
```

Description

The **dagsar_set_scanning_mode()** function allows the scanning mode to be turned on and off. During the scanning mode the system will gather information on the available connections which have data available to them and are currently unconfigured or deactivated.

Before turning the scanning mode on the **dagsar_init_scanning_mode()** function must be called. When scanning mode is set to on, no other scanning related functions can be called. Scanning places an extra load on the system which can have a detrimental effect on the performance of the AAL reassembler. For this reason it is not recommended to use the scanning mode whilst in normal operation. `scan_error` is not a valid option for the card to be set to.

After scanning mode has been initialized, turned on and then turned off the scanned connections can be received using the **dagsar_get_scanned_connections_number()** and **dagsar_get_scanned_connection()** functions.

Arguments

dagfd The file descriptor for the DAG card as returned from `dag_open()`. This card also should have been initialized via `dagemma_open_conn()`.

scan_mode The scanning mode to be set.

Return Codes

Code	Description
0	Function was successful
!0	Function was unsuccessful

Code	Description
-1	Scanning operation could not be completed (only DAG3.7I)
-2	Scanning could not be stopped. (only DAG3.7I)
-3	Timeout communicating with the XScale.
-4	Scanning has not been initialized. (only DAG3.7I)
-5	Scanning could not be started. (only DAG3.7I)
-6	Message not transmitted
-7	Message not responded to correctly

dagsar_get_scanning_mode

```
scanning_mode_t dagsar_get_scanning_mode
    uint32_t     dagfd
```

Description

The **dagsar_get_scanning_mode()** function returns the current status of the scanning mode as set with the **dagsar_set_scanning_mode()** function.

Arguments

dagfd The file descriptor for the DAG card as returned from `dag_open()`. This card also should have been initialized via `dagemma_open_conn()`.

Return Codes

Code	Description
scan_on	Scanning mode is on
scan_off	Scanning mode is off
scan_error	Scanning mode is undefined
Invalid scanning mode	Function was unsuccessful

Code	Description
-3	Timeout communicating with the XScale.
-6	Message not transmitted
-7	Message not responded to correctly

dagsar_get_scanned_connections_number

```
uint32_t dagsar_get_scanned_connections_number
        uint32_t    dagfd
```

Description

The **dagsar_get_scanned_connections()** function returns the number of connections that the host has information about after scanning. These connections can then be accessed using the function **dagsar_get_scanned_connection()**.

Scanned connections are only available after scanning has been performed and prior to the **dagsar_init_scanning_mode()** function being called.

Arguments

dagfd The file descriptor for the DAG card as returned from **dag_open()**. This card also should have been initialized via **dagama_open_conn()**.

Return Codes

Code	Description
Any	The number of scanned connections available

Code	Description
-1	Scanning mode is set to on.

dagsar_get_scanned_connection

```
uint32_t dagsar_get_scanned_connection
    uint32_t          dagfd
    uint32_t          connection_number
    connection_info_t *pConnection_info
```

Description

The **dagsar_get_scanned_connection()** function allows information about a connection which was identified during scanning to be retrieved. Connection information is indexed from zero to one less than the value returned by the **dagsar_get_scanned_connections()** function. Connection information returned includes the connection number, VCI, VPI and the Interface number. These identifiers can then be used to configure the connection.

Scanned connections are only available after scanning has been performed and prior to the **dagsar_init_scanning_mode()** function being called.

Arguments

- dagfd** The file descriptor for the DAG card as returned from `dag_open()`. This card also should have been initialized via `dagama_open_conn()`.
- connection_number** The index of the connection information to be retrieved.
- pConnection_info** Pointer to the structure that will contain the connection information after a successful call to the function. This pointer must already be pointing to a valid `connection_info` structure, and any memory used must be maintained and deallocated by the user.

Return Codes

Code	Description
0	Function was successful
!0	Function was unsuccessful

Code	Description
-1	Scanning mode is set to on.
-2	connection_number requested is out of range.
-3	Error attempting to access table entry (only DAG3.7T)
-4	Connection info pointer was not initialized. (only DAG7.1S)

dagsar_set_trailer_strip_mode (only DAG7.1S)

```
uint32_t dagsar_set_trailer_strip_mode
uint32_t      dagfd
trailer_strip_mode_t  strip_mode
```

Description

The **dagsar_set_trailer_strip_mode()** function sets the current AAL5 trailer stripping mode, by default the trailer and padding of AAL5 frames are removed by the reassembler prior to being presented to the host. Using this function the behavior can be changed to the preserve the padding bytes and the trailer on the end of the reassembled frame.

DAG 3.7T

This function is not available for the DAG3.7T.

Arguments

- dagfd** The file descriptor for the DAG card as returned from `dag_open()`. This card also should have been initialized via `dagama_open_conn()`.
- strip_mode** The trailer strip mode to set, possible values are `trailer_on` or `trailer_off`. If `trailer_on` is specified then the AAL5 trailer and padding bytes remain on the reassembled frames. If `trailer_off` is specified then the padding and trailer bytes are stripped from the reassembled frame, this is the default.

Return Codes

Code	Description
0	Function was successful
!0	Function was unsuccessful

Code	Description
-3	Timeout communicating with the XScale.
-6	Message not transmitted
-7	Message not responded to correctly
-15	Communication timed-out when connecting to the packet processing engines.
-18	A generic error occurred inside the packet processing engines.

dagsar_get_trailer_strip_mode (only DAG7.1S)

```
trailer_strip_mode_t dagsar_get_trailer_strip_mode
    uint32_t          dagfd
```

Description

The `dagsar_get_trailer_strip_mode()` function retrieves the current AAL5 trailer strip mode.

DAG 3.7T

This function is not available for the DAG3.7T.

Arguments

dagfd The file descriptor for the DAG card as returned from `dag_open()`. This card also should have been initialized via `dagama_open_conn()`.

Return Codes

Code	Description
trailer_on	The trailer and padding bytes remain on the reassembled AAL5 frames.
trailer_off	The trailer and padding bytes are stripped from the reassembled AAL5 frames.
trailer_error	A generic error occurred while trying to retrieve the current AAL5 trailer strip mode.

Data Structures, Attributes, Defines and Enums

sar_mode_t

The `sar_mode_t` enumeration is defined in the `dagsarapi.h` file and is in the form

```
typedef enum {
    sar_aal0,
    sar_aal2,
    sar_aal5,
    sar_error
} sar_mode_t;
```

Of these modes `sar_aal0`, `sar_aal2` and `sar_aal5` are currently valid modes. `sar_error` is used to indicate a virtual connection is in an error mode.

net_mode_t

The `net_mode_t` enumeration is defined in the `dagsarapi.h` file and is in the form

```
typedef enum
{
    nni,
    uni
} net_mode_t;
```

These modes can be set using the `dagsar_set_net_mode()` function.

stats_t

The `stats_t` enumeration is defined in the `dagsarapi.h` file and is in the form

```
typedef enum
{
    dropped_cells,
    filtered_cells,
    dag71s_stats_rx_pkt_0 = 0x100,
    dag71s_stats_rx_pkt_1,
    dag71s_stats_rx_pkt_2,
    dag71s_stats_rx_pkt_3,
    dag71s_stats_filter_drop_0,
    dag71s_stats_filter_drop_1,
    dag71s_stats_filter_drop_2,
    dag71s_stats_filter_drop_3,
    dag71s_stats_to_host_0,
    dag71s_stats_to_host_1,
    dag71s_stats_to_host_2,
    dag71s_stats_to_host_3,
    dag71s_stats_hash_collisions,
    dag71s_stats_datapath_resets,
    dag71s_stats_cmds_exec,
    dag71s_stats_crc_error_0,
    dag71s_stats_crc_error_1,
    dag71s_stats_crc_error_2,
    dag71s_stats_crc_error_3,
    dag71s_stats_loss_counter
} stats_t;
```

This defines the currently available statistics that can be received from the AAL reassembler. A definition of what these statistics represent can be found in the `dagsar` function definition section with the information for the function `dagsar_get_stats()`.

filter_action_t

The filter_action_t enumeration is defined in the dagsarapi.h file and is in the form

```
typedef enum
{
    sar_accept,
    sar_reject
}filter_action_t;
```

This defines the action which can be taken with a cell which is identified as fitting the set filter requirements. These actions can be set using the **dagsar_set_filter_bitmask()** function.

filter_operations_t (only DAG3.7T)

The filter_operations_t structure is defined in the aal_config_msg.h file and is in the form

```
typedef enum
{
    DAG_EQ = 0, /* 0 equal */
    DAG_NEQ, /* 1 not equal */
    DAG_LE, /* 2 less than or equal */
    DAG_GE, /* 3 greater than or equal */
    DAG_LT, /* 4 less than */
    DAG_GT, /* 5 greater than */
    DAG_AND, /* 6 bitwise and */
    DAG_OR, /* 7 bitwise or */
    DAG_XOR, /* 8 bitwise exclusive-or */

    DAG_NUM_FILTER_OPERATIONS /* this has to be the last in list */
}connection_info_t;
```

This defines the actions which can be used in extended filtering to compare the masked data value with the match value. This is only for an operational filter, the History filter always compares the masked data value with the previous masked data value, and if they are equal, then action is taken.

dag_filter_level_t (only DAG3.7T)

The dag_filter_level_t structure is defined in the aal_config_msg.h file and is in the form

```
typedef enum
{
    DAG_FILTER_LEVEL_CHANNEL = 0,
    DAG_FILTER_LEVEL_LINE,
    DAG_FILTER_LEVEL_BOARD,

    DAG_NUM_FILTER_LEVELS /* this has to be the last in list */
}DAG_filter_level_t;
```

This defines the level on which a filter is to be set.

list_operations_t (only DAG3.7T)

The list_operations_t structure is defined in the aal_config_msg.h file and is in the form

```
typedef enum
{
    DAG_OR_LIST = 0,
    DAG_AND_LIST
}list_operations_t;
```

This allows subfilter lists to be added in a form where either all of the filters need to be true to pass (DAG_AND_LIST) or any of the filters need to be true to pass (DAG_OR_LIST).

scanning_mode_t

The `scanning_mode_t` enumeration is defined in the `dagsarapi.h` file and is in the form

```
typedef enum
{
    scan_on,
    scan_off,
    scan_error
}scanning_mode_t;
```

This defines the scanning modes which can be set and received using the **`dagsar_set_scanning_mode()`** and **`dagsar_get_scanning_mode()`** functions. Although `scan_error` is defined, it should not be used as a mode to be set.

connection_info_t

The `connection_info_t` structure is defined in the `dagsarapi.h` file and is in the form

```
typedef struct
{
    uint32_t iface;
    uint32_t channel;
    uint32_t vpi;
    uint32_t vci;
}connection_info_t;
```

This structure defines the information which will be returned from a call to the **`dagsar_get_scanned_connection()`** function. On the DAG3.7T card the interface is not a valid identifier. This connection information can then be used to configure or activate a channel. The channel member of the structure is the same as the connection number given in the Multi-channel ERF header.

trailer_strip_mode_t

The `trailer_strip_mode_t` enumeration is defined in the `dagsarapi.h` file and is in the form

```
typedef enum
{
    trailer_on    = 0,
    trailer_off   = 1,
    trailer_error = 2
}trailer_strip_mode_t;
```

These modes can be set using the **`dagsar_set_trailer_strip_mode()`** function

Introduction

Endace DAG cards produce trace files in their own native format, known as the Endace “Extensible Record Format” (ERF). The ERF type used depends on the type of connection you are using to capture the data.

The SAR API supports the following ERF types:

ERF type	Description
3	TYPE_ATM - ATM cell
4	TYPE_AAL5 - reassembled AAL5 frame
7	TYPE_MC_ATM - Multi-channel ATM cell
9	TYPE_MC_AAL5 - Multi-channel AAL5
12	TYPE_MC_AAL2 - Multi-channel AAL2 frame
18	TYPE_AAL2 - Reassembled AAL2 Frame Record

For further details about these ERF formats, see *EDM011-01 Endace Extensible Record Format*, which is available from the Support section of the Endace website at www/endace.com/support.

