endace
accelerated

Universal Counter API

EDM04-25

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.

Published by:

Endace Technology® Ltd
Level 9
85 Alexandra Street

PO Box 19246
Hamilton 3244
New Zealand

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

International Locations

New Zealand
Endace Technology Ltd
Building 7, Lambie Drive
PO Box 76802
Manukau City 2104
New Zealand
Phone: +64 9 262 7260
Fax: +64 9 262 7261

Americas
Endace Network Systems Inc
14425 Penrose Place
Suite 225
Chantilly, VA 20151
United States of America
Phone: +1 703 964 3740
Fax: +1 703 378 0602

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 2008 Endace Technology Ltd. All rights reserved.

Contents

1. Purpose

This document describes the implementation of the universal statistics and counters interface (CSI). This interface facilitates the reading of the various counters and statistic registers of any type of DAG card. The counters and registers characteristics are printed using command from dagconfig.

Please be aware that this document is subject to change as additional functionality becomes available.

2. Overview

The interface covers the DRB space and is implemented by the firmware. The software implementation considers this interface as a firmware component/module.

CSI(counter statistics interface) enables the automatic identification by software of the presence of counters or informational bits from a predefined set. Each counter and information bit in this predefined set will have a set functionality. This will allow different combinations of counters and functional bits to be used in different cards without the requirement of additional software.

Each card can have multiple CSI blocks, thus allowing them to be placed in firmware modules when required and be easily added or removed depending on the requirements of the image. It also allows different accessing methods to be used depending on the counters.

Goals:

- To make an easy translation of the existing firmware module statistics to unified counter blocks interface. Which can be automatically recognized by the software
- To give both the firmware and software unique counter and deterministic Id's to prevent name changes and duplications across different cards and images.
- Extend the possibility for customer specific statistics with out software changes
- Extend the possibility for debugging statistics and counters with out need of special software support.
- Give the option for accumulated counters in near future and to backwards compatible with the software

Function definitions are described in later chapters of this document.

The main enumeration table will store entry(ies) of type CSI per 'counter statistics interface' which will point to a CSI block in the DRB address space. The enumeration entry will have different versions depending on the CSI format (type of data access: direct v0, or indirect v1).

3.1 Component

Each CSI block has a description field indicating:

- the number of counters,
- the type of the counters in the block (firmware module based or functional based),
- a 'latch and clear' set up.

Global latch and clear can be implemented at later stage through a single write-only DRB register instantiated in each CSI block.

This CSI block is implemented as a new component in dagconfig with attributes.

Its attributes are:

- Counter Statistics Interface type (kUint32AttributeCSIType),
- Number of counters in CSI (kUint32AttributeNbCounters),
- Latch & Clear set up (kBooleanAttributeLatchClear),
- Counter description base address (kUint32AttributeCounterDescBaseAdd),
- Counter value base address (kUint32AttributeCounterValueBaseAdd).

The post-initialization function creates the subcomponent(s) counter and initializes their state structure.

3.2 Subcomponent

Each counter is considered as a sub component of the CSI block component.

Each individual counter will have a 32-bit description entry containing:

- The counter ID which is unique and depending on the function is implemented when applicable,
- A sub-function ID which covers multiple streams, filters or interfaces ports,
- A “block value” type (counter value or address), this value type determines whether it is the counter value (0) or the address (1) where the counter value is stored,
- A 'Latch and Clear' information,
- The size of the counter can be 32 bits or 64 bits,
- type of access Indirect or Direct,
- Base address of the counter value.

The counter is implemented as a new sub-component of the CSI block component with these attributes:

- Counter ID (kUint32AttributeCounterID),
- Sub-function (kUint32AttributeSubFunction),
- Value type (kBooleanAttributeValueType),
- Latch and Clear information (kBooleanAttributeLatchClear),
- Counter size (kUint32AttributeCounterSize),
- Type of access (kBooleanAttributeAccess),
- Counter value (kUint32AttributeCounterValue),
- Sub-function (KUnit32AttributeSubFunctionsType)

The state structure contains:

- the index of the subcomponent,
- the address or offset (from the DRB base) of the description field,
- the address or offset (from the DRB base) of the counter value.

```
typedef struct
{
   uint32_t mIndex;
   uint32_t mValueOffset;
   uint32_t mDescrOffset;
   uint32_t* mValueAddress;
   uint32_t* mDescrAddress;

   } counter_state_t;
```

A specific function is implemented to read and set up each attribute.

Command:

The command used to print the counters is : `dagconfig -u`

4. Function Definitions

4.1 Component implementation

The interface is implemented as a component called “counter_interface” in file counter_interface_component.c (/lib/libdagconf/components/)

The usual function for a component has been implemented:

<component name>_get_new_component,

<component name>_post_initialize,

<component name>_reset,

<component name>_default,

<component name>_dispose,

<component name>_update_register_base.

4.2 Printing of counters and statistic registers

In order to print the various counters and statistics, a new functions has been created “print_univ_counters”in file counter_printing.c (tools/dagconfig/)

Others files modified to implement the component:

```
/tools/dagconfig/process_cmdline.c
/tools/dagconfig/process_cmdline.h
/tools/dagconfig/dagconfig.c
/lib/libdagconf/cards/dagx_impl.c
```

4.3 Structures

Structure of one counter, dag_counter_value_t:

- dag_counter_type_t typeID: indicates the type ID,
- int size : indicates the size of counter (32 or 64 bits),
- dag_subfct_type_t subfct: indicates the type of sub-function,
- int lc: indicates if there is a latch and clear bit to read the register,
- int value_type: indicates whether it is the counter value (0) or the address where the counter value is stored (1),
- uint64_t value: indicates the counter value,

```
typedef struct
{
   dag_counter_type_t typeID;  (see below.)
   int size;
   dag_subfct_type_t subfct;   (see below.)
   uint32_t interface_number
   int lc;
   int value_type; /* Only available for direct register */
   uint64_t value;
} dag_counter_value_t;

typedef enum
{
   kIDSubfctPort = 0x00,
   kIDSubfctStream = 0x01,
   kIDSubfctFilter = 0x02,
   kIDSubfctGeneral = 0x03,

} dag_subfct_type_t;


typedef enum
{
   kIDCounterInvalid = 0x0,
   kIDCounterRXFrame = 0x01,
   kIDCounterRXByte= 0x02,
   kIDCounterRXShort = 0x03,
   kIDCounterRXLong = 0x04,
   kIDCounterRXError = 0x05,
   kIDCounterRXFCS = 0x06,
   kIDCounterRXAbort = 0x07,
   kIDCounterTXFrame = 0x08,
   kIDCounterTXByte = 0x09,
   kIDCounterDIP4Error = 0x0A,
   kIDCounterDIP4PlError = 0x0B,
   kIDCounterBurstError = 0x0C,
   kIDCounterPlError = 0x0D,
   kIDCounterDebug = 0x0E,
   kIDCounterFilter = 0x0F,
   kIDCounterB1Error = 0x10,
   kIDCounterB2Error = 0x11,
   kIDCounterB3Error = 0x12,
   kIDCounterRXErr = 0x13,
   kIDCounterSpaceError = 0x14,
```



```
    kIDCounterContWdError = 0x15,
    kIDCounterPlContError = 0x16,
    kIDCounterTRDip4Error = 0x17,
    kIDCounterResvWd = 0x18,
    kIDCounterAddrError = 0x19,
    kIDCounterOOFPeriod = 0x1A,
    kIDCounterNbOOF = 0x1B,
    kIDCounterTXOOFPeriod = 0x1C,
    kIDCounterTXNbOOF = 0x1D,
    kIDCounterTXError = 0x1E,
    kIDCounterStatFrError = 0x1F,
    kIDCounterDip2Error = 0x20,
    kIDCounterPatternError = 0x21,
    kIDCounterRXStreamPacket = 0x22,
    kIDCounterRXStreamByte = 0x23,
    kIDCounterTXStreamPacket = 0x24,
    kIDCounterTXStreamByte = 0x25,
    kIDCounterPortDrop = 0x26,
    kIDCounterStreamDrop = 0x27,
    kIDCounterSubStreamDrop = 0x28,
    kIDCounterFilterDrop = 0x29,
} dag_counter_type_t;
```

4.3 Functions

4.3.1 dag_config_get_number_block function

Purpose	Return the number of block(s) of this card
Declared In	dag_config.h
Prototype	uint32_t dag_config_get_number_block(dag_card_ref_t card_ref)
Parameters	→ card_ref Reference of the DAG card
Returns	Number of block (counter statistic interface) of the DAG card.
Comments	

4.3.2 dag_config_get_number_counters function

Purpose	Return the number of counter(s) for a particular block
Declared In	dag_config.h
Prototype	uint32_t dag_config_get_number_counters(dag_card_ref_t card_ref, dag_block_type_t block_type)
Parameters	→ card_ref Reference of the DAG card → block_type Type of csi block
Returns	Number of statistic counter(s) for the block “block type”
Comments	

4.3.3 dag_config_get_number_all_counters function

Purpose	Return the total number of counter(s) of this card
Declared In	dag_config.h
Prototype	uint32_t dag_config_get_number_counters(dag_card_ref_t card_ref)
Parameters	→ card_ref Reference of the DAG card
Returns	Number of statistic counter(s) of the DAG card.
Comments	

4.3.4 dag_config_get_counter_id_subfct function

Purpose	Return the id and the sub-function of counters in a specific block.
Declared In	dag_config.h
Prototype	uint32_t dag_config_get_counter_id_subfct(dag_card_ref_t card_ref, dag_block_type_t block_type, dag_counter_value_t counter_id[], uint32_t size)
Parameters	→ card_ref Reference of the DAG card → block_type Type of csi block → counter_id[] returned array of dag_counter_value_t structure. → size Size of counter_id array.
Returns	Number of statistic counter(s) found for the block "block type".
Comments	

4.3.5 dag_config_get_all_block_id function

Purpose	Return all block ids.
Declared In	dag_config.h
Prototype	uint32_t dag_config_get_all_block_id(dag_card_ref_t card_ref, uint32_t block_id[], uint32_t size)
Parameters	→ card_ref Reference of the DAG card → block_type Type of csi block → block_id[] returned array of uint32_t. Contains all block ids. → size Size of block_id array.
Returns	Number of statistic counter(s) found for the block "block type".
Comments	

4.3.6 dag_config_latch_clear_all function

Purpose	Latch and clear all the csi blocks.
Declared In	dag_config.h
Prototype	void dag_config_latch_clear_all(dag_card_ref_t card_ref)
Parameters	→ card_ref Reference of the DAG card
Returns	N/A
Comments	This function is called by dag_config_read_all_counters, dag_config_read_counter and print_univ_counters (counter_printing.c).

4.3.7 dag_config_latch_clear_block function

Purpose	Latch and clear a specific csi block.
Declared In	dag_config.h
Prototype	void dag_config_latch_clear_all(dag_card_ref_t card_ref, dag_block_type_t block_type)
Parameters	→ card_ref Reference of the DAG card → block_type Type of csi block
Returns	N/A
Comments	

4.3.8 dag_config_read_single_block function

Purpose	Return the value of all counters in a specific csi block.
Declared In	dag_config.h
Prototype	uint32_t dag_config_read_single_block(dag_card_ref_t card_ref, dag_block_type_t block_type, dag_counter_value_t countersTab[], uint32_t size, int lc)
Parameters	→ card_ref Reference of the DAG card → block_type Type of csi block → countersTab Table of counter's structures → int size Size of countersTab → int lc Latch and clear option (0 = no latch and clear, 1 = latch and clear the block before reading the values)
Returns	Return the number of counters in the specific csi block
Comments	

4.3.9 dag_config_read_all_counters function

Purpose	Read all counters of the card and stock their parameters in a table
Declared In	dag_config.h
Prototype	uint32_t dag_config_read_all_counters(dag_card_ref_t card_ref, dag_counter_value_t countersTab[], uint32_t size, int lc)
Parameters	→ card_ref Reference of the DAG card → countersTab Table of counter's structures → int size Size of countersTab → int lc Latch and clear option (0 = no latch and clear, 1 = latch and clear the block before reading the values)
Returns	Return the number of counters
Comments	

4.3.10 dag_config_read_single_counter function

Purpose	Get the value of single counters on the card
Declared In	dag_config.c
Prototype	uint64_t dag_config_read_single_counter(dag_card_ref_t card_ref, dag_block_type_t block_type, dag_counter_type_t counter_type, dag_subfct_type_t subfct_type)
Parameters	→ card_ref Reference of the DAG card → block_type Type of csi block → counter_type Type ID of counter → subfct_type Type of sub-function
Returns	Return the value of a specific counter
Comments	

Version History

Revision	Data of Change	Description of Change	Revision Originator
1.0	4-Jan-07	Initial revision	Patricia LERUS
2.0	11-July-08	Updated Subfunction Type.	Karthik Sharma.
2.1	18 July 2008	Updated to Endace template	Sarah Stubbs