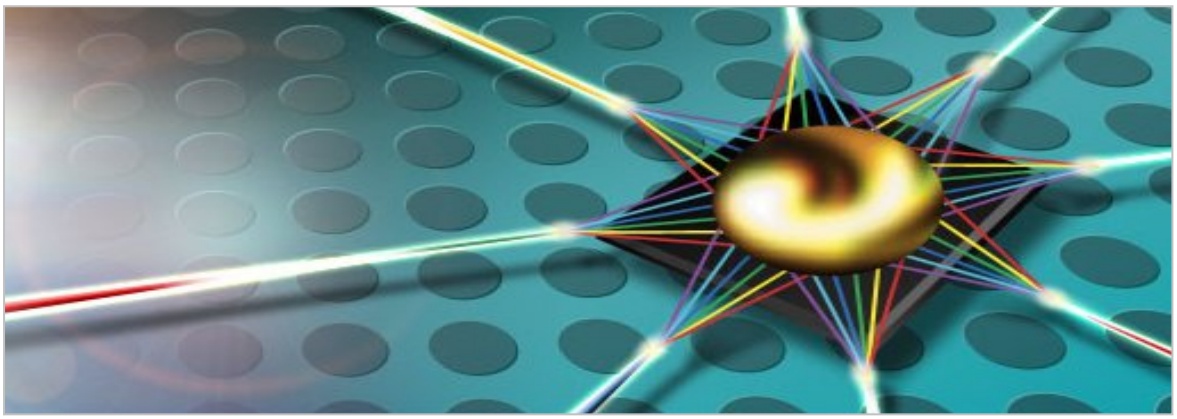
NPS-400

EZdp Reference Manual

Data Plane Environment Library for NPS-400 Network Processors

Document Version 1.9



Document Number: 27-8153-10

The information contained is proprietary and confidential.

EZchip Semiconductor Ltd.
1 Hatamar Street, P.O. Box 527
Yokneam 20692, Israel

EZchip Semiconductor Inc.
2700 Zanker Road, Suite 150
San Jose, CA 95134, USA

Preface

EZchip Semiconductor Inc., 2700 Zanker Road, Suite 150, San Jose, CA 95134, USA, Tel: (408) 520-3700, Fax: (408) 520-3701

EZchip Semiconductor Ltd., 1 Hatamar Street, PO Box 527, Yokneam 20692, Israel, Tel: +972-4-959-6666, Fax: +972-4-959-4166

Email: info@ezchip.com, Web: http://www.ezchip.com

EZchip welcomes your comments on this publication. Please address them to: supportNP@ezchip.com.

About this Manual

This document describes the EZchip Data Plane Services library (EZdp) and its related APIs. The EZdp library provides an application programming interface (API) for data-plane applications running on NPS network processors, abstracting the complexities of the underlying CTOP core instruction set and various hardware accelerators.

This manual is intended for software developers who plan to develop data-plane applications for products using the EZchip NPS-400 network processor. To use this manual, you should be familiar with the network processor architecture.

The *EZdk Release Notes* may contain information supplemental to this document.

Related Documents

For additional information refer to:

DOCUMENT	CONTENT
NPS-400 Architectural Specifications	Overview of the architecture, feature set and functionality of the NPS-400 network processor.
NPS-400 Programming Manual	Overview of the software programming model and concepts for the EZchip NPS-400 network processor.

This Document

The following is a brief description of the contents of each section:

CHAPTER	NAME	DESCRIPTION
Section 1	**Overview**	Provides overview of the data-plane services library (EZdp).
Section 2	**Directory Structure**	Describes the directory structure of EZdp.
Section 3	**API Organization**	Lists the groups of data-plane API routines.
Section 4	**API Overview**	Provides an overview of each of the API routines/commands in the groups.
Section 5	**Reference**	Lists each API routine in the group followed by its structures and enumerations in alphabetical order.

Revision History

REVISION	DATE	DESCRIPTION OF MODIFICATION
1.9	Sept. 6, 2015	Updated throughout. Refers to EZdk version 1.9a. **ezdp_atomic.h** ▪ Removed ezdp_atomic_xchg8_ext_addr and ezdp_atomic_xchg16_ext_addr. ▪ Added ezdp_get_mem_section_info and ezdp_mem_section_info_str. **ezdp_job.h** ▪ Added ezdp_valid_tm_queue_depth_handle. **ezdp_pci.h** ▪ ezdp_set_pci_msgq_read_index replaced ezdp_update_pci_msgq_read_index and ezdp_update_pci_msgq_read_index_async. **ezdp_processor.h** ▪ ezdp_get_hw_thread_id renamed ezdp_get_thread_id. ▪ ezdp_get_hw_core_id renamed ezdp_get_core_id. ▪ ezdp_get_hw_cluster_id renamed ezdp_get_cluster_id. **ezdp_search_prm.h** ▪ Added ezdp_prm_lookup_alg_tcam. **ezdp_security.h** ▪ Added ezdp_security_block_size.
1.8a	July 6, 2015	Refers to EZdk version 1.8a release. ezdp.h ▪ Section 5.2.1.4 Function Documentation was missing from the document. ezdp_math.h ▪ ezdp_add_checksum, ezdp_sub_checksum: Note corrected: The checksum calculation assumes an even (2 byte aligned) offset. ezdp_memory.h ▪ ezdp_calc_checksum_ext_addr, ezdp_calc_checksum: Note corrected: The checksum calculation assumes an even (2 byte aligned) offset. If the pointer is odd, the checksum result should be swapped. ezdp_search.h ▪ ezdp_modify_hash_entry, ezdp_scan_hash_slot: Hash is full indication removed. ▪ ezdp_lookup_alg_tcam: Input parameter added: *priority_ptr* - pointer to returned priority. ezdp_search_prm.h ▪ ezdp_prm_lookup_table_entry: *table_base_addr* replaced *prm_lookup_desc* input parameter. ▪ ezdp_prm_lookup_hash_entry, ezdp_prm_locate_hash_entry: *hash_base_addr* replaced *prm_lookup_desc* input parameter. ▪ ezdp_prm_add_hash_entry: bool replaced with uint32_t. Return is 0 (success), ENOMEM (hash is full). Use ezdp_get_err_msg() API to get the detailed error message of the failure. ▪ ezdp_prm_modify_hash_entry: bool replaced with void. No return. ▪ ezdp_prm_lookup_ultra_ip_entry: *uip_base_addr* replaced *prm_lookup_desc* input parameter.
1.8	Mar. 29, 2015	Updated throughout. Refers to EZdk version 1.8a beta.
1.7	Nov. 9, 2014	Updated throughout. Refers to EZdk version 1.7a.
1.6	July 22, 2014	Updated throughout. Refers to EZdk version 1.6a.
1.5	Mar. 3, 2014	Updated throughout. Refers to EZdk version 1.5a.
1.4	Oct. 31, 2013	Updated throughout. Refers to EZdk version 1.4a.
1.3	July 10, 2013	Updated throughout. Refers to EZdk version 1.3a.
1.2	Apr. 18, 2013	Updated throughout. Refers to EZdk version 1.2a.
1.1	Jan. 17, 2013	Initial release. Refers to EZdk version 1.1a.

Contents

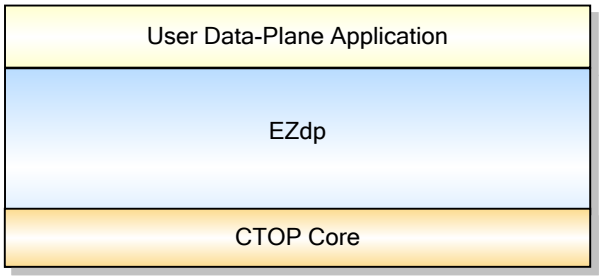
1. Overview

The EZchip data-plane services library (EZdp) provides an application programming interface (API) for data-plane applications running on NPS network processors, abstracting the complexities of the underlying CTOP core instruction set and various hardware accelerators.

Using the API routines outlined in this document, programmers can write C (ANSI) code for data-plane applications, including tasks such as packet processing, modification and forwarding; network protocol decoding; classification; etc.

The data-plane APIs are provided as a static library (EZdp) which is linked with the user's data-plane application. The data-plane APIs are performed in the context of the calling user-application tasks.

Figure 1. DPE overview



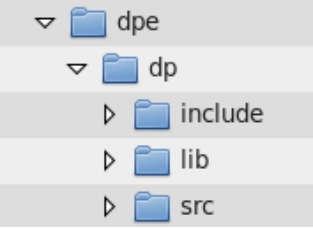
The data-plane services library provides APIs for utilizing the NPS CTOP core optimized instruction set. These provide convenient C-based wrappers for invoking the optimized instructions from C-based application code.

In addition, the data-plane services library provides both synchronous and asynchronous APIs for invoking various hardware accelerations external to the CTOP core. Synchronous commands internally perform implicit hardware scheduling, allowing the CTOP to process another hardware thread while the original request is being serviced. This provides a simple and convenient programming model, while still being efficient. Asynchronous commands are also provided for more advanced scenarios, allowing fine-grained control of the hardware scheduling. As a convention, the asynchronous API routine names end with the suffix "_async".

📖 For more information on the software programming model and concepts for the EZchip NPS-400 network processor, see the *NPS-400 Programming Manual*.

2. Directory Structure

The figure and table below show the directory structure after installing the EZdp library.



FOLDER	CONTENTS
/EZdk	
/dpe	Data plane environment libraries
/dp	Data-plane application services library
/include	Include files
/lib	Binary files
/src	Source files

3. API Organization

The data-plane APIs are arranged in groups of routines with common functionality, with each group is defined in a separate header file.

Following is a list of the main header files and their functionality:

Files	Description
ezdp.h	EZdp library initialization and version information.
ezdp_atomic.h	API routines for performing atomic operations.
ezdp_counter.h	API routines for counter operations.
ezdp_decode.h	API routines for performing decoding of frame data for standard protocol headers.
ezdp_dma.h	API routines for performing DMA operations.
ezdp_frame.h	API routines for operating with frame buffers.
ezdp_job.h	API routines for operating with jobs.
ezdp_lock.h	API routines for lock operations.
ezdp_math.h	API routines for advanced ALU operations, including arithmetic operations, logical operations, and bit-manipulation operations.
ezdp_memory.h	API routines for operating with memory addresses.
ezdp_pci.h	API routines for operating with the PCI Express interface.
ezdp_pool.h	API routines for operating with pools (user-defined index pool, memory pool).
ezdp_processor.h	API routines for controlling the CTOP processors.
ezdp_queue.h	API routines for queue operations.
ezdp_search.h	API routines for operating on search structures.
ezdp_search_prm.h	API routines for advanced low-level operations on search structures.
ezdp_security.h	API routines for operating security accelerators.
ezdp_string.h	API routines for manipulating character arrays.
ezdp_time.h	API routines for retrieving network and system time.

4. API Overview

4.1 EZdp Library Management (ezdp.h)

The EZdp API provides API routines for initialization and running of the data-plane application.

API Routine	Async	Description
ezdp_sync_cp	--	Sync until CP configuration is completed.
ezdp_init_global	--	Initialize the data-plane application; should be called once per data-plane executable.
ezdp_init_local	--	Initialize the data-plane process; should be called once per data-plane process (e.g. after each fork).
ezdp_run	--	Run a data plane application.

The **Async** column indicates which of the synchronous commands also have matching asynchronous commands. As a convention, the asynchronous API routine names end with the suffix "_async".

In addition, the EZdp API provides the following general services.

API Routine	Async	Description
ezdp_get_version	--	Return the version information of the ezdp library.
ezdp_get_err_msg	--	Return specific error message.
ezdp_get_mem_section_info	--	Return sizes of the memory sections.
ezdp_mem_section_info_str	--	Return printable string of the memory sizes.

4.2 Atomic Operations (ezdp_atomic.h)

EZdp provides API routines for performing atomic operations. These operations function on either extended addresses or summarized addresses.

📖 For more information on extended and summarized addresses, see the *NPS-400 Programming Manual*.

API Routine	Async	Description
ezdp_atomic_read8_ext_addr	--	Atomically read an 8-bit value from an extended address.
ezdp_atomic_read16_ext_addr	--	Atomically read a16-bit value from an extended address.
ezdp_atomic_read32_ext_addr	--	Atomically read a 32-bit value from an extended address.
ezdp_atomic_read32_sum_addr	+	Atomically read a 32-bit value from a summarized address.
ezdp_atomic_read64_sum_addr	+	Atomically read a 64-bit value from a summarized address.
ezdp_atomic_write8_ext_addr	+	Atomically write an 8-bit value to an extended address.
ezdp_atomic_write16_ext_addr	+	Atomically write a 16-bit value to an extended address.
ezdp_atomic_write32_ext_addr	+	Atomically write a 32-bit value to an extended address.
ezdp_atomic_write32_sum_addr	+	Atomically write a 32-bit value to a summarized address.
ezdp_atomic_write64_sum_addr	+	Atomically write a 64-bit value to a summarized address.
ezdp_atomic_xchg32_ext_addr	--	Atomically exchange a 32-bit value in an extended address.
ezdp_atomic_xchg32_sum_addr	--	Atomically exchange a 32-bit value in a summarized address.
ezdp_atomic_cmpxchg8_ext_addr	--	Atomically compare and exchange an 8-bit value in an extended address.
ezdp_atomic_cmpxchg16_ext_addr	--	Atomically compare and exchange a16-bit value in an extended address.
ezdp_atomic_cmpxchg32_ext_addr	--	Atomically compare and exchange a 32-bit value in an extended address.
ezdp_atomic_cmpxchg32_sum_addr	--	Atomic test and set a 32-bit value in a summarized address.
ezdp_atomic_read_and_tst8_ext_addr	--	Atomic read, test and set an 8-bit value in an extended address.
ezdp_atomic_read_and_tst16_ext_addr	--	Atomic read, test and set a16-bit value in an extended address.
ezdp_atomic_read_and_tst32_ext_addr	--	Atomic read, test and set a 32-bit value in an extended address.
ezdp_atomic_read_and_tst32_sum_addr	--	Atomic read, test and set a 32-bit value in a summarized address.
ezdp_atomic_read_and_clear8_ext_addr	--	Atomically read and clear an 8-bit value in an extended address.

API Routine	Async	Description
ezdp_atomic_read_and_clear16_ext_addr	--	Atomically read and clear a 16-bit value in an extended address.
ezdp_atomic_read_and_clear32_ext_addr	--	Atomically read and clear a 32-bit value in an extended address.
ezdp_atomic_read_and_clear32_sum_addr	--	Atomically read and clear a 32-bit value in a summarized address.
ezdp_atomic_read_and_clear64_sum_addr	--	Atomically read and clear a 64-bit value in a summarized address.
ezdp_atomic_add8_ext_addr	+	Atomically perform an 8-bit logical ADD operation on an extended address.
ezdp_atomic_add16_ext_addr	+	Atomically perform a 16-bit logical ADD operation on an extended address.
ezdp_atomic_add32_ext_addr	+	Atomically perform a 32-bit logical ADD operation on an extended address.
ezdp_atomic_read_and_add8_ext_addr	--	Atomically read and perform an 8-bit logical ADD operation on an extended address.
ezdp_atomic_read_and_add16_ext_addr	--	Atomically read and perform a 16-bit logical ADD operation on an extended address.
ezdp_atomic_read_and_add32_ext_addr	--	Atomically read and perform a 32-bit logical ADD operation on an extended address.
ezdp_atomic_add32_sum_addr	+	Atomically perform a 32-bit logical ADD operation on a summarized address.
ezdp_atomic_add64_sum_addr	+	Atomically perform a 64-bit logical ADD operation on a summarized address.
ezdp_atomic_read_and_add8_sum_addr	--	Atomically read and perform an 8-bit logical ADD operation on a summarized address.
ezdp_atomic_read_and_add16_sum_addr	--	Atomically read and perform a 16-bit logical ADD operation on a summarized address.
ezdp_atomic_read_and_add32_sum_addr	--	Atomically read and perform a 32-bit logical ADD operation on a summarized address.
ezdp_atomic_read_and_add64_sum_addr	--	Atomically read and perform a 64-bit logical ADD operation on a summarized address.
ezdp_atomic_dual_add32_ext_addr	+	Atomically perform a dual ADD operation to two 32-bit variables pointed to by the extended address.
ezdp_atomic_dual_add32_sum_addr	+	Atomically perform a dual ADD operation to two 32-bit variables pointed to by the summarized address.
ezdp_atomic_dual_add64_sum_addr	+	Atomically perform a dual ADD operation to two 64-bit variables pointed to by the summarized address.
ezdp_atomic_read_and_dual_add32_sum_addr	--	Atomically read and perform a dual ADD operation to two 32-bit variables pointed to by the summarized address.
ezdp_atomic_read_and_dual_add64_sum_addr	--	Atomically read and perform a dual ADD operation to two 64-bit variables pointed to by the summarized address.
ezdp_atomic_read_and_inc8_ext_addr	--	Atomically read and increment an 8-bit value in an extended address.
ezdp_atomic_read_and_inc16_ext_addr	--	Atomically read and increment a 16-bit value in an extended address.
ezdp_atomic_read_and_inc32_ext_addr	--	Atomically read and increment a 32-bit value in an extended address.

API Routine	Async	Description
ezdp_atomic_read_and_inc32_sum_addr	--	Atomically read and increment a 32-bit value in a summarized address.
ezdp_atomic_read_and_inc64_sum_addr	--	Atomically read and increment a 64-bit value in a summarized address.
ezdp_atomic_read_and_dec8_ext_addr	--	Atomically read and decrement conditionally by one 8-bit value in an extended address (zero value does not underflow).
ezdp_atomic_read_and_dec16_ext_addr	--	Atomically read and decrement conditionally by one 16-bit value in an extended address (zero value does not underflow).
ezdp_atomic_read_and_dec32_ext_addr	--	Atomically read and decrement conditionally by one 32-bit value in an extended address (zero value does not underflow).
ezdp_atomic_read_and_dec32_sum_addr	--	Atomically read and decrement conditionally by one 32-bit value in a summarized address (zero value does not underflow).
ezdp_atomic_read_and_dec64_sum_addr	--	Atomically read and decrement conditionally by one 64-bit value in a summarized address (zero value does not underflow).
ezdp_atomic_read_and_inc32_cond_ext_addr	--	Atomically read and increment conditionally a 32-bit value in an extended address.
ezdp_atomic_read_and_inc32_cond_sum_addr	--	Atomically read and increment conditionally a 32-bit value in a summarized address.
ezdp_atomic_and8_ext_addr	+	Atomically perform an 8-bit logical AND operation on an extended address.
ezdp_atomic_and16_ext_addr	+	Atomically perform a 16-bit logical AND operation on an extended address.
ezdp_atomic_and32_ext_addr	+	Atomically perform a 32-bit logical AND operation on an extended address.
ezdp_atomic_read_and_and8_ext_addr	--	Atomically read and perform an 8-bit logical AND operation on an extended address.
ezdp_atomic_read_and_and16_ext_addr	--	Atomically read and perform a 16-bit logical AND operation on an extended address.
ezdp_atomic_read_and_and32_ext_addr	--	Atomically read and perform a 32-bit logical AND operation on an extended address.
ezdp_atomic_and32_sum_addr	+	Atomically perform a 32-bit logical AND operation on a summarized address.
ezdp_atomic_read_and_and32_sum_addr	--	Atomically read and perform a 32-bit logical AND operation on a summarized address.
ezdp_atomic_or8_ext_addr	+	Atomically perform an 8-bit logical OR operation on an extended address.
ezdp_atomic_or16_ext_addr	+	Atomically perform a 16-bit logical OR operation on an extended address.
ezdp_atomic_or32_ext_addr	+	Atomically perform a 32-bit logical OR operation on an extended address.
ezdp_atomic_read_and_or8_ext_addr	--	Atomically read and perform an 8-bit logical OR operation on an extended address.
ezdp_atomic_read_and_or16_ext_addr	--	Atomically read and perform a 16-bit logical OR operation on an extended address.
ezdp_atomic_read_and_or32_ext_addr	--	Atomically read and perform a 32-bit logical OR operation on an extended address.

API Routine	Async	Description
ezdp_atomic_or8_sum_addr	+	Atomically perform an 8-bit logical OR operation on a summarized address.
ezdp_atomic_or16_sum_addr	+	Atomically perform a 16-bit logical OR operation on a summarized address.
ezdp_atomic_or32_sum_addr	+	Atomically perform a 32-bit logical OR operation on a summarized address.
ezdp_atomic_read_and_or32_sum_addr	--	Atomically read and perform a 32-bit logical OR operation on a summarized address.
ezdp_atomic_xor8_ext_addr	+	Atomically perform an 8-bit logical XOR operation on an extended address.
ezdp_atomic_xor16_ext_addr	+	Atomically perform a 16-bit logical XOR operation on an extended address.
ezdp_atomic_xor32_ext_addr	+	Atomically perform a 32-bit logical XOR operation on an extended address.
ezdp_atomic_read_and_xor8_ext_addr	--	Atomically read and perform an 8-bit logical XOR operation on an extended address.
ezdp_atomic_read_and_xor16_ext_addr	--	Atomically read and perform a 16-bit logical XOR operation on an extended address.
ezdp_atomic_read_and_xor32_ext_addr	--	Atomically read and perform a 32-bit logical XOR operation on an extended address.
ezdp_atomic_xor32_sum_addr	+	Atomically perform a 32-bit logical XOR operation on a summarized address.
ezdp_atomic_read_and_xor32_sum_addr	--	Atomically read and perform a 32-bit logical XOR operation on a summarized address.

4.3 Counter Operations (ezdp_counter.h)

EZdp provides API routines for operating on counters. Separate routines are provided for each counter type: single, double, bitwise, token bucket, watchdog and posted counters. In addition, routines for reading counter messages from the counter message queues are provided.

4.3.1 On-demand Counter Operations

EZdp provides the following API routines for operating with on-demand counters.

API Routine	Async	Description
ezdp_write_single_ctr_cfg	+	Configure single counter and its initial value.
ezdp_read_single_ctr_cfg	+	Read single counter configuration.
ezdp_write_single_ctr	+	Initialize single counter with the value specified.
ezdp_xchg_single_ctr	--	Write single counter with the value specified and read previous counter value.
ezdp_read_single_ctr	--	Read single counter value.
ezdp_inc_single_ctr	+	Increment single counter by the value specified.
ezdp_read_and_inc_single_ctr	--	Increment single counter by the value specified and read previous counter value.
ezdp_dec_single_ctr	+	Decrement single counter by the value specified.
ezdp_read_and_dec_single_ctr	--	Decrement single counter by the value specified and read previous counter value.
ezdp_reset_single_ctr	+	Reset single counter to zero.
ezdp_read_and_reset_single_ctr	--	Reset single counter to zero and read previous counter value.
ezdp_cond_dec_single_ctr	+	Conditionally decrement single counter by the value specified.
ezdp_read_and_cond_dec_single_ctr	--	Conditionally decrement single counter by the value specified and read previous counter value.
ezdp_prefetch_single_ctr	+	Prefetch single counter into the local cache.
ezdp_write_dual_ctr_cfg	+	Configure dual counter and its initial values (byte and event).
ezdp_read_dual_ctr_cfg	+	Read dual counter configuration.
ezdp_read_dual_ctr	--	Read dual counter values (byte and event).
ezdp_inc_dual_ctr	+	Increment dual counter with the values specified (byte and event).
ezdp_read_and_inc_dual_ctr	--	Increment dual counter with the values specified (byte and event) and read previous counter values.
ezdp_dec_dual_ctr	+	Decrement dual counter's byte value by the value specified and event value by 1.
ezdp_read_and_dec_dual_ctr	--	Decrement dual counter's byte value by the value specified and event value by 1, and read previous counter values.
ezdp_reset_dual_ctr	+	Reset dual counter values (byte and event) to zero.
ezdp_read_and_reset_dual_ctr	--	Reset dual counter values (byte and event) to zero and read previous counter values.
ezdp_prefetch_dual_ctr	+	Prefetch dual counter into the local cache.

API Routine	Async	Description
ezdp_write_bitwise_ctr_cfg	+	Configure bitwise counter and its initial value.
ezdp_read_bitwise_ctr_cfg	+	Read bitwise counter configuration.
ezdp_write_bits_bitwise_ctr	+	Write the value to the selected bits in the bitwise counter.
ezdp_xchg_bits_bitwise_ctr	--	Write the value to the selected bits in the bitwise counter and read previous counter value.
ezdp_read_bitwise_ctr	--	Read bitwise counter value.
ezdp_read_bits_bitwise_ctr	--	Read the selected bits from the bitwise counter.
ezdp_inc_bits_bitwise_ctr	+	Increment the selected bits in the bitwise counter by the value specified.
ezdp_read_and_inc_bits_bitwise_ctr	--	Increment the selected bits in the bitwise counter by the value specified and read previous counter value.
ezdp_dec_bits_bitwise_ctr	+	Decrement the selected bits in the bitwise counter by the value specified.
ezdp_read_and_dec_bits_bitwise_ctr	--	Decrement the selected bits in the bitwise counter by the value specified and read previous counter value.
ezdp_reset_bitwise_ctr	+	Reset bitwise counter value to zero.
ezdp_read_and_reset_bitwise_ctr	--	Reset bitwise counter value to zero and read previous counter value.
ezdp_set_bits_bitwise_ctr	+	Set the selected bits in the bitwise counter according to the value specified.
ezdp_read_and_set_bits_bitwise_ctr	--	Set the selected bits in the bitwise counter according to the value specified and read previous counter value.
ezdp_clear_bits_bitwise_ctr	+	Clear the selected bits in the bitwise counter according to the value specified.
ezdp_read_and_clear_bits_bitwise_ctr	--	Clear the selected bits in the bitwise counter according to the value specified and read previous counter value.
ezdp_read_and_cond_write_bits_bitwise_ctr	--	Read, compare and conditionally set the specified bits in the bitwise counter with the value specified.
ezdp_prefetch_bitwise_ctr	+	Prefetch bitwise counter into the local cache
ezdp_write_tb_ctr_cfg	+	Configure token bucket counter.
ezdp_read_tb_ctr_cfg	--	Read token bucket counter configuration.
ezdp_update_tb_ctr	+	Update a Token Bucket with the specified value (e.g. packet length).
ezdp_read_tb_ctr	+	Get the resulting color after updating a Token Bucket with the specified value (e.g. packet length).
ezdp_check_tb_ctr	+	Get the resulting color after updating a Token Bucket with the specified value (e.g. packet length), without updating it.
ezdp_inc_tb_ctr	+	Force increment token buckets with the specified value (e.g. packet length).
ezdp_read_and_inc_tb_ctr	--	Force increment token buckets with the specified value (e.g. packet length) and get resulting color and bucket states.
ezdp_dec_tb_ctr	+	Force decrement token buckets with the specified value (e.g. packet length).
ezdp_read_and_dec_tb_ctr	--	Force decrement token buckets with the specified value (e.g. packet length) and get resulting color and bucket states.

API Routine	Async	Description
ezdp_prefetch_tb_ctr	+	Prefetch token bucket counter into the local cache.
ezdp_write_hier_tb_ctr_cfg	+	Configure hierarchical token bucket counter.
ezdp_read_hier_tb_ctr_cfg	--	Read hierarchical token bucket counter configuration
ezdp_inc_hier_tb_ctr	+	Increment hierarchical token bucket counter accumulator(s) by the value specified.
ezdp_read_and_inc_hier_tb_ctr	--	Increment hierarchical token bucket counter accumulator(s) by the value specified and read previous counter value.
ezdp_update_hier_tb_ctr	+	Update hierarchical token bucket counter with state, app bits or clear accumulators.
ezdp_read_and_update_hier_tb_ctr	--	Update hierarchical token bucket counter with state, app bits or clear accumulators and read previous counter value.
ezdp_change_state_hier_tb_ctr	--	Change hierarchical token bucket state to Ph1
ezdp_write_watchdog_ctr_cfg	+	Configure watchdog counter.
ezdp_read_watchdog_ctr_cfg	--	Read watchdog counter configuration.
ezdp_start_watchdog_ctr	+	Start the watchdog counter.
ezdp_inc_watchdog_ctr	+	Increment the watchdog counter events by one.
ezdp_check_watchdog_ctr	+	Check the number of events in the watchdog counter.
ezdp_prefetch_watchdog_ctr	+	Prefetch watchdog counter into the local cache.
ezdp_init_ctr_msg_queue_des	--	Initialize counter message queue descriptor.
ezdp_read_ctr_msg	--	Read counter message from message queue.

4.3.2 Posted Counter Operations

EZdp provides the following API routines for operating with posted counters.

API Routine	Async	Description
ezdp_write_posted_ctr	+	Initialize posted counter with the value specified.
ezdp_dual_write_posted_ctr	+	Initialize two successive posted counters with the value specified.
ezdp_add_posted_ctr	+	Add signed value to posted counter.
ezdp_dual_add_posted_ctr	+	Add signed values to two successive posted counters.
ezdp_report_posted_ctr	--	Generate posted counter value report.
ezdp_report_and_clear_posted_ctr	--	Generate posted counter value report and reset the counter to zero.
ezdp_dual_report_posted_ctr	--	Generate posted counter value report for two successive counters.
ezdp_dual_report_and_clear_posted_ctr	--	Generate posted counter value report for two successive counters and reset both counter values to zero.
ezdp_reset_posted_ctr	+	Reset posted counter.
ezdp_dual_reset_posted_ctr	+	Reset two successive posted counters
ezdp_init_posted_ctr_msg_queue_desc	--	Initialize posted counter message queue descriptor.
ezdp_read_posted_ctr_msg	--	Read posted counter message from message queue.

4.4 Frame Data Decoding (ezdp_decode.h)

EZdp provides API routines for performing decoding of frame data for standard protocol headers.

The decoding operations are split into two types. The first type parses and decodes multi-byte frame data stored in the core's local memory (CMEM). These operations receiving a pointer to the start of the data in the CMEM and the size/length of the header to parse, and may take several cycles.

API Routine	Async	Description
ezdp_decode_mac	+	Parse and decode an Ethernet header.
ezdp_decode_ipv4	+	Parse and decode an IPv4 header.
ezdp_decode_ipv6	+	Parse and decode an IPv6 header.
ezdp_decode_mpls	+	Parse and decode an MPLS header.
ezdp_decode_mpls_label	+	Parse and decode an MPLS label.
ezdp_decode_tcp	--	Parse and decode a TCP header.

The second type does not parse data in the CMEM, but instead decodes a single, fixed-size value stored in a variable/register.

API Routine	Async	Description
ezdp_decode_ip_protocol	--	Decode an IP protocol value.
ezdp_decode_eth_type	--	Decode an Ethernet type value.

4.5 DMA Operations (ezdp_dma.h)

EZdp provides API routines for performing DMA operations. These operations function on either extended addresses or summarized addresses.

📖 For more information on extended and summarized addresses, see the *NPS-400 Programming Manual*.

API Routine	Async	Description
ezdp_copy_data_by_ext_addr	+	Copy data between two extended addresses.
ezdp_load_data_from_ext_addr	+	Copy data from an extended address to CMEM.
ezdp_load_16_byte_data_from_ext_addr	+	Copy 16 bytes from an extended address to CMEM.
ezdp_load_32_byte_data_from_ext_addr	+	Copy 32 bytes from an extended address to CMEM.
ezdp_store_data_to_ext_addr	+	Copy data from CMEM to an extended address.
ezdp_store_16_byte_data_to_ext_addr	+	Copy 16 bytes from CMEM to an extended address.
ezdp_store_32_byte_data_to_ext_addr	+	Copy 32 bytes from CMEM to an extended address.
ezdp_load_data_from_sum_addr	+	Load data from a summarized address to CMEM.
ezdp_load_16_byte_data_from_sum_addr	+	Load 16 bytes from a summarized address to CMEM
ezdp_load_32_byte_data_from_sum_addr	+	Load 32 bytes from a summarized address to CMEM
ezdp_store_data_to_sum_addr	+	Store data from CMEM to summarized address.
ezdp_store_16_byte_data_to_sum_addr	+	Store 16 bytes from CMEM to a summarized address
ezdp_store_32_byte_data_to_sum_addr	+	Store 32 bytes from CMEM to a summarized address

4.6 Frame Buffer Management (ezdp_frame.h)

EZdp provides API routines for operating with frame buffers. All frame buffer API routines operate on a buffer descriptor data structure (as defined in ezdp_frame_defs.h), which represents a handle to a frame buffer in either internal or external memory.

4.6.1 Resource Management

EZdp provides API routines for allocating/freeing frame buffers from the hardware buffer pools. The API supports simple and optimized operations for the common case of allocating/freeing a single buffer, and in addition also supports more advanced operations for allocating/freeing multiple buffers in a single command (up to 8 buffers per command).

API Routine	Async	Description
ezdp_alloc_buf	--	Allocate a single frame buffer.
ezdp_free_buf	+	Free a single frame buffer.
ezdp_alloc_multi_buf	+	Allocate multiple frame buffers.
ezdp_buf_alloc_failed	--	Check if allocation of the buffer failed.
ezdp_free_multi_buf	+	Free multiple frame buffers.
ezdp_read_free_buf	--	The number of buffers available to be obtained.
ezdp_rebudget_buf	+	Update the budget to which buffers are credited.

4.6.2 DMA Operations

EZdp provides API routines for performing DMA operations on frame data buffers. The API supports operations to load data from a frame buffer to the core's local memory (CMEM), store data from the CMEM to the frame buffer, and copy data between two frame buffers or between a frame buffer and an extended memory address.

API Routine	Async	Description
ezdp_copy_frame_data	+	Copy data between two frame buffers.
ezdp_copy_frame_data_to_ext_addr	+	Copy data from a frame buffer to an extended address.
ezdp_copy_frame_data_from_ext_addr	+	Copy data from an extended address to a frame buffer.
ezdp_clone_frame_data	+	Copy data between two frame buffers, with the same source and destination offset (optimized).
ezdp_load_frame_data	+	Copy data from a frame buffer to the CMEM.
ezdp_store_frame_data	+	Copy data from CMEM to a frame buffer.

Similarly, EZdp provides APIs to perform DMA operations on link-buffer-descriptor (LBD) data residing in frame buffers. These require separate/dedicated APIs as they use different memory error protection mechanisms.

API Routine	Async	Description
ezdp_copy_frame_lbd	+	Copy LBD data between two frame buffers.
ezdp_copy_frame_lbd_to_ext_addr	+	Copy LBD data from a frame buffer to an extended address.
ezdp_copy_frame_lbd_from_ext_addr	+	Copy LBD data from an extended address to a frame buffer.
ezdp_clone_frame_lbd	+	Copy LBD between two frame buffers, with the same source and destination offset (optimized).
ezdp_load_frame_lbd	+	Copy LBD data from a frame buffer to CMEM.
ezdp_store_frame_lbd	+	Copy LBD data from CMEM to a frame buffer.

4.6.3 Multicast Reference Counters

A dedicated multicast reference counter is maintained for each frame buffer. EZdp provides API routines for operating with the frame buffer's multicast reference counters.

API Routine	Async	Description
ezdp_alloc_mc_buf	--	Allocate a single frame buffer and set its multicast reference counter.
ezdp_free_mc_buf	--	Free a multicast frame buffer.
ezdp_write_mc_buf_counter	+	Set a frame buffer's multicast reference counter.
ezdp_read_mc_buf_counter	--	Get a frame buffer's multicast reference counter.
ezdp_atomic_read_and_inc_mc_buf_counter	--	Atomically read and increment a frame buffer's multicast reference counter.
ezdp_atomic_read_and_dec_mc_buf_counter	--	Atomically read and conditionally decrement a frame buffer's multicast reference counter.

4.6.4 Additional Operations

The following services are provided for operating with frame data structures:

API Routine	Async	Description
ezdp_calc_frame_data_checksum	--	Calculate checksum of frame data buffer.
ezdp_buf_data_len	--	Calculate the length of the buffer based on header offset and free bytes.
ezdp_lbd_length	--	Calculate LBD buffer length according to BD count.
ezdp_calc_header_offset	--	Calculate optimized frame header offset.

4.6.5 Frame Iterator Operations

The following services are provided for frame iterator operations:

API Routine	Async	Description
ezdp_get_first_buf	--	Gets first buffer from frame.
ezdp_get_next_buf	--	Get next buffer from frame.
ezdp_init_frame	--	Create a new frame by setting its frame descriptor params and init frame iterator.
ezdp_append_buf	--	Add newly allocated (by user) buffer to frame pointed by iterator.
ezdp_sync_frame	--	Store last LBD line to memory, if required.

4.6.6 TM Internal Memory Buffer Management (TM Data Cache)

The following additional services are provided for TM IMEM buffer management:

API Routine	Async	Description
ezdp_inc_tm_imem_buf_ctr	+	Increment TM IMEM buffer counter.
ezdp_dec_tm_imem_buf_ctr	+	Decrement TM IMEM buffer counter.
ezdp_read_tm_imem_buf_ctr	--	Number of IMEM buffers used by frame in TM.

4.7 Job Management (ezdp_job.h)

EZdp provides API routines for operating with jobs. The job management API routines operate on a job ID which represents a handle to a job descriptor residing in the internal memory. In addition, some of the APIs operate on the job descriptor data structure itself (as defined in ezdp_job_defs.h), which represents an active job.

The job management API routines support various levels of abstraction and control. The API provides both low-level routines to enable the data-plane application fine-grained control of the job management, and higher-level API routines that are suitable for most use-cases and provide a more convenient API to perform the required operations.

4.7.1 Resource Management

EZdp provides API routines for allocating/freeing jobs from the hardware job pool in the Processor Manager Unit (PMU). Note that in most cases, the data-plane application does not create or free jobs, but rather only receives existing jobs which are created by the NPS hardware, performs operations on these jobs, and then sends the jobs back to the NPS hardware which later frees the job resources.

API Routine	Async	Description
ezdp_alloc_job_id	+	Allocate a new job from the PMU.
ezdp_alloc_multi_job_id	+	Allocate multiple new jobs from the PMU.
ezdp_job_alloc_failed	--	Check if allocation of the job failed.
ezdp_free_job_id	+	Recycle a job to the PMU.
ezdp_read_free_job	--	The number of jobs available to be obtained.
ezdp_rebudget_job	+	Update the budget to which jobs are credited.

4.7.2 DMA Operations

EZdp provides API routines for performing DMA operations on job descriptors.

API Routine	Async	Description
ezdp_load_job	+	Copy the job descriptor from IMEM to CMEM.
ezdp_store_job	+	Copy the job descriptor from CMEM to IMEM.
ezdp_store_job_container	+	Copy the job container descriptor from CMEM to IMEM.

These include operations to load job descriptor data from the internal memory to the core's local memory (CMEM), and store data from the CMEM to the internal memory. These APIs receive a job id representing the job descriptor in internal memory and a pointer to the job descriptor data in the CMEM.

4.7.3 Receiving Jobs

EZdp provides API routines for receiving jobs from the NPS hardware Processor Management Unit (PMU).

API Routine	Async	Description
ezdp_request_job_id	--	Request a new job from the PMU.
ezdp_wait_for_job_id	--	Suspend execution until a job request completes.
ezdp_cancel_job_request	--	Cancel a job request from the PMU.
ezdp_receive_job	--	Request a new job from the PMU and load it to CMEM.

4.7.4 Transmitting Jobs

EZdp provides API routines for transmitting jobs to the network interfaces.

API Routine	Async	Description
ezdp_send_job_id_to_tm	+	Transmit the job via the Traffic Manager (TM).
ezdp_send_job_to_tm	--	Store the job descriptor and transmit the job via the Traffic Manager (TM).
ezdp_send_job_id_to_interface	+	Transmit the job directly to an output queue channel, bypassing the TM.
ezdp_send_job_to_interface	--	Store the job descriptor and transmit the job directly to an output queue channel, bypassing the TM.

4.7.5 Moving Job to Another Queue

EZdp provides API routines for moving jobs to another queue in the Processor Management Unit (PMU).

API Routine	Async	Description
ezdp_send_job_id_to_queue	+	Dispatch the job to another queue in the PMU.
ezdp_send_job_to_queue	--	Store the job descriptor and dispatch the job to another queue in the PMU.
ezdp_update_job_id_queue	--	Move the job to another PMU queue without dispatching it.
ezdp_update_job_queue	--	Store the job descriptor and move the job to another PMU queue without dispatching it.

4.7.6 Discarding Jobs

EZdp provides API routines for discarding a job, including all its associated frame buffer resources.

API Routine	Async	Description
ezdp_discard_job_id	+	Discard a job and all its associated frame resources.
ezdp_discard_job	--	Store the job descriptor and discard the job and all its associated frame resources.

4.7.7 Job Containers

EZdp provides API routines for operating with job containers.

API Routine	Async	Description
ezdp_send_job_id_container	+	Send request to PMU to distribute the job container.
ezdp_send_job_container	--	Store the job container and send request to PMU to distribute it.
ezdp_container_job_count	--	Return the number of the jobs that are in the job container.
ezdp_container_info	--	Set the info according to the number of jobs in container.

4.7.8 Inter Process Communication

EZdp provides API routines for Inter Process Communication (IPC).

API Routine	Async	Description
ezdp_notify_cpu	--	Notify target CPU.
ezdp_notice_pending	--	Check if there is a new notice.
ezdp_clear_notice	--	Clear notice indication.
ezdp_wait_for_notice	--	Suspend execution until receiving new notification for CPU.
ezdp_check_notice	--	Check if there is a new notice and clear new notice indication.
ezdp_wait_for_event	--	Suspend execution until job request completed or new notice received.
ezdp_notifier	--	Register notifier function.
ezdp_handle_notice	--	Handle notice.

4.7.9 System Congestion Status

EZdp provides API routines for reading system congestion status from various sources.

API Routine	Async	Description
ezdp_read_congestion_status	--	Read priority drop congestion status.
ezdp_read_flow_control_status	--	Read flow control status.
ezdp_read_pmu_input_queue_congestion	--	Read PMU input queue congestion level.
ezdp_read_global_budget	--	Read global budget counter value.
ezdp_read_pmu_input_queue_status	--	Read PMU input queue status from system info.
ezdp_read_pmu_tm_output_queue_status	--	Read PMU TM output queue status from system info.
ezdp_read_pmu_discard_output_queue_status	--	Read PMU discard output queue status from system info.
ezdp_read_pmu_tm_bypass_output_queue_status	--	Read PMU TM bypass output queue status from system info.
ezdp_read_pmu_app_schlr_status	--	Read PMU application scheduler status from system info.
ezdp_read_pmu_group_schlr_status	--	Read PMU group scheduler status from system info.
ezdp_init_tm_reporting_desc	--	Initialize TM queue depth descriptor.
ezdp_calc_tm_queue_depth_handle	--	Calculate TM queue depth handle.
ezdp_get_tm_queue_depth	--	Get entity queue depth.
ezdp_valid_tm_queue_depth_handle	--	Validate tm_handle received from CP/DP API.

4.8 Lock Operations (ezdp_lock.h)

EZdp provides API routines for spin lock functions.

API Routine	Async	Description
ezdp_init_spinlock_ext_addr	--	Initialize resources required for a spin lock.
ezdp_init_spinlock_sum_addr	--	Initialize resources required for a spin lock.
ezdp_lock_spinlock	--	Lock a spin lock.
ezdp_try_lock_spinlock	--	Lock a spin lock with limited number of attempts.
ezdp_unlock_spinlock	--	Release a spin lock which was locked.

EZdp provides API routines for qlock functions.

API Routine	Async	Description
ezdp_init_qlock	--	Initialize queue lock structure.
ezdp_destroy_qlock	--	Get queue lock address.
ezdp_alloc_qlock_slot	--	Allocate queue lock slot.
ezdp_free_qlock_slot	--	Free queue lock slot.
ezdp_lock_qlock	--	Try to lock queue lock.
ezdp_order_lock_qlock	--	Try to lock queue lock (with order).
ezdp_enqueue_qlock	--	Enqueue data to queue lock.
ezdp_dequeue_qlock	--	Dequeue data from queue lock.
ezdp_try_unlock_qlock	--	Try to unlock queue lock.

4.9 ALU Operations (ezdp_math.h)

EZdp provides API routines for advanced ALU operations, including arithmetic operations, logical operations, and bit-manipulation operations. All bit manipulation APIs operate on 32-bit variables.

4.9.1 Arithmetic and Logical Operations

EZdp provides the following API routines for performing advanced mathematical operations. These APIs support mathematical operations on specific bits within 32-bit values in an optimized manner.

API Routine	Async	Description
ezdp_add	--	Add selected bits of src1 to selected bits of src2.
ezdp_sub	--	Subtract selected bits of src2 from selected bits of src1.
ezdp_and	--	Perform logical 'AND' between selected bits of src1 and src2.
ezdp_or	--	Perform logical 'OR' between selected bits of src1 and src2.
ezdp_not	--	Perform logical 'NOT' between selected bits of src1 and src2.
ezdp_xor	--	Perform logical 'XOR' between selected bits of src1 and src2.
ezdp_fxor8	--	Apply an 8 bit 'folded xor' operation on selected bits of src1 and src2.
ezdp_fxor16	--	Apply a 16 bit 'folded xor' operation on selected bits of src1 and selected bits of src2.
ezdp_shift_left	--	Perform a 'shift left' operation on a set of bits in src1, with shift size selected by 5 adjacent bits of src2.
ezdp_shift_right	--	Perform a 'shift right' operation on a set of bits in src1, with shift size selected by 5 adjacent bits of src2.
ezdp_count_bits	--	Count the number of bits with value of '1' in a set of bits in src.
ezdp_div	--	Divide 8 selected bits of src1 by 4 selected bits of src2.
ezdp_mod	--	Modulus 8 selected bits of src1 by 4 selected bits of src2.
ezdp_pow_of_2	--	Calculate the value of 2^exp and merge into any position in dst.
ezdp_merge_pow_of_2	--	Calculate the value of 2^exp and merge into any position in src.

4.9.2 Bit Manipulation Operations

EZdp provides the following API routines for performing advanced bit-manipulation operations. These APIs support extracting, inserting, and merging of multiple bits within 32 bit values in an optimized manner.

API Routine	Async	Description
ezdp_set_bit	--	Set a single bit in src to 'one'.
ezdp_clear_bit	--	Clear a single bit in src (sets to 'zero')
ezdp_find_first_one	--	Find the position of the first 'one' in the range src[src_pos+size-1 : src_pos] (from lsb to msb).
ezdp_find_first_zero	--	Find the position of the first 'zero' in the range src[src_pos+size-1 : src_pos] (from lsb to msb).
ezdp_get_bitfield	--	Get a set of adjacent bits from src and place into any position in dst.
ezdp_merge_bitfield	--	Get a set of adjacent bits from src2 and merge into any position in src1.
ezdp_get_2_bitfields	--	Get 2 sets of adjacent bits from src1 and src2 and place into two locations in dst.
ezdp_merge_2_bitfields	--	Get 2 sets of adjacent bits from src1 and src2 and merge into two locations in src1.
ezdp_get_bit	--	Get any bit from src and place in any position in dst.
ezdp_merge_bit	--	Get any bit from src2 and merge it any position in src1.
ezdp_get_2_bits	--	Get two separate bits from src and place in two separate locations in dst.
ezdp_merge_2_bits	--	Get two separate bits from src2 and merge into two separate locations in src1.
ezdp_get_3_bits	--	Get three separate bits from src and place in three separate locations in dst.
ezdp_merge_3_bits	--	Get three separate bits from src2 and merge into three separate locations in src1.
ezdp_get_4_bits	--	Get four separate bits from src and place in four separate locations in dst.
ezdp_merge_4_bits	--	Get four separate bits from src2 and merge into four separate locations in src1.
ezdp_combine_4_bits	--	Get four separate bits from src and place into 4 adjacent bits in dst.
ezdp_combine_merge_4_bits	--	Get four separate bits from src2 to merge into 4 adjacent bits in src1.
ezdp_split_4_bits	--	Get four adjacent bits from src and place in four separate positions in destination.
ezdp_split_merge_4_bits	--	Get four adjacent bits from src2 and merge into four separate positions in src1.
ezdp_get_4_bytes	--	Extract any four bytes from src1 and src2.
ezdp_reflect_bits	--	Perform bit swap in resolution of 1, 2 or 4 bytes.

4.9.3 Hash Operations

EZdp provides the following API routines for calculating advanced hash values.

API Routine	Async	Description
ezdp_hash	--	General purpose hash function.
ezdp_hash32	--	General purpose hash function for 32-bit input.
ezdp_hash64	--	General purpose hash function for 64-bit input.
ezdp_bulk_hash	--	General purpose hash function for up to 64-byte input.
ezdp_calc_crc16	--	Perform CRC16 calculation
ezdp_calc_crc32	--	Perform CRC32 calculation
ezdp_add_checksum	--	Add value to checksum.
ezdp_sub_checksum	--	Subtract value from checksum.

4.10 Memory Operations (ezdp_memory.h)

EZdp provides API routines for operating with memory addresses.

API Routine	Async	Description
ezdp_calc_checksum_ext_addr	--	Calculate checksum of data on extended address.
ezdp_calc_checksum	--	Calculate checksum on a block of memory in CMEM.
ezdp_is_null_sum_addr	--	Check if a summarized address is null.
ezdp_calc_sum_addr	--	Calculate summarized address from address descriptor and key.
ezdp_sum_addr_to_ext_addr	--	Calculate extended address from summarized address.
ezdp_ext_addr_to_sum_addr	--	Calculate summarized address from extended address.
ezdp_calc_sum_addr_offset	--	Calculate offset of a summarized address from extended address.
ezdp_scramble_sum_addr	--	Scramble given summarized address.
ezdp_scramble_ext_addr	--	Scramble given extended address.

4.11 PCI Interface Operations (ezdp_pci.h)

EZdp provides API routines for operating with the PCI Express interface.

Note: The PCI Interface APIs are preliminary and not supported in the current EZdk release. They are provided here as reference only.

4.11.1 PCI Message Queue Operations

API Routine	Async	Description
ezdp_init_pci_queue_desc	--	Initialize and get PCI message queue description.
ezdp_get_pci_msg	--	Get message from PCI queue according to given index.
ezdp_set_pci_msgq_read_index	+	Set read index of PCI message queue.
ezdp_get_pci_msgq_write_index	--	Get write index of PCI message queue.
ezdp_get_pci_msgq_read_index	--	Get read index of PCI message queue.

4.11.2 Copy Operations

API Routine	Async	Description
ezdp_copy_frame_data_to_pci	+	Copy data from a frame buffer to PCI address.
ezdp_copy_frame_data_from_pci	+	Copy data from PCI address to a frame buffer.
ezdp_load_data_from_pci	+	Copy data from a PCI address to CMEM.
ezdp_store_data_to_pci	+	Copy data from CMEM to PCI address.
ezdp_copy_pci_data_to_ext_addr	+	Copy PCI data to extended addresses.
ezdp_copy_pci_data_from_ext_addr	+	Copy extended address data to PCI.
ezdp_translate_pci_addr	+	PCI address translation request; result will be saved in CMEM.
ezdp_translate_pci_addr_to_ext_addr	+	Translate PCI address to extended addresses.
ezdp_send_message_to_pci	+	Copy data from CMEM to PCI address.
ezdp_send_interrupt_to_pci	+	Send interrupt message to PCI.

4.11.3 Configuration Space Operations

API Routine	Async	Description
ezdp_get_pci_ctrl_reg	--	Get the value of the PCIe vendor specific configuration space register.
ezdp_set_pci_ctrl_reg	--	Set the value of the PCIe vendor specific configuration space register.

4.12 Pool Operations (ezdp_pool.h)

EZdp provides the following API routines for operating with pools, including user-defined index pool and memory pool. The API supports simple and optimized operations for the common case of allocating/freeing a single index, and in addition also supports more advanced operations for allocating/freeing multiple indexes in a single command (up to 8 indexes per command).

4.12.1 Index Pool Operations

API Routine	Async	Description
ezdp_alloc_index	--	Allocate a single index from an index pool.
ezdp_free_index	+	Free a single index from an index pool.
ezdp_alloc_multi_index	+	Allocate multiple indexes from an index pool.
ezdp_free_multi_index	+	Free multiple indexes from an index pool.
ezdp_read_free_indexes	--	The number of indexes available to be obtained.

4.12.2 Memory Pool Operations

API Routine	Async	Description
ezdp_init_memory_pool	--	Initialize a memory pool.
ezdp_alloc_obj	--	Allocate a single object from a memory pool.
ezdp_free_obj	--	Free a single object from a memory pool.
ezdp_get_obj	--	Get object based on object id.
ezdp_read_free_objs	--	The number of objects available to be obtained.

4.13 Processor Control Operations (ezdp_processor.h)

EZdp provides API routines for controlling the CTOP processors.

4.13.1 Identification Operations

EZdp provides the following API routines for CTOP processor identification.

API Routine	Async	Description
ezdp_get_cpu_id	--	Get the logical ID of the processor that the process is running on (0-4095).
ezdp_get_thread_id	--	Get the ID of the thread (within the core) that the process is running on (0-15).
ezdp_get_core_id	--	Get the ID of the core (within the cluster) that the process is running on (0-15).
ezdp_get_cluster_id	--	Get the ID of the cluster that the process is running on (0-15).
ezdp_calc_cpu_id	--	Calculate the logical ID of a processor.

4.13.2 Synchronization and Scheduling Operations

EZdp provides the following API routines for CTOP processor synchronization and scheduling.

API Routine	Async	Description
ezdp_sync	--	Relinquish the execution unit until all outstanding transactions complete.
ezdp_rsync	--	Relinquish the execution unit until all outstanding read transactions complete.
ezdp_mb	--	Wait until all outstanding memory accesses complete.
ezdp_rmb	--	Wait until all outstanding memory read accesses complete.
ezdp_wmb	--	Wait until all outstanding memory write accesses complete.

4.14 Queue Operations (ezdp_queue.h)

EZdp provides the following API routines for operating with queues: ring (array queue) and list queue.

4.14.1 Ring (Array Queue) Operations

API Routine	Async	Description
ezdp_init_ring	--	Initialize ring.
ezdp_ring_empty	--	Check if ring is empty.
ezdp_ring_full	--	Check if array_queue is full.
ezdp_ring_length	--	Return the number of entries in ring.
ezdp_enqueue_ring	--	Insert new entry to ring.
ezdp_dequeue_ring	--	Remove a head entry from ring.

4.14.2 List Queue Operations

API Routine	Async	Description
ezdp_init_list	--	Initialize list
ezdp_list_empty	--	Check if a list is empty.
ezdp_enqueue_list	--	Insert a new entry into a list.
ezdp_dequeue_list	--	Remove a head entry from a list.
ezdp_peek_list	--	Peek at head entry of a list.
ezdp_destroy_list	--	Destroy a list.

4.15 Search Structure Operations (ezdp_search.h)

EZdp provides API routines for operating on search structures such as lookup operations and entry management operations (e.g. adding, updating and deleting entries). Separate routines are provided for each of the supported search structure types (direct table, hash, UltraIP and TCAM).

4.15.1 Direct Table Structures

EZdp provides the following API routines for operating with direct table structures.

API Routine	Async	Description
ezdp_init_table_struct_desc	--	Initialize the structure descriptor for a table structure.
ezdp_validate_table_struct_desc	--	Validate the table structure parameters.
ezdp_lookup_table_entry	--	Lookup an entry in a table structure.
ezdp_add_table_entry	--	Add an entry in a table structure.
ezdp_modify_table_entry	--	Modify an existing entry in a table structure.
ezdp_update_table_entry	--	Update an entry in a table structure. Add the entry if it does not exist, otherwise modify it.
ezdp_delete_table_entry	--	Delete an entry from a table structure.

4.15.2 Hash Structures

EZdp provides the following API routines for operating with hash structures.

API Routine	Async	Description
ezdp_init_hash_struct_desc	--	Initialize the structure descriptor for a hash structure.
ezdp_validate_hash_struct_desc	--	Validate the hash structure parameters
ezdp_lookup_hash_entry	--	Lookup an entry in a hash structure.
ezdp_lookup_hash_entry_ctx	--	Lookup an entry in a hash structure while maintaining operation context.
ezdp_add_hash_entry	--	Add an entry in a hash structure.
ezdp_modify_hash_entry	--	Modify an existing entry in a hash structure.
ezdp_update_hash_entry	--	Update an entry in a hash structure. Add the entry if it does not exist, otherwise modify it.
ezdp_delete_hash_entry	--	Delete an entry from a hash structure.
ezdp_scan_hash_slot	--	Scan a hash slot.
get_hash_entry_key	--	Get hash key from entry.

4.15.3 UltraIP Structures

EZdp provides the following API routines for operating with UltraIP structures.

API Routine	Async	Description
ezdp_init_ultra_ip_struct_desc	--	Initialize the structure descriptor for an UltraIP structure.
ezdp_validate_ultra_ip_struct_desc	--	Validate the UltraIP structure parameters.
ezdp_lookup_ultra_ip_entry	--	Lookup an entry in an UltraIP structure.

4.15.4 TCAM Structures

EZdp provides the following API routines for operating with the ternary CAMs (TCAMs).

API Routine	Async	Description
ezdp_lookup_int_tcam	+	Lookup an entry in an internal TCAM.
ezdp_lookup_ext_tcam	+	Lookup an entry in an external TCAM.

4.15.5 Algorithmic TCAM Structures

EZdp provides the following API routines for operating with algorithmic TCAM search structures.

API Routine	Async	Description
ezdp_init_alg_tcam_struct_desc	--	Initialize the structure descriptor for an algorithmic TCAM structure.
ezdp_validate_alg_tcam_struct_desc	--	Validate the algorithmic TCAM structure parameters.
ezdp_lookup_alg_tcam	--	Lookup an entry in an algorithmic TCAM structure.

4.16 Primitive Search Structure Operations (ezdp_search_prm.h)

EZdp provides API routines for advanced low-level operations on search structures.

4.16.1 Direct Table Structures

EZdp provides the following API routines for operating with direct table structures.

API Routine	Async	Description
ezdp_prm_lock_table_line	--	Lock a table entry
ezdp_prm_trylock_table_line	--	Try to lock a table entry
ezdp_prm_unlock_table_line	--	Release a table entry lock
ezdp_prm_get_table_base_addr	--	Get search base address (used for lookup) from table struct descriptor
ezdp_prm_lookup_table_entry	--	Lookup an entry in a table structure.
ezdp_prm_update_table_entry	--	Add a new entry in a table structure. Override the existing entry.
ezdp_prm_delete_table_entry	--	Delete an entry from a table structure.

4.16.2 Hash Structures

EZdp provides the following API routines for operating with direct table structures.

API Routine	Async	Description
ezdp_prm_hash_key32	--	Calculate hash value for keys of up to 32 bits.
ezdp_prm_hash_key64	--	Calculate hash value for keys of up to 64 bits.
ezdp_prm_hash_bulk_key	--	Calculate hash value for keys > 8 bytes.
ezdp_prm_lock_hash_slot	--	Lock a hash slot according to the hashed key.
ezdp_prm_trylock_hash_slot	--	Try to lock a hash slot according to the hashed key.
ezdp_prm_unlock_hash_slot	--	Release a hash slot lock.
ezdp_prm_get_hash_base_addr	--	Get search base address (used for lookup) from hash struct descriptor.
ezdp_prm_lookup_hash_entry	--	Lookup an entry in a hash structure.
ezdp_prm_locate_hash_entry	--	Lookup an entry location in a hash structure.
ezdp_prm_add_hash_entry	--	Add a new entry in a hash structure.
ezdp_prm_modify_hash_entry	--	Modify an existing entry in a hash structure. entry_ptr should be updated with the new result.
ezdp_prm_delete_hash_entry	--	Delete an existing entry from a hash structure.
ezdp_prm_get_hash_first_entry	--	Get first entry of a hash slot.
ezdp_prm_get_hash_next_entry	--	Get next entry of a hash slot.
ezdp_prm_compress_hash_entry	--	Compress a hash entry with other entries in the same hash slot, if possible.

4.16.3 UltraIP Structures

EZdp provides the following API routines for operating with UltraIP structures.

API Routine	Async	Description
ezdp_prm_get_ultra_ip_base_addr	--	Get search base address (used for lookup) from UltraIP struct descriptor
ezdp_prm_lookup_ultra_ip_entry	--	Lookup an entry in an UltraIP structure.

4.16.4 Algorithmic TCAM Structures

EZdp provides the following API routines for operating with algorithmic TCAM search structures.

API Routine	Async	Description
ezdp_prm_lookup_alg_tcam	--	Lookup an entry in an algorithmic TCAM structure.

4.17 Security Operations (ezdp_security.h)

EZdp provides API routines for operating NPS security accelerators.

API Routine	Async	Description
ezdp_encrypt	+	Encrypt a data segment.
ezdp_decrypt	+	Decrypt a data segment.
ezdp_mac_calculation	+	Calculate the message authentication code (MAC) on a data segment.
ezdp_start_hmac_calculation	+	Start a hash-based message authentication code (MAC) calculation.
ezdp_end_hmac_calculation	+	Complete a hash-based message authentication code (MAC) calculation.
ezdp_generate_security_initial_vector	+	Generate security initial vector.
ezdp_end_gcm_mac_calculation	+	Complete a GCM hash-based message authentication code (MAC) calculation.
ezdp_expand_security_key	+	Expands the key in the security context memory.
ezdp_write_security_state	+	Copy the security state data from CMEM to the security context memory.
ezdp_read_security_state	+	Copy the security state data from the security context memory to CMEM.
ezdp_security_state_size	--	Return the state size.
ezdp_write_security_key	+	Copy the security key from CMEM to the security context memory.
ezdp_read_security_key	+	Copy the security key from the security context memory to CMEM.
ezdp_security_key_size	--	Return the key size.
ezdp_write_security_mac	+	Copy the security mac from CMEM to the security context memory.
ezdp_read_security_mac	+	Copy the security mac from the security context memory to CMEM.
ezdp_security_mac_size	--	Return the MAC size.
ezdp_write_security_initial_vector	+	Copy the security initial vector from CMEM to the security context memory.
ezdp_read_security_initial_vector	+	Copy the security initial vector from the security context memory to CMEM.
ezdp_security_initial_vector_size	--	Return the initial vector size.
ezdp_write_security_context	+	Copy the security context from CMEM to the security context memory.
ezdp_read_security_context	+	Copy the security context from the security context memory to CMEM.
ezdp_security_block_size	--	Return the algorithmic engine minimal block size.

4.18 String Operations (ezdp_string.h)

EZdp provides API routines for manipulating character arrays.

API Routine	Async	Description
ezdp_mem_copy	--	Copy a block of memory.
ezdp_mem_set	--	Set a block of memory to the specified value.
ezdp_mem_cmp	--	Compare two blocks of memory in CMEM.
ezdp_mem_cmp_byte_skip	--	Compare two blocks of memory in CMEM, skipping intermediate bytes.

4.19 Time Operations (ezdp_time.h)

EZdp provides API routines for retrieving network and system time.

API Routine	Async	Description
ezdp_get_system_tick	+	Get system tick (in core cycles).
ezdp_get_real_time_clock	+	Get real time clock.

5. Reference

The pages that follow list the API routines as well as their structures and enumerations.

Table of Contents

Data Structure Index

Data Structures

Here are the data structures with brief descriptions:

File Index

File List

Here is a list of all files with brief descriptions:

Data Structure Documentation

ezdp_1588_header Struct Reference

1588 header format definition

Data Fields

- union {
- struct {
- unsigned __pad0__: 7
- *Reserved bits 25 to 31.* unsigned __pad1__: 24
- *< Determine the action to process for an expanded job* }
- union {
- struct ezdp_1step_1588_header one_step
- *Job transmit request info.* struct ezdp_2step_1588_header two_step
- *Job dispatch request info.* } u
- };

Detailed Description

1588 header format definition

Field Documentation

unsigned ezdp_1588_header::__pad0__

Reserved bits 25 to 31.

unsigned ezdp_1588_header::__pad1__

Determine the action to process for an expanded job

Reserved bits 0 to 23

struct ezdp_1step_1588_header ezdp_1588_header::one_step `[read]`

Job transmit request info.

struct ezdp_2step_1588_header ezdp_1588_header::two_step `[read]`

Job dispatch request info.

union { ... } ezdp_1588_header::u

union { ... }

The documentation for this struct was generated from the following file:

- dpe/dp/include/ezdp_frame_defs.h

ezdp_1step_1588_header Struct Reference

1-step 1588 header format definition

Data Fields

- union {
- uint32_t raw_data [EZDP_1STEP_1588_HEADER_WORD_COUNT]
- struct {
- unsigned __pad0__: EZDP_1STEP_1588_HEADER_RESERVED28_31_SIZE
- *Reserved bits 28 to 31.* unsigned correction_odd_start: EZDP_1STEP_1588_HEADER_CORRECTION_ODD_START_SIZE
- *Correction field start from odd byte.* unsigned wrap_around_condition: EZDP_1STEP_1588_HEADER_WRAP_AROUND_CONDITION_SIZE
- *Wrap-around condition detected.* unsigned inject_checksum_flag: EZDP_1STEP_1588_HEADER_INJECT_CHECKSUM_FLAG_SIZE
- *Checksum injection required.* unsigned __pad1__: EZDP_1STEP_1588_HEADER_RESERVED24_SIZE
- *Reserved bit 24.* unsigned __pad2__: EZDP_1STEP_1588_HEADER_RESERVED16_23_SIZE
- *Reserved bits 16 to 23.* uint16_t checksum
- *Intermediate/Partial checksum field.* uint16_t checksum_offset
- *Offset of checksum field inside PTP frame.* uint16_t correction_offset
- *Offset of correction field inside the PTP frame.* uint64_t correction
- *Intermediate/Partial correction field.* }
- };

Detailed Description

1-step 1588 header format definition

Field Documentation

uint32_t ezdp_1step_1588_header::raw_data[EZDP_1STEP_1588_HEADER_WORD_COUNT]

unsigned ezdp_1step_1588_header::__pad0__

Reserved bits 28 to 31.

unsigned ezdp_1step_1588_header::correction_odd_start

Correction field start from odd byte.

unsigned ezdp_1step_1588_header::wrap_around_condition

Wrap-around condition detected.

unsigned ezdp_1step_1588_header::inject_checksum_flag

Checksum injection required.

unsigned ezdp_1step_1588_header::__pad1__

Reserved bit 24.

unsigned ezdp_1step_1588_header::__pad2__

Reserved bits 16 to 23.

uint16_t ezdp_1step_1588_header::checksum

Intermediate/Partial checksum field.

uint16_t ezdp_1step_1588_header::checksum_offset

Offset of checksum field inside PTP frame.

uint16_t ezdp_1step_1588_header::correction_offset

Offset of correction field inside the PTP frame.

uint64_t ezdp_1step_1588_header::correction

Intermediate/Partial correction field.

union { ... }

The documentation for this struct was generated from the following file:

- dpe/dp/include/ezdp_frame_defs.h

ezdp_2step_1588_header Struct Reference

2-step 1588 header format definition

Data Fields

- union {
- uint32_t raw_data [EZDP_2STEP_1588_HEADER_WORD_COUNT]
- struct {
- unsigned __pad0__: EZDP_2STEP_1588_HEADER_RESERVED24_31_SIZE
- *Reserved bits 24 to 31.* unsigned __pad1__: EZDP_2STEP_1588_HEADER_RESERVED24_SIZE
- *Reserved for TS mode flag.* unsigned __pad2__: EZDP_2STEP_1588_HEADER_RESERVED0_23_SIZE
- *Reserved bits 0 to 23.* unsigned __pad3__: EZDP_2STEP_1588_HEADER_RESERVED32_63_SIZE
- *Reserved bits 32 to 63.* uint8_t free_bytes
- *This field indicates how many free bytes are left at the end of the frame header data buffer.* uint8_t header_offset
- *This is the frame header starting point in the first frame data buffer.* unsigned __pad4__: EZDP_2STEP_1588_HEADER_RESERVED76_77_SIZE
- *Reserved bits 76 to 77.* unsigned class_of_service: EZDP_2STEP_1588_HEADER_CLASS_OF_SERVICE_SIZE
- *Frame class of service grade.* unsigned __pad5__: EZDP_2STEP_1588_HEADER_RESERVED74_75_SIZE
- *Reserved bits 74 to 75.* unsigned buf_budget_id: EZDP_2STEP_1588_HEADER_BUF_BUDGET_ID_SIZE
- *Budget group ID.* struct ezdp_buffer_desc buf_desc
- *Pointer to a 256B buffer located either in IMEM or EMEM.* }
- };

Detailed Description

2-step 1588 header format definition

Field Documentation

uint32_t ezdp_2step_1588_header::raw_data[EZDP_2STEP_1588_HEADER_WORD_COUNT]

unsigned ezdp_2step_1588_header::__pad0__

Reserved bits 24 to 31.

unsigned ezdp_2step_1588_header::__pad1__

Reserved for TS mode flag.

unsigned ezdp_2step_1588_header::__pad2__

Reserved bits 0 to 23.

unsigned ezdp_2step_1588_header::__pad3__

Reserved bits 32 to 63.

uint8_t ezdp_2step_1588_header::free_bytes

This field indicates how many free bytes are left at the end of the frame header data buffer.

This field is valid only for STANDARD frames.

uint8_t ezdp_2step_1588_header::header_offset

This is the frame header starting point in the first frame data buffer.

To include embedded LBD in the same buffer of the first frame data (EMBEDDED_BD frame type), the header_offset must be at least the max configured embedded LBD size (16B or 32B). The gaps from beginning of the buffer or from max embedded LBD size to header_offset can be utilized by frame context or can be used for optimized header modification (increasing or decreasing).

unsigned ezdp_2step_1588_header::__pad4__

Reserved bits 76 to 77.

unsigned ezdp_2step_1588_header::class_of_service

Frame class of service grade.

unsigned ezdp_2step_1588_header::__pad5__

Reserved bits 74 to 75.

unsigned ezdp_2step_1588_header::buf_budget_id

Budget group ID.

Budget identifies an allocated IMEM or EMEM buffer resource control operation. By default a budget group ID is associated with an Rx port ID from which the frame was received.

struct ezdp_buffer_desc ezdp_2step_1588_header::buf_desc `[read]`

Pointer to a 256B buffer located either in IMEM or EMEM.

The type to which the buffer points is determined by frame type.

union { ... }

The documentation for this struct was generated from the following file:

- dpe/dp/include/ezdp_frame_defs.h

ezdp_app_schlr_status Struct Reference

PMU application scheduler status (based on PMU system info).

Data Fields

- union {
- ezdp_app_schlr_status_t raw_data
- struct {
- unsigned enable: EZDP_APP_SCHLR_STATUS_ENABLE_SIZE
- *The application scheduler is enabled.* unsigned busy: EZDP_APP_SCHLR_STATUS_BUSY_SIZE
- *The application scheduler has pending requests (busy).* unsigned __pad0__: EZDP_APP_SCHLR_STATUS_RESERVED13_SIZE
- *Reserved bit 13.* unsigned dispatched_job: EZDP_APP_SCHLR_STATUS_DISPATCHED_JOB_SIZE
- *The number of jobs dispatched from the application scheduler.* }
- };

Detailed Description

PMU application scheduler status (based on PMU system info).
There are 8 application schedulers in each PMU side.

Field Documentation

ezdp_app_schlr_status_t ezdp_app_schlr_status::raw_data

unsigned ezdp_app_schlr_status::enable

The application scheduler is enabled.

unsigned ezdp_app_schlr_status::busy

The application scheduler has pending requests (busy).

unsigned ezdp_app_schlr_status::__pad0__

Reserved bit 13.

unsigned ezdp_app_schlr_status::dispatched_job

The number of jobs dispatched from the application scheduler.

Defined as number of jobs that were dispatched from the application scheduler for processing and are waiting for "job done".

union { ... }

The documentation for this struct was generated from the following file:

- dpe/dp/include/ezdp_job_defs.h

ezdp_bitwise_ctr_cfg Struct Reference

On-demand bitwise counter configuration definition.

Data Fields

- union {
- uint32_t raw_data [EZDP_BITWISE_CTR_CFG_WORD_COUNT]
- struct {
- unsigned __pad0__: EZDP_BITWISE_CTR_CFG_ECC_SIZE
- *ECC.* unsigned __pad1__: EZDP_BITWISE_CTR_CFG_SUB_TYPE_SIZE
- *Counter sub type (bitwise=9).* unsigned __pad2__: EZDP_BITWISE_CTR_CFG_RESERVED0_18_SIZE
- *Reserved bits 0 to 18.* unsigned __pad3__: EZDP_BITWISE_CTR_CFG_RESERVED32_63_SIZE
- *Reserved bits 32 to 63.* uint64_t data
- *Counter data value.* }
- };

Detailed Description

On-demand bitwise counter configuration definition.

Field Documentation

uint32_t ezdp_bitwise_ctr_cfg::raw_data[EZDP_BITWISE_CTR_CFG_WORD_COUNT]

unsigned ezdp_bitwise_ctr_cfg::__pad0__

ECC.

unsigned ezdp_bitwise_ctr_cfg::__pad1__

Counter sub type (bitwise=9).

unsigned ezdp_bitwise_ctr_cfg::__pad2__

Reserved bits 0 to 18.

unsigned ezdp_bitwise_ctr_cfg::__pad3__

Reserved bits 32 to 63.

uint64_t ezdp_bitwise_ctr_cfg::data

Counter data value.

union { ... }

The documentation for this struct was generated from the following file:

- dpe/dp/include/ezdp_counter_defs.h

ezdp_buffer_desc Struct Reference

Buffer descriptor (BD) data structure.

Data Fields

- union {
- ezdp_buffer_desc_t raw_data
- struct {
- unsigned valid_data_buf: EZDP_BUFFER_DESC_VALID_DATA_BUF_SIZE
- *The data buffer is valid.* unsigned __pad0__: EZDP_BUFFER_DESC_RESERVED28_29_SIZE
- *< Indicate whether index points to buffer in IMEM or EMEM address space.* unsigned id: EZDP_BUFFER_DESC_ID_SIZE
- *This ID provides an index pointer to a 256B buffer.* }
- };

Detailed Description

Buffer descriptor (BD) data structure.

Field Documentation

ezdp_buffer_desc_t ezdp_buffer_desc::raw_data

unsigned ezdp_buffer_desc::valid_data_buf

The data buffer is valid.

NOTE: FD with non valid data buffers can't be send to tm or interface or discarded.

unsigned ezdp_buffer_desc::__pad0__

Indicate whether index points to buffer in IMEM or EMEM address space.

Reserved bits 28 to 29

unsigned ezdp_buffer_desc::id

This ID provides an index pointer to a 256B buffer.

The buffer is located either in IMEM or in EMEM address space and it is interpreted according to mem_type.

union { ... }

The documentation for this struct was generated from the following file:

- dpe/dp/include/ezdp_frame_defs.h

ezdp_buffer_info Struct Reference

Buffer descriptor info.

Data Fields

- union {
- uint8_t free_bytes
- };

Detailed Description

Buffer descriptor info.

Field Documentation

uint8_t ezdp_buffer_info::free_bytes

union { ... }

The documentation for this struct was generated from the following file:

- dpe/dp/include/ezdp_frame_defs.h

ezdp_congestion_status Struct Reference

System priority drop congestion status.

Data Fields

- union {
- ezdp_congestion_status_t raw_data
- struct {
- unsigned __pad0__: EZDP_CONGESTION_STATUS_RESERVED14_15_SIZE
- *Reserved bits 14-15.* unsigned __pad1__: EZDP_CONGESTION_STATUS_RESERVED11_SIZE
- *< Indicate the port RX congestion level received from RxIF.* unsigned job_guarantee: EZDP_CONGESTION_STATUS_JOB_GUARANTEE_SIZE
- *Indicate that job are still guarantee level.* unsigned __pad2__: EZDP_CONGESTION_STATUS_RESERVED7_SIZE
- *< Indicate the job budget congestion level received from FCU.* unsigned emem_buf_guarantee: EZDP_CONGESTION_STATUS_EMEM_BUF_GUARANTEE_SIZE
- *Indicate that EMEM buffers are still guarantee level.* unsigned __pad3__: EZDP_CONGESTION_STATUS_RESERVED3_SIZE
- *< Indicate the EMEM buffer budget congestion level received from FCU.* unsigned imem_buf_guarantee: EZDP_CONGESTION_STATUS_IMEM_BUF_GUARANTEE_SIZE
- *Indicate that IMEM buffers are still guarantee level.* }
- };

Detailed Description

System priority drop congestion status.

Field Documentation

ezdp_congestion_status_t ezdp_congestion_status::raw_data

unsigned ezdp_congestion_status::__pad0__

Reserved bits 14-15.

unsigned ezdp_congestion_status::__pad1__

< Indicate the port RX congestion level received from RxIF.

Not applicable to channel or group. Applicable only for physical ports. Reserved bit 11.

unsigned ezdp_congestion_status::job_guarantee

Indicate that job are still guarantee level.

unsigned ezdp_congestion_status::__pad2__

< Indicate the job budget congestion level received from FCU.

Reserved bit 7.

unsigned ezdp_congestion_status::emem_buf_guarantee

Indicate that EMEM buffers are still guarantee level.

unsigned ezdp_congestion_status::__pad3__

< Indicate the EMEM buffer budget congestion level received from FCU.

Reserved bit 3.

unsigned ezdp_congestion_status::imem_buf_guarantee

Indicate that IMEM buffers are still guarantee level.

union { ... }

The documentation for this struct was generated from the following file:

- dpe/dp/include/ezdp_job_defs.h

ezdp_ctr_msg Struct Reference

Counter message queue definition.

Data Fields

- union {
- uint32_t raw_data [EZDP_CTR_MSG_WORD_COUNT]
- struct {
- unsigned __pad0__: EZDP_CTR_MSG_ECC_SIZE
- *ECC.* unsigned __pad1__: EZDP_CTR_MSG_RESERVED8_23_SIZE
- *reserved bits 8-23* unsigned overrun_error_condition: EZDP_CTR_MSG_OVERRUN_ERROR_CONDITION_SIZE
- *Queue was overrun and old messages are lost.* unsigned __pad2__: EZDP_CTR_MSG_RESERVED6_SIZE
- *reserved bit 6* unsigned overflow: EZDP_CTR_MSG_OVERFLOW_SIZE
- *Counter overflow.* struct ezdp_sum_addr sum_addr
- *< Counter message type* union {
- uint64_t single_ctr_value
- *On-demand counter value.* struct ezdp_dual_ctr dual_ctr_value
- *On-demand dual counter value.* }
- }
- };

Detailed Description

Counter message queue definition.

Field Documentation

uint32_t ezdp_ctr_msg::raw_data[EZDP_CTR_MSG_WORD_COUNT]

unsigned ezdp_ctr_msg::__pad0__

ECC.

unsigned ezdp_ctr_msg::__pad1__

reserved bits 8-23

unsigned ezdp_ctr_msg::overrun_error_condition

Queue was overrun and old messages are lost.

unsigned ezdp_ctr_msg::__pad2__

reserved bit 6

unsigned ezdp_ctr_msg::overflow

Counter overflow.

The counter has lost coherency

struct ezdp_sum_addr ezdp_ctr_msg::sum_addr [read]

< Counter message type

< Counter message type counter summarize address

uint64_t ezdp_ctr_msg::single_ctr_value

On-demand counter value.

struct ezdp_dual_ctr ezdp_ctr_msg::dual_ctr_value [read]

On-demand dual counter value.

union { ... }

The documentation for this struct was generated from the following file:

- dpe/dp/include/ezdp_counter_defs.h

ezdp_decode_eth_type_retval Struct Reference

Decode ip protocol return value struct definition.

Data Fields

- union {
- ezdp_decode_eth_type_retval_t raw_data
- struct {
- unsigned __pad0__: EZDP_DECODE_ETH_TYPE_RETVAL_RESERVED13_31_SIZE
- *Reserved bits 13 to 31.* unsigned other: EZDP_DECODE_ETH_TYPE_RETVAL_OTHER_SIZE
- *Ethernet type is not one of the decoded types.* unsigned pppoe_discovery: EZDP_DECODE_ETH_TYPE_RETVAL_PPPOE_DISCOVERY_SIZE
- *Ethernet type is PPPoE Discovery Stage, value 0x8863.* unsigned pppoe_session: EZDP_DECODE_ETH_TYPE_RETVAL_PPPOE_SESSION_SIZE
- *Ethernet type is PPPoE Session Stage, value 0x8864.* unsigned user_def1: EZDP_DECODE_ETH_TYPE_RETVAL_USER_DEF1_SIZE
- *Ethernet type is equal to user defined value 1.* unsigned user_def0: EZDP_DECODE_ETH_TYPE_RETVAL_USER_DEF0_SIZE
- *Ethernet type is equal to user defined value 0.* unsigned length: EZDP_DECODE_ETH_TYPE_RETVAL_LENGTH_SIZE
- *Ethernet type is less than or equal to 0x0600 and indicates that this field is length.* unsigned ipv6: EZDP_DECODE_ETH_TYPE_RETVAL_IPV6_SIZE
- *Ethernet type is Internet Protocol, Version 6 (IPv6), value 0x86dd.* unsigned mpls_multicast: EZDP_DECODE_ETH_TYPE_RETVAL_MPLS_MULTICAST_SIZE
- *Ethernet type is MPLS multicast, value 0x8848.* unsigned mpls_unicast: EZDP_DECODE_ETH_TYPE_RETVAL_MPLS_UNICAST_SIZE
- *Ethernet type is MPLS unicast, value 0x8847.* unsigned arp: EZDP_DECODE_ETH_TYPE_RETVAL_ARP_SIZE
- *Ethernet type is Address Resolution Protocol (ARP), value 0x0806.* unsigned eth_88a8: EZDP_DECODE_ETH_TYPE_RETVAL_ETH_88A8_SIZE
- *Ethernet type is Provider Bridging (IEEE 802.1ad) and Shortest Path Bridging IEEE 802.1aq, value 0x88a8.* unsigned eth_8100: EZDP_DECODE_ETH_TYPE_RETVAL_ETH_8100_SIZE
- *Ethernet type is VLAN-tagged frame (IEEE 802.1Q) and Shortest Path Bridging IEEE 802.1aq, value 0x8100.* unsigned ipv4: EZDP_DECODE_ETH_TYPE_RETVAL_IPV4_SIZE
- *Ethernet type is Internet Protocol Version 4 (IPv4), value 0x0800.* }
- };

Detailed Description

Decode ip protocol return value struct definition.

Field Documentation

ezdp_decode_eth_type_retval_t ezdp_decode_eth_type_retval::raw_data

unsigned ezdp_decode_eth_type_retval::__pad0__

Reserved bits 13 to 31.

unsigned ezdp_decode_eth_type_retval::other

Ethernet type is not one of the decoded types.

unsigned ezdp_decode_eth_type_retval::pppoe_discovery

Ethernet type is PPPoE Discovery Stage, value 0x8863.

unsigned ezdp_decode_eth_type_retval::pppoe_session

Ethernet type is PPPoE Session Stage, value 0x8864.

unsigned ezdp_decode_eth_type_retval::user_def1

Ethernet type is equal to user defined value 1.

unsigned ezdp_decode_eth_type_retval::user_def0

Ethernet type is equal to user defined value 0.

unsigned ezdp_decode_eth_type_retval::length

Ethernet type is less than or equal to 0x0600 and indicates that this field is length.

unsigned ezdp_decode_eth_type_retval::ipv6

Ethernet type is Internet Protocol, Version 6 (IPv6), value 0x86dd.

unsigned ezdp_decode_eth_type_retval::mpls_multicast

Ethernet type is MPLS multicast, value 0x8848.

unsigned ezdp_decode_eth_type_retval::mpls_unicast

Ethernet type is MPLS unicast, value 0x8847.

unsigned ezdp_decode_eth_type_retval::arp

Ethernet type is Address Resolution Protocol (ARP), value 0x0806.

unsigned ezdp_decode_eth_type_retval::eth_88a8

Ethernet type is Provider Bridging (IEEE 802.1ad) and Shortest Path Bridging IEEE 802.1aq, value 0x88a8.

unsigned ezdp_decode_eth_type_retval::eth_8100

Ethernet type is VLAN-tagged frame (IEEE 802.1Q) and Shortest Path Bridging IEEE 802.1aq, value 0x8100.

unsigned ezdp_decode_eth_type_retval::ipv4

Ethernet type is Internet Protocol Version 4 (IPv4), value 0x0800.

union { ... }

The documentation for this struct was generated from the following file:

- dpe/dp/include/ezdp_decode_defs.h

ezdp_decode_ip_next_protocol Struct Reference

IP protocol type flags.

Data Fields

- union {
- ezdp_decode_ip_next_protocol_t raw_data
- struct {
- unsigned other: EZDP_DECODE_IP_NEXT_PROTOCOL_OTHER_SIZE
- *Protocol is not one of the decode types.* unsigned icmp_igmp: EZDP_DECODE_IP_NEXT_PROTOCOL_ICMP_IGMP_SIZE
- *Control frame - ICMP/IGMP.* unsigned ipv6: EZDP_DECODE_IP_NEXT_PROTOCOL_IPV6_SIZE
- *IPv6 protocol.* unsigned ipv4: EZDP_DECODE_IP_NEXT_PROTOCOL_IPV4_SIZE
- *IPv4 protocol.* unsigned gre: EZDP_DECODE_IP_NEXT_PROTOCOL_GRE_SIZE
- *GRE protocol.* unsigned mpls: EZDP_DECODE_IP_NEXT_PROTOCOL_MPLS_SIZE
- *MPLS protocol.* unsigned udp: EZDP_DECODE_IP_NEXT_PROTOCOL_UDP_SIZE
- *UDP protocol.* unsigned tcp: EZDP_DECODE_IP_NEXT_PROTOCOL_TCP_SIZE
- *TCP protocol.* }
- };

Detailed Description

IP protocol type flags.

Field Documentation

ezdp_decode_ip_next_protocol_t ezdp_decode_ip_next_protocol::raw_data

unsigned ezdp_decode_ip_next_protocol::other

Protocol is not one of the decode types.

unsigned ezdp_decode_ip_next_protocol::icmp_igmp

Control frame - ICMP/IGMP.

unsigned ezdp_decode_ip_next_protocol::ipv6

IPv6 protocol.

unsigned ezdp_decode_ip_next_protocol::ipv4

IPv4 protocol.

unsigned ezdp_decode_ip_next_protocol::gre

GRE protocol.

unsigned ezdp_decode_ip_next_protocol::mpls

MPLS protocol.

unsigned ezdp_decode_ip_next_protocol::udp

UDP protocol.

unsigned ezdp_decode_ip_next_protocol::tcp

TCP protocol.

union { ... }

The documentation for this struct was generated from the following file:

- dpe/dp/include/ezdp_decode_defs.h

ezdp_decode_ip_protocol_retval Struct Reference

Decode ip protocol return value struct definition.

Data Fields

- union {
- ezdp_decode_ip_protocol_retval_t raw_data
- struct {
- unsigned __pad0__: EZDP_DECODE_IP_PROTOCOL_RETVAL_RESERVED14_31_SIZE
- *Reserved bits 14 to 31.* unsigned other: EZDP_DECODE_IP_PROTOCOL_RETVAL_OTHER_SIZE
- *Protocol is other.* unsigned ah_prot: EZDP_DECODE_IP_PROTOCOL_RETVAL_AH_PROT_SIZE
- *AH (Authentication Header) protocol.* unsigned esp_prot: EZDP_DECODE_IP_PROTOCOL_RETVAL_ESP_PROT_SIZE
- *ESP (Encapsulating Security Payload) protocol.* unsigned def_ip_prot_3: EZDP_DECODE_IP_PROTOCOL_RETVAL_DEF_IP_PROT_3_SIZE
- *User defined protocol 3.* unsigned def_ip_prot_2: EZDP_DECODE_IP_PROTOCOL_RETVAL_DEF_IP_PROT_2_SIZE
- *User defined protocol 2.* unsigned def_ip_prot_1: EZDP_DECODE_IP_PROTOCOL_RETVAL_DEF_IP_PROT_1_SIZE
- *User defined protocol 1.* unsigned def_ip_prot_0: EZDP_DECODE_IP_PROTOCOL_RETVAL_DEF_IP_PROT_0_SIZE
- *User defined protocol 0.* unsigned icmp_igmp: EZDP_DECODE_IP_PROTOCOL_RETVAL_ICMP_IGMP_SIZE
- *Control frame - ICMP/IGMP.* unsigned ipv6: EZDP_DECODE_IP_PROTOCOL_RETVAL_IPV6_SIZE
- *Protocol is IPv6.* unsigned ipv4: EZDP_DECODE_IP_PROTOCOL_RETVAL_IPV4_SIZE
- *Protocol is IPv4.* unsigned gre: EZDP_DECODE_IP_PROTOCOL_RETVAL_GRE_SIZE
- *Protocol is GRE.* unsigned mpls: EZDP_DECODE_IP_PROTOCOL_RETVAL_MPLS_SIZE
- *Protocol is MPLS.* unsigned udp: EZDP_DECODE_IP_PROTOCOL_RETVAL_UDP_SIZE
- *Protocol is UDP.* unsigned tcp: EZDP_DECODE_IP_PROTOCOL_RETVAL_TCP_SIZE
- *Protocol is TCP.* }
- };

Detailed Description

Decode ip protocol return value struct definition.

Field Documentation

ezdp_decode_ip_protocol_retval_t ezdp_decode_ip_protocol_retval::raw_data

unsigned ezdp_decode_ip_protocol_retval::__pad0__

Reserved bits 14 to 31.

unsigned ezdp_decode_ip_protocol_retval::other

Protocol is other.

unsigned ezdp_decode_ip_protocol_retval::ah_prot

AH (Authentication Header) protocol.

unsigned ezdp_decode_ip_protocol_retval::esp_prot

ESP (Encapsulating Security Payload) protocol.

unsigned ezdp_decode_ip_protocol_retval::def_ip_prot_3

User defined protocol 3.

unsigned ezdp_decode_ip_protocol_retval::def_ip_prot_2

User defined protocol 2.

unsigned ezdp_decode_ip_protocol_retval::def_ip_prot_1

User defined protocol 1.

unsigned ezdp_decode_ip_protocol_retval::def_ip_prot_0

User defined protocol 0.

unsigned ezdp_decode_ip_protocol_retval::icmp_igmp

Control frame - ICMP/IGMP.

unsigned ezdp_decode_ip_protocol_retval::ipv6

Protocol is IPv6.

unsigned ezdp_decode_ip_protocol_retval::ipv4

Protocol is IPv4.

unsigned ezdp_decode_ip_protocol_retval::gre

Protocol is GRE.

unsigned ezdp_decode_ip_protocol_retval::mpls

Protocol is MPLS.

unsigned ezdp_decode_ip_protocol_retval::udp

Protocol is UDP.

unsigned ezdp_decode_ip_protocol_retval::tcp

Protocol is TCP.

union { ... }

The documentation for this struct was generated from the following file:

- dpe/dp/include/ezdp_decode_defs.h

ezdp_decode_ipv4_control Struct Reference

IPv4 addresses decoding result.

Data Fields

- union {
- ezdp_decode_ipv4_control_t raw_data
- struct {
- unsigned user_config2: EZDP_DECODE_IPV4_CONTROL_USER_CONFIG2_SIZE
- *Destination IP is equal to user configured DIP 2 masked with user configured DIP mask 2.* unsigned user_config1: EZDP_DECODE_IPV4_CONTROL_USER_CONFIG1_SIZE
- *Destination IP is equal to user configured DIP 1 masked with user configured DIP mask 1.* unsigned user_config0: EZDP_DECODE_IPV4_CONTROL_USER_CONFIG0_SIZE
- *Destination IP is equal to user configured DIP 0 masked with user configured DIP mask 0.* unsigned igmp: EZDP_DECODE_IPV4_CONTROL_IGMP_SIZE
- *Destination IP is IGMP.* unsigned icmp: EZDP_DECODE_IPV4_CONTROL_ICMP_SIZE
- *Destination IP is ICMP.* unsigned __pad0__: EZDP_DECODE_IPV4_CONTROL_RESERVED_2_SIZE
- *Reserved bit 2.* unsigned internetwork_multicast_range: EZDP_DECODE_IPV4_CONTROL_INTERNETWORK_MULTICAST_RANGE_SIZE
- *Destination IP is 0xE0-00-01-xx.* unsigned link_local_multicast_range: EZDP_DECODE_IPV4_CONTROL_LINK_LOCAL_MULTICAST_RANGE_SIZE
- *Destination IP is 0xE0-00-00-xx.* }
- };

Detailed Description

IPv4 addresses decoding result.

Field Documentation

ezdp_decode_ipv4_control_t ezdp_decode_ipv4_control::raw_data

unsigned ezdp_decode_ipv4_control::user_config2

Destination IP is equal to user configured DIP 2 masked with user configured DIP mask 2.

unsigned ezdp_decode_ipv4_control::user_config1

Destination IP is equal to user configured DIP 1 masked with user configured DIP mask 1.

unsigned ezdp_decode_ipv4_control::user_config0

Destination IP is equal to user configured DIP 0 masked with user configured DIP mask 0.

unsigned ezdp_decode_ipv4_control::igmp

Destination IP is IGMP.

unsigned ezdp_decode_ipv4_control::icmp

Destination IP is ICMP.

unsigned ezdp_decode_ipv4_control::__pad0__

Reserved bit 2.

unsigned ezdp_decode_ipv4_control::internetwork_multicast_range

Destination IP is 0xE0-00-01-xx.

unsigned ezdp_decode_ipv4_control::link_local_multicast_range

Destination IP is 0xE0-00-00-xx.

union { ... }

The documentation for this struct was generated from the following file:

- dpe/dp/include/ezdp_decode_defs.h

ezdp_decode_ipv4_errors Struct Reference

IPv4 header decode error flags.

Data Fields

- union {
- ezdp_decode_ipv4_errors_t raw_data
- struct {
- unsigned __pad0__: EZDP_DECODE_IPV4_ERRORS_RESERVED9_15_SIZE
- *Reserved bits 9 to 15.* unsigned decode_error: EZDP_DECODE_IPV4_ERRORS_DECODE_ERROR_SIZE
- *Aggregated decode error flag.* unsigned sip_equal_dip: EZDP_DECODE_IPV4_ERRORS_SIP_EQUAL_DIP_SIZE
- *Destination IP equals source IP.* unsigned checksum_error: EZDP_DECODE_IPV4_ERRORS_CHECKSUM_ERROR_SIZE
- *Header checksum error.* unsigned not_ipv4_version: EZDP_DECODE_IPV4_ERRORS_NOT_IPV4_VERSION_SIZE
- *Incorrect version.* unsigned header_length_gt_frame_length: EZDP_DECODE_IPV4_ERRORS_HEADER_LENGTH_GT_FRAME_LENGTH_SIZE
- *IPv4 header length field value is greater than frame length.* unsigned total_length_gt_frame_length: EZDP_DECODE_IPV4_ERRORS_TOTAL_LENGTH_GT_FRAME_LENGTH_SIZE
- *IPv4 header total length field value is greater than frame length.* unsigned header_length_lt_5: EZDP_DECODE_IPV4_ERRORS_HEADER_LENGTH_LT_5_SIZE
- *IPv4 header length is less than 5.* unsigned sip_is_zero: EZDP_DECODE_IPV4_ERRORS_SIP_IS_ZERO_SIZE
- *Source IP is zero.* unsigned sip_is_multicast: EZDP_DECODE_IPV4_ERRORS_SIP_IS_MULTICAST_SIZE
- *Source IP is multicast (SIP is greater than 0xE0-00-00-00).* }
- };

Detailed Description

IPv4 header decode error flags.

Field Documentation

ezdp_decode_ipv4_errors_t ezdp_decode_ipv4_errors::raw_data

unsigned ezdp_decode_ipv4_errors::__pad0__

Reserved bits 9 to 15.

unsigned ezdp_decode_ipv4_errors::decode_error

Aggregated decode error flag.

Indicate at least one decode error

unsigned ezdp_decode_ipv4_errors::sip_equal_dip

Destination IP equals source IP.

unsigned ezdp_decode_ipv4_errors::checksum_error

Header checksum error.

unsigned ezdp_decode_ipv4_errors::not_ipv4_version

Incorrect version.

unsigned ezdp_decode_ipv4_errors::header_length_gt_frame_length

IPv4 header length field value is greater than frame length.

unsigned ezdp_decode_ipv4_errors::total_length_gt_frame_length

IPv4 header total length field value is greater than frame length.

unsigned ezdp_decode_ipv4_errors::header_length_lt_5

IPv4 header length is less than 5.

unsigned ezdp_decode_ipv4_errors::sip_is_zero

Source IP is zero.

unsigned ezdp_decode_ipv4_errors::sip_is_multicast

Source IP is multicast (SIP is greater than 0xE0-00-00-00).

union { ... }

The documentation for this struct was generated from the following file:

- dpe/dp/include/ezdp_decode_defs.h

ezdp_decode_ipv4_result Struct Reference

Decode IPv4 result.

Data Fields

- union {
- uint32_t raw_data [EZDP_DECODE_IPV4_RESULT_WORD_COUNT]
- struct {
- struct ezdp_decode_ipv4_control control
- *Decode control.* struct ezdp_decode_ipv4_errors error_codes
- *Decode error codes.* unsigned first_fragment: EZDP_DECODE_IPV4_RESULT_FIRST_FRAGMENT_SIZE
- *First fragment flag.* unsigned __pad0__: EZDP_DECODE_IPV4_RESULT_RESERVED_2_6_SIZE
- *Reserved bits 2 to 6.* unsigned user_config_sip: EZDP_DECODE_IPV4_RESULT_USER_CONFIG_SIP_SIZE
- *Source IP (SIP) is equal to user configured SIP masked with user configured SIP mask.* unsigned option_exist: EZDP_DECODE_IPV4_RESULT_OPTION_EXIST_SIZE
- *IPv4 header length is greater than 5 - options exist.* unsigned __pad1__: EZDP_DECODE_IPV4_RESULT_RESERVED_56_63_SIZE
- *Reserved bits 56 to 63.* struct ezdp_decode_ip_next_protocol next_protocol
- *Next protocol struct.* uint16_t sip_dip_hash
- *sip+dip hash* }
- };

Detailed Description

Decode IPv4 result.

Field Documentation

uint32_t ezdp_decode_ipv4_result::raw_data[EZDP_DECODE_IPV4_RESULT_WORD_COUNT]

struct ezdp_decode_ipv4_control ezdp_decode_ipv4_result::control [read]

Decode control.

struct ezdp_decode_ipv4_errors ezdp_decode_ipv4_result::error_codes [read]

Decode error codes.

unsigned ezdp_decode_ipv4_result::first_fragment

First fragment flag.

unsigned ezdp_decode_ipv4_result::__pad0__

Reserved bits 2 to 6.

unsigned ezdp_decode_ipv4_result::user_config_sip

Source IP (SIP) is equal to user configured SIP masked with user configured SIP mask.

unsigned ezdp_decode_ipv4_result::option_exist

IPv4 header length is greater than 5 - options exist.

unsigned ezdp_decode_ipv4_result::__pad1__

Reserved bits 56 to 63.

struct ezdp_decode_ip_next_protocol ezdp_decode_ipv4_result::next_protocol `[read]`

Next protocol struct.

uint16_t ezdp_decode_ipv4_result::sip_dip_hash

sip+dip hash

union { ... }

The documentation for this struct was generated from the following file:

- dpe/dp/include/ezdp_decode_defs.h

ezdp_decode_ipv4_retval Struct Reference

Decode IPv4 return value struct definition.

Data Fields

- union {
- ezdp_decode_ipv4_retval_t raw_data
- struct {
- struct ezdp_decode_ipv4_control control
- *Decode control.* struct ezdp_decode_ipv4_errors error_codes
- *Decode error codes.* unsigned first_fragment: EZDP_DECODE_IPV4_RETVAL_FIRST_FRAGMENT_SIZE
- *First fragment flag.* unsigned __pad0__: EZDP_DECODE_IPV4_RETVAL_RESERVED_2_6_SIZE
- *Reserved bits 2 to 6.* unsigned user_config_sip: EZDP_DECODE_IPV4_RETVAL_USER_CONFIG_SIP_SIZE
- *Source IP (SIP) is equal to user configured SIP masked with user configured SIP mask.* unsigned option_exist: EZDP_DECODE_IPV4_RETVAL_OPTION_EXIST_SIZE
- *IPv4 header length is greater than 5 - options exist.* }
- };

Detailed Description

Decode IPv4 return value struct definition.

Field Documentation

ezdp_decode_ipv4_retval_t ezdp_decode_ipv4_retval::raw_data

struct ezdp_decode_ipv4_control ezdp_decode_ipv4_retval::control [read]

Decode control.

struct ezdp_decode_ipv4_errors ezdp_decode_ipv4_retval::error_codes [read]

Decode error codes.

unsigned ezdp_decode_ipv4_retval::first_fragment

First fragment flag.

unsigned ezdp_decode_ipv4_retval::__pad0__

Reserved bits 2 to 6.

unsigned ezdp_decode_ipv4_retval::user_config_sip

Source IP (SIP) is equal to user configured SIP masked with user configured SIP mask.

unsigned ezdp_decode_ipv4_retval::option_exist

IPv4 header length is greater than 5 - options exist.

union { ... }

The documentation for this struct was generated from the following file:

- dpe/dp/include/ezdp_decode_defs.h

ezdp_decode_ipv6_control Struct Reference

IPv6 addresses decoding result.

Data Fields

- union {
- ezdp_decode_ipv6_control_t raw_data
- struct {
- unsigned __pad0__: EZDP_DECODE_IPV6_CONTROL_RESERVED7_8_SIZE
- *Reserved bits 7 to 8.* unsigned dip_is_wellknown_multicast: EZDP_DECODE_IPV6_CONTROL_DIP_IS_WELLKNOWN_MULTICAST_SIZE
- *Destination IP is FF*...dip.byte[1].bit[4]=0.* unsigned dip_is_multicast: EZDP_DECODE_IPV6_CONTROL_DIP_IS_MULTICAST_SIZE
- *Destination IP is FF*...dip.byte[1].bit[4]=1.* unsigned __pad1__: EZDP_DECODE_IPV6_CONTROL_RESERVED_3_SIZE
- *Reserved bit 3.* unsigned solicited_node_multicast_range: EZDP_DECODE_IPV6_CONTROL_SOLICITED_NODE_MULTICAST_RANGE_SIZE
- *Destination IP is ff02::01:ffxx:xxxx.* unsigned internetwork_multicast_range: EZDP_DECODE_IPV6_CONTROL_INTERNETWORK_MULTICAST_RANGE_SIZE
- *Destination IP is ff02::01xx.* unsigned link_local_multicast_range: EZDP_DECODE_IPV6_CONTROL_LINK_LOCAL_MULTICAST_RANGE_SIZE
- *Destination IP is ff02::xx.* }
- };

Detailed Description

IPv6 addresses decoding result.

Field Documentation

ezdp_decode_ipv6_control_t ezdp_decode_ipv6_control::raw_data

unsigned ezdp_decode_ipv6_control::__pad0__

Reserved bits 7 to 8.

unsigned ezdp_decode_ipv6_control::dip_is_wellknown_multicast

Destination IP is FF*...dip.byte[1].bit[4]=0.

unsigned ezdp_decode_ipv6_control::dip_is_multicast

Destination IP is FF*...dip.byte[1].bit[4]=1.

unsigned ezdp_decode_ipv6_control::__pad1__

Reserved bit 3.

unsigned ezdp_decode_ipv6_control::solicited_node_multicast_range

Destination IP is ff02::01:ffxx:xxxx.

unsigned ezdp_decode_ipv6_control::internetwork_multicast_range

Destination IP is ff02::01xx.

unsigned ezdp_decode_ipv6_control::link_local_multicast_range

Destination IP is ff02::xx.

union { ... }

The documentation for this struct was generated from the following file:

- dpe/dp/include/ezdp_decode_defs.h

ezdp_decode_ipv6_errors Struct Reference

IPv6 header decode error flags.

Data Fields

- union {
- ezdp_decode_ipv6_errors_t raw_data
- struct {
- unsigned __pad0__: EZDP_DECODE_IPV6_ERRORS_RESERVED10_15_SIZE
- *Reserved bits 10 to 15.* unsigned sip_is_multicast: EZDP_DECODE_IPV6_ERRORS_SIP_IS_MULTICAST_SIZE
- *SIP is multicast (first octet is 0xFF).* unsigned payload_missing: EZDP_DECODE_IPV6_ERRORS_PAYLOAD_MISSING_SIZE
- *Payload Length is zero when next header field is not hop-by-hop (0x00).* unsigned decode_error: EZDP_DECODE_IPV6_ERRORS_DECODE_ERROR_SIZE
- *Aggregated decode error flag.* unsigned sip_equal_dip: EZDP_DECODE_IPV6_ERRORS_SIP_EQUAL_DIP_SIZE
- *Destination IP equals source IP.* unsigned dip_is_one: EZDP_DECODE_IPV6_ERRORS_DIP_IS_ONE_SIZE
- *Destination IP is one.* unsigned dip_is_zero: EZDP_DECODE_IPV6_ERRORS_DIP_IS_ZERO_SIZE
- *Destination IP is zero.* unsigned sip_is_one: EZDP_DECODE_IPV6_ERRORS_SIP_IS_ONE_SIZE
- *Source IP is one.* unsigned sip_is_zero: EZDP_DECODE_IPV6_ERRORS_SIP_IS_ZERO_SIZE
- *Source IP is zero.* unsigned not_ipv6_version: EZDP_DECODE_IPV6_ERRORS_NOT_IPV6_VERSION_SIZE
- *Incorrect version.* unsigned payload_gt_frame_length: EZDP_DECODE_IPV6_ERRORS_PAYLOAD_GT_FRAME_LENGTH_SIZE
- *Payload length greater than frame length.* }
- };

Detailed Description

IPv6 header decode error flags.

Field Documentation

ezdp_decode_ipv6_errors_t ezdp_decode_ipv6_errors::raw_data

unsigned ezdp_decode_ipv6_errors::__pad0__

Reserved bits 10 to 15.

unsigned ezdp_decode_ipv6_errors::sip_is_multicast

SIP is multicast (first octet is 0xFF).

unsigned ezdp_decode_ipv6_errors::payload_missing

Payload Length is zero when next header field is not hop-by-hop (0x00).

unsigned ezdp_decode_ipv6_errors::decode_error

Aggregated decode error flag.

Indicate at least one decode error

unsigned ezdp_decode_ipv6_errors::sip_equal_dip

Destination IP equals source IP.

unsigned ezdp_decode_ipv6_errors::dip_is_one

Destination IP is one.

unsigned ezdp_decode_ipv6_errors::dip_is_zero

Destination IP is zero.

unsigned ezdp_decode_ipv6_errors::sip_is_one

Source IP is one.

unsigned ezdp_decode_ipv6_errors::sip_is_zero

Source IP is zero.

unsigned ezdp_decode_ipv6_errors::not_ipv6_version

Incorrect version.

unsigned ezdp_decode_ipv6_errors::payload_gt_frame_length

Payload length greater than frame length.

union { ... }

The documentation for this struct was generated from the following file:

- dpe/dp/include/ezdp_decode_defs.h

ezdp_decode_ipv6_result Struct Reference

Decode IPv6 result.

Data Fields

- union {
- uint32_t raw_data [EZDP_DECODE_IPV6_RESULT_WORD_COUNT]
- struct {
- struct ezdp_decode_ipv6_errors error_codes
- *Error codes.* unsigned __pad0__: EZDP_DECODE_IPV6_RESULT_RESERVED_15_SIZE
- *Reserved bit 15.* unsigned global_addresses: EZDP_DECODE_IPV6_RESULT_GLOBAL_ADDRESSES_SIZE
- *Both addresses scope are global.* unsigned site_local_address: EZDP_DECODE_IPV6_RESULT_SITE_LOCAL_ADDRESS_SIZE
- *One of addresses scope is site local.* unsigned link_local_address: EZDP_DECODE_IPV6_RESULT_LINK_LOCAL_ADDRESS_SIZE
- *One of addresses scope is link local.* unsigned __pad1__: EZDP_DECODE_IPV6_RESULT_RESERVED9_11_SIZE
- *Reserved bits 9 to 11.* unsigned options_exist: EZDP_DECODE_IPV6_RESULT_OPTIONS_EXIST_SIZE
- *Options exist.* struct ezdp_decode_ipv6_control control
- *Control struct.* unsigned __pad2__: EZDP_DECODE_IPV6_RESULT_RESERVED_56_63_SIZE
- *Reserved bits 56 to 63.* struct ezdp_decode_ip_next_protocol next_protocol
- *Next protocol struct.* uint16_t sip_dip_hash
- *sip+dip hash* }
- };

Detailed Description

Decode IPv6 result.

Field Documentation

uint32_t ezdp_decode_ipv6_result::raw_data[EZDP_DECODE_IPV6_RESULT_WORD_COUNT]

struct ezdp_decode_ipv6_errors ezdp_decode_ipv6_result::error_codes [read]

Error codes.

unsigned ezdp_decode_ipv6_result::__pad0__

Reserved bit 15.

unsigned ezdp_decode_ipv6_result::global_addresses

Both addresses scope are global.

unsigned ezdp_decode_ipv6_result::site_local_address

One of addresses scope is site local.

unsigned ezdp_decode_ipv6_result::link_local_address

One of addresses scope is link local.

unsigned ezdp_decode_ipv6_result::__pad1__

Reserved bits 9 to 11.

unsigned ezdp_decode_ipv6_result::options_exist

Options exist.

struct ezdp_decode_ipv6_control ezdp_decode_ipv6_result::control `[read]`

Control struct.

unsigned ezdp_decode_ipv6_result::__pad2__

Reserved bits 56 to 63.

struct ezdp_decode_ip_next_protocol ezdp_decode_ipv6_result::next_protocol `[read]`

Next protocol struct.

uint16_t ezdp_decode_ipv6_result::sip_dip_hash

sip+dip hash

union { ... }

The documentation for this struct was generated from the following file:

- dpe/dp/include/ezdp_decode_defs.h

ezdp_decode_ipv6_retval Struct Reference

Decode IPv4 return value struct definition.

Data Fields

- union {
- ezdp_decode_ipv6_retval_t raw_data
- struct {
- struct ezdp_decode_ipv6_errors error_codes
- *Error codes.* unsigned __pad0__: EZDP_DECODE_IPV6_RETVAL_RESERVED_15_SIZE
- *Reserved bit 15.* unsigned global_addresses: EZDP_DECODE_IPV6_RETVAL_GLOBAL_ADDRESSES_SIZE
- *Both addresses scope are global.* unsigned site_local_address: EZDP_DECODE_IPV6_RETVAL_SITE_LOCAL_ADDRESS_SIZE
- *One of addresses scope is site local.* unsigned link_local_address: EZDP_DECODE_IPV6_RETVAL_LINK_LOCAL_ADDRESS_SIZE
- *One of addresses scope is link local.* unsigned __pad1__: EZDP_DECODE_IPV6_RETVAL_RESERVED9_11_SIZE
- *Reserved bits 9 to 11.* unsigned options_exist: EZDP_DECODE_IPV6_RETVAL_OPTIONS_EXIST_SIZE
- *Options exist.* struct ezdp_decode_ipv6_control control
- *Control struct.* }
- };

Detailed Description

Decode IPv4 return value struct definition.

Field Documentation

ezdp_decode_ipv6_retval_t ezdp_decode_ipv6_retval::raw_data

struct ezdp_decode_ipv6_errors ezdp_decode_ipv6_retval::error_codes [read]

Error codes.

unsigned ezdp_decode_ipv6_retval::__pad0__

Reserved bit 15.

unsigned ezdp_decode_ipv6_retval::global_addresses

Both addresses scope are global.

unsigned ezdp_decode_ipv6_retval::site_local_address

One of addresses scope is site local.

unsigned ezdp_decode_ipv6_retval::link_local_address

One of addresses scope is link local.

unsigned ezdp_decode_ipv6_retval::__pad1__

Reserved bits 9 to 11.

unsigned ezdp_decode_ipv6_retval::options_exist

Options exist.

struct ezdp_decode_ipv6_control ezdp_decode_ipv6_retval::control `[read]`

Control struct.

union { ... }

The documentation for this struct was generated from the following file:

- dpe/dp/include/ezdp_decode_defs.h

ezdp_decode_mac_control Struct Reference

MAC addresses decoding result.

Data Fields

- union {
- ezdp_decode_mac_control_t raw_data
- struct {
- unsigned __pad0__: EZDP_DECODE_MAC_CONTROL_RESERVED13_15_SIZE
- *Reserved bits 13 to 15.* unsigned smac_equals_dmac: EZDP_DECODE_MAC_CONTROL_SMAC_EQUALS_DMAC_SIZE
- *Source MAC equals destination MAC.* unsigned user_config3: EZDP_DECODE_MAC_CONTROL_USER_CONFIG3_SIZE
- *Destination MAC (DMAC) is equal to user configured DMAC 3 masked with user configured DMAC mask 3.* unsigned user_config2: EZDP_DECODE_MAC_CONTROL_USER_CONFIG2_SIZE
- *Destination MAC (DMAC) is equal to user configured DMAC 2 masked with user configured DMAC mask 2.* unsigned user_config1: EZDP_DECODE_MAC_CONTROL_USER_CONFIG1_SIZE
- *Destination MAC (DMAC) is equal to user configured DMAC 1 masked with user configured DMAC mask 1.* unsigned user_config0: EZDP_DECODE_MAC_CONTROL_USER_CONFIG0_SIZE
- *Destination MAC (DMAC) is equal to user configured DMAC 0 masked with user configured DMAC mask 0.* unsigned ipv6_multicast: EZDP_DECODE_MAC_CONTROL_IPV6_MULTICAST_SIZE
- *DMAC is 0x33-33-xx-xx-xx-xx.* unsigned ipv4_multicast: EZDP_DECODE_MAC_CONTROL_IPV4_MULTICAST_SIZE
- *DMAC is 0x01-00-5E-xx-xx-xx.* unsigned vrrp_mac: EZDP_DECODE_MAC_CONTROL_VRRP_MAC_SIZE
- *DMAC is 0x00-00-5E-00-01-xx.* unsigned mac_control_other: EZDP_DECODE_MAC_CONTROL_MAC_CONTROL_OTHER_SIZE
- *DMAC is 0x01-80-C2-[01-FF]-[01-FF]-[01-FF].* unsigned mac_control_lsb_2x: EZDP_DECODE_MAC_CONTROL_MAC_CONTROL_LSB_2X_SIZE
- *DMAC is 0x01-80-C2-00-00-2x.* unsigned mac_control_lsb_1x: EZDP_DECODE_MAC_CONTROL_MAC_CONTROL_LSB_1X_SIZE
- *DMAC is 0x01-80-C2-00-00-1x.* unsigned mac_control_lsb_0x: EZDP_DECODE_MAC_CONTROL_MAC_CONTROL_LSB_0X_SIZE
- *DMAC is 0x01-80-C2-00-00-0x.* unsigned my_mac: EZDP_DECODE_MAC_CONTROL_MY_MAC_SIZE
- *Destination MAC is my MAC (DMAC is less than or equal to Low MAC and greater than or equal to High MAC, where Low and High MACs are user configured).* }
- };

Detailed Description

MAC addresses decoding result.

Field Documentation

ezdp_decode_mac_control_t ezdp_decode_mac_control::raw_data

unsigned ezdp_decode_mac_control::__pad0__

Reserved bits 13 to 15.

unsigned ezdp_decode_mac_control::smac_equals_dmac

Source MAC equals destination MAC.

unsigned ezdp_decode_mac_control::user_config3

Destination MAC (DMAC) is equal to user configured DMAC 3 masked with user configured DMAC mask 3.

unsigned ezdp_decode_mac_control::user_config2

Destination MAC (DMAC) is equal to user configured DMAC 2 masked with user configured DMAC mask 2.

unsigned ezdp_decode_mac_control::user_config1

Destination MAC (DMAC) is equal to user configured DMAC 1 masked with user configured DMAC mask 1.

unsigned ezdp_decode_mac_control::user_config0

Destination MAC (DMAC) is equal to user configured DMAC 0 masked with user configured DMAC mask 0.

unsigned ezdp_decode_mac_control::ipv6_multicast

DMAC is 0x33-33-xx-xx-xx-xx.

unsigned ezdp_decode_mac_control::ipv4_multicast

DMAC is 0x01-00-5E-xx-xx-xx.

unsigned ezdp_decode_mac_control::vrrp_mac

DMAC is 0x00-00-5E-00-01-xx.

unsigned ezdp_decode_mac_control::mac_control_other

DMAC is 0x01-80-C2-[01-FF]-[01-FF]-[01-FF].

unsigned ezdp_decode_mac_control::mac_control_lsb_2x

DMAC is 0x01-80-C2-00-00-2x.

unsigned ezdp_decode_mac_control::mac_control_lsb_1x

DMAC is 0x01-80-C2-00-00-1x.

unsigned ezdp_decode_mac_control::mac_control_lsb_0x

DMAC is 0x01-80-C2-00-00-0x.

unsigned ezdp_decode_mac_control::my_mac

Destination MAC is my MAC (DMAC is less than or equal to Low MAC and greater than or equal to High MAC, where Low and High MACs are user configured).

union { ... }

The documentation for this struct was generated from the following file:

- dpe/dp/include/ezdp_decode_defs.h

ezdp_decode_mac_errors Struct Reference

MAC header decode error flags.

Data Fields

- union {
- ezdp_decode_mac_errors_t raw_data
- struct {
- unsigned __pad0__: EZDP_DECODE_MAC_ERRORS_RESERVED5_7_SIZE
- *Reserved bits 5 to 7.* unsigned decode_error: EZDP_DECODE_MAC_ERRORS_DECODE_ERROR_SIZE
- *Aggregated decode error flag.* unsigned ip_version_mismatch_in_pppoe: EZDP_DECODE_MAC_ERRORS_IP_VERSION_MISMATCH_IN_PPPOE_SIZE
- *IP version mismatch in PPPoE.* unsigned dmac_is_zero: EZDP_DECODE_MAC_ERRORS_DMAC_IS_ZERO_SIZE
- *Destination MAC is zero.* unsigned smac_is_zero: EZDP_DECODE_MAC_ERRORS_SMAC_IS_ZERO_SIZE
- *Source MAC is zero.* unsigned smac_is_not_unicast: EZDP_DECODE_MAC_ERRORS_SMAC_IS_NOT_UNICAST_SIZE
- *Source MAC is not unicast.* }
- };

Detailed Description

MAC header decode error flags.

Field Documentation

ezdp_decode_mac_errors_t ezdp_decode_mac_errors::raw_data

unsigned ezdp_decode_mac_errors::__pad0__

Reserved bits 5 to 7.

unsigned ezdp_decode_mac_errors::decode_error

Aggregated decode error flag.

Indicate at least one decode error

unsigned ezdp_decode_mac_errors::ip_version_mismatch_in_pppoe

IP version mismatch in PPPoE.

unsigned ezdp_decode_mac_errors::dmac_is_zero

Destination MAC is zero.

unsigned ezdp_decode_mac_errors::smac_is_zero

Source MAC is zero.

unsigned ezdp_decode_mac_errors::smac_is_not_unicast

Source MAC is not unicast.

union { ... }

The documentation for this struct was generated from the following file:

- dpe/dp/include/ezdp_decode_defs.h

ezdp_decode_mac_protocol_type Struct Reference

Ethernet type definition.

Data Fields

- union {
- ezdp_decode_mac_protocol_type_t raw_data
- struct {
- unsigned __pad0__: EZDP_DECODE_MAC_PROTOCOL_TYPE_RESERVED_15_SIZE
- *Reserved bit 15.* unsigned user_config_vlan2: EZDP_DECODE_MAC_PROTOCOL_TYPE_USER_CONFIG_VLAN2_SIZE
- *Ethernet type is equal to user configured vlan type 2.* unsigned pppoe_discovery: EZDP_DECODE_MAC_PROTOCOL_TYPE_PPPOE_DISCOVERY_SIZE
- *Ethernet type is PPPoE Discovery Stage, value 0x8863.* unsigned pppoe_session: EZDP_DECODE_MAC_PROTOCOL_TYPE_PPPOE_SESSION_SIZE
- *Ethernet type is PPPoE Session Stage, value 0x8864.* unsigned user_config3: EZDP_DECODE_MAC_PROTOCOL_TYPE_USER_CONFIG3_SIZE
- *Ethernet type is equal to user configured type 3.* unsigned user_config2: EZDP_DECODE_MAC_PROTOCOL_TYPE_USER_CONFIG2_SIZE
- *Ethernet type is equal to user configured type 2.* unsigned user_config1: EZDP_DECODE_MAC_PROTOCOL_TYPE_USER_CONFIG1_SIZE
- *Ethernet type is equal to user configured type 1.* unsigned user_config0: EZDP_DECODE_MAC_PROTOCOL_TYPE_USER_CONFIG0_SIZE
- *Ethernet type is equal to user configured type 0.* unsigned length: EZDP_DECODE_MAC_PROTOCOL_TYPE_LENGTH_SIZE
- *Ethernet type is less or equal 0x0600 and indicate that this field is length.* unsigned ipv6: EZDP_DECODE_MAC_PROTOCOL_TYPE_IPV6_SIZE
- *Ethernet type is Internet Protocol, Version 6 (IPv6), value 0x86dd.* unsigned mpls_multicast: EZDP_DECODE_MAC_PROTOCOL_TYPE_MPLS_MULTICAST_SIZE
- *Ethernet type is MPLS multicast, value 0x8848.* unsigned mpls_unicast: EZDP_DECODE_MAC_PROTOCOL_TYPE_MPLS_UNICAST_SIZE
- *Ethernet type is MPLS unicast, value 0x8847.* unsigned arp: EZDP_DECODE_MAC_PROTOCOL_TYPE_ARP_SIZE
- *Ethernet type is Address Resolution Protocol (ARP), value 0x0806.* unsigned user_config_vlan1: EZDP_DECODE_MAC_PROTOCOL_TYPE_USER_CONFIG_VLAN1_SIZE
- *Ethernet type is equal to user configured vlan type 1.* unsigned user_config_vlan0: EZDP_DECODE_MAC_PROTOCOL_TYPE_USER_CONFIG_VLAN0_SIZE
- *Ethernet type is equal to user configured vlan type 0.* unsigned ipv4: EZDP_DECODE_MAC_PROTOCOL_TYPE_IPV4_SIZE
- *Ethernet type is Internet Protocol Version 4 (IPv4), value 0x0800.* }
- };

Detailed Description

Ethernet type definition.

Field Documentation

ezdp_decode_mac_protocol_type_t ezdp_decode_mac_protocol_type::raw_data

unsigned ezdp_decode_mac_protocol_type::__pad0__

Reserved bit 15.

unsigned ezdp_decode_mac_protocol_type::user_config_vlan2

Ethernet type is equal to user configured vlan type 2.

unsigned ezdp_decode_mac_protocol_type::pppoe_discovery

Ethernet type is PPPoE Discovery Stage, value 0x8863.

unsigned ezdp_decode_mac_protocol_type::pppoe_session

Ethernet type is PPPoE Session Stage, value 0x8864.

unsigned ezdp_decode_mac_protocol_type::user_config3

Ethernet type is equal to user configured type 3.

unsigned ezdp_decode_mac_protocol_type::user_config2

Ethernet type is equal to user configured type 2.

unsigned ezdp_decode_mac_protocol_type::user_config1

Ethernet type is equal to user configured type 1.

unsigned ezdp_decode_mac_protocol_type::user_config0

Ethernet type is equal to user configured type 0.

unsigned ezdp_decode_mac_protocol_type::length

Ethernet type is less or equal 0x0600 and indicate that this field is length.

unsigned ezdp_decode_mac_protocol_type::ipv6

Ethernet type is Internet Protocol, Version 6 (IPv6), value 0x86dd.

unsigned ezdp_decode_mac_protocol_type::mpls_multicast

Ethernet type is MPLS multicast, value 0x8848.

unsigned ezdp_decode_mac_protocol_type::mpls_unicast

Ethernet type is MPLS unicast, value 0x8847.

unsigned ezdp_decode_mac_protocol_type::arp

Ethernet type is Address Resolution Protocol (ARP), value 0x0806.

unsigned ezdp_decode_mac_protocol_type::user_config_vlan1

Ethernet type is equal to user configured vlan type 1.

unsigned ezdp_decode_mac_protocol_type::user_config_vlan0

Ethernet type is equal to user configured vlan type 0.

unsigned ezdp_decode_mac_protocol_type::ipv4

Ethernet type is Internet Protocol Version 4 (IPv4), value 0x0800.

union { ... }

The documentation for this struct was generated from the following file:

- dpe/dp/include/ezdp_decode_defs.h

ezdp_decode_mac_result Struct Reference

Decode MAC result.

Data Fields

- union {
- uint32_t raw_data [EZDP_DECODE_MAC_RESULT_WORD_COUNT]
- struct {
- unsigned __pad0__: EZDP_DECODE_MAC_RESULT_RESERVED_31_SIZE
- *Reserved bit 31.* unsigned number_of_tags: EZDP_DECODE_MAC_RESULT_NUMBER_OF_TAGS_SIZE
- *Number of tags.* unsigned __pad1__: EZDP_DECODE_MAC_RESULT_RESERVED26_27_SIZE
- *Reserved bits 26 to 27.* unsigned ipv6_in_pppoe: EZDP_DECODE_MAC_RESULT_IPV6_IN_PPPOE_SIZE
- *IPv6 in PPPoE.* unsigned ipv4_in_pppoe: EZDP_DECODE_MAC_RESULT_IPV4_IN_PPPOE_SIZE
- *IPv4 in PPPoE.* struct ezdp_decode_mac_errors error_codes
- *Error codes.* struct ezdp_decode_mac_control control
- *control struct* struct ezdp_decode_mac_protocol_type tag2_protocol_type
- *Tag2 protocol type.* struct ezdp_decode_mac_protocol_type tag1_protocol_type
- *Tag1 protocol type.* struct ezdp_decode_mac_protocol_type last_tag_protocol_type
- *Last tag protocol type.* struct ezdp_decode_mac_protocol_type tag3_protocol_type
- *Tag3 protocol type.* unsigned __pad2__: EZDP_DECODE_MAC_RESULT_RESERVED120_127_SIZE
- *Reserved bits 120 to 127.* uint8_t layer2_size
- *Layer2 size.* uint16_t da_sa_hash
- *SMAC+DMAC hash.* }
- };

Detailed Description

Decode MAC result.

Field Documentation

uint32_t ezdp_decode_mac_result::raw_data[EZDP_DECODE_MAC_RESULT_WORD_COUNT]

unsigned ezdp_decode_mac_result::__pad0__

Reserved bit 31.

unsigned ezdp_decode_mac_result::number_of_tags

Number of tags.

unsigned ezdp_decode_mac_result::__pad1__

Reserved bits 26 to 27.

unsigned ezdp_decode_mac_result::ipv6_in_pppoe

IPv6 in PPPoE.

unsigned ezdp_decode_mac_result::ipv4_in_pppoe

IPv4 in PPPoE.

struct ezdp_decode_mac_errors ezdp_decode_mac_result::error_codes `[read]`

Error codes.

struct ezdp_decode_mac_control ezdp_decode_mac_result::control `[read]`

control struct

struct ezdp_decode_mac_protocol_type ezdp_decode_mac_result::tag2_protocol_type `[read]`

Tag2 protocol type.

struct ezdp_decode_mac_protocol_type ezdp_decode_mac_result::tag1_protocol_type `[read]`

Tag1 protocol type.

struct ezdp_decode_mac_protocol_type ezdp_decode_mac_result::last_tag_protocol_type `[read]`

Last tag protocol type.

struct ezdp_decode_mac_protocol_type ezdp_decode_mac_result::tag3_protocol_type `[read]`

Tag3 protocol type.

unsigned ezdp_decode_mac_result::__pad2__

Reserved bits 120 to 127.

uint8_t ezdp_decode_mac_result::layer2_size

Layer2 size.

uint16_t ezdp_decode_mac_result::da_sa_hash

SMAC+DMAC hash.

union { ... }

The documentation for this struct was generated from the following file:

- dpe/dp/include/ezdp_decode_defs.h

ezdp_decode_mac_retval Struct Reference

Decode MAC return value struct definition.

Data Fields

- union {
- ezdp_decode_mac_retval_t raw_data
- struct {
- unsigned __pad0__: EZDP_DECODE_MAC_RETVAL_RESERVED_31_SIZE
- *Reserved bit 31.* unsigned number_of_tags: EZDP_DECODE_MAC_RETVAL_NUMBER_OF_TAGS_SIZE
- *Number of tags.* unsigned __pad1__: EZDP_DECODE_MAC_RETVAL_RESERVED26_27_SIZE
- *Reserved bits 26 to 27.* unsigned ipv6_in_pppoe: EZDP_DECODE_MAC_RETVAL_IPV6_IN_PPPOE_SIZE
- *IPv6 in PPPoE.* unsigned ipv4_in_pppoe: EZDP_DECODE_MAC_RETVAL_IPV4_IN_PPPOE_SIZE
- *IPv4 in PPPoE.* struct ezdp_decode_mac_errors error_codes
- *Error codes.* struct ezdp_decode_mac_control control
- *control struct* }
- };

Detailed Description

Decode MAC return value struct definition.

Field Documentation

ezdp_decode_mac_retval_t ezdp_decode_mac_retval::raw_data

unsigned ezdp_decode_mac_retval::__pad0__

Reserved bit 31.

unsigned ezdp_decode_mac_retval::number_of_tags

Number of tags.

unsigned ezdp_decode_mac_retval::__pad1__

Reserved bits 26 to 27.

unsigned ezdp_decode_mac_retval::ipv6_in_pppoe

IPv6 in PPPoE.

unsigned ezdp_decode_mac_retval::ipv4_in_pppoe

IPv4 in PPPoE.

struct ezdp_decode_mac_errors ezdp_decode_mac_retval::error_codes `[read]`

Error codes.

struct ezdp_decode_mac_control ezdp_decode_mac_retval::control `[read]`

control struct

union { ... }

The documentation for this struct was generated from the following file:

- dpe/dp/include/ezdp_decode_defs.h

ezdp_decode_mpls_label_result Struct Reference

ezdp_decode_mpls_label_result struct for ezdp

Data Fields

- union {
- ezdp_decode_mpls_label_result_t raw_data
- struct {
- unsigned __pad0__: EZDP_DECODE_MPLS_LABEL_RESULT_RESERVED10_31_SIZE
- *Reserved bits 10 to 31.* unsigned stop_bit: EZDP_DECODE_MPLS_LABEL_RESULT_STOP_BIT_SIZE
- *Stop bit = exception_bit | (~exception_bit and ~reserved_label).* unsigned exception_bit: EZDP_DECODE_MPLS_LABEL_RESULT_EXCEPTION_BIT_SIZE
- *Expection bit = Or_reduce (((first 8 bits of retval) ^ inv_from_host) and mask_from_host).* unsigned user_config3: EZDP_DECODE_MPLS_LABEL_RESULT_USER_CONFIG3_SIZE
- *Label is equal to user configured label 3.* unsigned user_config2: EZDP_DECODE_MPLS_LABEL_RESULT_USER_CONFIG2_SIZE
- *Label is equal to user configured label 2.* unsigned user_config1: EZDP_DECODE_MPLS_LABEL_RESULT_USER_CONFIG1_SIZE
- *Label is equal to user configured label 1.* unsigned user_config0: EZDP_DECODE_MPLS_LABEL_RESULT_USER_CONFIG0_SIZE
- *Label is equal to user configured label 0.* unsigned ttl_is_one: EZDP_DECODE_MPLS_LABEL_RESULT_TTL_IS_ONE_SIZE
- *TTL is one.* unsigned ttl_is_zero: EZDP_DECODE_MPLS_LABEL_RESULT_TTL_IS_ZERO_SIZE
- *TTL is zero.* unsigned reserved_label: EZDP_DECODE_MPLS_LABEL_RESULT_RESERVED_LABEL_SIZE
- *Reserved label - label value is less than 16.* unsigned end_of_stack: EZDP_DECODE_MPLS_LABEL_RESULT_END_OF_STACK_SIZE
- *Last label on MPLS stack.* }
- };

Detailed Description

ezdp_decode_mpls_label_result struct for ezdp

Field Documentation

ezdp_decode_mpls_label_result_t ezdp_decode_mpls_label_result::raw_data

unsigned ezdp_decode_mpls_label_result::__pad0__

Reserved bits 10 to 31.

unsigned ezdp_decode_mpls_label_result::stop_bit

Stop bit = exception_bit | (~exception_bit and ~reserved_label).

unsigned ezdp_decode_mpls_label_result::exception_bit

Expection bit = Or_reduce (((first 8 bits of retval) ^ inv_from_host) and mask_from_host).

unsigned ezdp_decode_mpls_label_result::user_config3

Label is equal to user configured label 3.

unsigned ezdp_decode_mpls_label_result::user_config2

Label is equal to user configured label 2.

unsigned ezdp_decode_mpls_label_result::user_config1

Label is equal to user configured label 1.

unsigned ezdp_decode_mpls_label_result::user_config0

Label is equal to user configured label 0.

unsigned ezdp_decode_mpls_label_result::ttl_is_one

TTL is one.

unsigned ezdp_decode_mpls_label_result::ttl_is_zero

TTL is zero.

unsigned ezdp_decode_mpls_label_result::reserved_label

Reserved label - label value is less than 16.

unsigned ezdp_decode_mpls_label_result::end_of_stack

Last label on MPLS stack.

union { ... }

The documentation for this struct was generated from the following file:

- dpe/dp/include/ezdp_decode_defs.h

ezdp_decode_mpls_label_retval Struct Reference

Decode MPLS label return value struct definition.

Data Fields

- union {
- ezdp_decode_mpls_label_retval_t raw_data
- struct {
- unsigned __pad0__: EZDP_DECODE_MPLS_LABEL_RETVAL_RESERVED10_31_SIZE
- *Reserved bits 10 to 31.* unsigned stop_bit: EZDP_DECODE_MPLS_LABEL_RETVAL_STOP_BIT_SIZE
- *Stop bit = exception_bit | (~exception_bit and ~reserved_label).* unsigned exception_bit: EZDP_DECODE_MPLS_LABEL_RETVAL_EXCEPTION_BIT_SIZE
- *Expection bit = or_reduce (((first 8 bits of retval) ^ inv_from_host) and mask_from_host).* unsigned user_config3: EZDP_DECODE_MPLS_LABEL_RETVAL_USER_CONFIG3_SIZE
- *Label is equal to user configured label 3.* unsigned user_config2: EZDP_DECODE_MPLS_LABEL_RETVAL_USER_CONFIG2_SIZE
- *Label is equal to user configured label 2.* unsigned user_config1: EZDP_DECODE_MPLS_LABEL_RETVAL_USER_CONFIG1_SIZE
- *Label is equal to user configured label 1.* unsigned user_config0: EZDP_DECODE_MPLS_LABEL_RETVAL_USER_CONFIG0_SIZE
- *Label is equal to user configured label 0.* unsigned ttl_is_one: EZDP_DECODE_MPLS_LABEL_RETVAL_TTL_IS_ONE_SIZE
- *TTL is one.* unsigned ttl_is_zero: EZDP_DECODE_MPLS_LABEL_RETVAL_TTL_IS_ZERO_SIZE
- *TTL is zero.* unsigned reserved_label: EZDP_DECODE_MPLS_LABEL_RETVAL_RESERVED_LABEL_SIZE
- *Reserved label - label value is less than 16.* unsigned end_of_stack: EZDP_DECODE_MPLS_LABEL_RETVAL_END_OF_STACK_SIZE
- *Last label on MPLS stack.* }
- };

Detailed Description

Decode MPLS label return value struct definition.

Field Documentation

ezdp_decode_mpls_label_retval_t ezdp_decode_mpls_label_retval::raw_data

unsigned ezdp_decode_mpls_label_retval::__pad0__

Reserved bits 10 to 31.

unsigned ezdp_decode_mpls_label_retval::stop_bit

Stop bit = exception_bit | (~exception_bit and ~reserved_label).

unsigned ezdp_decode_mpls_label_retval::exception_bit

Expection bit = or_reduce (((first 8 bits of retval) ^ inv_from_host) and mask_from_host).

unsigned ezdp_decode_mpls_label_retval::user_config3

Label is equal to user configured label 3.

unsigned ezdp_decode_mpls_label_retval::user_config2

Label is equal to user configured label 2.

unsigned ezdp_decode_mpls_label_retval::user_config1

Label is equal to user configured label 1.

unsigned ezdp_decode_mpls_label_retval::user_config0

Label is equal to user configured label 0.

unsigned ezdp_decode_mpls_label_retval::ttl_is_one

TTL is one.

unsigned ezdp_decode_mpls_label_retval::ttl_is_zero

TTL is zero.

unsigned ezdp_decode_mpls_label_retval::reserved_label

Reserved label - label value is less than 16.

unsigned ezdp_decode_mpls_label_retval::end_of_stack

Last label on MPLS stack.

union { ... }

The documentation for this struct was generated from the following file:

- dpe/dp/include/ezdp_decode_defs.h

ezdp_decode_mpls_result Struct Reference

ezdp_decode_mpls_result struct for ezdp

Data Fields

- union {
- ezdp_decode_mpls_result_t raw_data
- struct {
- unsigned __pad0__: EZDP_DECODE_MPLS_RESULT_RESERVED28_31_SIZE
- *Reserved bits 28 to 31.* unsigned label4_ttl_is_one: EZDP_DECODE_MPLS_RESULT_LABEL4_TTL_IS_ONE_SIZE
- *Label4 TTL is one.* unsigned label3_ttl_is_one: EZDP_DECODE_MPLS_RESULT_LABEL3_TTL_IS_ONE_SIZE
- *Label3 TTL is one.* unsigned label2_ttl_is_one: EZDP_DECODE_MPLS_RESULT_LABEL2_TTL_IS_ONE_SIZE
- *Label2 TTL is one.* unsigned label1_ttl_is_one: EZDP_DECODE_MPLS_RESULT_LABEL1_TTL_IS_ONE_SIZE
- *Label1 TTL is one.* unsigned __pad1__: EZDP_DECODE_MPLS_RESULT_RESERVED20_23_SIZE
- *Reserved bits 20 to 23.* unsigned label4_ttl_is_zero: EZDP_DECODE_MPLS_RESULT_LABEL4_TTL_IS_ZERO_SIZE
- *Label4 TTL is zero.* unsigned label3_ttl_is_zero: EZDP_DECODE_MPLS_RESULT_LABEL3_TTL_IS_ZERO_SIZE
- *Label3 TTL is zero.* unsigned label2_ttl_is_zero: EZDP_DECODE_MPLS_RESULT_LABEL2_TTL_IS_ZERO_SIZE
- *Label2 TTL is zero.* unsigned label1_ttl_is_zero: EZDP_DECODE_MPLS_RESULT_LABEL1_TTL_IS_ZERO_SIZE
- *Label1 TTL is zero.* unsigned __pad2__: EZDP_DECODE_MPLS_RESULT_RESERVED10_15_SIZE
- *Reserved bits 10 to 15.* unsigned last_entry_in_stack: EZDP_DECODE_MPLS_RESULT_LAST_ENTRY_IN_STACK_SIZE
- *Last entry in stack.* unsigned __pad3__: EZDP_DECODE_MPLS_RESULT_RESERVED1_7_SIZE
- *Reserved bits 1 to 7.* unsigned decode_error: EZDP_DECODE_MPLS_RESULT_DECODE_ERROR_SIZE
- *Error flag: decode failed.* }
- };

Detailed Description

ezdp_decode_mpls_result struct for ezdp

Field Documentation

ezdp_decode_mpls_result_t ezdp_decode_mpls_result::raw_data

unsigned ezdp_decode_mpls_result::__pad0__

Reserved bits 28 to 31.

unsigned ezdp_decode_mpls_result::label4_ttl_is_one

Label4 TTL is one.

unsigned ezdp_decode_mpls_result::label3_ttl_is_one

Label3 TTL is one.

unsigned ezdp_decode_mpls_result::label2_ttl_is_one

Label2 TTL is one.

unsigned ezdp_decode_mpls_result::label1_ttl_is_one

Label1 TTL is one.

unsigned ezdp_decode_mpls_result::__pad1__

Reserved bits 20 to 23.

unsigned ezdp_decode_mpls_result::label4_ttl_is_zero

Label4 TTL is zero.

unsigned ezdp_decode_mpls_result::label3_ttl_is_zero

Label3 TTL is zero.

unsigned ezdp_decode_mpls_result::label2_ttl_is_zero

Label2 TTL is zero.

unsigned ezdp_decode_mpls_result::label1_ttl_is_zero

Label1 TTL is zero.

unsigned ezdp_decode_mpls_result::__pad2__

Reserved bits 10 to 15.

unsigned ezdp_decode_mpls_result::last_entry_in_stack

Last entry in stack.

unsigned ezdp_decode_mpls_result::__pad3__

Reserved bits 1 to 7.

unsigned ezdp_decode_mpls_result::decode_error

Error flag: decode failed.

union { ... }

The documentation for this struct was generated from the following file:

- dpe/dp/include/ezdp_decode_defs.h

ezdp_decode_mpls_retval Struct Reference

Decode MPLS return value struct definition.

Data Fields

- union {
- ezdp_decode_mpls_retval_t raw_data
- struct {
- unsigned __pad0__: EZDP_DECODE_MPLS_RETVAL_RESERVED28_31_SIZE
- *Reserved bits 28 to 31.* unsigned label4_ttl_is_one: EZDP_DECODE_MPLS_RETVAL_LABEL4_TTL_IS_ONE_SIZE
- *Label4 TTL is one.* unsigned label3_ttl_is_one: EZDP_DECODE_MPLS_RETVAL_LABEL3_TTL_IS_ONE_SIZE
- *Label3 TTL is one.* unsigned label2_ttl_is_one: EZDP_DECODE_MPLS_RETVAL_LABEL2_TTL_IS_ONE_SIZE
- *Label2 TTL is one.* unsigned label1_ttl_is_one: EZDP_DECODE_MPLS_RETVAL_LABEL1_TTL_IS_ONE_SIZE
- *Label1 TTL is one.* unsigned __pad1__: EZDP_DECODE_MPLS_RETVAL_RESERVED20_23_SIZE
- *Reserved bits 20 to 23.* unsigned label4_ttl_is_zero: EZDP_DECODE_MPLS_RETVAL_LABEL4_TTL_IS_ZERO_SIZE
- *Label4 TTL is zero.* unsigned label3_ttl_is_zero: EZDP_DECODE_MPLS_RETVAL_LABEL3_TTL_IS_ZERO_SIZE
- *Label3 TTL is zero.* unsigned label2_ttl_is_zero: EZDP_DECODE_MPLS_RETVAL_LABEL2_TTL_IS_ZERO_SIZE
- *Label2 TTL is zero.* unsigned label1_ttl_is_zero: EZDP_DECODE_MPLS_RETVAL_LABEL1_TTL_IS_ZERO_SIZE
- *Label1 TTL is zero.* unsigned __pad2__: EZDP_DECODE_MPLS_RETVAL_RESERVED10_15_SIZE
- *Reserved bits 10 to 15.* unsigned last_entry_in_stack: EZDP_DECODE_MPLS_RETVAL_LAST_ENTRY_IN_STACK_SIZE
- *Last entry in stack.* unsigned __pad3__: EZDP_DECODE_MPLS_RETVAL_RESERVED1_7_SIZE
- *Reserved bits 1 to 7.* unsigned decode_error: EZDP_DECODE_MPLS_RETVAL_DECODE_ERROR_SIZE
- *Error flag: decode failed.* }
- };

Detailed Description

Decode MPLS return value struct definition.

Field Documentation

ezdp_decode_mpls_retval_t ezdp_decode_mpls_retval::raw_data

unsigned ezdp_decode_mpls_retval::__pad0__

Reserved bits 28 to 31.

unsigned ezdp_decode_mpls_retval::label4_ttl_is_one

Label4 TTL is one.

unsigned ezdp_decode_mpls_retval::label3_ttl_is_one

Label3 TTL is one.

unsigned ezdp_decode_mpls_retval::label2_ttl_is_one

Label2 TTL is one.

unsigned ezdp_decode_mpls_retval::label1_ttl_is_one

Label1 TTL is one.

unsigned ezdp_decode_mpls_retval::__pad1__

Reserved bits 20 to 23.

unsigned ezdp_decode_mpls_retval::label4_ttl_is_zero

Label4 TTL is zero.

unsigned ezdp_decode_mpls_retval::label3_ttl_is_zero

Label3 TTL is zero.

unsigned ezdp_decode_mpls_retval::label2_ttl_is_zero

Label2 TTL is zero.

unsigned ezdp_decode_mpls_retval::label1_ttl_is_zero

Label1 TTL is zero.

unsigned ezdp_decode_mpls_retval::__pad2__

Reserved bits 10 to 15.

unsigned ezdp_decode_mpls_retval::last_entry_in_stack

Last entry in stack.

unsigned ezdp_decode_mpls_retval::__pad3__

Reserved bits 1 to 7.

unsigned ezdp_decode_mpls_retval::decode_error

Error flag: decode failed.

union { ... }

The documentation for this struct was generated from the following file:

- dpe/dp/include/ezdp_decode_defs.h

ezdp_decode_tcp_errors Struct Reference

TCP header decode error flags.

Data Fields

- union {
- ezdp_decode_tcp_errors_t raw_data
- struct {
- unsigned __pad0__: EZDP_DECODE_TCP_ERRORS_RESERVED3_7_SIZE
- *Reserved bits 3 to 7.* unsigned decode_error: EZDP_DECODE_TCP_ERRORS_DECODE_ERROR_SIZE
- *Aggregated decode error flag.* unsigned syn_and_fin_eq_1: EZDP_DECODE_TCP_ERRORS_SYN_AND_FIN_EQ_1_SIZE
- *syn==1 and fin==1* unsigned data_offset_lt_5: EZDP_DECODE_TCP_ERRORS_DATA_OFFSET_LT_5_SIZE
- *Data offset is less than 5.* }
- };

Detailed Description

TCP header decode error flags.

Field Documentation

ezdp_decode_tcp_errors_t ezdp_decode_tcp_errors::raw_data

unsigned ezdp_decode_tcp_errors::__pad0__

Reserved bits 3 to 7.

unsigned ezdp_decode_tcp_errors::decode_error

Aggregated decode error flag.

Indicate at least one decode error

unsigned ezdp_decode_tcp_errors::syn_and_fin_eq_1

syn==1 and fin==1

unsigned ezdp_decode_tcp_errors::data_offset_lt_5

Data offset is less than 5.

union { ... }

The documentation for this struct was generated from the following file:

- dpe/dp/include/ezdp_decode_defs.h

ezdp_decode_tcp_retval Struct Reference

Decode TCP return value struct definition.

Data Fields

- union {
- ezdp_decode_tcp_retval_t raw_data
- struct {
- unsigned __pad0__: EZDP_DECODE_TCP_RETVAL_RESERVED24_31_SIZE
- *Reserved bits 24 to 31.* unsigned __pad1__: EZDP_DECODE_TCP_RETVAL_RESERVED22_23_SIZE
- *Reserved bits 22 to 23.* unsigned data_offset: EZDP_DECODE_TCP_RETVAL_DATA_OFFSET_SIZE
- *Data offset.* unsigned __pad2__: EZDP_DECODE_TCP_RETVAL_RESERVED9_15_SIZE
- *Reserved bits 9 to 15.* unsigned options_exist: EZDP_DECODE_TCP_RETVAL_OPTIONS_EXIST_SIZE
- *Options exist.* struct ezdp_decode_tcp_errors error_codes
- *Decode error codes.* }
- };

Detailed Description

Decode TCP return value struct definition.

Field Documentation

ezdp_decode_tcp_retval_t ezdp_decode_tcp_retval::raw_data

unsigned ezdp_decode_tcp_retval::__pad0__

Reserved bits 24 to 31.

unsigned ezdp_decode_tcp_retval::__pad1__

Reserved bits 22 to 23.

unsigned ezdp_decode_tcp_retval::data_offset

Data offset.

unsigned ezdp_decode_tcp_retval::__pad2__

Reserved bits 9 to 15.

unsigned ezdp_decode_tcp_retval::options_exist

Options exist.

struct ezdp_decode_tcp_errors ezdp_decode_tcp_retval::error_codes [read]

Decode error codes.

union { ... }

The documentation for this struct was generated from the following file:

- dpe/dp/include/ezdp_decode_defs.h

ezdp_driver_desc Struct Reference

TX/RX descriptor.

Data Fields

- union {
- uint32_t raw_data [EZDP_DRIVER_DESC_WORD_COUNT]
- struct {
- uint64_t buf_data_addr
- *Address of Buffer data.* uint32_t len
- *Buffer data length.* uint16_t sub_type
- *TODO: define.* struct ezdp_driver_desc_flags flags
- *marks ring descriptor/buffer pair state and type* uint8_t total
- *The total number of buffers in an buffer block.* }
- };

Detailed Description

TX/RX descriptor.

Field Documentation

uint32_t ezdp_driver_desc::raw_data[EZDP_DRIVER_DESC_WORD_COUNT]

uint64_t ezdp_driver_desc::buf_data_addr

Address of Buffer data.

uint32_t ezdp_driver_desc::len

Buffer data length.

uint16_t ezdp_driver_desc::sub_type

TODO: define.

struct ezdp_driver_desc_flags ezdp_driver_desc::flags [read]

marks ring descriptor/buffer pair state and type

uint8_t ezdp_driver_desc::total

The total number of buffers in an buffer block.

union { ... }

The documentation for this struct was generated from the following file:

- dpe/dp/include/ezdp_pci_defs.h

ezdp_driver_desc_flags Struct Reference

TX/RX descriptor flags structure.

Data Fields

- union {
- ezdp_driver_desc_flags_t raw_data
- struct {
- unsigned type: EZDP_DRIVER_DESC_FLAGS_TYPE_SIZE
- *TODO: define.* unsigned error: EZDP_DRIVER_DESC_FLAGS_ERROR_SIZE
- *TODO: define.* unsigned owner: EZDP_DRIVER_DESC_FLAGS_OWNER_SIZE
- *indicates whether the descriptor belongs to the driver (OWNER=1) or the NIC (OWNER=0)* unsigned data: EZDP_DRIVER_DESC_FLAGS_DATA_SIZE
- *indicates whether the descriptor is a message (DATA=0) or a data (DATA=1)* }
- };

Detailed Description

TX/RX descriptor flags structure.

Field Documentation

ezdp_driver_desc_flags_t ezdp_driver_desc_flags::raw_data

unsigned ezdp_driver_desc_flags::type

TODO: define.

unsigned ezdp_driver_desc_flags::error

TODO: define.

unsigned ezdp_driver_desc_flags::owner

indicates whether the descriptor belongs to the driver (OWNER=1) or the NIC (OWNER=0)

unsigned ezdp_driver_desc_flags::data

indicates whether the descriptor is a message (DATA=0) or a data (DATA=1)

union { ... }

The documentation for this struct was generated from the following file:

- dpe/dp/include/ezdp_pci_defs.h

ezdp_dual_add32_result Struct Reference

The result of the atomic dual add32 instruction.

Data Fields

- union {
- uint32_t raw_data [EZDP_DUAL_ADD32_RESULT_WORD_COUNT]
- struct {
- int32_t original_value2
- *The original value of variable 2.* int32_t original_value1
- *The original value of variable 1.* }
- };

Detailed Description

The result of the atomic dual add32 instruction.

Field Documentation

uint32_t ezdp_dual_add32_result::raw_data[EZDP_DUAL_ADD32_RESULT_WORD_COUNT]

int32_t ezdp_dual_add32_result::original_value2

The original value of variable 2.

int32_t ezdp_dual_add32_result::original_value1

The original value of variable 1.

union { ... }

The documentation for this struct was generated from the following file:

- dpe/dp/include/ezdp_memory_defs.h

ezdp_dual_add64_result Struct Reference

The result of the atomic dual add64 instruction.

Data Fields

- union {
- uint32_t raw_data [EZDP_DUAL_ADD64_RESULT_WORD_COUNT]
- struct {
- int64_t original_value2
- *The original value of variable 2.* int64_t original_value1
- *The original value of variable 1.* }
- };

Detailed Description

The result of the atomic dual add64 instruction.

Field Documentation

uint32_t ezdp_dual_add64_result::raw_data[EZDP_DUAL_ADD64_RESULT_WORD_COUNT]

int64_t ezdp_dual_add64_result::original_value2

The original value of variable 2.

int64_t ezdp_dual_add64_result::original_value1

The original value of variable 1.

union { ... }

The documentation for this struct was generated from the following file:

- dpe/dp/include/ezdp_memory_defs.h

ezdp_dual_ctr Struct Reference

On-demand dual counter value.

Data Fields

- union {
- uint32_t raw_data [EZDP_DUAL_CTR_WORD_COUNT]
- struct {
- uint64_t byte
- *Byte counter value.* uint64_t event
- *Event counter value.* }
- };

Detailed Description

On-demand dual counter value.

Field Documentation

uint32_t ezdp_dual_ctr::raw_data[EZDP_DUAL_CTR_WORD_COUNT]

uint64_t ezdp_dual_ctr::byte

Byte counter value.

uint64_t ezdp_dual_ctr::event

Event counter value.

union { ... }

The documentation for this struct was generated from the following file:

- dpe/dp/include/ezdp_counter_defs.h

ezdp_dual_ctr_cfg Struct Reference

On-demand dual counter configuration definition.

Data Fields

- union {
- uint32_t raw_data [EZDP_DUAL_CTR_CFG_WORD_COUNT]
- struct {
- unsigned __pad0__: EZDP_DUAL_CTR_CFG_ECC_SIZE
- *ECC.* unsigned __pad1__: EZDP_DUAL_CTR_CFG_RESERVED19_23_SIZE
- *Reserved bits 19 to 23 (sub type - double=14).* unsigned byte_report_exceeded: EZDP_DUAL_CTR_CFG_BYTE_REPORT_EXCEEDED_SIZE
- *Number of bits threshold on byte counter.* unsigned __pad2__: EZDP_DUAL_CTR_CFG_CLR_ON_GC_SIZE
- *Clear counter when generating garbage collection message.* unsigned enable_exceed_message: EZDP_DUAL_CTR_CFG_ENABLE_EXCEED_MESSAGE_SIZE
- *Enable threshold exceed message.* unsigned event_report_exceeded: EZDP_DUAL_CTR_CFG_EVENT_REPORT_EXCEEDED_SIZE
- *Number of bits threshold on event counter.* unsigned byte_value_size: EZDP_DUAL_CTR_CFG_BYTE_VALUE_SIZE_SIZE
- *Byte counter size.* unsigned __pad3__: EZDP_DUAL_CTR_CFG_RESERVED0_SIZE
- *Reserved bit 0 (overflow flag).* struct ezdp_dual_ctr value
- *Counter value.* }
- };

Detailed Description

On-demand dual counter configuration definition.

Field Documentation

uint32_t ezdp_dual_ctr_cfg::raw_data[EZDP_DUAL_CTR_CFG_WORD_COUNT]

unsigned ezdp_dual_ctr_cfg::__pad0__

ECC.

unsigned ezdp_dual_ctr_cfg::__pad1__

Reserved bits 19 to 23 (sub type - double=14).

unsigned ezdp_dual_ctr_cfg::byte_report_exceeded

Number of bits threshold on byte counter.

unsigned ezdp_dual_ctr_cfg::__pad2__

Clear counter when generating garbage collection message.

Should always be set to 1.

unsigned ezdp_dual_ctr_cfg::enable_exceed_message

Enable threshold exceed message.

unsigned ezdp_dual_ctr_cfg::event_report_exceeded

Number of bits threshold on event counter.

unsigned ezdp_dual_ctr_cfg::byte_value_size

Byte counter size.

unsigned ezdp_dual_ctr_cfg::__pad3__

Reserved bit 0 (overflow flag).

struct ezdp_dual_ctr ezdp_dual_ctr_cfg::value `[read]`

Counter value.

union { ... }

The documentation for this struct was generated from the following file:

- dpe/dp/include/ezdp_counter_defs.h

ezdp_dual_ctr_result Struct Reference

On-demand dual value counter result value.

Data Fields

- union {
- uint32_t raw_data [EZDP_DUAL_CTR_RESULT_WORD_COUNT]
- struct {
- unsigned __pad0__: EZDP_DUAL_CTR_RESULT_RESERVED31_SIZE
- *Reserved bit 31.* unsigned byte_value_msb: EZDP_DUAL_CTR_RESULT_BYTE_VALUE_MSB_SIZE
- *MSB byte counter value.* unsigned byte_value_lsb: EZDP_DUAL_CTR_RESULT_BYTE_VALUE_LSB_SIZE
- *LSB byte counter value.* unsigned event_value: EZDP_DUAL_CTR_RESULT_EVENT_VALUE_SIZE
- *Event counter value.* }
- };

Detailed Description

On-demand dual value counter result value.

Field Documentation

uint32_t ezdp_dual_ctr_result::raw_data[EZDP_DUAL_CTR_RESULT_WORD_COUNT]

unsigned ezdp_dual_ctr_result::__pad0__

Reserved bit 31.

unsigned ezdp_dual_ctr_result::byte_value_msb

MSB byte counter value.

unsigned ezdp_dual_ctr_result::byte_value_lsb

LSB byte counter value.

unsigned ezdp_dual_ctr_result::event_value

Event counter value.

union { ... }

The documentation for this struct was generated from the following file:

- dpe/dp/include/ezdp_counter_defs.h

ezdp_ext_addr Struct Reference

Extended address definition.

Data Fields

- union {
- uint32_t raw_data [EZDP_EXT_ADDR_WORD_COUNT]
- struct {
- unsigned __pad0__: EZDP_EXT_ADDR_RESERVED16_31_SIZE
- *Reserved bits 16 to 31.* unsigned __pad1__: EZDP_EXT_ADDR_RESERVED14_15_SIZE
- *Reserved bits 14-15.* unsigned msid: EZDP_EXT_ADDR_MSID_SIZE
- *< Type of the MSID* unsigned __pad2__: EZDP_EXT_ADDR_RESERVED4_7_SIZE
- *Reserved bits 4-7.* unsigned address_msb: EZDP_EXT_ADDR_ADDRESS_MSB_SIZE
- *4 msb of 36 bits extended address* uint32_t address
- *32 lsb of 36 bit extended address* }
- };

Detailed Description

Extended address definition.

Field Documentation

uint32_t ezdp_ext_addr::raw_data[EZDP_EXT_ADDR_WORD_COUNT]

unsigned ezdp_ext_addr::__pad0__

Reserved bits 16 to 31.

unsigned ezdp_ext_addr::__pad1__

Reserved bits 14-15.

unsigned ezdp_ext_addr::msid

Type of the MSID

MSID select

unsigned ezdp_ext_addr::__pad2__

Reserved bits 4-7.

unsigned ezdp_ext_addr::address_msb

4 msb of 36 bits extended address

uint32_t ezdp_ext_addr::address

32 lsb of 36 bit extended address

union { ... }

The documentation for this struct was generated from the following file:

- dpe/dp/include/ezdp_memory_defs.h

ezdp_ext_linked_buffers_desc Struct Reference

Extended linked buffers descriptor.

Data Fields

- struct ezdp_linked_buffers_desc_line line [EZDP_EXTENDED_LBD]

Array of 16 linked buffers lines, which contains up to 48 buffers.

Detailed Description

Extended linked buffers descriptor.

May hold up to 48 buffs Applicable only in EZDP_EXT_FRAME frame type

Field Documentation

struct ezdp_linked_buffers_desc_line ezdp_ext_linked_buffers_desc::line[EZDP_EXTENDED_LBD] `[read]`

Array of 16 linked buffers lines, which contains up to 48 buffers.

The documentation for this struct was generated from the following file:

- dpe/dp/include/ezdp_frame_defs.h

ezdp_flow_control_status Struct Reference

Flow control status.

Data Fields

- union {
- ezdp_flow_control_status_t raw_data
- struct {
- unsigned __pad0__: EZDP_FLOW_CONTROL_STATUS_RESERVED4_7_SIZE
- *Reserved bits 4 to 7.* unsigned enable: EZDP_FLOW_CONTROL_STATUS_ENABLE_SIZE
- *Indicate if flow control element is enabled.* }
- };

Detailed Description

Flow control status.

Field Documentation

ezdp_flow_control_status_t ezdp_flow_control_status::raw_data

unsigned ezdp_flow_control_status::__pad0__

Reserved bits 4 to 7.

unsigned ezdp_flow_control_status::enable

Indicate if flow control element is enabled.

union { ... }

The documentation for this struct was generated from the following file:

- dpe/dp/include/ezdp_job_defs.h

ezdp_frame_desc Struct Reference

Frame descriptor data structure.

Data Fields

- union {
- uint32_t raw_data [EZDP_FRAME_DESC_WORD_COUNT]
- struct {
- uint8_t ecc
- *Eight-bit ECC field protecting bytes 0x1-0xf frame descriptor data.* unsigned transmit_confirmation_flag: EZDP_FRAME_DESC_TRANSMIT_CONFIRMATION_FLAG_SIZE
- *< The frame type field defines buffer organization and identifies which buffer type is pointed by buffer (BD).* unsigned timestamp_flag: EZDP_FRAME_DESC_TIMESTAMP_FLAG_SIZE
- *On receive path, this flag indicates presence of timestamp value in ezdp_job_desc.rx_info.* unsigned transmit_keep_buf_flag: EZDP_FRAME_DESC_TRANSMIT_KEEP_BUF_FLAG_SIZE
- *On receive path, this flag is echoed on confirmation responses.* unsigned gross_checksum_flag: EZDP_FRAME_DESC_GROSS_CHECKSUM_FLAG_SIZE
- *On receive path, the checksum flag indicates the checksum has been calculated for the frame.* unsigned __pad0__: EZDP_FRAME_DESC_RESERVED0_1_SIZE
- *Reserved bits 0 to 1.* unsigned __pad1__: EZDP_FRAME_DESC_RESERVED14_15_SIZE
- *Reserved bits 14 to 15.* unsigned class_of_service: EZDP_FRAME_DESC_CLASS_OF_SERVICE_SIZE
- *Frame class of service grade.* unsigned __pad2__: EZDP_FRAME_DESC_RESERVED10_11_SIZE
- *Reserved bits 10 to 11.* unsigned buf_budget_id: EZDP_FRAME_DESC_BUF_BUDGET_ID_SIZE
- *Budget group ID.* uint16_t frame_length
- *Total frame data length in bytes.* uint8_t data_buf_count
- *This field indicates the number of buffers (BDs) that hold the frame (include empty and null buffers).* uint8_t header_offset
- *This is the frame header starting point in the first frame data buffer.* struct ezdp_buffer_desc buf_desc
- *Pointer to a 256B buffer located either in IMEM or EMEM.* uint8_t free_bytes
- *This field indicates how many free bytes are left at the end of the frame header data buffer.* uint8_t logical_id
- *Logical user info assigned by configuration to interfaces, Can be used to identify interfaces or group of same interface type.* unsigned __pad3__: EZDP_FRAME_DESC_RESERVED106_110_SIZE
- *< Multicast control* unsigned job_budget_id: EZDP_FRAME_DESC_JOB_BUDGET_ID_SIZE
- *Job budget group ID.* }
- };

Detailed Description

Frame descriptor data structure.

Field Documentation

uint32_t ezdp_frame_desc::raw_data[EZDP_FRAME_DESC_WORD_COUNT]

uint8_t ezdp_frame_desc::ecc

Eight-bit ECC field protecting bytes 0x1-0xf frame descriptor data.

This field can be used by the SW if frame descriptor is not saved in DDR.

unsigned ezdp_frame_desc::transmit_confirmation_flag

< The frame type field defines buffer organization and identifies which buffer type is pointed by buffer (BD).

On receive path, this flag indicates that received job is a confirmation response. On transmit path, this flag is used as a request to return job confirmation when transmission is done. NOTE: When timestamp flag is on in additional to transmit_confirmation flag, the confirmation frame (FD) will contain timestamp timer value captured on the MAC. This option is used for two-step 1588.

unsigned ezdp_frame_desc::timestamp_flag

On receive path, this flag indicates presence of timestamp value in ezdp_job_desc.rx_info.

For two-step 1588 confirmation frame this flag also indicate success of adding 1588 timestamp. Or in the other words, timestamp_flag will be off if adding captured 1588 timestamp failed. On transmit path, this flag is used as a request to add captured 1588 timestamp to header by MAC. For two-step 1588 packet, where confirmation response is required from the MAC, the transmit_confirmation flag should also be on.

unsigned ezdp_frame_desc::transmit_keep_buf_flag

On receive path, this flag is echoed on confirmation responses.

On transmit path, this flag indicates to keep the frame buffers. It is SW responsibility to free frame buffers to prevent memory leakage. When FD is sent to the PMU flush queue TKB flag is implied to be zero (usage of PMU flush queue forces buffer release operation regardless of FD[TKB] state).

unsigned ezdp_frame_desc::gross_checksum_flag

On receive path, the checksum flag indicates the checksum has been calculated for the frame.

On transmit path, this flag is reserved on NPS-400.

unsigned ezdp_frame_desc:: __pad0__

Reserved bits 0 to 1.

unsigned ezdp_frame_desc:: __pad1__

Reserved bits 14 to 15.

unsigned ezdp_frame_desc::class_of_service

Frame class of service grade.

unsigned ezdp_frame_desc:: __pad2__

Reserved bits 10 to 11.

unsigned ezdp_frame_desc::buf_budget_id

Budget group ID.

Budget identifies an allocated IMEM or EMEM buffer resource control operation. By default a budget group ID is associated with an Rx port ID from which the frame was received.

uint16_t ezdp_frame_desc::frame_length

Total frame data length in bytes.

Represents the actual frame data received/transmitted on the wire starting from L2 frame header. The four-layer 2-byte CRC may be included (if the MAC is configured for keeping CRC bytes on Rx and/or not adding CRC bytes on Tx) or excluded (if the MAC is configured to strip CRC on Rx and/or add CRC on Tx). frame_length excludes all overheads such as frame context and other bytes preceding the header_offset position. Value of 0x3FFF represents that frame data length is greater or equal to 16383 bytes, therefore max length that can be accurately represented by an exact value is 0x3FFE (16382 bytes).

uint8_t ezdp_frame_desc::data_buf_count

This field indicates the number of buffers (BDs) that hold the frame (include empty and null buffers).

Note that for an Extended frame the value does not include the buffer where the linked buffers descriptors is stored.

uint8_t ezdp_frame_desc::header_offset

This is the frame header starting point in the first frame data buffer.

To include embedded LBD in the same buffer of the first frame data (EMBEDDED_BD frame type), the header_offset must be at least the max configured embedded LBD size (16B or 32B). The gaps from beginning of the buffer or from max embedded LBD size to header_offset can be utilized by frame context or can be used for optimized header modification (increasing or decreasing).

struct ezdp_buffer_desc ezdp_frame_desc::buf_desc `[read]`

Pointer to a 256B buffer located either in IMEM or EMEM.

The type to which the buffer points is determined by frame type.

uint8_t ezdp_frame_desc::free_bytes

This field indicates how many free bytes are left at the end of the frame header data buffer.

This field is valid only for STANDARD frames.

uint8_t ezdp_frame_desc::logical_id

Logical user info assigned by configuration to interfaces, Can be used to identify interfaces or group of same interface type.

unsigned ezdp_frame_desc::__pad3__

< Multicast control

Reserved bits 106 to 110

unsigned ezdp_frame_desc::job_budget_id

Job budget group ID.

Budget identifies an allocated job resource control operations. By default a budget group ID is associated with an Rx port ID from which the frame was received.

union { ... }

The documentation for this struct was generated from the following file:

- dpe/dp/include/ezdp_frame_defs.h

ezdp_group_schlr_status Struct Reference

PMU group scheduler status (based on PMU system info).

Data Fields

- union {
- ezdp_group_schlr_status_t raw_data
- struct {
- unsigned __pad0__: EZDP_GROUP_SCHLR_STATUS_RESERVED13_15_SIZE
- *Reserved bits 13 to 15.* unsigned dispatched_job: EZDP_GROUP_SCHLR_STATUS_DISPATCHED_JOB_SIZE
- *The number of jobs dispatched from the group scheduler.* }
- };

Detailed Description

PMU group scheduler status (based on PMU system info).

There are 16 group schedulers in each PMU side.

Field Documentation

ezdp_group_schlr_status_t ezdp_group_schlr_status::raw_data

unsigned ezdp_group_schlr_status::__pad0__

Reserved bits 13 to 15.

unsigned ezdp_group_schlr_status::dispatched_job

The number of jobs dispatched from the group scheduler.

Defined as number of jobs that were dispatched from the group scheduler for processing and are waiting for "job done".

union { ... }

The documentation for this struct was generated from the following file:

- dpe/dp/include/ezdp_job_defs.h

ezdp_hier_tb_ctr_cfg Struct Reference

Statistic hierarchical token bucket counter config structure (write cfg usage).

Data Fields

- union {
- uint32_t raw_data [EZDP_HIER_TB_CTR_CFG_WORD_COUNT]
- struct {
- unsigned ctr0_fail_threshold: EZDP_HIER_TB_CTR_CFG_CTR0_FAIL_THRESHOLD_SIZE
- *Fail threshold for first front end counter/accumulator.* unsigned __pad0__: EZDP_HIER_TB_CTR_CFG_RESERVED22_26_SIZE
- *reserved bit 22-26.* unsigned ctr_sum_fail_threshold: EZDP_HIER_TB_CTR_CFG_CTR_SUM_FAIL_THRESHOLD_SIZE
- *Fail threshold for sum of first and second front end counters/accumulators.* unsigned ctr_sum_updt_threshold: EZDP_HIER_TB_CTR_CFG_CTR_SUM_UPDT_THRESHOLD_SIZE
- *Update threshold for sum of first and second front end counters/accumulators.* unsigned ctr1_fail_threshold: EZDP_HIER_TB_CTR_CFG_CTR1_FAIL_THRESHOLD_SIZE
- *Fail threshold for second front end counter/accumulator.* unsigned ctr1_updt_threshold: EZDP_HIER_TB_CTR_CFG_CTR1_UPDT_THRESHOLD_SIZE
- *Update threshold for second front end counter/accumulator.* unsigned __pad1__: EZDP_HIER_TB_CTR_CFG_RESERVED0_1_SIZE
- *reserved bits 0-1* unsigned __pad2__: EZDP_HIER_TB_CTR_CFG_RESERVED63_SIZE
- *reserved bits 63.* unsigned timestamp_threshold: EZDP_HIER_TB_CTR_CFG_TIMESTAMP_THRESHOLD_SIZE
- *Selects one of 4 sets of timestamp thresholds for this counter.* unsigned ctr0_updt_threshold: EZDP_HIER_TB_CTR_CFG_CTR0_UPDT_THRESHOLD_SIZE
- *Update threshold for second front end counter/accumulator.* unsigned app_bits: EZDP_HIER_TB_CTR_CFG_APP_BITS_SIZE
- *The application specific bits of the counter.* }
- };

Detailed Description

Statistic hierarchical token bucket counter config structure (write cfg usage).

Field Documentation

uint32_t ezdp_hier_tb_ctr_cfg::raw_data[EZDP_HIER_TB_CTR_CFG_WORD_COUNT]

unsigned ezdp_hier_tb_ctr_cfg::ctr0_fail_threshold

Fail threshold for first front end counter/accumulator.

Possible values: 0 - 17.

unsigned ezdp_hier_tb_ctr_cfg::__pad0__

reserved bit 22-26.

unsigned ezdp_hier_tb_ctr_cfg::ctr_sum_fail_threshold

Fail threshold for sum of first and second front end counters/accumulators.

unsigned ezdp_hier_tb_ctr_cfg::ctr_sum_updt_threshold

Update threshold for sum of first and second front end counters/accumulators.
Possible values: 0 - 17.

unsigned ezdp_hier_tb_ctr_cfg::ctr1_fail_threshold

Fail threshold for second front end counter/accumulator.
Possible values: 0 - 17.

unsigned ezdp_hier_tb_ctr_cfg::ctr1_updt_threshold

Update threshold for second front end counter/accumulator.
Possible values: 0 - 17.

unsigned ezdp_hier_tb_ctr_cfg::__pad1__

reserved bits 0-1

unsigned ezdp_hier_tb_ctr_cfg::__pad2__

reserved bits 63.

unsigned ezdp_hier_tb_ctr_cfg::timestamp_threshold

Selects one of 4 sets of timestamp thresholds for this counter.

unsigned ezdp_hier_tb_ctr_cfg::ctr0_updt_threshold

Update threshold for second front end counter/accumulator.
Possible values: 0 - 17.

unsigned ezdp_hier_tb_ctr_cfg::app_bits

The application specific bits of the counter.

union { ... }

The documentation for this struct was generated from the following file:

- dpe/dp/include/ezdp_counter_defs.h

ezdp_hier_tb_result Struct Reference

Hierarchical token bucket counter result value definition.

Data Fields

- union {
- uint32_t raw_data [EZDP_HIER_TB_RESULT_WORD_COUNT]
- struct {
- unsigned fail: EZDP_HIER_TB_RESULT_FAIL_SIZE
- *Operation Failed.* unsigned update_task: EZDP_HIER_TB_RESULT_UPDATE_TASK_SIZE
- *BE update procedure is required.* unsigned ctr1: EZDP_HIER_TB_RESULT_CTR1_SIZE
- *< The state of the counter.* unsigned __pad0__: EZDP_HIER_TB_RESULT_RESERVED0_9_SIZE
- *reserved bits 0-9* unsigned __pad1__: EZDP_HIER_TB_RESULT_RESERVED56_63_SIZE
- *reserved bits 56-63* unsigned app_bits: EZDP_HIER_TB_RESULT_APP_BITS_SIZE
- *The application specific bits of the counter, as was configured by config API.* unsigned __pad2__: EZDP_HIER_TB_RESULT_RESERVED82_95_SIZE
- *reserved bits 82-95* unsigned ctr0: EZDP_HIER_TB_RESULT_CTR0_SIZE
- *Value of first front end counter/accumulator (pre colored green).* }
- };

Detailed Description

Hierarchical token bucket counter result value definition.

Field Documentation

uint32_t ezdp_hier_tb_result::raw_data[EZDP_HIER_TB_RESULT_WORD_COUNT]

unsigned ezdp_hier_tb_result::fail

Operation Failed.

unsigned ezdp_hier_tb_result::update_task

BE update procedure is required.

unsigned ezdp_hier_tb_result::ctr1

< The state of the counter.

Value of first front end counter/accumulator (pre colored yellow)

unsigned ezdp_hier_tb_result::__pad0__

reserved bits 0-9

unsigned ezdp_hier_tb_result::__pad1__

reserved bits 56-63

unsigned ezdp_hier_tb_result::app_bits

The application specific bits of the counter, as was configured by config API.

unsigned ezdp_hier_tb_result::__pad2__

reserved bits 82-95

unsigned ezdp_hier_tb_result::ctr0

Value of first front end counter/accumulator (pre colored green).

union { ... }

The documentation for this struct was generated from the following file:

- dpe/dp/include/ezdp_counter_defs.h

ezdp_hier_tb_ug_app_bits Struct Reference

Application bits of Hierarchical token bucket for ultra green feature.

Data Fields

- union {
- ezdp_hier_tb_ug_app_bits_t raw_data
- struct {
- unsigned __pad0__: EZDP_HIER_TB_UG_APP_BITS_RESERVED24_31_SIZE
- *bits out of app_bit size.* unsigned app_bits: EZDP_HIER_TB_UG_APP_BITS_APP_BITS_SIZE
- *The application specific bits of the counter.* unsigned eigth_mode_ret_bits: EZDP_HIER_TB_UG_APP_BITS_EIGTH_MODE_RET_BITS_SIZE
- *Returned bits for 8 bit mode Note: Will be updated while updating app_bits.* unsigned color_state_y: EZDP_HIER_TB_UG_APP_BITS_COLOR_STATE_Y_SIZE
- *State of the buckets for pre-color yellow.* unsigned color_state_g: EZDP_HIER_TB_UG_APP_BITS_COLOR_STATE_G_SIZE
- *State of the buckets for pre-color green.* }
- };

Detailed Description

Application bits of Hierarchical token bucket for ultra green feature.

Field Documentation

ezdp_hier_tb_ug_app_bits_t ezdp_hier_tb_ug_app_bits::raw_data

unsigned ezdp_hier_tb_ug_app_bits::__pad0__

bits out of app_bit size.

reserved bits 24-31

unsigned ezdp_hier_tb_ug_app_bits::app_bits

The application specific bits of the counter.

This field is part of Hybrid hier_tb (ultra green feature

unsigned ezdp_hier_tb_ug_app_bits::eigth_mode_ret_bits

Returned bits for 8 bit mode Note: Will be updated while updating app_bits.

unsigned ezdp_hier_tb_ug_app_bits::color_state_y

State of the buckets for pre-color yellow.

0x3 mean ultra yellow Note: Will be updated while updating app_bits

unsigned ezdp_hier_tb_ug_app_bits::color_state_g

State of the buckets for pre-color green.

0x3 mean ultra greenNote: Will be updated while updating app_bits

union { ... }

The documentation for this struct was generated from the following file:

- dpe/dp/include/ezdp_counter_defs.h

ezdp_hier_tb_update Struct Reference

Hierarchical token bucket update counter definition.

Data Fields

- union {
- ezdp_hier_tb_update_t raw_data
- struct {
- unsigned set_active_state: EZDP_HIER_TB_UPDATE_SET_ACTIVE_STATE_SIZE
- *Set active state (non conditional).* unsigned clr_ctr: EZDP_HIER_TB_UPDATE_CLR_CTR_SIZE
- *Clear the front end counters value.* unsigned set_app_bits: EZDP_HIER_TB_UPDATE_SET_APP_BITS_SIZE
- *Set the app specific bits.* unsigned cond_set_active_state: EZDP_HIER_TB_UPDATE_COND_SET_ACTIVE_STATE_SIZE
- *Set active state only if both front end counters are 0.* unsigned __pad0__: EZDP_HIER_TB_UPDATE_RESERVED24_27_SIZE
- *reserved 24-27* unsigned app_bits: EZDP_HIER_TB_UPDATE_APP_BITS_SIZE
- *The new application specific bits of the counter.* }
- };

Detailed Description

Hierarchical token bucket update counter definition.

Field Documentation

ezdp_hier_tb_update_t ezdp_hier_tb_update::raw_data

unsigned ezdp_hier_tb_update::set_active_state

Set active state (non conditional).

unsigned ezdp_hier_tb_update::clr_ctr

Clear the front end counters value.

unsigned ezdp_hier_tb_update::set_app_bits

Set the app specific bits.

unsigned ezdp_hier_tb_update::cond_set_active_state

Set active state only if both front end counters are 0.

unsigned ezdp_hier_tb_update::__pad0__

reserved 24-27

unsigned ezdp_hier_tb_update::app_bits

The new application specific bits of the counter.

Applicable only when set_app_bits is 1.

union { ... }

The documentation for this struct was generated from the following file:

- dpe/dp/include/ezdp_counter_defs.h

ezdp_input_queue_status Struct Reference

PMU physical input queue status definition (based on PMU system info).

Data Fields

- union {
- uint32_t raw_data [EZDP_INPUT_QUEUE_STATUS_WORD_COUNT]
- struct {
- unsigned __pad0__: EZDP_INPUT_QUEUE_STATUS_RESERVED19_31_SIZE
- *Reserved bits 19 to 31.* unsigned ready: EZDP_INPUT_QUEUE_STATUS_READY_SIZE
- *The queue is ready to accept traffic.* uint16_t dispatched_job
- *< Indicate the queue congestion level.* uint16_t size
- *The total number of jobs in this list.* uint16_t outstanding_job
- *The number of outstanding jobs.* }
- };

Detailed Description

PMU physical input queue status definition (based on PMU system info).

Field Documentation

uint32_t ezdp_input_queue_status::raw_data[EZDP_INPUT_QUEUE_STATUS_WORD_COUNT]

unsigned ezdp_input_queue_status::__pad0__

Reserved bits 19 to 31.

unsigned ezdp_input_queue_status::ready

The queue is ready to accept traffic.

uint16_t ezdp_input_queue_status::dispatched_job

< Indicate the queue congestion level.

The number of dispatched jobs from the queue. Defined as number of jobs that were dispatched from the queue for processing and are waiting for "job done".

uint16_t ezdp_input_queue_status::size

The total number of jobs in this list.

This include jobs which not yet dispatched, jobs which are waiting for "job done" and jobs which are waiting to get to head of the queue. The number of not dispatched job can be calculated by size - outstanding_job.

uint16_t ezdp_input_queue_status::outstanding_job

The number of outstanding jobs.

Defined as number of jobs that were dispatched from the queue and were not yet dequeued (either waiting for "job done" or waiting to get to head of the queue).

union { ... }

The documentation for this struct was generated from the following file:

- dpe/dp/include/ezdp_job_defs.h

ezdp_job_container_cmd_desc Struct Reference

Job container request.

Data Fields

- union {
- ezdp_job_container_cmd_desc_t raw_data
- struct {
- uint16_t job_id
- *Pointer to job that needs to be expanded by the container.* union {
- struct {
- unsigned __pad0__: EZDP_JOB_CONTAINER_CMD_DESC_RESERVED0_11_SIZE
- *< Determine the action to process for an expanded job* }
- union {
- struct ezdp_job_transmit_cmd_info transmit_info
- *Job transmit request info.* struct ezdp_job_queue_cmd_info queue_info
- *Job dispatch request info.* struct ezdp_job_discard_cmd_info discard_info
- *Job discard request info.* } u
- }
- }
- };

Detailed Description

Job container request.

Field Documentation

ezdp_job_container_cmd_desc_t ezdp_job_container_cmd_desc::raw_data

uint16_t ezdp_job_container_cmd_desc::job_id

Pointer to job that needs to be expanded by the container.

NOTE: It must never point to another job container

unsigned ezdp_job_container_cmd_desc::__pad0__

< Determine the action to process for an expanded job

Reserved bits 0 to 11

struct ezdp_job_transmit_cmd_info ezdp_job_container_cmd_desc::transmit_info [read]

Job transmit request info.

struct ezdp_job_queue_cmd_info ezdp_job_container_cmd_desc::queue_info [read]

Job dispatch request info.

struct ezdp_job_discard_cmd_info ezdp_job_container_cmd_desc::discard_info `[read]`

Job discard request info.

union { ... } ezdp_job_container_cmd_desc::u

union { ... }

The documentation for this struct was generated from the following file:

- dpe/dp/include/ezdp_job_defs.h

ezdp_job_container_desc Struct Reference

Job container descriptor.

Data Fields

- union {
- uint32_t raw_data [EZDP_JOB_CONTAINER_DESC_WORD_COUNT]
- struct {
- unsigned __pad0__: EZDP_JOB_CONTAINER_DESC_RESERVED29_31_SIZE
- *Reserved bits 29 to 31.* ezdp_job_container_info_t info: EZDP_JOB_CONTAINER_DESC_INFO_SIZE
- *Number of job requests in the job container.* unsigned job_budget_id: EZDP_JOB_CONTAINER_DESC_JOB_BUDGET_ID_SIZE
- *Job budget group ID.* unsigned __pad1__: EZDP_JOB_CONTAINER_DESC_RESERVED0_15_SIZE
- *Reserved bits 0 to 15.* struct ezdp_job_container_cmd_desc job_commands [EZDP_JOB_CONTAINER_DESC_MAX_NUM_OF_JOBS]
- *Job requests array.* }
- };

Detailed Description

Job container descriptor.

Field Documentation

uint32_t ezdp_job_container_desc::raw_data[EZDP_JOB_CONTAINER_DESC_WORD_COUNT]

unsigned ezdp_job_container_desc::__pad0__

Reserved bits 29 to 31.

ezdp_job_container_info_t ezdp_job_container_desc::info

Number of job requests in the job container.

unsigned ezdp_job_container_desc::job_budget_id

Job budget group ID.

Budget identifies an allocated job resource control operation.

unsigned ezdp_job_container_desc::__pad1__

Reserved bits 0 to 15.

struct ezdp_job_container_cmd_desc ezdp_job_container_desc::job_commands[EZDP_JOB_CONTAINER_DESC_MAX_NUM_OF_JOBS] [read]

Job requests array.

union { ... }

The documentation for this struct was generated from the following file:

- dpe/dp/include/ezdp_job_defs.h

ezdp_job_desc Struct Reference

job descriptor data structure

Data Fields

- struct ezdp_frame_desc frame_desc
- *Frame descriptor associated with the job.* union {
- struct ezdp_job_rx_info rx_info
- *Job receive info.* struct ezdp_job_tx_info tx_info
- *Job transmit info.* };

Detailed Description

job descriptor data structure

Field Documentation

struct ezdp_frame_desc ezdp_job_desc::frame_desc [read]

Frame descriptor associated with the job.

struct ezdp_job_rx_info ezdp_job_desc::rx_info [read]

Job receive info.

Applicable when job is received by the process or when sending or updating PMU queue.

struct ezdp_job_tx_info ezdp_job_desc::tx_info [read]

Job transmit info.

Applicable when job is sent to TM.

union { ... }

The documentation for this struct was generated from the following file:

- dpe/dp/include/ezdp_job_defs.h

ezdp_job_discard_cmd_info Struct Reference

Job container discard request info.

Data Fields

- union {
- ezdp_job_discard_cmd_info_t raw_data
- struct {
- unsigned __pad0__: EZDP_JOB_DISCARD_CMD_INFO_RESERVED11_15_SIZE
- *Reserved bits 11 to 15.* unsigned side: EZDP_JOB_DISCARD_CMD_INFO_SIDE_SIZE
- *PMU side.* unsigned __pad1__: EZDP_JOB_DISCARD_CMD_INFO_RESERVED0_9_SIZE
- *Reserved bits 0 to 9.* }
- };

Detailed Description

Job container discard request info.

Field Documentation

ezdp_job_discard_cmd_info_t ezdp_job_discard_cmd_info::raw_data

unsigned ezdp_job_discard_cmd_info::__pad0__

Reserved bits 11 to 15.

unsigned ezdp_job_discard_cmd_info::side

PMU side.

Define which side should handle the job done request (PMU implements the queuing and JD read operation)

unsigned ezdp_job_discard_cmd_info::__pad1__

Reserved bits 0 to 9.

union { ... }

The documentation for this struct was generated from the following file:

- dpe/dp/include/ezdp_job_defs.h

ezdp_job_queue_cmd_info Struct Reference

Job container send to queue request info.

Data Fields

- union {
- ezdp_job_queue_cmd_info_t raw_data
- struct {
- unsigned __pad0__: EZDP_JOB_QUEUE_CMD_INFO_RESERVED8_15_SIZE
- *Reserved bits 8 to 15.* unsigned side: EZDP_JOB_QUEUE_CMD_INFO_SIDE_SIZE
- *PMU target queue side.* unsigned target_queue: EZDP_JOB_QUEUE_CMD_INFO_TARGET_QUEUE_SIZE
- *PMU target queue.* }
- };

Detailed Description

Job container send to queue request info.

Field Documentation

ezdp_job_queue_cmd_info_t ezdp_job_queue_cmd_info::raw_data

unsigned ezdp_job_queue_cmd_info::__pad0__

Reserved bits 8 to 15.

unsigned ezdp_job_queue_cmd_info::side

PMU target queue side.

unsigned ezdp_job_queue_cmd_info::target_queue

PMU target queue.

union { ... }

The documentation for this struct was generated from the following file:

- dpe/dp/include/ezdp_job_defs.h

ezdp_job_rx_confirmation_info Struct Reference

Info field for incoming job from TX confirmation ports.

Data Fields

- union {
- uint32_t raw_data [EZDP_JOB_RX_CONFIRMATION_INFO_WORD_COUNT]
- struct {
- unsigned __pad0__: EZDP_JOB_RX_CONFIRMATION_INFO_RESERVED8_31_SIZE
- *Reserved bits 8 to 31.* uint8_t timestamp_sec
- *The timestamp seconds counter is sampled at one with the nanoseconds counter, providing together the standard 5B timestamp.* uint32_t timestamp_nsec
- *The timestamp nanoseconds counter.* }
- };

Detailed Description

Info field for incoming job from TX confirmation ports.

Field Documentation

uint32_t ezdp_job_rx_confirmation_info::raw_data[EZDP_JOB_RX_CONFIRMATION_INFO_WORD_COUNT]

unsigned ezdp_job_rx_confirmation_info::__pad0__

Reserved bits 8 to 31.

uint8_t ezdp_job_rx_confirmation_info::timestamp_sec

The timestamp seconds counter is sampled at one with the nanoseconds counter, providing together the standard 5B timestamp.

Applicable only when frame descriptor time_stamp flag is on.

uint32_t ezdp_job_rx_confirmation_info::timestamp_nsec

The timestamp nanoseconds counter.

Together with timestamp_sec provide the standard 5B timestemp. Applicable only when frame descriptor time_stamp flag is on. The Tx MAC samples the timestamp at the beginning of transmission time and embeds the measurement in the response job, that can be interpreted by SW to calculate 1588 timestamp.

union { ... }

The documentation for this struct was generated from the following file:

- dpe/dp/include/ezdp_job_defs.h

ezdp_job_rx_info Struct Reference

Job receive info.

Data Fields

- union {
- uint32_t raw_data [EZDP_JOB_RX_INFO_WORD_COUNT]
- struct {
- union {
- struct ezdp_job_rx_interface_info interface_info
- *External interface information.* struct ezdp_job_rx_loopback_info loopback_info
- *Loopback information.* struct ezdp_job_rx_confirmation_info confirmation_info
- *Confirmation information.* struct ezdp_job_rx_timer_info timer_info
- *Timer information.* struct ezdp_job_rx_user_info user_info
- *User information.* }
- unsigned __pad0__: EZDP_JOB_RX_INFO_RESERVED112_127_SIZE
- *Reserved bits 112 to 127.* uint16_t gross_checksum
- *Checksum value starts at header_offset and continues to end of frame, excluding bytes stripped by the MAC (e.g.* uint16_t seq_number
- *Sequence number of the job in a physical queue.* unsigned is_service_ready: EZDP_JOB_RX_INFO_IS_SERVICE_READY_SIZE
- *This flag indicate that service, requested in dispatch command, is ready.* unsigned seq_number_valid: EZDP_JOB_RX_INFO_SEQ_NUMBER_VALID_SIZE
- *Indicates that the seq_number field is valid.* unsigned __pad1__: EZDP_JOB_RX_INFO_RESERVED108_109_SIZE
- *Reserved bits 108 to 109.* unsigned __pad2__: EZDP_JOB_RX_INFO_RESERVED104_107_SIZE
- *Reserved bits 104 to 107.* unsigned side: EZDP_JOB_RX_INFO_SIDE_SIZE
- *The side of the PMU.* unsigned source_queue: EZDP_JOB_RX_INFO_SOURCE_QUEUE_SIZE
- *The Processor Manager Unit queue identification, initialized by Processor Manager Unit for incoming job according to configuration or by SW on the dispatch, in order to inform the next pipeline processing stage.* }
- };

Detailed Description

Job receive info.

Field Documentation

uint32_t ezdp_job_rx_info::raw_data[EZDP_JOB_RX_INFO_WORD_COUNT]

struct ezdp_job_rx_interface_info ezdp_job_rx_info::interface_info [read]

External interface information.

struct ezdp_job_rx_loopback_info ezdp_job_rx_info::loopback_info [read]

Loopback information.

struct ezdp_job_rx_confirmation_info ezdp_job_rx_info::confirmation_info [read]

Confirmation information.

struct ezdp_job_rx_timer_info ezdp_job_rx_info::timer_info `[read]`

Timer information.

struct ezdp_job_rx_user_info ezdp_job_rx_info::user_info `[read]`

User information.

unsigned ezdp_job_rx_info::__pad0__

Reserved bits 112 to 127.

uint16_t ezdp_job_rx_info::gross_checksum

Checksum value starts at header_offset and continues to end of frame, excluding bytes stripped by the MAC (e.g.

four CRC bytes at the end of the frame). NOTE: This field is valid only when the checksum flag is set.

uint16_t ezdp_job_rx_info::seq_number

Sequence number of the job in a physical queue.

Valid only when seq_number_valid flag is ON.

unsigned ezdp_job_rx_info::is_service_ready

This flag indicate that service, requested in dispatch command, is ready.

unsigned ezdp_job_rx_info::seq_number_valid

Indicates that the seq_number field is valid.

Sequence number is set by Processor Manager Unit when frame is received from port, or when it updates sequence number based on dispatch request with sequence numbering service.

unsigned ezdp_job_rx_info::__pad1__

Reserved bits 108 to 109.

unsigned ezdp_job_rx_info::__pad2__

Reserved bits 104 to 107.

unsigned ezdp_job_rx_info::side

The side of the PMU.

unsigned ezdp_job_rx_info::source_queue

The Processor Manager Unit queue identification, initialized by Processor Manager Unit for incoming job according to configuration or by SW on the dispatch, in order to inform the next pipeline processing stage.

union { ... }

The documentation for this struct was generated from the following file:

- dpe/dp/include/ezdp_job_defs.h

ezdp_job_rx_interface_info Struct Reference

Info field for incoming job from external RX interfaces.

Data Fields

- union {
- uint32_t raw_data [EZDP_JOB_RX_INTERFACE_INFO_WORD_COUNT]
- struct {
- unsigned imem_buf_count: EZDP_JOB_RX_INTERFACE_INFO_IMEM_BUF_COUNT_SIZE
- *< Indicate the PMU queue depth from which the job was sent.* unsigned __pad0__: EZDP_JOB_RX_INTERFACE_INFO_RESERVED15_SIZE
- *Reserved bit 15.* unsigned __pad1__: EZDP_JOB_RX_INTERFACE_INFO_RESERVED14_SIZE
- *Reserved bit 14.* unsigned icu_succ_parsing_flag: EZDP_JOB_RX_INTERFACE_INFO_ICU_SUCC_PARSING_FLAG_SIZE
- *If this flag is cleared the L2-L3 frame header could not be parsed by ICU, and therefore global_congestion_level is invalid and frame CoS contains default value.* unsigned truncation_flag: EZDP_JOB_RX_INTERFACE_INFO_TRUNCATION_FLAG_SIZE
- *Indicating the frame data was truncated on the Rx path and the data is illegal.* unsigned __pad2__: EZDP_JOB_RX_INTERFACE_INFO_RESERVED11_SIZE
- *Reserved bit 11.* unsigned __pad3__: EZDP_JOB_RX_INTERFACE_INFO_RESERVED10_SIZE
- *Reserved bit 10.* unsigned crc_ok_flag: EZDP_JOB_RX_INTERFACE_INFO_CRC_OK_FLAG_SIZE
- *CRC was checked by MAC and was found OK.* unsigned crc_checked_flag: EZDP_JOB_RX_INTERFACE_INFO_CRC_CHECKED_FLAG_SIZE
- *CRC checked by the MAC.* uint8_t timestamp_sec
- *The timestamp seconds counter is sampled at one with the nanoseconds counter, providing together the standard 5B timestamp.* uint32_t timestamp_nsec
- *The timestamp nanoseconds counter.* }
- };

Detailed Description

Info field for incoming job from external RX interfaces.

Field Documentation

uint32_t ezdp_job_rx_interface_info::raw_data[EZDP_JOB_RX_INTERFACE_INFO_WORD_COUNT]

unsigned ezdp_job_rx_interface_info::imem_buf_count

< Indicate the PMU queue depth from which the job was sent.

< Indicate the job budget congestion level reported by flow control at reception time. < Indicate the EMEM buffer budget congestion level reported by flow control at reception time. < Indicate the IMEM buffer budget congestion level reported by flow control at reception time. < Indicate the global congestion level of the system. Based on configuration and congestion level of the IMEM buffers budget, EMEM buffers budget, job budget. This is the number of leading frame buffers allocated by Rx NDMA from IMEM. The remaining buffers (if exist) are allocated in EMEM.

unsigned ezdp_job_rx_interface_info::__pad0__

Reserved bit 15.

unsigned ezdp_job_rx_interface_info::__pad1__

Reserved bit 14.

unsigned ezdp_job_rx_interface_info::icu_succ_parsing_flag

If this flag is cleared the L2-L3 frame header could not be parsed by ICU, and therefore global_congestion_level is invalid and frame CoS contains default value.

unsigned ezdp_job_rx_interface_info::truncation_flag

Indicating the frame data was truncated on the Rx path and the data is illegal.

unsigned ezdp_job_rx_interface_info::__pad2__

Reserved bit 11.

unsigned ezdp_job_rx_interface_info::__pad3__

Reserved bit 10.

unsigned ezdp_job_rx_interface_info::crc_ok_flag

CRC was checked by MAC and was found OK.

Applicable only if checked by MAC flag is on.

unsigned ezdp_job_rx_interface_info::crc_checked_flag

CRC checked by the MAC.

uint8_t ezdp_job_rx_interface_info::timestamp_sec

The timestamp seconds counter is sampled at one with the nanoseconds counter, providing together the standard 5B timestamp.

Applicable only when frame descriptor time_stamp flag is on. The timestamp is sampled by the Rx MAC when the frame is speculated to be within the PTP length range (based on frame length) to minimize overheads.

uint32_t ezdp_job_rx_interface_info::timestamp_nsec

The timestamp nanoseconds counter.

Together with timestamp_sec provide the standard 5B timestemp. Applicable only when frame descriptor time_stamp flag is on. The timestamp is sampled by the Rx MAC when the frame is speculated to be within the PTP length range (based on frame length) to minimize overheads.

union { ... }

The documentation for this struct was generated from the following file:

- dpe/dp/include/ezdp_job_defs.h

ezdp_job_rx_loopback_info Struct Reference

Info field for incoming job from loopback ports.

Data Fields

- union {
- uint32_t raw_data [EZDP_JOB_RX_LOOPBACK_INFO_WORD_COUNT]
- struct {
- uint16_t replication_id
- *The multicast replication id is a running number from 0 to the replication_count (include replication_count), provided by a SW unicast frame sent to the TM loopback port.* unsigned __pad0__: EZDP_JOB_RX_LOOPBACK_INFO_RESERVED0_15_SIZE
- *Reserved bits 0 to 15.* unsigned __pad1__: EZDP_JOB_RX_LOOPBACK_INFO_RESERVED32_63_SIZE
- *Reserved bits 32 to 63.* }
- };

Detailed Description

Info field for incoming job from loopback ports.

Field Documentation

uint32_t ezdp_job_rx_loopback_info::raw_data[EZDP_JOB_RX_LOOPBACK_INFO_WORD_COUNT]

uint16_t ezdp_job_rx_loopback_info::replication_id

The multicast replication id is a running number from 0 to the replication_count (include replication_count), provided by a SW unicast frame sent to the TM loopback port.

unsigned ezdp_job_rx_loopback_info::__pad0__

Reserved bits 0 to 15.

unsigned ezdp_job_rx_loopback_info::__pad1__

Reserved bits 32 to 63.

union { ... }

The documentation for this struct was generated from the following file:

- dpe/dp/include/ezdp_job_defs.h

ezdp_job_rx_timer_info Struct Reference

Info field for incoming timer job (PMU Timer).

Data Fields

- union {
- uint32_t raw_data [EZDP_JOB_RX_TIMER_INFO_WORD_COUNT]
- struct {
- unsigned __pad0__: EZDP_JOB_RX_TIMER_INFO_RESERVED16_31_SIZE
- *Reserved 16 to 31.* uint8_t timer_id
- *Job originating unique timer ID.* unsigned __pad1__: EZDP_JOB_RX_TIMER_INFO_RESERVED0_8_SIZE
- *Reserved bits 0 to 7.* uint32_t event_id
- *The timer event number represents a sequentially advancing event in the interval generated by the timer.* }
- };

Detailed Description

Info field for incoming timer job (PMU Timer).

Field Documentation

uint32_t ezdp_job_rx_timer_info::raw_data[EZDP_JOB_RX_TIMER_INFO_WORD_COUNT]

unsigned ezdp_job_rx_timer_info::__pad0__

Reserved 16 to 31.

uint8_t ezdp_job_rx_timer_info::timer_id

Job originating unique timer ID.

unsigned ezdp_job_rx_timer_info::__pad1__

Reserved bits 0 to 7.

uint32_t ezdp_job_rx_timer_info::event_id

The timer event number represents a sequentially advancing event in the interval generated by the timer.

union { ... }

The documentation for this struct was generated from the following file:

- dpe/dp/include/ezdp_job_defs.h

ezdp_job_rx_user_info Struct Reference

Info field for incoming frame job from generic user forwarding.

Data Fields

- union {
- uint32_t raw_data [EZDP_JOB_RX_USER_INFO_WORD_COUNT]
- struct {
- uint32_t user_data_info0
- *User data info0.* uint32_t user_data_info1
- *User data info1.* }
- };

Detailed Description

Info field for incoming frame job from generic user forwarding.

Field Documentation

uint32_t ezdp_job_rx_user_info::raw_data[EZDP_JOB_RX_USER_INFO_WORD_COUNT]

uint32_t ezdp_job_rx_user_info::user_data_info0

User data info0.

uint32_t ezdp_job_rx_user_info::user_data_info1

User data info1.

union { ... }

The documentation for this struct was generated from the following file:

- dpe/dp/include/ezdp_job_defs.h

ezdp_job_transmit_cmd_info Struct Reference

Job container send out request info.

Data Fields

- union {
- ezdp_job_transmit_cmd_info_t raw_data
- struct {
- unsigned __pad0__: EZDP_JOB_TRANSMIT_CMD_INFO_RESERVED12_15_SIZE
- *Reserved bits 12 to 15.* unsigned side: EZDP_JOB_TRANSMIT_CMD_INFO_SIDE_SIZE
- *< Define the send mode* unsigned output_channel: EZDP_JOB_TRANSMIT_CMD_INFO_OUTPUT_CHANNEL_SIZE
- *Output channel id.* }
- };

Detailed Description

Job container send out request info.

Field Documentation

ezdp_job_transmit_cmd_info_t ezdp_job_transmit_cmd_info::raw_data

unsigned ezdp_job_transmit_cmd_info::__pad0__

Reserved bits 12 to 15.

unsigned ezdp_job_transmit_cmd_info::side

< Define the send mode

The TM/PMU side. Define which side should handle the job done request (PMU implements the queuing and JD read operation)

unsigned ezdp_job_transmit_cmd_info::output_channel

Output channel id.

Define direct mapping to destination port. Also used to select (by configuration) to one of the 4 PMU TM bypass queues that will hold the packet. Applicable only for TM bypass mode

union { ... }

The documentation for this struct was generated from the following file:

- dpe/dp/include/ezdp_job_defs.h

ezdp_job_tx_info Struct Reference

Info field for transmitting frame job (TM mode is full or tm qos bypass).

Data Fields

- union {
- uint32_t raw_data [EZDP_JOB_TX_INFO_WORD_COUNT]
- struct {
- unsigned wred_color: EZDP_JOB_TX_INFO_WRED_COLOR_SIZE
- *Policer WRED Color field.* unsigned __pad0__: EZDP_JOB_TX_INFO_RESERVED25_29_SIZE
- *Reserved bits 25 to 29.* unsigned packet_switch_id_select: EZDP_JOB_TX_INFO_PACKET_SWITCH_ID_SELECT_SIZE
- *Selects an entry in packet switch table.* union {
- uint16_t replication_count
- *Defines the number of repeated job descriptors generated by the PMU replication logic.* uint16_t user_info
- *User information that gets echoed back to the job created by the loopback port.* uint16_t explicit_packet_switch_id
- *Select an entry from the PSID table.* uint16_t dest_queue
- *Define which PMU queue should be selected for the frame arriving from a loopback port.* }
- unsigned __pad1__: EZDP_JOB_TX_INFO_RESERVED61_SIZE
- *< Controls the outgoing packet drop policy.* unsigned qos_bypass: EZDP_JOB_TX_INFO_QOS_BYPASS_SIZE
- *TM QOS bypass.* unsigned __pad2__: EZDP_JOB_TX_INFO_RESERVED59_SIZE
- *Reserved bit 59.* unsigned __pad3__: EZDP_JOB_TX_INFO_RESERVED52_55_SIZE
- *< Selects traffic manager output queue mapping mode.* unsigned side: EZDP_JOB_TX_INFO_SIDE_SIZE
- *Side to forward the frame to.* unsigned flow_id: EZDP_JOB_TX_INFO_FLOW_ID_SIZE
- *Flow ID field.* unsigned stat_code_profile1: EZDP_JOB_TX_INFO_STAT_CODE_PROFILE1_SIZE
- *Statistics Code Profile 1 field.* unsigned stat_code_profile2: EZDP_JOB_TX_INFO_STAT_CODE_PROFILE2_SIZE
- *Statistics Code Profile 2 field.* unsigned __pad4__: EZDP_JOB_TX_INFO_RESERVED88_90_SIZE
- *Reserved bits 88 to 90.* unsigned stat_stream_id: EZDP_JOB_TX_INFO_STAT_STREAM_ID_SIZE
- *Statistic stream id field.* unsigned wred_flow_template_profile: EZDP_JOB_TX_INFO_WRED_FLOW_TEMPLATE_PROFILE_SIZE
- *Policer WRED flow template profile id field.* unsigned wred_class_template_profile: EZDP_JOB_TX_INFO_WRED_CLASS_TEMPLATE_PROFILE_SIZE
- *Policer WRED class template profile id field.* uint8_t wred_flow_scale_profile
- *Policer WRED flow absolute scaling profile id field.* uint8_t wred_class_scale_profile
- *Policer WRED class absolute scaling profile id field.* unsigned __pad5__: EZDP_JOB_TX_INFO_RESERVED102_103_SIZE
- *Reserved bits 102 to 103.* unsigned inter_packet_gap_control: EZDP_JOB_TX_INFO_INTER_PACKET_GAP_CONTROL_SIZE
- *Select the inter-packet gap emulation mode.* unsigned inter_packet_gap: EZDP_JOB_TX_INFO_INTER_PACKET_GAP_SIZE
- *The Inter Packet Gap emulation field provides per frame overhead for TM IPG emulation logic.* }
- };

Detailed Description

Info field for transmitting frame job (TM mode is full or tm qos bypass).

Field Documentation

uint32_t ezdp_job_tx_info::raw_data[EZDP_JOB_TX_INFO_WORD_COUNT]

unsigned ezdp_job_tx_info::wred_color

Policer WRED Color field.

Defines the color of the packet, which is used at TM enqueue point for implementation of the WRED algorithm.

unsigned ezdp_job_tx_info::__pad0__

Reserved bits 25 to 29.

unsigned ezdp_job_tx_info::packet_switch_id_select

Selects an entry in packet switch table.

Used together with 9 bits from L1 entry to get the entry id in packet switch table.

uint16_t ezdp_job_tx_info::replication_count

Defines the number of repeated job descriptors generated by the PMU replication logic.

Applicable only when destination port is loopback and multicast_mode is REPLICATION. The replication logic counts from zero to replication_count (meaning that replication_count determines the extra jobs created in addition to the first loopback job). Single job with no further repetition is a legal scenario (identified by replication_num=0 and multicast_mode=REPLICATION).

uint16_t ezdp_job_tx_info::user_info

User information that gets echoed back to the job created by the loopback port.

Applicable only when destination port is loopback and no multicast_mode = REPLICATION port is not configured to override PMU queue selection. Applicable only when destination port is loopback.

uint16_t ezdp_job_tx_info::explicit_packet_switch_id

Select an entry from the PSID table.

uint16_t ezdp_job_tx_info::dest_queue

Define which PMU queue should be selected for the frame arriving from a loopback port.

Applicable only when destination port is loopback and configured to enable override PMU queue section from SW. NOTE: The job is inserted to the same PMU side to which loopback is block to.

unsigned ezdp_job_tx_info::__pad1__

< Controls the outgoing packet drop policy.

Reserved bit 61

unsigned ezdp_job_tx_info::qos_bypass

TM QOS bypass.

When flag is on, the packet runs through TM, bypassing the per flow queue, directly to the packet switching table selection. Only EXPLISIT and BASE packet switch modes are available when flag is ON. TM control fields flow_id, WRED controls and WRED statistics are not applicable.

unsigned ezdp_job_tx_info::__pad2__

Reserved bit 59.

unsigned ezdp_job_tx_info::__pad3__

< Selects traffic manager output queue mapping mode.

The TM port switching table selects an output channel, and the output channel further selects the side of a Tx port it is assigned to. Reserved bits 52 to 55.

unsigned ezdp_job_tx_info::side

Side to forward the frame to.

unsigned ezdp_job_tx_info::flow_id

Flow ID field.

The flow ID uniquely selects an L4 entity in the target TM, and TM configuration topology further maps it through TM scheduling levels down to the target port.

unsigned ezdp_job_tx_info::stat_code_profile1

Statistics Code Profile 1 field.

Used in TM WRED statistics reporting, participates in generation of a 5-bit policer drop code in either topology based statistics or stream id based statistics.

unsigned ezdp_job_tx_info::stat_code_profile2

Statistics Code Profile 2 field.

Participates together with Statistics Code Profile 1 in generation of a 5-bit policer drop code in either topology based statistics or stream ID based statistics. It provides 0-3 bits replacing bits from the aggregated policer result when mapping stream ID based reporting. Additionally it participates in mapping the topology based reporting together with profile 1 code and pass/drop decision.

unsigned ezdp_job_tx_info::__pad4__

Reserved bits 88 to 90.

unsigned ezdp_job_tx_info::stat_stream_id

Statistic stream id field.

The stat_stream_id field select the block to be reported, where the reporting code selects the counter in the block. It points to the base location where a continuous block with statistic counters associated with the packet can be updated. On WRED reporting, together with policer codes, statistic code profile 1 and statistic code profile 2 stream based reporting code is generated, mapping one counter in a block of up to 32 counters for the reported stream.

unsigned ezdp_job_tx_info::wred_flow_template_profile

Policer WRED flow template profile id field.

This field selects one of sixteen RED behavior templates to be used by the WRED algorithm at the flow level (L4). Each template holds eight profiles (one per color), totaling 128 templates. A single entry in the template provides per priority the relative percentage of different ranges of the WRED graph for that priority.

unsigned ezdp_job_tx_info::wred_class_template_profile

Policer WRED class template profile id field.

This field selects one of sixteen RED behavior templates to be used by the WRED algorithm at the class level (L3). Each template holds eight profiles (one per color), totaling 128 templates. A single entry in the template provides per priority the relative percentage of different ranges of the WRED graph for that priority.

uint8_t ezdp_job_tx_info::wred_flow_scale_profile

Policer WRED flow absolute scaling profile id field.

The policer_flow_scale_index (PFAI) selects an absolute scaling factor index from a 256-entry table, used in combination with the selected policer_flow_template_index to define WRED algorithm behavior at the flow level (L4). Normalized regions provided per priority by the template are scaled to describe the final WRED graphs.

uint8_t ezdp_job_tx_info::wred_class_scale_profile

Policer WRED class absolute scaling profile id field.

This field selects an absolute scaling factor index from a 256-entry table, used in combination with the selected policer_class_template_index to define WRED algorithm behavior at the class level (L3). Normalized regions provided per priority by the template are scaled to describe the final WRED graphs.

unsigned ezdp_job_tx_info::__pad5__

Reserved bits 102 to 103.

unsigned ezdp_job_tx_info::inter_packet_gap_control

Select the inter-packet gap emulation mode.

unsigned ezdp_job_tx_info::inter_packet_gap

The Inter Packet Gap emulation field provides per frame overhead for TM IPG emulation logic.

The inter_packet_gap_enum field comprises 5-bit mantissa (-16 to 15) and one bit for exponent selection mode. (TM configuration has global exponent configuration to scale the mantissa by x1, x4, ,16 or x128.) The covered range is: (-16..15) x (1/4/16/128) bytes. It affects packet length taking into account parts that are added or excluded from shaping and fairness algorithms as well as from statistics reporting. The TM can independently use inter_packet_gap_enum for altering frame length on shaper accounting and for statistics reporting. Additionally inter_packet_gap_enum can affect TM WFQ scheduling at each level independently. Besides emulating physical media characteristics (such as Ethernet inter-packet gap bytes), inter_packet_gap_enum may account for parts of a packet that are either ignored or added in TM algorithms (for example, it may consider only layer 3 IP traffic and use inter_packet_gap_enum to exclude layer 2 header as well as four CRC bytes from the packet length for shaping and scheduling). inter_packet_gap_enum can also be used to account for proprietary header bytes.

union { ... }

The documentation for this struct was generated from the following file:

- dpe/dp/include/ezdp_job_defs.h

ezdp_large_linked_buffers_desc Struct Reference

Large linked buffers descriptor.

Data Fields

- struct ezdp_linked_buffers_desc_line line [EZDP_LARGE_LBD]

Array of 2 linked buffers lines, which contains up to 6 buffers.

Detailed Description

Large linked buffers descriptor.

May hold up to 6 buffs

Field Documentation

struct ezdp_linked_buffers_desc_line ezdp_large_linked_buffers_desc::line[EZDP_LARGE_LBD] [read]

Array of 2 linked buffers lines, which contains up to 6 buffers.

The documentation for this struct was generated from the following file:

- dpe/dp/include/ezdp_frame_defs.h

ezdp_linked_buffers_desc Struct Reference

A generic linked buffers descriptor.

Data Fields

- struct ezdp_linked_buffers_desc_line line [0]

Linked buffers lines.

Detailed Description

A generic linked buffers descriptor.

Good for pointers

Field Documentation

struct ezdp_linked_buffers_desc_line ezdp_linked_buffers_desc::line[0] [read]

Linked buffers lines.

The documentation for this struct was generated from the following file:

- dpe/dp/include/ezdp_frame_defs.h

ezdp_linked_buffers_desc_line Struct Reference

LBD Line data structure.

Data Fields

- uint8_t ecc
- *ECC.* struct ezdp_buffer_info buf_info [EZDP_LINKED_BUFFER_DESC_LINE_NUMBER_OF_BUFFERS_DESC]
- *Array of 3 buffers info.* struct ezdp_buffer_desc buf_desc [EZDP_LINKED_BUFFER_DESC_LINE_NUMBER_OF_BUFFERS_DESC]

Array of 3 buffers.

Detailed Description

LBD Line data structure.

Field Documentation

uint8_t ezdp_linked_buffers_desc_line::ecc

ECC.

struct ezdp_buffer_info ezdp_linked_buffers_desc_line::buf_info[EZDP_LINKED_BUFFER_DESC_LINE_NUMBER_OF_BUFFERS_DESC] [read]

Array of 3 buffers info.

struct ezdp_buffer_desc ezdp_linked_buffers_desc_line::buf_desc[EZDP_LINKED_BUFFER_DESC_LINE_NUMBER_OF_BUFFERS_DESC] [read]

Array of 3 buffers.

The documentation for this struct was generated from the following file:

- dpe/dp/include/ezdp_frame_defs.h

ezdp_list_cfg Struct Reference

list queue configuration data structure

Data Fields

- ezdp_sum_addr_t head
- *Head address of the list.* ezdp_sum_addr_t tail
- *Tail address of the list.* ezdp_mem_pool_t queue_memory_pool

Memory pool for the queue elements.

Detailed Description

list queue configuration data structure

Field Documentation

ezdp_sum_addr_t ezdp_list_cfg::head

Head address of the list.

ezdp_sum_addr_t ezdp_list_cfg::tail

Tail address of the list.

ezdp_mem_pool_t ezdp_list_cfg::queue_memory_pool

Memory pool for the queue elements.

The documentation for this struct was generated from the following file:

- dpe/dp/include/ezdp_queue_defs.h

ezdp_lookup_ext_tcam_16B_data_result_element Struct Reference

Lookup external tcam 16 Byte associated data only result.

Data Fields

- union {
- uint32_t raw_data [EZDP_LOOKUP_EXT_TCAM_16B_DATA_RESULT_ELEMENT_WORD_COUNT]
- struct {
- unsigned valid: EZDP_LOOKUP_EXT_TCAM_4B_DATA_RESULT_ELEMENT_VALID_SIZE
- *Result valid.* unsigned match: EZDP_LOOKUP_EXT_TCAM_4B_DATA_RESULT_ELEMENT_MATCH_SIZE
- *Match.* unsigned lookup_error: EZDP_LOOKUP_EXT_TCAM_16B_DATA_RESULT_ELEMENT_LOOKUP_ERROR_SIZE
- *< Result element type.* unsigned truncated: EZDP_LOOKUP_EXT_TCAM_16B_DATA_RESULT_ELEMENT_TRUNCATED_SIZE
- *Truncated response indication - result returned was truncated due to being larger than result_len.* unsigned __pad0__: EZDP_LOOKUP_EXT_TCAM_16B_DATA_RESULT_ELEMENT_RESERVED24_SIZE
- *Reserved bit 24.* uint8_t assoc_data [EZDP_LOOKUP_EXT_TCAM_16B_DATA_RESULT_ELEMENT_ASSOC_DATA_COUNT]
- *120 bit Associated data* }
- };

Detailed Description

Lookup external tcam 16 Byte associated data only result.

Field Documentation

uint32_t ezdp_lookup_ext_tcam_16B_data_result_element::raw_data[EZDP_LOOKUP_EXT_TCAM_16B_DATA_RESULT_ELEMENT_WORD_COUNT]

unsigned ezdp_lookup_ext_tcam_16B_data_result_element::valid

Result valid.

Must be set by user. Recommend to set to 1 to match ezdp definition

unsigned ezdp_lookup_ext_tcam_16B_data_result_element::match

Match.

Must be set by user. Recommend to set to 1 to match ezdp definition

unsigned ezdp_lookup_ext_tcam_16B_data_result_element::lookup_error

< Result element type.

Must be set by user. Recommend to set to EZDP_USER_DEFINED_ASSOC_DATA1 or EZDP_USER_DEFINED_ASSOC_DATA2 or EZDP_USER_DEFINED_ASSOC_DATA3 to match other response types definition Lookup error. Will be on if any error occurred. Applicable for the first result element only.

unsigned ezdp_lookup_ext_tcam_16B_data_result_element::truncated

Truncated response indication - result returned was truncated due to being larger than result_len.

Applicable for the first result element only

unsigned ezdp_lookup_ext_tcam_16B_data_result_element::__pad0__

Reserved bit 24.

uint8_t ezdp_lookup_ext_tcam_16B_data_result_element::assoc_data[EZDP_LOOKUP_EXT_TCAM_16B_DATA_RESULT_ELEMENT_ASSOC_DATA_COUNT]

120 bit Associated data

union { ... }

The documentation for this struct was generated from the following file:

- dpe/dp/include/ezdp_search_defs.h

ezdp_lookup_ext_tcam_32B_data_result_element Struct Reference

Lookup external tcam 32 Byte associated data only result.

Data Fields

- union {
- uint32_t raw_data [EZDP_LOOKUP_EXT_TCAM_32B_DATA_RESULT_ELEMENT_WORD_COUNT]
- struct {
- unsigned valid: EZDP_LOOKUP_EXT_TCAM_4B_DATA_RESULT_ELEMENT_VALID_SIZE
- *Result valid.* unsigned match: EZDP_LOOKUP_EXT_TCAM_4B_DATA_RESULT_ELEMENT_MATCH_SIZE
- *Match.* unsigned lookup_error: EZDP_LOOKUP_EXT_TCAM_32B_DATA_RESULT_ELEMENT_LOOKUP_ERROR_SIZE
- *< Result element type.* unsigned truncated: EZDP_LOOKUP_EXT_TCAM_32B_DATA_RESULT_ELEMENT_TRUNCATED_SIZE
- *Truncated response indication - result returned was truncated due to being larger than result_len.* unsigned __pad0__: EZDP_LOOKUP_EXT_TCAM_32B_DATA_RESULT_ELEMENT_RESERVED24_SIZE
- *Reserved bit 24.* uint8_t assoc_data [EZDP_LOOKUP_EXT_TCAM_32B_DATA_RESULT_ELEMENT_ASSOC_DATA_COUNT]
- *248 bit Associated data* }
- };

Detailed Description

Lookup external tcam 32 Byte associated data only result.

Field Documentation

uint32_t ezdp_lookup_ext_tcam_32B_data_result_element::raw_data[EZDP_LOOKUP_EXT_TCAM_32B_DATA_RESULT_ELEMENT_WORD_COUNT]

unsigned ezdp_lookup_ext_tcam_32B_data_result_element::valid

Result valid.

Must be set by user. Recommend to set to 1 to match ezdp definition

unsigned ezdp_lookup_ext_tcam_32B_data_result_element::match

Match.

Must be set by user. Recommend to set to 1 to match ezdp definition

unsigned ezdp_lookup_ext_tcam_32B_data_result_element::lookup_error

< Result element type.

Must be set by user. Recommend to set to EZDP_USER_DEFINED_ASSOC_DATA1 or EZDP_USER_DEFINED_ASSOC_DATA2 or EZDP_USER_DEFINED_ASSOC_DATA3 to match other response types definition Lookup error. Will be on if any error occurred. Applicable for the first result element only.

unsigned ezdp_lookup_ext_tcam_32B_data_result_element::truncated

Truncated response indication - result returned was truncated due to being larger than result_len.

Applicable for the first result element only

unsigned ezdp_lookup_ext_tcam_32B_data_result_element::__pad0__

Reserved bit 24.

uint8_t ezdp_lookup_ext_tcam_32B_data_result_element::assoc_data[EZDP_LOOKUP_EXT_TCAM_32B_DATA_RESULT_ELEMENT_ASSOC_DATA_COUNT]

248 bit Associated data

union { ... }

The documentation for this struct was generated from the following file:

- dpe/dp/include/ezdp_search_defs.h

ezdp_lookup_ext_tcam_4B_data_result_element Struct Reference

Lookup external tcam 4 Byte associated data only result.

Data Fields

- union {
- ezdp_lookup_ext_tcam_4B_data_result_element_t raw_data
- struct {
- unsigned valid: EZDP_LOOKUP_EXT_TCAM_4B_DATA_RESULT_ELEMENT_VALID_SIZE
- *Result valid.* unsigned match: EZDP_LOOKUP_EXT_TCAM_4B_DATA_RESULT_ELEMENT_MATCH_SIZE
- *Match.* unsigned lookup_error: EZDP_LOOKUP_EXT_TCAM_4B_DATA_RESULT_ELEMENT_LOOKUP_ERROR_SIZE
- *< Result element type.* unsigned truncated: EZDP_LOOKUP_EXT_TCAM_4B_DATA_RESULT_ELEMENT_TRUNCATED_SIZE
- *Truncated response indication - result returned was truncated due to being larger than result_len.* unsigned __pad0__: EZDP_LOOKUP_EXT_TCAM_4B_DATA_RESULT_ELEMENT_RESERVED24_SIZE
- *Reserved bit 24.* uint8_t assoc_data [EZDP_LOOKUP_EXT_TCAM_4B_DATA_RESULT_ELEMENT_ASSOC_DATA_COUNT]
- *24 bit Associated data* }
- };

Detailed Description

Lookup external tcam 4 Byte associated data only result.

Field Documentation

ezdp_lookup_ext_tcam_4B_data_result_element_t ezdp_lookup_ext_tcam_4B_data_result_element::raw_data

unsigned ezdp_lookup_ext_tcam_4B_data_result_element::valid

Result valid.

Must be set by user. Recommend to set to 1 to match ezdp definition

unsigned ezdp_lookup_ext_tcam_4B_data_result_element::match

Match.

Must be set by user. Recommend to set to 1 to match ezdp definition

unsigned ezdp_lookup_ext_tcam_4B_data_result_element::lookup_error

< Result element type.

Must be set by user. Recommend to set to EZDP_USER_DEFINED_ASSOC_DATA1 or EZDP_USER_DEFINED_ASSOC_DATA2 or EZDP_USER_DEFINED_ASSOC_DATA3 to match other response types definition Lookup error. Will be on if any error occurred. Applicable for the first result element only.

unsigned ezdp_lookup_ext_tcam_4B_data_result_element::truncated

Truncated response indication - result returned was truncated due to being larger than result_len.

Applicable for the first result element only

unsigned ezdp_lookup_ext_tcam_4B_data_result_element::__pad0__

Reserved bit 24.

uint8_t ezdp_lookup_ext_tcam_4B_data_result_element::assoc_data[EZDP_LOOKUP_EXT_TCAM_4B_DATA_RESULT_ELEMENT_ASSOC_DATA_COUNT]

24 bit Associated data

union { ... }

The documentation for this struct was generated from the following file:

- dpe/dp/include/ezdp_search_defs.h

ezdp_lookup_ext_tcam_8B_data_result_element Struct Reference

Lookup external tcam 8 Byte associated data only result.

Data Fields

- union {
- uint32_t raw_data [EZDP_LOOKUP_EXT_TCAM_8B_DATA_RESULT_ELEMENT_WORD_COUNT]
- struct {
- unsigned valid: EZDP_LOOKUP_EXT_TCAM_4B_DATA_RESULT_ELEMENT_VALID_SIZE
- *Result valid.* unsigned match: EZDP_LOOKUP_EXT_TCAM_4B_DATA_RESULT_ELEMENT_MATCH_SIZE
- *Match.* unsigned lookup_error: EZDP_LOOKUP_EXT_TCAM_8B_DATA_RESULT_ELEMENT_LOOKUP_ERROR_SIZE
- *< Result element type.* unsigned truncated: EZDP_LOOKUP_EXT_TCAM_8B_DATA_RESULT_ELEMENT_TRUNCATED_SIZE
- *Truncated response indication - result returned was truncated due to being larger than result_len.* unsigned __pad0__: EZDP_LOOKUP_EXT_TCAM_8B_DATA_RESULT_ELEMENT_RESERVED24_SIZE
- *Reserved bit 24.* uint8_t assoc_data [EZDP_LOOKUP_EXT_TCAM_8B_DATA_RESULT_ELEMENT_ASSOC_DATA_COUNT]
- *56 bit Associated data* }
- };

Detailed Description

Lookup external tcam 8 Byte associated data only result.

Field Documentation

uint32_t ezdp_lookup_ext_tcam_8B_data_result_element::raw_data[EZDP_LOOKUP_EXT_TCAM_8B_DATA_RESULT_ELEMENT_WORD_COUNT]

unsigned ezdp_lookup_ext_tcam_8B_data_result_element::valid

Result valid.

Must be set by user. Recommend to set to 1 to match ezdp definition

unsigned ezdp_lookup_ext_tcam_8B_data_result_element::match

Match.

Must be set by user. Recommend to set to 1 to match ezdp definition

unsigned ezdp_lookup_ext_tcam_8B_data_result_element::lookup_error

< Result element type.

Must be set by user. Recommend to set to EZDP_USER_DEFINED_ASSOC_DATA1 or EZDP_USER_DEFINED_ASSOC_DATA2 or EZDP_USER_DEFINED_ASSOC_DATA3 to match other response types definition Lookup error. Will be on if any error occurred. Applicable for the first result element only.

unsigned ezdp_lookup_ext_tcam_8B_data_result_element::truncated

Truncated response indication - result returned was truncated due to being larger than result_len.

Applicable for the first result element only

unsigned ezdp_lookup_ext_tcam_8B_data_result_element::__pad0__

Reserved bit 24.

uint8_t ezdp_lookup_ext_tcam_8B_data_result_element::assoc_data[EZDP_LOOKUP_EXT_TCAM_8B_DATA_RESULT_ELEMENT_ASSOC_DATA_COUNT]

56 bit Associated data

union { ... }

The documentation for this struct was generated from the following file:

- dpe/dp/include/ezdp_search_defs.h

ezdp_lookup_ext_tcam_index_16B_data_result_element Struct Reference

Lookup external tcam index result with 16 Byte associated data.

Data Fields

- union {
- uint32_t raw_data [EZDP_LOOKUP_EXT_TCAM_INDEX_16B_DATA_RESULT_ELEMENT_WORD_COUNT]
- struct {
- unsigned valid: EZDP_LOOKUP_EXT_TCAM_INDEX_16B_DATA_RESULT_ELEMENT_VALID_SIZE
- *Result valid.* unsigned match: EZDP_LOOKUP_EXT_TCAM_INDEX_16B_DATA_RESULT_ELEMENT_MATCH_SIZE
- *Match.* unsigned lookup_error: EZDP_LOOKUP_EXT_TCAM_INDEX_16B_DATA_RESULT_ELEMENT_LOOKUP_ERROR_SIZE
- *< Result element type.* unsigned truncated: EZDP_LOOKUP_EXT_TCAM_INDEX_16B_DATA_RESULT_ELEMENT_TRUNCATED_SIZE
- *Truncated response indication - result returned was truncated due to being larger than result_len.* unsigned __pad0__: EZDP_LOOKUP_EXT_TCAM_INDEX_16B_DATA_RESULT_ELEMENT_RESERVED23_24_SIZE
- *Reserved bits 23 to 24.* unsigned device_id: EZDP_LOOKUP_EXT_TCAM_INDEX_16B_DATA_RESULT_ELEMENT_DEVICE_ID_SIZE
- *Responding device id.* unsigned index: EZDP_LOOKUP_EXT_TCAM_INDEX_16B_DATA_RESULT_ELEMENT_INDEX_SIZE
- *compare result index* uint8_t assoc_data [EZDP_LOOKUP_EXT_TCAM_INDEX_16B_DATA_RESULT_ELEMENT_ASSOC_DATA_COUNT]
- *16B Associated data* }
- };

Detailed Description

Lookup external tcam index result with 16 Byte associated data.

Field Documentation

uint32_t ezdp_lookup_ext_tcam_index_16B_data_result_element::raw_data[EZDP_LOOKUP_EXT_TCAM_INDEX_16B_DATA_RESULT_ELEMENT_WORD_COUNT]

unsigned ezdp_lookup_ext_tcam_index_16B_data_result_element::valid

Result valid.

unsigned ezdp_lookup_ext_tcam_index_16B_data_result_element::match

Match.

unsigned ezdp_lookup_ext_tcam_index_16B_data_result_element::lookup_error

< Result element type.

Lookup error. Will be on if any error occurred. For the exact error look in the error flags returned in ezdp_lookup_ext_tcam_retval_t. Applicable for the first result element only.

unsigned ezdp_lookup_ext_tcam_index_16B_data_result_element::truncated

Truncated response indication - result returned was truncated due to being larger than result_len.

Applicable for the first result element only

unsigned ezdp_lookup_ext_tcam_index_16B_data_result_element::__pad0__

Reserved bits 23 to 24.

unsigned ezdp_lookup_ext_tcam_index_16B_data_result_element::device_id

Responding device id.

unsigned ezdp_lookup_ext_tcam_index_16B_data_result_element::index

compare result index

uint8_t ezdp_lookup_ext_tcam_index_16B_data_result_element::assoc_data[EZDP_LOOKUP_EXT_TCAM_INDEX_16B_DATA_RESULT_ELEMENT_ASSOC_DATA_COUNT]

16B Associated data

union { ... }

The documentation for this struct was generated from the following file:

- dpe/dp/include/ezdp_search_defs.h

ezdp_lookup_ext_tcam_index_32B_data_result_element Struct Reference

Lookup external tcam index result with 32 Byte associated data.

Data Fields

- union {
- uint32_t raw_data [EZDP_LOOKUP_EXT_TCAM_INDEX_32B_DATA_RESULT_ELEMENT_WORD_COUNT]
- struct {
- unsigned valid: EZDP_LOOKUP_EXT_TCAM_INDEX_32B_DATA_RESULT_ELEMENT_VALID_SIZE
- *Result valid.* unsigned match: EZDP_LOOKUP_EXT_TCAM_INDEX_32B_DATA_RESULT_ELEMENT_MATCH_SIZE
- *Match.* unsigned lookup_error: EZDP_LOOKUP_EXT_TCAM_INDEX_32B_DATA_RESULT_ELEMENT_LOOKUP_ERROR_SIZE
- *< Result element type.* unsigned truncated: EZDP_LOOKUP_EXT_TCAM_INDEX_32B_DATA_RESULT_ELEMENT_TRUNCATED_SIZE
- *Truncated response indication - result returned was truncated due to being larger than result_len.* unsigned __pad0__: EZDP_LOOKUP_EXT_TCAM_INDEX_32B_DATA_RESULT_ELEMENT_RESERVED23_24_SIZE
- *Reserved bits 23 to 24.* unsigned device_id: EZDP_LOOKUP_EXT_TCAM_INDEX_32B_DATA_RESULT_ELEMENT_DEVICE_ID_SIZE
- *Responding device id.* unsigned index: EZDP_LOOKUP_EXT_TCAM_INDEX_32B_DATA_RESULT_ELEMENT_INDEX_SIZE
- *compare result index* uint8_t assoc_data [EZDP_LOOKUP_EXT_TCAM_INDEX_32B_DATA_RESULT_ELEMENT_ASSOC_DATA_COUNT]
- *32B Associated data* }
- };

Detailed Description

Lookup external tcam index result with 32 Byte associated data.

Field Documentation

uint32_t ezdp_lookup_ext_tcam_index_32B_data_result_element::raw_data[EZDP_LOOKUP_EXT_TCAM_INDEX_32B_DATA_RESULT_ELEMENT_WORD_COUNT]

unsigned ezdp_lookup_ext_tcam_index_32B_data_result_element::valid

Result valid.

unsigned ezdp_lookup_ext_tcam_index_32B_data_result_element::match

Match.

unsigned ezdp_lookup_ext_tcam_index_32B_data_result_element::lookup_error

< Result element type.

Lookup error. Will be on if any error occurred. For the exact error look in the error flags returned in ezdp_lookup_ext_tcam_retval_t. Applicable for the first result element only.

unsigned ezdp_lookup_ext_tcam_index_32B_data_result_element::truncated

Truncated response indication - result returned was truncated due to being larger than result_len.

Applicable for the first result element only

unsigned ezdp_lookup_ext_tcam_index_32B_data_result_element::__pad0__

Reserved bits 23 to 24.

unsigned ezdp_lookup_ext_tcam_index_32B_data_result_element::device_id

Responding device id.

unsigned ezdp_lookup_ext_tcam_index_32B_data_result_element::index

compare result index

uint8_t ezdp_lookup_ext_tcam_index_32B_data_result_element::assoc_data[EZDP_LOOKUP_EXT_TCAM_INDEX_32B_DATA_RESULT_ELEMENT_ASSOC_DATA_COUNT]

32B Associated data

union { ... }

The documentation for this struct was generated from the following file:

- dpe/dp/include/ezdp_search_defs.h

ezdp_lookup_ext_tcam_index_4B_data_result_element Struct Reference

Lookup external tcam index result with 4 Byte associated data.

Data Fields

- union {
- uint32_t raw_data [EZDP_LOOKUP_EXT_TCAM_INDEX_4B_DATA_RESULT_ELEMENT_WORD_COUNT]
- struct {
- unsigned valid: EZDP_LOOKUP_EXT_TCAM_INDEX_4B_DATA_RESULT_ELEMENT_VALID_SIZE
- *Result valid.* unsigned match: EZDP_LOOKUP_EXT_TCAM_INDEX_4B_DATA_RESULT_ELEMENT_MATCH_SIZE
- *Match.* unsigned lookup_error: EZDP_LOOKUP_EXT_TCAM_INDEX_4B_DATA_RESULT_ELEMENT_LOOKUP_ERROR_SIZE
- *< Result element type.* unsigned truncated: EZDP_LOOKUP_EXT_TCAM_INDEX_4B_DATA_RESULT_ELEMENT_TRUNCATED_SIZE
- *Truncated response indication - result returned was truncated due to being larger than result_len.* unsigned __pad0__: EZDP_LOOKUP_EXT_TCAM_INDEX_4B_DATA_RESULT_ELEMENT_RESERVED23_24_SIZE
- *Reserved bits 23 to 24.* unsigned device_id: EZDP_LOOKUP_EXT_TCAM_INDEX_4B_DATA_RESULT_ELEMENT_DEVICE_ID_SIZE
- *Responding device id.* unsigned index: EZDP_LOOKUP_EXT_TCAM_INDEX_4B_DATA_RESULT_ELEMENT_INDEX_SIZE
- *compare result index* uint8_t assoc_data [EZDP_LOOKUP_EXT_TCAM_INDEX_4B_DATA_RESULT_ELEMENT_ASSOC_DATA_COUNT]
- *4B Associated data* }
- };

Detailed Description

Lookup external tcam index result with 4 Byte associated data.

Field Documentation

uint32_t ezdp_lookup_ext_tcam_index_4B_data_result_element::raw_data[EZDP_LOOKUP_EXT_TCAM_INDEX_4B_DATA_RESULT_ELEMENT_WORD_COUNT]

unsigned ezdp_lookup_ext_tcam_index_4B_data_result_element::valid

Result valid.

unsigned ezdp_lookup_ext_tcam_index_4B_data_result_element::match

Match.

unsigned ezdp_lookup_ext_tcam_index_4B_data_result_element::lookup_error

< Result element type.

Lookup error. Will be on if any error occurred. For the exact error look in the error flags returned in ezdp_lookup_ext_tcam_retval_t. Applicable for the first result element only.

unsigned ezdp_lookup_ext_tcam_index_4B_data_result_element::truncated

Truncated response indication - result returned was truncated due to being larger than result_len.

Applicable for the first result element only

unsigned ezdp_lookup_ext_tcam_index_4B_data_result_element::__pad0__

Reserved bits 23 to 24.

unsigned ezdp_lookup_ext_tcam_index_4B_data_result_element::device_id

Responding device id.

unsigned ezdp_lookup_ext_tcam_index_4B_data_result_element::index

compare result index

uint8_t ezdp_lookup_ext_tcam_index_4B_data_result_element::assoc_data[EZDP_LOOKUP_EXT_TCAM_INDEX_4B_DATA_RESULT_ELEMENT_ASSOC_DATA_COUNT]

4B Associated data

union { ... }

The documentation for this struct was generated from the following file:

- dpe/dp/include/ezdp_search_defs.h

ezdp_lookup_ext_tcam_index_8B_data_result_element Struct Reference

Lookup external tcam index result with 8 Byte associated data.

Data Fields

- union {
- uint32_t raw_data [EZDP_LOOKUP_EXT_TCAM_INDEX_8B_DATA_RESULT_ELEMENT_WORD_COUNT]
- struct {
- unsigned valid: EZDP_LOOKUP_EXT_TCAM_INDEX_8B_DATA_RESULT_ELEMENT_VALID_SIZE
- *Result valid.* unsigned match: EZDP_LOOKUP_EXT_TCAM_INDEX_8B_DATA_RESULT_ELEMENT_MATCH_SIZE
- *Match.* unsigned lookup_error: EZDP_LOOKUP_EXT_TCAM_INDEX_8B_DATA_RESULT_ELEMENT_LOOKUP_ERROR_SIZE
- *< Result element type.* unsigned truncated: EZDP_LOOKUP_EXT_TCAM_INDEX_8B_DATA_RESULT_ELEMENT_TRUNCATED_SIZE
- *Truncated response indication - result returned was truncated due to being larger than result_len.* unsigned __pad0__: EZDP_LOOKUP_EXT_TCAM_INDEX_8B_DATA_RESULT_ELEMENT_RESERVED23_24_SIZE
- *Reserved bits 23 to 24.* unsigned device_id: EZDP_LOOKUP_EXT_TCAM_INDEX_8B_DATA_RESULT_ELEMENT_DEVICE_ID_SIZE
- *Responding device id.* unsigned index: EZDP_LOOKUP_EXT_TCAM_INDEX_8B_DATA_RESULT_ELEMENT_INDEX_SIZE
- *compare result index* uint8_t assoc_data [EZDP_LOOKUP_EXT_TCAM_INDEX_8B_DATA_RESULT_ELEMENT_ASSOC_DATA_COUNT]
- *8B Associated data* }
- };

Detailed Description

Lookup external tcam index result with 8 Byte associated data.

Field Documentation

uint32_t ezdp_lookup_ext_tcam_index_8B_data_result_element::raw_data[EZDP_LOOKUP_EXT_TCAM_INDEX_8B_DATA_RESULT_ELEMENT_WORD_COUNT]

unsigned ezdp_lookup_ext_tcam_index_8B_data_result_element::valid

Result valid.

unsigned ezdp_lookup_ext_tcam_index_8B_data_result_element::match

Match.

unsigned ezdp_lookup_ext_tcam_index_8B_data_result_element::lookup_error

< Result element type.

Lookup error. Will be on if any error occurred. For the exact error look in the error flags returned in ezdp_lookup_ext_tcam_retval_t. Applicable for the first result element only.

unsigned ezdp_lookup_ext_tcam_index_8B_data_result_element::truncated

Truncated response indication - result returned was truncated due to being larger than result_len.

Applicable for the first result element only

unsigned ezdp_lookup_ext_tcam_index_8B_data_result_element::__pad0__

Reserved bits 23 to 24.

unsigned ezdp_lookup_ext_tcam_index_8B_data_result_element::device_id

Responding device id.

unsigned ezdp_lookup_ext_tcam_index_8B_data_result_element::index

compare result index

uint8_t ezdp_lookup_ext_tcam_index_8B_data_result_element::assoc_data[EZDP_LOOKUP_EXT_TCAM_INDEX_8B_DATA_RESULT_ELEMENT_ASSOC_DATA_COUNT]

8B Associated data

union { ... }

The documentation for this struct was generated from the following file:

- dpe/dp/include/ezdp_search_defs.h

ezdp_lookup_ext_tcam_index_result_element Struct Reference

Lookup external tcam index result element.

Data Fields

- union {
- ezdp_lookup_ext_tcam_index_result_element_t raw_data
- struct {
- unsigned valid: EZDP_LOOKUP_EXT_TCAM_INDEX_RESULT_ELEMENT_VALID_SIZE
- *Result valid.* unsigned match: EZDP_LOOKUP_EXT_TCAM_INDEX_RESULT_ELEMENT_MATCH_SIZE
- *Match.* unsigned lookup_error: EZDP_LOOKUP_EXT_TCAM_INDEX_RESULT_ELEMENT_LOOKUP_ERROR_SIZE
- *< Result element type.* unsigned truncated: EZDP_LOOKUP_EXT_TCAM_INDEX_RESULT_ELEMENT_TRUNCATED_SIZE
- *Truncated response indication - result returned was truncated due to being larger than result_len.* unsigned any_match: EZDP_LOOKUP_EXT_TCAM_INDEX_RESULT_ELEMENT_ANY_MATCH_SIZE
- *Any match - there is at least one valid match.* unsigned __pad0__: EZDP_LOOKUP_EXT_TCAM_INDEX_RESULT_ELEMENT_RESERVED23_SIZE
- *Reserved bit 23.* unsigned device_id: EZDP_LOOKUP_EXT_TCAM_INDEX_RESULT_ELEMENT_DEVICE_ID_SIZE
- *Responding device id.* unsigned index: EZDP_LOOKUP_EXT_TCAM_INDEX_RESULT_ELEMENT_INDEX_SIZE
- *compare result index* }
- };

Detailed Description

Lookup external tcam index result element.

Field Documentation

ezdp_lookup_ext_tcam_index_result_element_t ezdp_lookup_ext_tcam_index_result_element::raw_data

unsigned ezdp_lookup_ext_tcam_index_result_element::valid

Result valid.

unsigned ezdp_lookup_ext_tcam_index_result_element::match

Match.

unsigned ezdp_lookup_ext_tcam_index_result_element::lookup_error

< Result element type.

Lookup error. Will be on if any error occurred. For the exact error look in the error flags returned in ezdp_lookup_ext_tcam_retval_t. Applicable for the first result element only.

unsigned ezdp_lookup_ext_tcam_index_result_element::truncated

Truncated response indication - result returned was truncated due to being larger than result_len.

Applicable for the first result element only

unsigned ezdp_lookup_ext_tcam_index_result_element::any_match

Any match - there is at least one valid match.

unsigned ezdp_lookup_ext_tcam_index_result_element::__pad0__

Reserved bit 23.

unsigned ezdp_lookup_ext_tcam_index_result_element::device_id

Responding device id.

unsigned ezdp_lookup_ext_tcam_index_result_element::index

compare result index

union { ... }

The documentation for this struct was generated from the following file:

- dpe/dp/include/ezdp_search_defs.h

ezdp_lookup_ext_tcam_retval Struct Reference

Lookup external tcam return value.

Data Fields

- union {
- ezdp_lookup_ext_tcam_retval_t raw_data
- struct {
- unsigned __pad0__: EZDP_LOOKUP_EXT_TCAM_RETVAL_RESERVED_BIT8_31_SIZE
- *Reserved bits 8 to 31.* unsigned lookup_error: EZDP_LOOKUP_EXT_TCAM_RETVAL_LOOKUP_ERROR_SIZE
- *Aggregated lookup error flag.* unsigned truncated: EZDP_LOOKUP_EXT_TCAM_RETVAL_TRUNCATED_SIZE
- *Truncated response indication - result returned was truncated due to being larger than result_len.* unsigned multi_match: EZDP_LOOKUP_EXT_TCAM_RETVAL_MULTI_MATCH_SIZE
- *Multi match indication - more than one result element is a match.* unsigned any_match: EZDP_LOOKUP_EXT_TCAM_RETVAL_ANY_MATCH_SIZE
- *Any match indication - at least one result element is a match.* unsigned time_out_error: EZDP_LOOKUP_EXT_TCAM_RETVAL_TIME_OUT_ERROR_SIZE
- *Time out error - no response received within configured time frame.* unsigned device_error: EZDP_LOOKUP_EXT_TCAM_RETVAL_DEVICE_ERROR_SIZE
- *External TCAM device error - errors returned by the NL12K unit.* unsigned mac_error: EZDP_LOOKUP_EXT_TCAM_RETVAL_MAC_ERROR_SIZE
- *Mac error - error identified by interlaken mac.* unsigned no_context_match_error: EZDP_LOOKUP_EXT_TCAM_RETVAL_NO_CONTEXT_MATCH_ERROR_SIZE
- *No context match error - returned result's context does not match request context.* }
- };

Detailed Description

Lookup external tcam return value.

Field Documentation

ezdp_lookup_ext_tcam_retval_t ezdp_lookup_ext_tcam_retval::raw_data

unsigned ezdp_lookup_ext_tcam_retval::__pad0__

Reserved bits 8 to 31.

unsigned ezdp_lookup_ext_tcam_retval::lookup_error

Aggregated lookup error flag.

Will be on if any error occurred out of the following: time_out_error, device_error, mac_error or no_context_match_error.

unsigned ezdp_lookup_ext_tcam_retval::truncated

Truncated response indication - result returned was truncated due to being larger than result_len.

unsigned ezdp_lookup_ext_tcam_retval::multi_match

Multi match indication - more than one result element is a match.

Applicable to index mode only.

unsigned ezdp_lookup_ext_tcam_retval::any_match

Any match indication - at least one result element is a match.

Applicable to index mode only.

unsigned ezdp_lookup_ext_tcam_retval::time_out_error

Time out error - no response received within configured time frame.

unsigned ezdp_lookup_ext_tcam_retval::device_error

External TCAM device error - errors returned by the NL12K unit.

unsigned ezdp_lookup_ext_tcam_retval::mac_error

Mac error - error identified by interlaken mac.

unsigned ezdp_lookup_ext_tcam_retval::no_context_match_error

No context match error - returned result's context does not match request context.

union { ... }

The documentation for this struct was generated from the following file:

- dpe/dp/include/ezdp_search_defs.h

ezdp_lookup_int_tcam_12B_data_result Struct Reference

Lookup internal tcam 12 byte associated data result.

Data Fields

- union {
- uint32_t raw_data [EZDP_LOOKUP_INT_TCAM_12B_DATA_RESULT_WORD_COUNT]
- struct {
- unsigned match: EZDP_LOOKUP_INT_TCAM_12B_DATA_RESULT_MATCH_SIZE
- *Match indication.* unsigned data0: EZDP_LOOKUP_INT_TCAM_12B_DATA_RESULT_DATA0_SIZE
- *31 msb of user defined associated data* uint32_t data1
- *Bytes 4 to 7 of user defined associated data.* uint32_t data2
- *Bytes 8 to 11 of user defined associated data.* }
- };

Detailed Description

Lookup internal tcam 12 byte associated data result.

Field Documentation

uint32_t ezdp_lookup_int_tcam_12B_data_result::raw_data[EZDP_LOOKUP_INT_TCAM_12B_DATA_RESULT_WORD_COUNT]

unsigned ezdp_lookup_int_tcam_12B_data_result::match

Match indication.

unsigned ezdp_lookup_int_tcam_12B_data_result::data0

31 msb of user defined associated data

uint32_t ezdp_lookup_int_tcam_12B_data_result::data1

Bytes 4 to 7 of user defined associated data.

uint32_t ezdp_lookup_int_tcam_12B_data_result::data2

Bytes 8 to 11 of user defined associated data.

union { ... }

The documentation for this struct was generated from the following file:

- dpe/dp/include/ezdp_search_defs.h

ezdp_lookup_int_tcam_16B_data_result Struct Reference

Lookup internal tcam 16 byte associated data result.

Data Fields

- union {
- uint32_t raw_data [EZDP_LOOKUP_INT_TCAM_16B_DATA_RESULT_WORD_COUNT]
- struct {
- unsigned match: EZDP_LOOKUP_INT_TCAM_16B_DATA_RESULT_MATCH_SIZE
- *Match indication.* unsigned data0: EZDP_LOOKUP_INT_TCAM_16B_DATA_RESULT_DATA0_SIZE
- *31 msb of user defined associated data* uint32_t data1
- *Bytes 4 to 7 of user defined associated data.* uint32_t data2
- *Bytes 8 to 11 of user defined associated data.* uint32_t data3
- *Bytes 12 to 15 of user defined associated data.* }
- };

Detailed Description

Lookup internal tcam 16 byte associated data result.

Field Documentation

uint32_t ezdp_lookup_int_tcam_16B_data_result::raw_data[EZDP_LOOKUP_INT_TCAM_16B_DATA_RESULT_WORD_COUNT]

unsigned ezdp_lookup_int_tcam_16B_data_result::match

Match indication.

unsigned ezdp_lookup_int_tcam_16B_data_result::data0

31 msb of user defined associated data

uint32_t ezdp_lookup_int_tcam_16B_data_result::data1

Bytes 4 to 7 of user defined associated data.

uint32_t ezdp_lookup_int_tcam_16B_data_result::data2

Bytes 8 to 11 of user defined associated data.

uint32_t ezdp_lookup_int_tcam_16B_data_result::data3

Bytes 12 to 15 of user defined associated data.

union { ... }

The documentation for this struct was generated from the following file:

- dpe/dp/include/ezdp_search_defs.h

ezdp_lookup_int_tcam_4B_data_result Struct Reference

Lookup internal tcam 4 byte associated data result.

Data Fields

- union {
- ezdp_lookup_int_tcam_4B_data_result_t raw_data
- struct {
- unsigned match: EZDP_LOOKUP_INT_TCAM_4B_DATA_RESULT_MATCH_SIZE
- *Match indication.* unsigned data: EZDP_LOOKUP_INT_TCAM_4B_DATA_RESULT_DATA_SIZE
- *31 bits of user defined associated data* }
- };

Detailed Description

Lookup internal tcam 4 byte associated data result.

Field Documentation

ezdp_lookup_int_tcam_4B_data_result_t ezdp_lookup_int_tcam_4B_data_result::raw_data

unsigned ezdp_lookup_int_tcam_4B_data_result::match

Match indication.

unsigned ezdp_lookup_int_tcam_4B_data_result::data

31 bits of user defined associated data

union { ... }

The documentation for this struct was generated from the following file:

- dpe/dp/include/ezdp_search_defs.h

ezdp_lookup_int_tcam_8B_data_result Struct Reference

Lookup internal tcam 8 byte associated data result.

Data Fields

- union {
- uint32_t raw_data [EZDP_LOOKUP_INT_TCAM_8B_DATA_RESULT_WORD_COUNT]
- struct {
- unsigned match: EZDP_LOOKUP_INT_TCAM_8B_DATA_RESULT_MATCH_SIZE
- *Match indication.* unsigned data0: EZDP_LOOKUP_INT_TCAM_8B_DATA_RESULT_DATA0_SIZE
- *31 msb of user defined associated data* uint32_t data1
- *32 lsb of user defined associated data* }
- };

Detailed Description

Lookup internal tcam 8 byte associated data result.

Field Documentation

uint32_t ezdp_lookup_int_tcam_8B_data_result::raw_data[EZDP_LOOKUP_INT_TCAM_8B_DATA_RESULT_WORD_COUNT]

unsigned ezdp_lookup_int_tcam_8B_data_result::match

Match indication.

unsigned ezdp_lookup_int_tcam_8B_data_result::data0

31 msb of user defined associated data

uint32_t ezdp_lookup_int_tcam_8B_data_result::data1

32 lsb of user defined associated data

union { ... }

The documentation for this struct was generated from the following file:

- dpe/dp/include/ezdp_search_defs.h

ezdp_lookup_int_tcam_result Struct Reference

Lookup ITCAM result definition.

Data Fields

- union {
- struct ezdp_lookup_int_tcam_standard_result standard [EZDP_LOOKUP_INT_TCAM_STANDARD_RESULT_MAX_NUM]
- *Result for standard internal TCAM lookup.* struct ezdp_lookup_int_tcam_4B_data_result assoc_4B_data
- *4 byte associated user data.* struct ezdp_lookup_int_tcam_8B_data_result assoc_8B_data
- *8 byte associated user data.* struct ezdp_lookup_int_tcam_12B_data_result assoc_12B_data
- *12 byte associated user data.* struct ezdp_lookup_int_tcam_16B_data_result assoc_16B_data
- *16 byte associated user data.* };

Detailed Description

Lookup ITCAM result definition.

Field Documentation

struct ezdp_lookup_int_tcam_standard_result ezdp_lookup_int_tcam_result::standard[EZDP_LOOKUP_INT_TCAM_STANDARD_RESULT_MAX_NUM] [read]

Result for standard internal TCAM lookup. Up to 4 results.

struct ezdp_lookup_int_tcam_4B_data_result ezdp_lookup_int_tcam_result::assoc_4B_data [read]

4 byte associated user data. Overrides first TCAM result.

struct ezdp_lookup_int_tcam_8B_data_result ezdp_lookup_int_tcam_result::assoc_8B_data [read]

8 byte associated user data. Overrides first 2 TCAM results.

struct ezdp_lookup_int_tcam_12B_data_result ezdp_lookup_int_tcam_result::assoc_12B_data [read]

12 byte associated user data.

Overrides first 3 TCAM results.

struct ezdp_lookup_int_tcam_16B_data_result ezdp_lookup_int_tcam_result::assoc_16B_data [read]

16 byte associated user data. Overrides all TCAM results.

union { ... }

The documentation for this struct was generated from the following file:

- dpe/dp/include/ezdp_search_defs.h

ezdp_lookup_int_tcam_retval Struct Reference

Lookup ITCAM retval definition.

Data Fields

- union {
- ezdp_lookup_int_tcam_retval_t raw_data
- *Retval 32 bit raw data.* struct ezdp_lookup_int_tcam_standard_result standard
- *Result for standard internal TCAM lookup.* struct ezdp_lookup_int_tcam_4B_data_result assoc_data
- *4 byte associated user data.* };

Detailed Description

Lookup ITCAM retval definition.

Field Documentation

ezdp_lookup_int_tcam_retval_t ezdp_lookup_int_tcam_retval::raw_data

Retval 32 bit raw data.

struct ezdp_lookup_int_tcam_standard_result ezdp_lookup_int_tcam_retval::standard [read]

Result for standard internal TCAM lookup.

struct ezdp_lookup_int_tcam_4B_data_result ezdp_lookup_int_tcam_retval::assoc_data [read]

4 byte associated user data.

union { ... }

The documentation for this struct was generated from the following file:

- dpe/dp/include/ezdp_search_defs.h

ezdp_lookup_int_tcam_standard_result Struct Reference

Lookup internal tcam standard result.

Data Fields

- union {
- ezdp_lookup_int_tcam_standard_result_t raw_data
- struct {
- unsigned match: EZDP_LOOKUP_INT_TCAM_STANDARD_RESULT_MATCH_SIZE
- *Match indication.* unsigned __pad0__: EZDP_LOOKUP_INT_TCAM_STANDARD_RESULT_RESERVED0_15_SIZE
- *Reserved bits 0 to 15.* unsigned index: EZDP_LOOKUP_INT_TCAM_STANDARD_RESULT_INDEX_SIZE
- *Lookup index result.* }
- };

Detailed Description

Lookup internal tcam standard result.

Field Documentation

ezdp_lookup_int_tcam_standard_result_t ezdp_lookup_int_tcam_standard_result::raw_data

unsigned ezdp_lookup_int_tcam_standard_result::match

Match indication.

unsigned ezdp_lookup_int_tcam_standard_result::__pad0__

Reserved bits 0 to 15.

unsigned ezdp_lookup_int_tcam_standard_result::index

Lookup index result.

union { ... }

The documentation for this struct was generated from the following file:

- dpe/dp/include/ezdp_search_defs.h

ezdp_lookup_retval Struct Reference

Lookup return value.

Data Fields

- union {
- ezdp_lookup_retval_t raw_data
- struct {
- unsigned mem_error: EZDP_LOOKUP_RETVAL_MEM_ERROR_SIZE
- *Memory error indication.* unsigned info: EZDP_LOOKUP_RETVAL_INFO_SIZE
- *Operation specific additional information is available.* unsigned success: EZDP_LOOKUP_RETVAL_SUCCESS_SIZE
- *Operation success indication - No memory error and match.* unsigned match: EZDP_LOOKUP_RETVAL_MATCH_SIZE
- *Match indication (in lookup).* unsigned data: EZDP_LOOKUP_RETVAL_DATA_SIZE
- *The first 28 bits of the lookup result.* }
- };

Detailed Description

Lookup return value.

Field Documentation

ezdp_lookup_retval_t ezdp_lookup_retval::raw_data

unsigned ezdp_lookup_retval::mem_error

Memory error indication.

unsigned ezdp_lookup_retval::info

Operation specific additional information is available.

Currently only used by UIP lookup to indicate availability of extended result.

unsigned ezdp_lookup_retval::success

Operation success indication - No memory error and match.

unsigned ezdp_lookup_retval::match

Match indication (in lookup).

unsigned ezdp_lookup_retval::data

The first 28 bits of the lookup result.

union { ... }

The documentation for this struct was generated from the following file:

- dpe/dp/include/ezdp_search_defs.h

ezdp_mem_pool_config Struct Reference

memory pool configuration data structure

Data Fields

- struct ezdp_sum_addr base_addr
- *The start address of the memory.* uint16_t index_pool_id
- *BMU index pool id to be used by memory pool.* uint16_t obj_size

The size of the memory element/object.

Detailed Description

memory pool configuration data structure

Field Documentation

struct ezdp_sum_addr ezdp_mem_pool_config::base_addr [read]

The start address of the memory.

uint16_t ezdp_mem_pool_config::index_pool_id

BMU index pool id to be used by memory pool.

NOTE: Pool id which should be configured/enabled to NPS

uint16_t ezdp_mem_pool_config::obj_size

The size of the memory element/object.

The documentation for this struct was generated from the following file:

- dpe/dp/include/ezdp_pool_defs.h

ezdp_mem_section_info Struct Reference

Data Fields

- uint32_t private_cmem_size
- *The size of the private cmem section rounded to multiply of 32 bytes.* uint32_t shared_cmem_size
- *The size of the shared cmem section rounded to multiply of 32 bytes.* uint32_t cache_size
- *The size of the cmem used for thread data cache.* uint32_t imem_private_data_size
- *The size of the imem private data section* . uint32_t imem_half_cluster_data_size
- *The size of the imem half cluster data section* . uint32_t imem_1_cluster_data_size
- *The size of the imem 1 cluster data section* . uint32_t imem_2_cluster_data_size
- *The size of the imem 2 cluster data section* . uint32_t imem_4_cluster_data_size
- *The size of the imem 4 cluster data section* . uint32_t imem_16_cluster_data_size
- *The size of the imem 16 cluster data section* . uint32_t imem_all_cluster_data_size
- *The size of the imem all cluster data section* . uint32_t emem_data_size
- *The size of the emem shared data section* . uint32_t imem_half_cluster_code_size
- *The size of the imem half cluster code section* . uint32_t imem_1_cluster_code_size
- *The size of the imem half cluster code section* . uint32_t imem_2_cluster_code_size
- *The size of the imem 2 cluster code section* . uint32_t imem_4_cluster_code_size
- *The size of the imem 4 cluster code section* . uint32_t imem_16_cluster_code_size
- *The size of the imem 16 cluster code section* . uint32_t imem_all_cluster_code_size

The size of the imem all cluster code section .

Field Documentation

uint32_t ezdp_mem_section_info::private_cmem_size

The size of the private cmem section rounded to multiply of 32 bytes.

uint32_t ezdp_mem_section_info::shared_cmem_size

The size of the shared cmem section rounded to multiply of 32 bytes.

uint32_t ezdp_mem_section_info::cache_size

The size of the cmem used for thread data cache.

Limitation: The total core data cache (thread_cache_size*number of threads) values is 0K, 1K, 2K, 4K, 8K and 16K The minimal thread data cache size is 256 bytes

uint32_t ezdp_mem_section_info::imem_private_data_size

The size of the imem private data section .

uint32_t ezdp_mem_section_info::imem_half_cluster_data_size

The size of the imem half cluster data section .

uint32_t ezdp_mem_section_info::imem_1_cluster_data_size

The size of the imem 1 cluster data section .

uint32_t ezdp_mem_section_info::imem_2_cluster_data_size

The size of the imem 2 cluster data section .

uint32_t ezdp_mem_section_info::imem_4_cluster_data_size

The size of the imem 4 cluster data section .

uint32_t ezdp_mem_section_info::imem_16_cluster_data_size

The size of the imem 16 cluster data section .

uint32_t ezdp_mem_section_info::imem_all_cluster_data_size

The size of the imem all cluster data section .

uint32_t ezdp_mem_section_info::emem_data_size

The size of the emem shared data section .

uint32_t ezdp_mem_section_info::imem_half_cluster_code_size

The size of the imem half cluster code section .

uint32_t ezdp_mem_section_info::imem_1_cluster_code_size

The size of the imem half cluster code section .

uint32_t ezdp_mem_section_info::imem_2_cluster_code_size

The size of the imem 2 cluster code section .

uint32_t ezdp_mem_section_info::imem_4_cluster_code_size

The size of the imem 4 cluster code section .

uint32_t ezdp_mem_section_info::imem_16_cluster_code_size

The size of the imem 16 cluster code section .

uint32_t ezdp_mem_section_info::imem_all_cluster_code_size

The size of the imem all cluster code section .

The documentation for this struct was generated from the following file:

- dpe/dp/include/ezdp.h

ezdp_output_queue_status Struct Reference

PMU output queue status definition (based on PMU system info).

Data Fields

- union {
- ezdp_output_queue_status_t raw_data
- struct {
- unsigned __pad0__: EZDP_OUTPUT_QUEUE_STATUS_RESERVED18_31_SIZE
- *Reserved bits 18 to 31.* unsigned congestion: EZDP_OUTPUT_QUEUE_STATUS_CONGESTION_SIZE
- *The queue in congestion.* unsigned ready: EZDP_OUTPUT_QUEUE_STATUS_READY_SIZE
- *The queue is ready to accept traffic.* uint16_t size
- *The total number of jobs in this list.* }
- };

Detailed Description

PMU output queue status definition (based on PMU system info).

Field Documentation

ezdp_output_queue_status_t ezdp_output_queue_status::raw_data

unsigned ezdp_output_queue_status::__pad0__

Reserved bits 18 to 31.

unsigned ezdp_output_queue_status::congestion

The queue in congestion.

The size of the queue pass the threshold.

unsigned ezdp_output_queue_status::ready

The queue is ready to accept traffic.

uint16_t ezdp_output_queue_status::size

The total number of jobs in this list.

union { ... }

The documentation for this struct was generated from the following file:

- dpe/dp/include/ezdp_job_defs.h

ezdp_pci_addr Struct Reference

PCI Address data structure.

Data Fields

- union {
- uint32_t raw_data [EZDP_PCI_ADDR_WORD_COUNT]
- struct {
- unsigned __pad0__: EZDP_PCI_ADDR_RESERVED29_30_SIZE
- *< Address type* unsigned virt_func_en: EZDP_PCI_ADDR_VIRT_FUNC_EN_SIZE
- *Virtual Function Enable.* unsigned phy_func: EZDP_PCI_ADDR_PHY_FUNC_SIZE
- *Physical function.* uint8_t virt_func
- *Virtual function.* unsigned __pad1__: EZDP_PCI_ADDR_RESERVED14_15_SIZE
- *Reserved bits 14-15.* unsigned msid: EZDP_PCI_ADDR_MSID_SIZE
- *< Type of the MSID* unsigned __pad2__: EZDP_PCI_ADDR_RESERVED4_7_SIZE
- *Reserved bits 4-7.* unsigned address_msb: EZDP_PCI_ADDR_ADDRESS_MSB_SIZE
- *3 msb of 36 bits PCI address* uint32_t address
- *32 lsb of 35 bit extended address* }
- };

Detailed Description

PCI Address data structure.

Field Documentation

uint32_t ezdp_pci_addr::raw_data[EZDP_PCI_ADDR_WORD_COUNT]

unsigned ezdp_pci_addr::__pad0__

Address type. Reserved bits 29-30

unsigned ezdp_pci_addr::virt_func_en

Virtual Function Enable.

unsigned ezdp_pci_addr::phy_func

Physical function.

uint8_t ezdp_pci_addr::virt_func

Virtual function.

unsigned ezdp_pci_addr::__pad1__

Reserved bits 14-15.

unsigned ezdp_pci_addr::msid

Type of the MSID. Select MSID of the structure

unsigned ezdp_pci_addr::__pad2__

Reserved bits 4-7.

unsigned ezdp_pci_addr::address_msb

3 msb of 36 bits PCI address

uint32_t ezdp_pci_addr::address

32 lsb of 35 bit extended address

union { ... }

The documentation for this struct was generated from the following file:

- dpe/dp/include/ezdp_memory_defs.h

ezdp_pci_info Struct Reference

PCI info for describing to which endpoint, physical function, virtual function and queue the frame is to be sent.

Data Fields

- union {
- ezdp_pci_info_t raw_data
- struct {
- unsigned __pad0__: EZDP_PCI_INFO_RESERVED16_32_SIZE
- *Reserved bits 16 to 32.* unsigned virt_func_en: EZDP_PCI_INFO_VIRT_FUNC_EN_SIZE
- *Enable/disable virtual function.* unsigned queue: EZDP_PCI_INFO_QUEUE_SIZE
- *Queue ID.* unsigned phys_func: EZDP_PCI_INFO_PHYS_FUNC_SIZE
- *Physical function.* unsigned endpoint: EZDP_PCI_INFO_ENDPOINT_SIZE
- *Destination end point (PCI device ID).* unsigned virt_func: EZDP_PCI_INFO_VIRT_FUNC_SIZE
- *Virtual function number.* }
- };

Detailed Description

PCI info for describing to which endpoint, physical function, virtual function and queue the frame is to be sent.

Field Documentation

ezdp_pci_info_t ezdp_pci_info::raw_data

unsigned ezdp_pci_info::__pad0__

Reserved bits 16 to 32.

unsigned ezdp_pci_info::virt_func_en

Enable/disable virtual function.

unsigned ezdp_pci_info::queue

Queue ID.

unsigned ezdp_pci_info::phys_func

Physical function.

unsigned ezdp_pci_info::endpoint

Destination end point (PCI device ID).

unsigned ezdp_pci_info::virt_func

Virtual function number.

union { ... }

The documentation for this struct was generated from the following file:

- dpe/dp/include/ezdp_pci_defs.h

ezdp_pci_msg Struct Reference

Message from PCI queue.

Data Fields

- union {
- uint32_t raw_data [EZDP_PCI_MSG_WORD_COUNT]
- struct {
- unsigned __pad0__: EZDP_PCI_MSG_ECC_SIZE
- *ECC.* struct ezdp_pci_msg_ctrl ctrl
- *< Message encoding* union {
- struct ezdp_pci_msg_payload_elbi elbi_payload
- *ELBI message payload.* struct ezdp_pci_msg_payload_ats ats_payload
- *ATS message payload.* struct ezdp_pci_msg_payload_msix msix_payload
- *MSIX message payload.* }
- }
- };

Detailed Description

Message from PCI queue.

Field Documentation

uint32_t ezdp_pci_msg::raw_data[EZDP_PCI_MSG_WORD_COUNT]

unsigned ezdp_pci_msg::__pad0__

ECC.

struct ezdp_pci_msg_ctrl ezdp_pci_msg::ctrl [read]

Message encoding. Message control configuration

struct ezdp_pci_msg_payload_elbi ezdp_pci_msg::elbi_payload [read]

ELBI message payload.

struct ezdp_pci_msg_payload_ats ezdp_pci_msg::ats_payload [read]

ATS message payload.

struct ezdp_pci_msg_payload_msix ezdp_pci_msg::msix_payload [read]

MSIX message payload.

union { ... }

The documentation for this struct was generated from the following file:

- dpe/dp/include/ezdp_pci_defs.h

ezdp_pci_msg_ctrl Struct Reference

PCI message control.

Data Fields

- union {
- ezdp_pci_msg_ctrl_t raw_data
- struct {
- unsigned virt_func_en: EZDP_PCI_MSG_CTRL_VIRT_FUNC_EN_SIZE
- *Virtual Function Enable.* unsigned bar_num: EZDP_PCI_MSG_CTRL_BAR_NUM_SIZE
- *BAR number.* unsigned __pad0__: EZDP_PCI_MSG_CTRL_RESERVED10_11_SIZE
- *Reserved bits 10-11.* unsigned phy_func: EZDP_PCI_MSG_CTRL_PHY_FUNC_SIZE
- *Physical function.* unsigned __pad1__: EZDP_PCI_MSG_CTRL_RESERVED8_SIZE
- *Reserved bit number 8.* unsigned virt_func: EZDP_PCI_MSG_CTRL_VIRT_FUNC_SIZE
- *Virtual function number.* }
- };

Detailed Description

PCI message control.

Field Documentation

ezdp_pci_msg_ctrl_t ezdp_pci_msg_ctrl::raw_data

unsigned ezdp_pci_msg_ctrl::virt_func_en

Virtual Function Enable.

unsigned ezdp_pci_msg_ctrl::bar_num

BAR number.

unsigned ezdp_pci_msg_ctrl::__pad0__

Reserved bits 10-11.

unsigned ezdp_pci_msg_ctrl::phy_func

Physical function.

unsigned ezdp_pci_msg_ctrl::__pad1__

Reserved bit number 8.

unsigned ezdp_pci_msg_ctrl::virt_func

Virtual function number.

union { ... }

The documentation for this struct was generated from the following file:

- dpe/dp/include/ezdp_pci_defs.h

ezdp_pci_msg_payload_ats Struct Reference

PCI ATS message payload.

Data Fields

- union {
- uint32_t raw_data [EZDP_PCI_MSG_PAYLOAD_ATS_WORD_COUNT]
- struct {
- unsigned __pad0__: EZDP_PCI_MSG_PAYLOAD_ATS_RESERVED_SIZE
- *Reserved Byte.* uint32_t data_msb
- *ATS MSB data.* uint32_t data_lsb
- *ATS LSB data.* }
- };

Detailed Description

PCI ATS message payload.

Field Documentation

uint32_t ezdp_pci_msg_payload_ats::raw_data[EZDP_PCI_MSG_PAYLOAD_ATS_WORD_COUNT]

unsigned ezdp_pci_msg_payload_ats::__pad0__

Reserved Byte.

uint32_t ezdp_pci_msg_payload_ats::data_msb

ATS MSB data.

uint32_t ezdp_pci_msg_payload_ats::data_lsb

ATS LSB data.

union { ... }

The documentation for this struct was generated from the following file:

- dpe/dp/include/ezdp_pci_defs.h

ezdp_pci_msg_payload_elbi Struct Reference

PCI ELBI message payload.

Data Fields

- union {
- uint32_t raw_data [EZDP_PCI_MSG_PAYLOAD_ELBI_WORD_COUNT]
- struct {
- unsigned __pad0__: EZDP_PCI_MSG_PAYLOAD_ELBI_RESERVED_SIZE
- *Reserved Byte.* uint32_t address
- *ELBI Address.* uint32_t data
- *ELBI data.* }
- };

Detailed Description

PCI ELBI message payload.

Field Documentation

uint32_t ezdp_pci_msg_payload_elbi::raw_data[EZDP_PCI_MSG_PAYLOAD_ELBI_WORD_COUNT]

unsigned ezdp_pci_msg_payload_elbi::__pad0__

Reserved Byte.

uint32_t ezdp_pci_msg_payload_elbi::address

ELBI Address.

uint32_t ezdp_pci_msg_payload_elbi::data

ELBI data.

union { ... }

The documentation for this struct was generated from the following file:

- dpe/dp/include/ezdp_pci_defs.h

ezdp_pci_msg_payload_msix Struct Reference

PCI MSIX message payload.

Data Fields

- union {
- uint32_t raw_data [EZDP_PCI_MSG_PAYLOAD_MSIX_WORD_COUNT]
- struct {
- unsigned __pad0__: EZDP_PCI_MSG_PAYLOAD_MSIX_RESERVED0_31_SIZE
- *Reserved bits 0 to 31.* unsigned __pad1__: EZDP_PCI_MSG_PAYLOAD_MSIX_RESERVED_32_63_SIZE
- *Reserved bits 32 to 63.* unsigned __pad2__: EZDP_PCI_MSG_PAYLOAD_MSIX_RESERVED66_95_SIZE
- *Reserved bits 66 to 95.* unsigned vector_index: EZDP_PCI_MSG_PAYLOAD_MSIX_VECTOR_INDEX_SIZE
- *Requested MSIX vector index.* }
- };

Detailed Description

PCI MSIX message payload.

Field Documentation

uint32_t ezdp_pci_msg_payload_msix::raw_data[EZDP_PCI_MSG_PAYLOAD_MSIX_WORD_COUNT]

unsigned ezdp_pci_msg_payload_msix::__pad0__

Reserved bits 0 to 31.

unsigned ezdp_pci_msg_payload_msix::__pad1__

Reserved bits 32 to 63.

unsigned ezdp_pci_msg_payload_msix::__pad2__

Reserved bits 66 to 95.

unsigned ezdp_pci_msg_payload_msix::vector_index

Requested MSIX vector index.

union { ... }

The documentation for this struct was generated from the following file:

- dpe/dp/include/ezdp_pci_defs.h

ezdp_posted_ctr_msg Struct Reference

Posted counter message queue definition.

Data Fields

- union {
- uint32_t raw_data [EZDP_POSTED_CTR_MSG_WORD_COUNT]
- struct {
- unsigned __pad0__: EZDP_POSTED_CTR_MSG_ECC_SIZE
- *ECC.* unsigned __pad1__: EZDP_POSTED_CTR_MSG_RESERVED8_23_SIZE
- *reserved bits 8-23* unsigned overrun_error_condition: EZDP_POSTED_CTR_MSG_OVERRUN_ERROR_CONDITION_SIZE
- *Queue was overrun and old messages are lost.* unsigned __pad2__: EZDP_POSTED_CTR_MSG_RESERVED5_6_SIZE
- *reserved bits 5-6* unsigned clear: EZDP_POSTED_CTR_MSG_CLEAR_SIZE
- *Counter was cleared in memory.* unsigned flush: EZDP_POSTED_CTR_MSG_FLUSH_SIZE
- *Flush was executed before reporting.* struct ezdp_sum_addr sum_addr
- *< Posted counter message type* uint64_t value
- *64 bit message value from a summarize address* }
- };

Detailed Description

Posted counter message queue definition.

Field Documentation

uint32_t ezdp_posted_ctr_msg::raw_data[EZDP_POSTED_CTR_MSG_WORD_COUNT]

unsigned ezdp_posted_ctr_msg::__pad0__

ECC.

unsigned ezdp_posted_ctr_msg::__pad1__

reserved bits 8-23

unsigned ezdp_posted_ctr_msg::overrun_error_condition

Queue was overrun and old messages are lost.

unsigned ezdp_posted_ctr_msg::__pad2__

reserved bits 5-6

unsigned ezdp_posted_ctr_msg::clear

Counter was cleared in memory.

unsigned ezdp_posted_ctr_msg::flush

Flush was executed before reporting.

struct ezdp_sum_addr ezdp_posted_ctr_msg::sum_addr [read]

< Posted counter message type

memory summarize address

uint64_t ezdp_posted_ctr_msg::value

64 bit message value from a summarize address

union { ... }

The documentation for this struct was generated from the following file:

- dpe/dp/include/ezdp_counter_defs.h

ezdp_ring_cfg Struct Reference

ring (array queue) configuration data structure

Data Fields

- struct ezdp_sum_addr base_addr
- *Base address to start of queue.* struct ezdp_sum_addr control_addr
- *Control address for managing the queue - must be in resolution of 16B.* uint32_t size

Maximum number of elements in array.

Detailed Description

ring (array queue) configuration data structure

Field Documentation

struct ezdp_sum_addr ezdp_ring_cfg::base_addr [read]

Base address to start of queue.

struct ezdp_sum_addr ezdp_ring_cfg::control_addr [read]

Control address for managing the queue - must be in resolution of 16B.

uint32_t ezdp_ring_cfg::size

Maximum number of elements in array.

Must be a power of 2 (16,32,64,128...)

The documentation for this struct was generated from the following file:

- dpe/dp/include/ezdp_queue_defs.h

ezdp_rtc Struct Reference

ezdp_rtc struct for ezdp

Data Fields

- union {
- uint32_t raw_data [EZDP_RTC_WORD_COUNT]
- struct {
- uint32_t sec
- *The real time clock in resolution of seconds.* uint32_t nsec
- *The real time clock in resolution of nano seconds.* }
- };

Detailed Description

ezdp_rtc struct for ezdp

Field Documentation

uint32_t ezdp_rtc::raw_data[EZDP_RTC_WORD_COUNT]

uint32_t ezdp_rtc::sec

The real time clock in resolution of seconds.

uint32_t ezdp_rtc::nsec

The real time clock in resolution of nano seconds.

union { ... }

The documentation for this struct was generated from the following file:

- dpe/dp/include/ezdp_time_defs.h

ezdp_security_handle Struct Reference

security handle configuration data structure

Data Fields

- union {
- ezdp_security_handle_t raw_data
- struct {
- unsigned __pad0__: EZDP_SECURITY_HANDLE_RESERVED24_31_SIZE
- < *Security algorithm type* unsigned __pad1__: EZDP_SECURITY_HANDLE_RESERVED8_15_SIZE
- *Reserved bits 8 to 15.* uint8_t context_id
- *Security context ID.* }
- };

Detailed Description

security handle configuration data structure

Field Documentation

ezdp_security_handle_t ezdp_security_handle::raw_data

unsigned ezdp_security_handle::__pad0__

< Security algorithm type

Reserved bits 24 to 31

unsigned ezdp_security_handle::__pad1__

Reserved bits 8 to 15.

uint8_t ezdp_security_handle::context_id

Security context ID.

union { ... }

The documentation for this struct was generated from the following file:

- dpe/dp/include/ezdp_security_defs.h

ezdp_single_ctr_cfg Struct Reference

On-demand single value counter configuration definition.

Data Fields

- union {
- uint32_t raw_data [EZDP_SINGLE_CTR_CFG_WORD_COUNT]
- struct {
- unsigned __pad0__: EZDP_SINGLE_CTR_CFG_ECC_SIZE
- *ECC.* unsigned __pad1__: EZDP_SINGLE_CTR_CFG_SUB_TYPE_SIZE
- *Counter sub type (long=0).* unsigned report_exceeded: EZDP_SINGLE_CTR_CFG_REPORT_EXCEEDED_SIZE
- *Number of bits threshold (1-58) for generating exceed message.* unsigned __pad2__: EZDP_SINGLE_CTR_CFG_ZERO_SIZE
- *Zero bit.* unsigned enable_exceed_message: EZDP_SINGLE_CTR_CFG_ENABLE_EXCEED_MESSAGE_SIZE
- *Enable threshold exceed message.* unsigned __pad3__: EZDP_SINGLE_CTR_CFG_RESERVED0_10_SIZE
- *Reserved bits 0 to 10.* unsigned __pad4__: EZDP_SINGLE_CTR_CFG_RESERVED32_63_SIZE
- *Reserved bits 32 to 63.* uint64_t value
- *Counter value.* }
- };

Detailed Description

On-demand single value counter configuration definition.

Field Documentation

uint32_t ezdp_single_ctr_cfg::raw_data[EZDP_SINGLE_CTR_CFG_WORD_COUNT]

unsigned ezdp_single_ctr_cfg::__pad0__

ECC.

unsigned ezdp_single_ctr_cfg::__pad1__

Counter sub type (long=0).

unsigned ezdp_single_ctr_cfg::report_exceeded

Number of bits threshold (1-58) for generating exceed message.

unsigned ezdp_single_ctr_cfg::__pad2__

Zero bit.

unsigned ezdp_single_ctr_cfg::enable_exceed_message

Enable threshold exceed message.

unsigned ezdp_single_ctr_cfg::__pad3__

Reserved bits 0 to 10.

unsigned ezdp_single_ctr_cfg::__pad4__

Reserved bits 32 to 63.

uint64_t ezdp_single_ctr_cfg::value

Counter value.

union { ... }

The documentation for this struct was generated from the following file:

- dpe/dp/include/ezdp_counter_defs.h

ezdp_small_linked_buffers_desc Struct Reference

Small linked buffers descriptor.May hold up to 3 buffs.

Data Fields

- struct ezdp_linked_buffers_desc_line line [EZDP_SMALL_LBD]

Array of 1 linked buffers line, which contains up to 3 buffers.

Detailed Description

Small linked buffers descriptor.May hold up to 3 buffs.

Field Documentation

struct ezdp_linked_buffers_desc_line ezdp_small_linked_buffers_desc::line[EZDP_SMALL_LBD] [read]

Array of 1 linked buffers line, which contains up to 3 buffers.

The documentation for this struct was generated from the following file:

- dpe/dp/include/ezdp_frame_defs.h

ezdp_sum_addr Struct Reference

Summarized Address data structure.

Data Fields

- union {
- ezdp_sum_addr_t raw_data
- struct {
- unsigned msid: EZDP_SUM_ADDR_MSID_SIZE
- < *Memory space type* unsigned element_index: EZDP_SUM_ADDR_ELEMENT_INDEX_SIZE
- *Index of the element within a specific structure.* }
- };

Detailed Description

Summarized Address data structure.

Field Documentation

ezdp_sum_addr_t ezdp_sum_addr::raw_data

unsigned ezdp_sum_addr::msid

< Memory space type

Memory system ID

unsigned ezdp_sum_addr::element_index

Index of the element within a specific structure.

union { ... }

The documentation for this struct was generated from the following file:

- dpe/dp/include/ezdp_memory_defs.h

ezdp_sum_addr_table_desc Struct Reference

Structure definition table entry data structure.

Data Fields

- union {
- ezdp_sum_addr_table_desc_t raw_data
- struct {
- unsigned key_shuff_en: EZDP_SUM_ADDR_TABLE_DESC_KEY_SHUFF_EN_SIZE
- *Enable/Disable key shuffling.* unsigned key_shuff_bits: EZDP_SUM_ADDR_TABLE_DESC_KEY_SHUFF_BITS_SIZE
- *Bitmap for table key shuffling.* unsigned __pad0__: EZDP_SUM_ADDR_TABLE_DESC_RESERVED25_26_SIZE
- *< Memory space type* unsigned key_size: EZDP_SUM_ADDR_TABLE_DESC_KEY_SIZE_SIZE
- *Table key size in bits (up to 32) / Hash1 size in bits (up to 32).* unsigned msid: EZDP_SUM_ADDR_TABLE_DESC_MSID_SIZE
- *Memory system ID.* uint16_t base_index
- *Base index of the structure This is the logical offset within a specific MSID, Index Resolution: IMEM: 1KB EMEM: 1MB.* }
- };

Detailed Description

Structure definition table entry data structure.

Field Documentation

ezdp_sum_addr_table_desc_t ezdp_sum_addr_table_desc::raw_data

unsigned ezdp_sum_addr_table_desc::key_shuff_en

Enable/Disable key shuffling.

unsigned ezdp_sum_addr_table_desc::key_shuff_bits

Bitmap for table key shuffling.

unsigned ezdp_sum_addr_table_desc::__pad0__

< Memory space type

Reserved bits 25 to 26

unsigned ezdp_sum_addr_table_desc::key_size

Table key size in bits (up to 32) / Hash1 size in bits (up to 32).

Used for masking the input key

unsigned ezdp_sum_addr_table_desc::msid

Memory system ID.

uint16_t ezdp_sum_addr_table_desc::base_index

Base index of the structure This is the logical offset within a specific MSID, Index Resolution: IMEM: 1KB EMEM: 1MB.

union { ... }

The documentation for this struct was generated from the following file:

- dpe/dp/include/ezdp_memory_defs.h

ezdp_tb_ctr_cfg Struct Reference

Token bucket counter configuration definition.

Data Fields

- union {
- uint32_t raw_data [EZDP_TB_CTR_CFG_WORD_COUNT]
- struct {
- unsigned __pad0__: EZDP_TB_CTR_CFG_RESERVED26_31_SIZE
- *Reserved bits 26 to 31.* unsigned coupling_flag: EZDP_TB_CTR_CFG_COUPLING_FLAG_SIZE
- *Enable/disable coupling flag.* unsigned color_aware: EZDP_TB_CTR_CFG_COLOR_AWARE_SIZE
- *Enable/disable color awareness.* unsigned excess_profile_id: EZDP_TB_CTR_CFG_EXCESS_PROFILE_ID_SIZE
- *< The marking algorithm used for this profile* unsigned commit_profile_id: EZDP_TB_CTR_CFG_COMMIT_PROFILE_ID_SIZE
- *Id of commit token bucket profile in the token bucket profile table.* unsigned __pad1__: EZDP_TB_CTR_CFG_RESERVED32_63_SIZE
- *Reserved bits 32 to 63.* unsigned __pad2__: EZDP_TB_CTR_CFG_RESERVED64_95_SIZE
- *Reserved bits 64 to 95.* unsigned __pad3__: EZDP_TB_CTR_CFG_RESERVED96_127_SIZE
- *Reserved bits 96 to 127.* }
- };

Detailed Description

Token bucket counter configuration definition.

Field Documentation

uint32_t ezdp_tb_ctr_cfg::raw_data[EZDP_TB_CTR_CFG_WORD_COUNT]

unsigned ezdp_tb_ctr_cfg::__pad0__

Reserved bits 26 to 31.

unsigned ezdp_tb_ctr_cfg::coupling_flag

Enable/disable coupling flag.

Relevant only for trTCM MEF algorithm.

unsigned ezdp_tb_ctr_cfg::color_aware

Enable/disable color awareness.

Relevant only for srTCM, trTCM and trTCM MEF algorithms.

unsigned ezdp_tb_ctr_cfg::excess_profile_id

< The marking algorithm used for this profile

Id of excess token bucket profile in the token bucket profile table

unsigned ezdp_tb_ctr_cfg::commit_profile_id

Id of commit token bucket profile in the token bucket profile table.

unsigned ezdp_tb_ctr_cfg::__pad1__

Reserved bits 32 to 63.

unsigned ezdp_tb_ctr_cfg::__pad2__

Reserved bits 64 to 95.

unsigned ezdp_tb_ctr_cfg::__pad3__

Reserved bits 96 to 127.

union { ... }

The documentation for this struct was generated from the following file:

- dpe/dp/include/ezdp_counter_defs.h

ezdp_tb_ctr_result Struct Reference

Token bucket counter result value definition.

Data Fields

- union {
- uint32_t raw_data [EZDP_TB_CTR_RESULT_WORD_COUNT]
- struct {
- unsigned __pad0__: EZDP_TB_CTR_RESULT_RESERVED0_31_SIZE
- *Reserved bits 0 to 31.* unsigned __pad1__: EZDP_TB_CTR_RESULT_RESERVED60_63_SIZE
- *Reserved bits 60 to 63.* unsigned empty_excess_bucket_ug: EZDP_TB_CTR_RESULT_EMPTY_EXCESS_BUCKET_UG_SIZE
- *Empty excess bucket ultra green Close to burst size indication after the force dec command.* unsigned empty_commit_bucket_ug: EZDP_TB_CTR_RESULT_EMPTY_COMMIT_BUCKET_UG_SIZE
- *Empty commit bucket ultra green Close to burst size indication after the force dec command.* unsigned empty_excess_bucket: EZDP_TB_CTR_RESULT_EMPTY_EXCESS_BUCKET_SIZE
- *Indicate that excess bucket is empty.* unsigned empty_commit_bucket: EZDP_TB_CTR_RESULT_EMPTY_COMMIT_BUCKET_SIZE
- *Indicate that commit bucket is empty.* unsigned __pad2__: EZDP_TB_CTR_RESULT_RESERVED34_57_SIZE
- *Reserved bits 34 to 57.* }
- };

Detailed Description

Token bucket counter result value definition.

Field Documentation

uint32_t ezdp_tb_ctr_result::raw_data[EZDP_TB_CTR_RESULT_WORD_COUNT]

unsigned ezdp_tb_ctr_result::__pad0__

Reserved bits 0 to 31.

unsigned ezdp_tb_ctr_result::__pad1__

Reserved bits 60 to 63.

unsigned ezdp_tb_ctr_result::empty_excess_bucket_ug

Empty excess bucket ultra green Close to burst size indication after the force dec command.

unsigned ezdp_tb_ctr_result::empty_commit_bucket_ug

Empty commit bucket ultra green Close to burst size indication after the force dec command.

unsigned ezdp_tb_ctr_result::empty_excess_bucket

Indicate that excess bucket is empty.

Note1: The value can be negative. Note2: Applicable to increment/decrement command only.

unsigned ezdp_tb_ctr_result::empty_commit_bucket

Indicate that commit bucket is empty.

Note1: The value can be negative. Note2: Applicable to increment/decrement command only.

unsigned ezdp_tb_ctr_result::__pad2__

Reserved bits 34 to 57.

union { ... }

The documentation for this struct was generated from the following file:

- dpe/dp/include/ezdp_counter_defs.h

ezdp_version Struct Reference

version info data structure

Data Fields

- const char * project_name
- *Project name.* const char * module_name
- *Module name.* uint32_t major_version
- *Major version.* uint32_t minor_version
- *Minor version.* uint8_t version_char
- *Version version.* const char * version_string
- *Version string.* uint8_t major_patch_version
- *Major patch version.* uint8_t minor_patch_version
- *Minor patch version.* uint8_t micro_patch_version
- *Micro patch version.* const int8_t * build_number
- *Build number.* const char * creation_date
- *Creation date.* const char * creation_time

Creation time.

Detailed Description

version info data structure

Field Documentation

const char* ezdp_version::project_name

Project name.

const char* ezdp_version::module_name

Module name.

uint32_t ezdp_version::major_version

Major version.

uint32_t ezdp_version::minor_version

Minor version.

uint8_t ezdp_version::version_char

Version version.

const char* ezdp_version::version_string

Version string.

uint8_t ezdp_version::major_patch_version

Major patch version.

uint8_t ezdp_version::minor_patch_version

Minor patch version.

uint8_t ezdp_version::micro_patch_version

Micro patch version.

const int8_t* ezdp_version::build_number

Build number.

const char* ezdp_version::creation_date

Creation date.

const char* ezdp_version::creation_time

Creation time.

The documentation for this struct was generated from the following file:

- dpe/dp/include/ezdp_version.h

ezdp_watchdog_accumulative_window_cfg Struct Reference

Watchdog accumulative window configuration definition.

Data Fields

- union {
- uint32_t raw_data [EZDP_WATCHDOG_ACCUMULATIVE_WINDOW_CFG_WORD_COUNT]
- struct {
- unsigned __pad0__: EZDP_WATCHDOG_ACCUMULATIVE_WINDOW_CFG_ALERT_SIZE
- *(Max alert) or (Min alert)* unsigned __pad1__: EZDP_WATCHDOG_ACCUMULATIVE_WINDOW_CFG_MIN_THRESHOLD_ALERT_SIZE
- *Check resulted in passing of the Min threshold.* unsigned __pad2__: EZDP_WATCHDOG_ACCUMULATIVE_WINDOW_CFG_MAX_THRESHOLD_ALERT_SIZE
- *Check resulted in passing of the Max threshold.* unsigned __pad3__: EZDP_WATCHDOG_ACCUMULATIVE_WINDOW_CFG_RESERVED5_28_SIZE
- *reserved bits 5-28* unsigned accumulative_events: EZDP_WATCHDOG_ACCUMULATIVE_WINDOW_CFG_ACCUMULATIVE_EVENTS_SIZE
- *Counts the number of lost events deficit as compared to per profile expected events in a given scan period.* unsigned __pad4__: EZDP_WATCHDOG_ACCUMULATIVE_WINDOW_CFG_RESERVED63_SIZE
- *reserved bit 63* unsigned __pad5__: EZDP_WATCHDOG_ACCUMULATIVE_WINDOW_CFG_PARITY_SIZE
- *The counter parity value.* unsigned profile_id: EZDP_WATCHDOG_ACCUMULATIVE_WINDOW_CFG_PROFILE_ID_SIZE
- *A pointer to one of 16 profiles, where each profile holds one set of minimum and maximum event thresholds.* unsigned valid: EZDP_WATCHDOG_ACCUMULATIVE_WINDOW_CFG_VALID_SIZE
- *Indicates if the counters value is valid.* unsigned __pad6__: EZDP_WATCHDOG_ACCUMULATIVE_WINDOW_CFG_INIT_BIT_SIZE
- *Raise after CTOP initialize the counter.* unsigned curr_events: EZDP_WATCHDOG_ACCUMULATIVE_WINDOW_CFG_CURR_EVENTS_SIZE
- *The current number of events in the corresponding session.* unsigned last_events: EZDP_WATCHDOG_ACCUMULATIVE_WINDOW_CFG_LAST_EVENTS_SIZE
- *The previous current event counter to calculate the new events.* }
- };

Detailed Description

Watchdog accumulative window configuration definition.

Field Documentation

uint32_t ezdp_watchdog_accumulative_window_cfg::raw_data[EZDP_WATCHDOG_ACCUMULATIVE_WINDOW_CFG_WORD_COUNT]

unsigned ezdp_watchdog_accumulative_window_cfg::__pad0__

(Max alert) or (Min alert)

unsigned ezdp_watchdog_accumulative_window_cfg::__pad1__

Check resulted in passing of the Min threshold.

unsigned ezdp_watchdog_accumulative_window_cfg::__pad2__

Check resulted in passing of the Max threshold.

unsigned ezdp_watchdog_accumulative_window_cfg::__pad3__

reserved bits 5-28

unsigned ezdp_watchdog_accumulative_window_cfg::accumulative_events

Counts the number of lost events deficit as compared to per profile expected events in a given scan period.

unsigned ezdp_watchdog_accumulative_window_cfg::__pad4__

reserved bit 63

unsigned ezdp_watchdog_accumulative_window_cfg::__pad5__

The counter parity value.

unsigned ezdp_watchdog_accumulative_window_cfg::profile_id

A pointer to one of 16 profiles, where each profile holds one set of minimum and maximum event thresholds.

unsigned ezdp_watchdog_accumulative_window_cfg::valid

Indicates if the counters value is valid.

unsigned ezdp_watchdog_accumulative_window_cfg::__pad6__

Raise after CTOP initialize the counter.
Reset after the first check

unsigned ezdp_watchdog_accumulative_window_cfg::curr_events

The current number of events in the corresponding session.

unsigned ezdp_watchdog_accumulative_window_cfg::last_events

The previous current event counter to calculate the new events.

union { ... }

The documentation for this struct was generated from the following file:

- dpe/dp/include/ezdp_counter_defs.h

ezdp_watchdog_ctr_cfg Struct Reference

Watchdog counter configuration definition.

Data Fields

- union {
- uint32_t raw_data [EZDP_WATCHDOG_CTR_CFG_WORD_COUNT]
- struct {
- unsigned __pad0__: EZDP_WATCHDOG_CTR_CFG_ECC_SIZE
- *ECC.* unsigned __pad1__: EZDP_WATCHDOG_CTR_CFG_SUB_TYPE_SIZE
- *Counter sub type (bitwise=9).* unsigned __pad2__: EZDP_WATCHDOG_CTR_CFG_RESERVED0_18_SIZE
- *Reserved bits 0 to 18.* unsigned __pad3__: EZDP_WATCHDOG_CTR_CFG_RESERVED32_63_SIZE
- *Reserved bits 32 to 63.* union {
- struct ezdp_watchdog_accumulative_window_cfg accumulative_window
- *Accumulative window mode configuration for WD counter.* struct ezdp_watchdog_sliding_window_cfg sliding_window
- *Sliding window mode configuration for WD counter.* }
- }
- };

Detailed Description

Watchdog counter configuration definition.

Field Documentation

uint32_t ezdp_watchdog_ctr_cfg::raw_data[EZDP_WATCHDOG_CTR_CFG_WORD_COUNT]

unsigned ezdp_watchdog_ctr_cfg::__pad0__

ECC.

unsigned ezdp_watchdog_ctr_cfg::__pad1__

Counter sub type (bitwise=9).

unsigned ezdp_watchdog_ctr_cfg::__pad2__

Reserved bits 0 to 18.

unsigned ezdp_watchdog_ctr_cfg::__pad3__

Reserved bits 32 to 63.

struct ezdp_watchdog_accumulative_window_cfg ezdp_watchdog_ctr_cfg::accumulative_window [read]

Accumulative window mode configuration for WD counter.

struct ezdp_watchdog_sliding_window_cfg ezdp_watchdog_ctr_cfg::sliding_window [read]

Sliding window mode configuration for WD counter.

union { ... }

The documentation for this struct was generated from the following file:

- dpe/dp/include/ezdp_counter_defs.h

ezdp_watchdog_ctr_check_result Struct Reference

Watchdog counter check result definition.

Data Fields

- union {
- uint32_t raw_data [EZDP_WATCHDOG_CTR_CHECK_RESULT_WORD_COUNT]
- struct {
- unsigned alert: EZDP_WATCHDOG_CTR_CHECK_RESULT_ALERT_SIZE
- *(Max alert) or (Min alert)* unsigned min_threshold_alert: EZDP_WATCHDOG_CTR_CHECK_RESULT_MIN_THRESHOLD_ALERT_SIZE
- *Check resulted in passing of the Min threshold.* unsigned max_threshold_alert: EZDP_WATCHDOG_CTR_CHECK_RESULT_MAX_THRESHOLD_ALERT_SIZE
- *Check resulted in passing of the Max threshold.* unsigned __pad0__: EZDP_WATCHDOG_CTR_CHECK_RESULT_RESERVED4_28_SIZE
- *reserved bits 4-28* unsigned __pad1__: EZDP_WATCHDOG_CTR_CHECK_RESULT_ACCUMULATIVE_EVENTS_SIZE
- *Counts the number of lost events deficit as compared to per profile expected events in a given scan period 5 MSB.* unsigned __pad2__: EZDP_WATCHDOG_CTR_CHECK_RESULT_WINDOW_RELATED_SIZE
- *Counts the number of lost events deficit as compared to per profile expected events in a given scan period.* unsigned __pad3__: EZDP_WATCHDOG_CTR_CHECK_RESULT_RESERVED62_SIZE
- *reserved bit - 62* unsigned __pad4__: EZDP_WATCHDOG_CTR_CHECK_RESULT_PROFILE_ID_SIZE
- *A pointer to one of 16 profiles, where each profile holds one set of minimum and maximum event thresholds.* unsigned __pad5__: EZDP_WATCHDOG_CTR_CHECK_RESULT_VALID_SIZE
- *Indicates if the counters value is valid.* unsigned __pad6__: EZDP_WATCHDOG_CTR_CHECK_RESULT_INIT_BIT_SIZE
- *Raise after CTOP initialize the counter.* unsigned __pad7__: EZDP_WATCHDOG_CTR_CHECK_RESULT_CURR_EVENTS_SIZE
- *The current number of events in the corresponding session.* unsigned __pad8__: EZDP_WATCHDOG_CTR_CHECK_RESULT_LAST_EVENTS_SIZE
- *The previous current event counter to calculate the new events.* }
- };

Detailed Description

Watchdog counter check result definition.

Field Documentation

uint32_t ezdp_watchdog_ctr_check_result::raw_data[EZDP_WATCHDOG_CTR_CHECK_RESULT_WORD_COUNT]

unsigned ezdp_watchdog_ctr_check_result::alert

(Max alert) or (Min alert)

unsigned ezdp_watchdog_ctr_check_result::min_threshold_alert

Check resulted in passing of the Min threshold.

unsigned ezdp_watchdog_ctr_check_result::max_threshold_alert

Check resulted in passing of the Max threshold.

unsigned ezdp_watchdog_ctr_check_result::__pad0__

reserved bits 4-28

unsigned ezdp_watchdog_ctr_check_result::__pad1__

Counts the number of lost events deficit as compared to per profile expected events in a given scan period 5 MSB.

unsigned ezdp_watchdog_ctr_check_result::__pad2__

Counts the number of lost events deficit as compared to per profile expected events in a given scan period.

unsigned ezdp_watchdog_ctr_check_result::__pad3__

reserved bit - 62

unsigned ezdp_watchdog_ctr_check_result::__pad4__

A pointer to one of 16 profiles, where each profile holds one set of minimum and maximum event thresholds.

unsigned ezdp_watchdog_ctr_check_result::__pad5__

Indicates if the counters value is valid.

unsigned ezdp_watchdog_ctr_check_result::__pad6__

Raise after CTOP initialize the counter.

Reset after the first check

unsigned ezdp_watchdog_ctr_check_result::__pad7__

The current number of events in the corresponding session.

unsigned ezdp_watchdog_ctr_check_result::__pad8__

The previous current event counter to calculate the new events.

union { ... }

The documentation for this struct was generated from the following file:

- dpe/dp/include/ezdp_counter_defs.h

ezdp_watchdog_ctr_start_result Struct Reference

Watchdog counter check result definition.

Data Fields

- union {
- uint32_t raw_data [EZDP_WATCHDOG_CTR_START_RESULT_WORD_COUNT]
- struct {
- uint32_t msb
- *Counter MSB.* uint32_t lsb
- *Counter LSB.* }
- };

Detailed Description

Watchdog counter check result definition.

Field Documentation

uint32_t ezdp_watchdog_ctr_start_result::raw_data[EZDP_WATCHDOG_CTR_START_RESULT_WORD_COUNT]

uint32_t ezdp_watchdog_ctr_start_result::msb

Counter MSB.

uint32_t ezdp_watchdog_ctr_start_result::lsb

Counter LSB.

union { ... }

The documentation for this struct was generated from the following file:

- dpe/dp/include/ezdp_counter_defs.h

ezdp_watchdog_sliding_window_cfg Struct Reference

Watchdog sliding window configuration definition.

Data Fields

- union {
- uint32_t raw_data [EZDP_WATCHDOG_SLIDING_WINDOW_CFG_WORD_COUNT]
- struct {
- unsigned __pad0__: EZDP_WATCHDOG_SLIDING_WINDOW_CFG_ALERT_SIZE
- *(Max alert) or (Min alert)* unsigned __pad1__: EZDP_WATCHDOG_SLIDING_WINDOW_CFG_MIN_THRESHOLD_ALERT_SIZE
- *Check resulted in passing of the Min threshold.* unsigned __pad2__: EZDP_WATCHDOG_SLIDING_WINDOW_CFG_MAX_THRESHOLD_ALERT_SIZE
- *Check resulted in passing of the Max threshold.* unsigned __pad3__: EZDP_WATCHDOG_SLIDING_WINDOW_CFG_RESERVED5_28_SIZE
- *reserved bits 5-28* unsigned valid_windows: EZDP_WATCHDOG_SLIDING_WINDOW_CFG_VALID_WINDOWS_SIZE
- *Counts the number of valid sliding windows.* unsigned __pad4__: EZDP_WATCHDOG_SLIDING_WINDOW_CFG_RESERVED63_SIZE
- *reserved bit 63* unsigned __pad5__: EZDP_WATCHDOG_SLIDING_WINDOW_CFG_PARITY_SIZE
- *The counter parity value.* unsigned profile_id: EZDP_WATCHDOG_SLIDING_WINDOW_CFG_PROFILE_ID_SIZE
- *A pointer to one of 16 profiles, where each profile holds one set of minimum and maximum event thresholds.* unsigned valid: EZDP_WATCHDOG_SLIDING_WINDOW_CFG_VALID_SIZE
- *Indicates if the counters value is valid.* unsigned __pad6__: EZDP_WATCHDOG_SLIDING_WINDOW_CFG_RESERVED56_SIZE
- *reserved bit 56* unsigned counters: EZDP_WATCHDOG_SLIDING_WINDOW_CFG_COUNTERS_SIZE
- *3/4/6/8/12/24 windows counters* }
- };

Detailed Description

Watchdog sliding window configuration definition.

Field Documentation

uint32_t ezdp_watchdog_sliding_window_cfg::raw_data[EZDP_WATCHDOG_SLIDING_WINDOW_CFG_WORD_COUNT]

unsigned ezdp_watchdog_sliding_window_cfg::__pad0__

(Max alert) or (Min alert)

unsigned ezdp_watchdog_sliding_window_cfg::__pad1__

Check resulted in passing of the Min threshold.

unsigned ezdp_watchdog_sliding_window_cfg::__pad2__

Check resulted in passing of the Max threshold.

unsigned ezdp_watchdog_sliding_window_cfg::__pad3__

reserved bits 5-28

unsigned ezdp_watchdog_sliding_window_cfg::valid_windows

Counts the number of valid sliding windows.

unsigned ezdp_watchdog_sliding_window_cfg::__pad4__

reserved bit 63

unsigned ezdp_watchdog_sliding_window_cfg::__pad5__

The counter parity value.

unsigned ezdp_watchdog_sliding_window_cfg::profile_id

A pointer to one of 16 profiles, where each profile holds one set of minimum and maximum event thresholds.

unsigned ezdp_watchdog_sliding_window_cfg::valid

Indicates if the counters value is valid.

unsigned ezdp_watchdog_sliding_window_cfg::__pad6__

reserved bit 56

unsigned ezdp_watchdog_sliding_window_cfg::counters

3/4/6/8/12/24 windows counters

union { ... }

The documentation for this struct was generated from the following file:

- dpe/dp/include/ezdp_counter_defs.h

File Documentation

dpe/dp/include/ezdp.h File Reference

Data Structures

- struct ezdp_mem_section_info

Defines

- #define EZDP_MEM_CFG_USE_ALTER_CMEM 0x1
- *MEM configuration flags.* #define EZDP_MEM_CFG_USE_ALTER_SHARED_CMEM 0x2
- *Indicate if alternative shared CMEM should be used.* #define EZDP_MEM_CFG_IMEM_PRIVATE_DATA_CACHABLE 0x4
- *Indicate if IMEM private memory should be cachable.* #define EZDP_MEM_CFG_IMEM_HALF_CLUSTER_DATA_CACHABLE 0x8
- *Indicate if IMEM half cluster memory should be cachable.* #define EZDP_MEM_CFG_IMEM_1_CLUSTER_DATA_CACHABLE 0x10
- *Indicate if IMEM 1 cluster memory should be cachable.* #define EZDP_MEM_CFG_IMEM_2_CLUSTER_DATA_CACHABLE 0x20
- *Indicate if IMEM 2 cluster memory should be cachable.* #define EZDP_MEM_CFG_IMEM_4_CLUSTER_DATA_CACHABLE 0x40
- *Indicate if IMEM 4 cluster memory should be cachable.* #define EZDP_MEM_CFG_IMEM_16_CLUSTER_DATA_CACHABLE 0x80
- *Indicate if IMEM 16 cluster memory should be cachable.* #define EZDP_MEM_CFG_IMEM_ALL_CLUSTER_DATA_CACHABLE 0x100
- *Indicate if IMEM all cluster memory should be cachable.* #define EZDP_MEM_CFG_EMEM_DATA_CACHABLE 0x200

Indicate if EMEM shared memory should be cachable. Typedefs

- typedef bool(* EZDP_MEM_CTOR_FUNC)(enum ezdp_data_mem_space data_ms_type, uintptr_t user_data)
- *Type definition for the user memory constructor handling.* typedef void(* EZDP_MAIN_FUNC)(void)

Type definition for the data plane application's main frame handling loop function. Enumerations

- enum ezdp_data_mem_space { EZDP_CMEM_DATA, EZDP_SHARED_CMEM_DATA, EZDP_IMEM_PRIVATE_DATA, EZDP_IMEM_HALF_CLUSTER_DATA, EZDP_IMEM_1_CLUSTER_DATA, EZDP_IMEM_2_CLUSTER_DATA, EZDP_IMEM_4_CLUSTER_DATA, EZDP_IMEM_16_CLUSTER_DATA, EZDP_IMEM_ALL_CLUSTER_DATA, EZDP_EMEM_DATA }
  memory space types.

Functions

- uint32_t ezdp_sync_cp (void)
- *Sync until CP is up.* uint32_t ezdp_init_global (uint32_t app_id)
- *Initialize the data-plane application, Should be called once per data-plane executable.* uint32_t ezdp_init_local (uint32_t app_id, int32_t cpu_id, EZDP_MEM_CTOR_FUNC mem_ctor_func, uintptr_t mem_ctor_data, uint32_t flags)
- *Initialize the data-plane process.* void ezdp_run (EZDP_MAIN_FUNC func, uint32_t wait_count)
- *Run a data plane application.* struct ezdp_version * ezdp_get_version (void)
- *Return the version information of the ezdp library.* char * ezdp_get_err_msg (void)
- *Return string of the last error.* void ezdp_get_mem_section_info (struct ezdp_mem_section_info *mem_info, uint32_t flags)
- *Return sizes of the memory sections.* const char * ezdp_mem_section_info_str (struct ezdp_mem_section_info *mem_info)

Return printable string of the memory sizes.

Define Documentation

#define EZDP_MEM_CFG_USE_ALTER_CMEM 0x1

MEM configuration flags.

Indicate if alternative private CMEM should be used. Used in order to determine the size of the CMEM section NOTE: alternative CMEM doesn't contain compile time init values

#define EZDP_MEM_CFG_USE_ALTER_SHARED_CMEM 0x2

Indicate if alternative shared CMEM should be used.

Used in order to determine the size of the CMEM section NOTE: alternative CMEM doesn't contain compile time init values

#define EZDP_MEM_CFG_IMEM_PRIVATE_DATA_CACHABLE 0x4

Indicate if IMEM private memory should be cachable.

#define EZDP_MEM_CFG_IMEM_HALF_CLUSTER_DATA_CACHABLE 0x8

Indicate if IMEM half cluster memory should be cachable.

#define EZDP_MEM_CFG_IMEM_1_CLUSTER_DATA_CACHABLE 0x10

Indicate if IMEM 1 cluster memory should be cachable.

#define EZDP_MEM_CFG_IMEM_2_CLUSTER_DATA_CACHABLE 0x20

Indicate if IMEM 2 cluster memory should be cachable.

#define EZDP_MEM_CFG_IMEM_4_CLUSTER_DATA_CACHABLE 0x40

Indicate if IMEM 4 cluster memory should be cachable.

#define EZDP_MEM_CFG_IMEM_16_CLUSTER_DATA_CACHABLE 0x80

Indicate if IMEM 16 cluster memory should be cachable.

#define EZDP_MEM_CFG_IMEM_ALL_CLUSTER_DATA_CACHABLE 0x100

Indicate if IMEM all cluster memory should be cachable.

#define EZDP_MEM_CFG_EMEM_DATA_CACHABLE 0x200

Indicate if EMEM shared memory should be cachable.

Typedef Documentation

typedef bool(* EZDP_MEM_CTOR_FUNC)(enum ezdp_data_mem_space data_ms_type, uintptr_t user_data)

Type definition for the user memory constructor handling.

typedef void(* EZDP_MAIN_FUNC)(void)

Type definition for the data plane application's main frame handling loop function.

Enumeration Type Documentation

enum ezdp_data_mem_space

memory space types.

Enumerator:

EZDP_CMEM_DATA Private CMEM data memory space.

EZDP_SHARED_CMEM_DATA Shared CMEM data memory space.

EZDP_IMEM_PRIVATE_DATA Thread private data memory space.

EZDP_IMEM_HALF_CLUSTER_DATA Sub cluster data memory space.

EZDP_IMEM_1_CLUSTER_DATA Single cluster data memory space.

EZDP_IMEM_2_CLUSTER_DATA Dual cluster data memory space.

EZDP_IMEM_4_CLUSTER_DATA Quad cluster data memory space.

EZDP_IMEM_16_CLUSTER_DATA 16 cluster data memory space.

EZDP_IMEM_ALL_CLUSTER_DATA All cluster data memory space.

EZDP_EMEM_DATA External memory shared data.

Function Documentation

uint32_t ezdp_sync_cp (void)

Sync until CP is up.

Wait until CP is up and till memory is configured and ready to use

Returns:

- 0 (operation success), The function may fail for any of the errors specified for routines: open,close,ioctl In such case the errno value of the failure routine is returned Use ezdp_get_err_msg() API to get the detail error message of the failure

uint32_t ezdp_init_global (uint32_t *app_id*)

Initialize the data-plane application, Should be called once per data-plane executable.

Used to allocate shared memory, which is later used in ezdp_init_local function

Parameters:

[in] *app_id* - Determines the application id

Returns:

- 0 (operation success), The function may fail for any of the errors specified for routines: open,mmap,ftruncate,ftok,semget,semop In such case the errno value of the failure routine is returned Use ezdp_get_err_msg() API to get the detail error message of the failure

uint32_t ezdp_init_local (uint32_t *app_id*, int32_t *cpu_id*, EZDP_MEM_CTOR_FUNC *mem_ctor_func*, uintptr_t *mem_ctor_data*, uint32_t *flags*)

Initialize the data-plane process.

Should be called once per data-plane process (e.g. after each fork).

Copy code and data to internal memories and configure cmem configuration and fmt mapping

Parameters:

[in] *app_id* - Determines to which application id this application belong to.
[in] *cpu_id* - Determines the logical processor to attach the process to.
[in] *mem_ctor_func* - Pointer constructor function. Will be called per memory space. Used to initialize the memory. Number of call to the constructor will be according to number of memory replications For example: for 4 cluster memory, construct will be called 4 types to allow initialize memory in in each quad cluster
[in] *mem_ctor_data* - Data to provide to construct function
[in] *flags* - Bitwise OR of one or more of the EZDP_MEM_CFG_* flags defined above

Note:

mem_ctor_func should be one for all cpus of the applications and initialize all application memories

Returns:

- 0 (operation success), EINVAL (illegal/invalid configuration), ENODEV (ezdp_init_global was not called) EPROCUNAVAIL (data-plane process already initialize) The function may also fail for any of the errors specified for routines: open, ioctl, close, malloc, mmap, mprotect, sched_setaffinity, semop. In such case the errno value of the failure routine is returned Use ezdp_get_err_msg() API to get the detail error message of the failure

void ezdp_run (EZDP_MAIN_FUNC *func*, uint32_t *wait_count*)

Run a data plane application.

The function receives the data plane application's main frame handling loop function as a parameter. It performs the required preparation for the data plane application and then executes the passed function.

Parameters:

[in] *func* - Data plane application function to run.
[in] *wait_count* - Number of additional threads to wait for till starting the execution of data plane application function

Returns:

none

struct ezdp_version* ezdp_get_version (void) [read]

Return the version information of the ezdp library.

Use ezdp_version_get_string to get the info string.

Returns:

The pointer to the version info of the ezdp library.

char* ezdp_get_err_msg (void)

Return string of the last error.

Returns pointer to string. This string provide more details about the error returns by the functions which return error.

Returns:

char* - The pointer to the error message string

void ezdp_get_mem_section_info (struct ezdp_mem_section_info * *mem_info*, uint32_t *flags*)

Return sizes of the memory sections.

Parameters:

[out] *mem_info* - Pointer to ezdp_mem_section_info to write respond
[in] *flags* - Bitwise OR of one or more of the EZDP_MEM_CFG_* flags defined above

Returns:

void

const char* ezdp_mem_section_info_str (struct ezdp_mem_section_info * *mem_info*)

Return printable string of the memory sizes.

Parameters:

[in] *mem_info* - Pointer to ezdp_mem_section_info to convert to string format

Returns:

point to string

dpe/dp/include/ezdp_atomic.h File Reference

Functions

- static __always_inline uint32_t ezdp_atomic_read8_ext_addr (struct ezdp_ext_addr *addr)
- *Atomically read an 8 bit value from an extended address.* static __always_inline uint32_t ezdp_atomic_read16_ext_addr (struct ezdp_ext_addr *addr)
- *Atomically read a 16 bit value from an extended address.* static __always_inline uint32_t ezdp_atomic_read32_ext_addr (struct ezdp_ext_addr *addr)
- *Atomically read a 32 bit value from an extended address.* static __always_inline uint32_t ezdp_atomic_read32_sum_addr (ezdp_sum_addr_t addr)
- *Atomically read a 32 bit value from a summarized address.* static __always_inline void ezdp_atomic_read32_sum_addr_async (ezdp_sum_addr_t addr, uint32_t __cmem *value)
- *Non-blocking/posted version of ezdp_atomic_read32_sum_addr().* static __always_inline uint32_t ezdp_atomic_read64_sum_addr (ezdp_sum_addr_t addr, uint64_t __cmem *value)
- *Atomically read a 64 bit value from a summarized address.* static __always_inline void ezdp_atomic_read64_sum_addr_async (ezdp_sum_addr_t addr, uint64_t __cmem *value)
- *Non-blocking/posted version of ezdp_atomic_read64_sum_addr().* static __always_inline void ezdp_atomic_write8_ext_addr (struct ezdp_ext_addr *addr, uint8_t value)
- *Atomically write an 8 bit value to an extended address.* static __always_inline void ezdp_atomic_write8_ext_addr_async (struct ezdp_ext_addr *addr, uint8_t value)
- *Non-blocking/posted version of ezdp_atomic_write8_ext_addr().* static __always_inline void ezdp_atomic_write16_ext_addr (struct ezdp_ext_addr *addr, uint16_t value)
- *Atomically write a 16 bit value to an extended address.* static __always_inline void ezdp_atomic_write16_ext_addr_async (struct ezdp_ext_addr *addr, uint16_t value)
- *Non-blocking/posted version of ezdp_atomic_write16_ext_addr().* static __always_inline void ezdp_atomic_write32_ext_addr (struct ezdp_ext_addr *addr, uint32_t value)
- *Atomically write a 32 bit value to an extended address.* static __always_inline void ezdp_atomic_write32_ext_addr_async (struct ezdp_ext_addr *addr, uint32_t value)
- *Non-blocking/posted version of ezdp_atomic_write16_ext_addr().* static __always_inline void ezdp_atomic_write32_sum_addr (ezdp_sum_addr_t addr, uint32_t value)
- *Atomically write a 32 bit value to a summarized address.* static __always_inline void ezdp_atomic_write32_sum_addr_async (ezdp_sum_addr_t addr, uint32_t value)
- *Non-blocking/posted version of ezdp_atomic_write16_sum_addr().* static __always_inline void ezdp_atomic_write64_sum_addr (ezdp_sum_addr_t addr, uint64_t value)
- *Atomically write a 64 bit value to a summarized address.* static __always_inline void ezdp_atomic_write64_sum_addr_async (ezdp_sum_addr_t addr, uint64_t value)
- *Non-blocking/posted version of ezdp_atomic_write64_sum_addr().* static __always_inline uint32_t ezdp_atomic_xchg32_ext_addr (struct ezdp_ext_addr *addr, uint32_t value)
- *Atomically exchange a 32 bit value in an extended address.* static __always_inline uint32_t ezdp_atomic_xchg32_sum_addr (ezdp_sum_addr_t addr, uint32_t value)
- *Atomically exchange a 32 bit value in a summarized address.* static __always_inline uint32_t ezdp_atomic_cmpxchg8_ext_addr (uint8_t compare_value, struct ezdp_ext_addr *addr, uint8_t value)
- *Atomically compare and exchange an 8 bit value in an extended address.* static __always_inline uint32_t ezdp_atomic_cmpxchg16_ext_addr (uint16_t compare_value, struct ezdp_ext_addr *addr, uint16_t value)
- *Atomically compare and exchange a 16 bit value in an extended address.* static __always_inline uint32_t ezdp_atomic_cmpxchg32_ext_addr (uint32_t compare_value, struct ezdp_ext_addr *addr, uint32_t value)
- *Atomically compare and exchange a 32 bit value in an extended address.* static __always_inline uint32_t ezdp_atomic_cmpxchg32_sum_addr (uint32_t compare_value, ezdp_sum_addr_t addr, uint32_t value)
- *Atomically compare and exchange a 32 bit value in a summarized address.* static __always_inline uint32_t ezdp_atomic_read_and_tst8_ext_addr (struct ezdp_ext_addr *addr, bool *fail_flag)
- *Atomic read, test and set an 8 bit value in an extended address.* static __always_inline uint32_t ezdp_atomic_read_and_tst16_ext_addr (struct ezdp_ext_addr *addr, bool *fail_flag)
- *Atomic read, test and set a 16 bit value in an extended address.* static __always_inline uint32_t ezdp_atomic_read_and_tst32_ext_addr (struct ezdp_ext_addr *addr, bool *fail_flag)
- *Atomic read, test and set a 32 bit value in an extended address.* static __always_inline uint32_t ezdp_atomic_read_and_tst32_sum_addr (ezdp_sum_addr_t addr, bool *fail_flag)

- *Atomic read, test and set a 32 bit value in a summarized address.* static __always_inline uint32_t ezdp_atomic_read_and_clear8_ext_addr (struct ezdp_ext_addr *addr)
- *Atomically read and clear an 8-bit value in an extended address.* static __always_inline uint32_t ezdp_atomic_read_and_clear16_ext_addr (struct ezdp_ext_addr *addr)
- *Atomically read and clear a 16-bit value in an extended address.* static __always_inline uint32_t ezdp_atomic_read_and_clear32_ext_addr (struct ezdp_ext_addr *addr)
- *Atomically read and clear a 32-bit value in an extended address.* static __always_inline uint32_t ezdp_atomic_read_and_clear32_sum_addr (ezdp_sum_addr_t addr)
- *Atomically read and clear a 32-bit value in a summarized address.* static __always_inline uint32_t ezdp_atomic_read_and_clear64_sum_addr (ezdp_sum_addr_t addr, uint64_t __cmem *orig_value)
- *Atomically read and clear a 64-bit value in a summarized address.* static __always_inline void ezdp_atomic_add8_ext_addr (struct ezdp_ext_addr *addr, int8_t value)
- *Atomically perform an 8 bit logical ADD operation on an extended address.* static __always_inline void ezdp_atomic_add8_ext_addr_async (struct ezdp_ext_addr *addr, int8_t value)
- *Non-blocking/posted version of ezdp_atomic_add8_ext_addr().* static __always_inline int32_t ezdp_atomic_read_and_add8_ext_addr (struct ezdp_ext_addr *addr, int8_t value, bool *overflow_flag)
- *Atomically read and perform an 8 bit logical ADD operation on an extended address.* static __always_inline void ezdp_atomic_add16_ext_addr (struct ezdp_ext_addr *addr, int16_t value)
- *Atomically perform a 16 bit logical ADD operation on an extended address.* static __always_inline void ezdp_atomic_add16_ext_addr_async (struct ezdp_ext_addr *addr, int16_t value)
- *Non-blocking/posted version of ezdp_atomic_add16_ext_addr().* static __always_inline int32_t ezdp_atomic_read_and_add16_ext_addr (struct ezdp_ext_addr *addr, int16_t value, bool *overflow_flag)
- *Atomically read and perform a 16 bit logical ADD operation on an extended address.* static __always_inline void ezdp_atomic_add32_ext_addr (struct ezdp_ext_addr *addr, int32_t value)
- *Atomically perform a 32 bit logical ADD operation on an extended address.* static __always_inline void ezdp_atomic_add32_ext_addr_async (struct ezdp_ext_addr *addr, int32_t value)
- *Non-blocking/posted version of ezdp_atomic_add32_ext_addr().* static __always_inline int32_t ezdp_atomic_read_and_add32_ext_addr (struct ezdp_ext_addr *addr, int32_t value, bool *overflow_flag)
- *Atomically read and perform a 32 bit logical ADD operation on an extended address.* static __always_inline void ezdp_atomic_add32_sum_addr (ezdp_sum_addr_t addr, int32_t value)
- *Atomically perform a 32 bit logical ADD operation on a summarized address.* static __always_inline void ezdp_atomic_add32_sum_addr_async (ezdp_sum_addr_t addr, int32_t value)
- *Non-blocking/posted version of ezdp_atomic_add32_sum_addr().* static __always_inline int32_t ezdp_atomic_read_and_add32_sum_addr (ezdp_sum_addr_t addr, int32_t value, bool *overflow_flag)
- *Atomically read and perform a 32 bit logical ADD operation on a summarized address.* static __always_inline void ezdp_atomic_add64_sum_addr (ezdp_sum_addr_t addr, int32_t value)
- *Atomically perform a 64 bit logical ADD operation on a summarized address.* static __always_inline void ezdp_atomic_add64_sum_addr_async (ezdp_sum_addr_t addr, int32_t value)
- *Non-blocking/posted version of ezdp_atomic_add64_sum_addr().* static __always_inline int32_t ezdp_atomic_read_and_add64_sum_addr (ezdp_sum_addr_t addr, int32_t value, int64_t __cmem *orig_value, bool *overflow_flag)
- *Atomically read and perform a 64 bit logical ADD operation on a summarized address.* static __always_inline void ezdp_atomic_dual_add32_ext_addr (struct ezdp_ext_addr *addr, int16_t value1, int16_t value2)
- *Atomically perform a dual ADD operation to the two 32-bit variables pointed to by the extended address.* static __always_inline void ezdp_atomic_dual_add32_ext_addr_async (struct ezdp_ext_addr *addr, int16_t value1, int16_t value2)
- *Non-blocking/posted version of ezdp_atomic_dual_add32_ext_addr().* static __always_inline void ezdp_atomic_dual_add32_sum_addr (ezdp_sum_addr_t addr, int16_t value1, int16_t value2)
- *Atomically perform a dual ADD operation to the two 32-bit variables pointed to by the summarized address.* static __always_inline void ezdp_atomic_dual_add32_sum_addr_async (ezdp_sum_addr_t addr, int16_t value1, int16_t value2)
- *Non-blocking/posted version of ezdp_atomic_dual_add32_sum_addr().* static __always_inline int32_t ezdp_atomic_read_and_dual_add32_sum_addr (ezdp_sum_addr_t addr, int16_t value1, int16_t value2, struct ezdp_dual_add32_result __cmem *orig_value, bool *overflow_flag)
- *Atomically read and perform a dual ADD operation to the two 32-bit variables pointed to by the summarized address.* static __always_inline void ezdp_atomic_dual_add64_sum_addr (ezdp_sum_addr_t addr, int16_t value1, int16_t value2)

- *Atomically perform dual ADD operation to the two 64-bit variables pointed to by the summarized address.* static __always_inline void ezdp_atomic_dual_add64_sum_addr_async (ezdp_sum_addr_t addr, int16_t value1, int16_t value2)
- *Non-blocking/posted version of ezdp_atomic_dual_add64_sum_addr().* static __always_inline int32_t ezdp_atomic_read_and_dual_add64_sum_addr (ezdp_sum_addr_t addr, int16_t value1, int16_t value2, struct ezdp_dual_add64_result __cmem *orig_value, bool *overflow_flag)
- *Atomically read and perform a dual ADD operation to the two 64-bit variables pointed to by the summarized address.* static __always_inline uint32_t ezdp_atomic_read_and_inc8_ext_addr (struct ezdp_ext_addr *addr, bool *zero_flag)
- *Atomically read and increment an 8 bit value in an extended address.* static __always_inline uint32_t ezdp_atomic_read_and_inc16_ext_addr (struct ezdp_ext_addr *addr, bool *zero_flag)
- *Atomically read and increment a 16 bit value in an extended address.* static __always_inline uint32_t ezdp_atomic_read_and_inc32_ext_addr (struct ezdp_ext_addr *addr, bool *zero_flag)
- *Atomically read and increment a 32 bit value in an extended address.* static __always_inline uint32_t ezdp_atomic_read_and_inc32_sum_addr (ezdp_sum_addr_t addr, bool *zero_flag)
- *Atomically read and increment a 32 bit value in a summarized address.* static __always_inline uint32_t ezdp_atomic_read_and_inc64_sum_addr (ezdp_sum_addr_t addr, uint64_t __cmem *orig_value, bool *zero_flag)
- *Atomically read and increment a 64 bit value in a summarized address.* static __always_inline uint32_t ezdp_atomic_read_and_dec8_ext_addr (struct ezdp_ext_addr *addr, bool *unaffected_flag)
- *Atomically read and decrement conditionally by one an 8-bit value in an extended address (zero value does not underflow).* static __always_inline uint32_t ezdp_atomic_read_and_dec16_ext_addr (struct ezdp_ext_addr *addr, bool *unaffected_flag)
- *Atomically read and decrement conditionally by one a 16-bit value in an extended address (zero value does not underflow).* static __always_inline uint32_t ezdp_atomic_read_and_dec32_ext_addr (struct ezdp_ext_addr *addr, bool *unaffected_flag)
- *Atomically read and decrement conditionally by one a 32-bit value in an extended address (zero value does not underflow).* static __always_inline uint32_t ezdp_atomic_read_and_dec32_sum_addr (ezdp_sum_addr_t addr, bool *unaffected_flag)
- *Atomically read and decrement conditionally by one a 32-bit value in a summarized address (zero value does not underflow).* static __always_inline uint32_t ezdp_atomic_read_and_dec64_sum_addr (ezdp_sum_addr_t addr, uint64_t __cmem *orig_value, bool *unaffected_flag)
- *Atomically read and decrement conditionally by one a 64-bit value in a summarized address (zero value does not underflow).* static __always_inline uint32_t ezdp_atomic_read_and_inc32_cond_ext_addr (struct ezdp_ext_addr *addr, bool res_mode, bool *unaffected_flag)
- *Atomically read and increment conditionally a 32-bit value in an extended address.* static __always_inline uint32_t ezdp_atomic_read_and_inc32_cond_sum_addr (ezdp_sum_addr_t addr, bool res_mode, bool *unaffected_flag)
- *Atomically read and increment conditionally a 32-bit value in a summarized address.* static __always_inline void ezdp_atomic_and8_ext_addr (struct ezdp_ext_addr *addr, uint8_t value)
- *Atomically perform an 8 bit logical AND operation on an extended address.* static __always_inline void ezdp_atomic_and8_ext_addr_async (struct ezdp_ext_addr *addr, uint8_t value)
- *Non-blocking/posted version of ezdp_atomic_and8_ext_addr().* static __always_inline uint32_t ezdp_atomic_read_and_and8_ext_addr (struct ezdp_ext_addr *addr, uint8_t value, bool *no_chng_flag)
- *Atomically read and perform an 8 bit logical AND operation on an extended address.* static __always_inline void ezdp_atomic_and16_ext_addr (struct ezdp_ext_addr *addr, uint16_t value)
- *Atomically perform a 16 bit logical AND operation on an extended address.* static __always_inline void ezdp_atomic_and16_ext_addr_async (struct ezdp_ext_addr *addr, uint16_t value)
- *Non-blocking/posted version of ezdp_atomic_and16_ext_addr().* static __always_inline uint32_t ezdp_atomic_read_and_and16_ext_addr (struct ezdp_ext_addr *addr, uint16_t value, bool *no_chng_flag)
- *Atomically read and perform a 16 bit logical AND operation on an extended address.* static __always_inline void ezdp_atomic_and32_ext_addr (struct ezdp_ext_addr *addr, uint32_t value)
- *Atomically perform a 32 bit logical AND operation on an extended address.* static __always_inline void ezdp_atomic_and32_ext_addr_async (struct ezdp_ext_addr *addr, uint32_t value)
- *Non-blocking/posted version of ezdp_atomic_and32_ext_addr_async().* static __always_inline uint32_t ezdp_atomic_read_and_and32_ext_addr (struct ezdp_ext_addr *addr, uint32_t value, bool *no_chng_flag)
- *Atomically read and perform a 32 bit logical AND operation on an extended address.* static __always_inline void ezdp_atomic_and32_sum_addr (ezdp_sum_addr_t addr, uint32_t value)

- *Atomically perform a 32 bit logical AND operation on a summarized address.* static __always_inline void ezdp_atomic_and32_sum_addr_async (ezdp_sum_addr_t addr, uint32_t value)
- *Non-blocking/posted version of ezdp_atomic_and32_sum_addr_async().* static __always_inline uint32_t ezdp_atomic_read_and_and32_sum_addr (ezdp_sum_addr_t addr, uint32_t value, bool *no_chng_flag)
- *Atomically read and perform a 32 bit logical AND operation on a summarized address.* static __always_inline void ezdp_atomic_or8_ext_addr (struct ezdp_ext_addr *addr, uint8_t value)
- *Atomically perform an 8 bit logical OR operation on an extended address.* static __always_inline void ezdp_atomic_or8_ext_addr_async (struct ezdp_ext_addr *addr, uint8_t value)
- *Non-blocking/posted version of ezdp_atomic_or8_ext_addr().* static __always_inline uint32_t ezdp_atomic_read_and_or8_ext_addr (struct ezdp_ext_addr *addr, uint8_t value, bool *no_chng_flag)
- *Atomically read and perform an 8 bit logical OR operation on an extended address.* static __always_inline void ezdp_atomic_or8_sum_addr (ezdp_sum_addr_t addr, uint8_t value)
- *Atomically perform an 8 bit logical OR operation on a summarized address.* static __always_inline void ezdp_atomic_or8_sum_addr_async (ezdp_sum_addr_t addr, uint8_t value)
- *Non-blocking/posted version of ezdp_atomic_or8_sum_addr().* static __always_inline void ezdp_atomic_or16_ext_addr (struct ezdp_ext_addr *addr, uint16_t value)
- *Atomically perform a 16 bit logical OR operation on an extended address.* static __always_inline void ezdp_atomic_or16_ext_addr_async (struct ezdp_ext_addr *addr, uint16_t value)
- *Non-blocking/posted version of ezdp_atomic_or16_ext_addr().* static __always_inline uint32_t ezdp_atomic_read_and_or16_ext_addr (struct ezdp_ext_addr *addr, uint16_t value, bool *no_chng_flag)
- *Atomically read and perform a 16 bit logical OR operation on an extended address.* static __always_inline void ezdp_atomic_or16_sum_addr (ezdp_sum_addr_t addr, uint16_t value)
- *Atomically perform a 16 bit logical OR operation on a summarized address.* static __always_inline void ezdp_atomic_or16_sum_addr_async (ezdp_sum_addr_t addr, uint16_t value)
- *Non-blocking/posted version of ezdp_atomic_or16_sum_addr().* static __always_inline void ezdp_atomic_or32_ext_addr (struct ezdp_ext_addr *addr, uint32_t value)
- *Atomically perform a 32 bit logical OR operation on an extended address.* static __always_inline void ezdp_atomic_or32_ext_addr_async (struct ezdp_ext_addr *addr, uint32_t value)
- *Non-blocking/posted version of ezdp_atomic_or32_ext_addr().* static __always_inline uint32_t ezdp_atomic_read_and_or32_ext_addr (struct ezdp_ext_addr *addr, uint32_t value, bool *no_chng_flag)
- *Atomically read and perform a 32 bit logical OR operation on an extended address.* static __always_inline void ezdp_atomic_or32_sum_addr (ezdp_sum_addr_t addr, uint32_t value)
- *Atomically perform a 32 bit logical OR operation on a summarized address.* static __always_inline void ezdp_atomic_or32_sum_addr_async (ezdp_sum_addr_t addr, uint32_t value)
- *Non-blocking/posted version of ezdp_atomic_or32_sum_addr().* static __always_inline uint32_t ezdp_atomic_read_and_or32_sum_addr (ezdp_sum_addr_t addr, uint32_t value, bool *no_chng_flag)
- *Atomically read and perform a 32 bit logical OR operation on a summarized address.* static __always_inline void ezdp_atomic_xor8_ext_addr (struct ezdp_ext_addr *addr, uint8_t value)
- *Atomically perform an 8 bit logical XOR operation on an extended address.* static __always_inline void ezdp_atomic_xor8_ext_addr_async (struct ezdp_ext_addr *addr, uint8_t value)
- *Non-blocking/posted version of ezdp_atomic_xor8_ext_addr().* static __always_inline uint32_t ezdp_atomic_read_and_xor8_ext_addr (struct ezdp_ext_addr *addr, uint8_t value, bool *no_id_flag)
- *Atomically read and perform an 8 bit logical XOR operation on an extended address.* static __always_inline void ezdp_atomic_xor16_ext_addr (struct ezdp_ext_addr *addr, uint16_t value)
- *Atomically perform a 16 bit logical XOR operation on an extended address.* static __always_inline void ezdp_atomic_xor16_ext_addr_async (struct ezdp_ext_addr *addr, uint16_t value)
- *Non-blocking/posted version of ezdp_atomic_xor8_ext_addr().* static __always_inline uint32_t ezdp_atomic_read_and_xor16_ext_addr (struct ezdp_ext_addr *addr, uint16_t value, bool *no_id_flag)
- *Atomically read and perform a 16 bit logical XOR operation on an extended address.* static __always_inline void ezdp_atomic_xor32_ext_addr (struct ezdp_ext_addr *addr, uint32_t value)
- *Atomically perform a 32 bit logical XOR operation on an extended address.* static __always_inline void ezdp_atomic_xor32_ext_addr_async (struct ezdp_ext_addr *addr, uint32_t value)
- *Non-blocking/posted version of ezdp_atomic_xor32_ext_addr().* static __always_inline uint32_t ezdp_atomic_read_and_xor32_ext_addr (struct ezdp_ext_addr *addr, uint32_t value, bool *no_id_flag)
- *Atomically read and perform a 32 bit logical XOR operation on an extended address.* static __always_inline void ezdp_atomic_xor32_sum_addr (ezdp_sum_addr_t addr, uint32_t value)
- *Atomically perform a 32 bit logical XOR operation on a summarized address.* static __always_inline void ezdp_atomic_xor32_sum_addr_async (ezdp_sum_addr_t addr, uint32_t value)

- *Non-blocking/posted version of ezdp_atomic_xor32_sum_addr().* static __always_inline uint32_t ezdp_atomic_read_and_xor32_sum_addr (ezdp_sum_addr_t addr, uint32_t value, bool *no_id_flag)

Atomically read and perform a 32 bit logical XOR operation on a summarized address.

Function Documentation

static __always_inline uint32_t ezdp_atomic_read8_ext_addr (struct ezdp_ext_addr * *addr*) [static]

Atomically read an 8 bit value from an extended address.

Parameters:

[in] *addr* - pointer to extended address

Returns:

Read value. The value is limited to 8 bits

static __always_inline uint32_t ezdp_atomic_read16_ext_addr (struct ezdp_ext_addr * *addr*) [static]

Atomically read a 16 bit value from an extended address.

Parameters:

[in] *addr* - pointer to extended address

Note:

Address must be 2-byte aligned.

Returns:

Read value. The value is limited to 16 bits

static __always_inline uint32_t ezdp_atomic_read32_ext_addr (struct ezdp_ext_addr * *addr*) [static]

Atomically read a 32 bit value from an extended address.

Parameters:

[in] *addr* - pointer to extended address

Note:

Address must be 4-byte aligned.

Returns:

Read value

static __always_inline uint32_t ezdp_atomic_read32_sum_addr (ezdp_sum_addr_t *addr*) [static]

Atomically read a 32 bit value from a summarized address.

Parameters:

[in] *addr* - summarized address

Returns:
Read value

static __always_inline void ezdp_atomic_read32_sum_addr_async (ezdp_sum_addr_t *addr*, uint32_t __cmem * *value*) [static]

Non-blocking/posted version of ezdp_atomic_read32_sum_addr().

Parameters:
[in] *addr* - summarized address
[out] *value* - pointer in CMEM to write the value in

Returns:
void

static __always_inline uint32_t ezdp_atomic_read64_sum_addr (ezdp_sum_addr_t *addr*, uint64_t __cmem * *value*) [static]

Atomically read a 64 bit value from a summarized address.

Parameters:
[in] *addr* - summarized address
[out] *value* - pointer in CMEM to write the value in

Returns:
The 32 MSB of the read value

static __always_inline void ezdp_atomic_read64_sum_addr_async (ezdp_sum_addr_t *addr*, uint64_t __cmem * *value*) [static]

Non-blocking/posted version of ezdp_atomic_read64_sum_addr().

Parameters:
[in] *addr* - summarized address
[out] *value* - pointer in CMEM to write the value in

Returns:
void

static __always_inline void ezdp_atomic_write8_ext_addr (struct ezdp_ext_addr * *addr*, uint8_t *value*) [static]

Atomically write an 8 bit value to an extended address.

Parameters:
[in] *addr* - pointer to extended address
[in] *value* - write value

Returns:
void

static __always_inline void ezdp_atomic_write8_ext_addr_async (struct ezdp_ext_addr * *addr*, uint8_t *value*) [static]

Non-blocking/posted version of ezdp_atomic_write8_ext_addr().

Parameters:

[in] *addr* - pointer to extended address
[in] *value* - write value

Returns:

void

static __always_inline void ezdp_atomic_write16_ext_addr (struct ezdp_ext_addr * *addr*, uint16_t *value*) [static]

Atomically write a 16 bit value to an extended address.

Parameters:

[in] *addr* - pointer to extended address
[in] *value* - write value

Note:

Address must be 2-byte aligned.

Returns:

void

static __always_inline void ezdp_atomic_write16_ext_addr_async (struct ezdp_ext_addr * *addr*, uint16_t *value*) [static]

Non-blocking/posted version of ezdp_atomic_write16_ext_addr().

Parameters:

[in] *addr* - pointer to extended address
[in] *value* - write value

Note:

Address must be 2-byte aligned.

Returns:

void

static __always_inline void ezdp_atomic_write32_ext_addr (struct ezdp_ext_addr * *addr*, uint32_t *value*) [static]

Atomically write a 32 bit value to an extended address.

Parameters:

[in] *addr* - pointer to extended address
[in] *value* - write value

Note:

Address must be 4-byte aligned.

Returns:
void

static __always_inline void ezdp_atomic_write32_ext_addr_async (struct ezdp_ext_addr * *addr*, uint32_t *value*) `[static]`

Non-blocking/posted version of ezdp_atomic_write16_ext_addr().

Parameters:
[in] *addr* - pointer to extended address
[in] *value* - write value

Note:
Address must be 4-byte aligned.

Returns:
void

static __always_inline void ezdp_atomic_write32_sum_addr (ezdp_sum_addr_t *addr*, uint32_t *value*) `[static]`

Atomically write a 32 bit value to a summarized address.

Parameters:
[in] *addr* - summarized address
[in] *value* - write value

Returns:
void

static __always_inline void ezdp_atomic_write32_sum_addr_async (ezdp_sum_addr_t *addr*, uint32_t *value*) `[static]`

Non-blocking/posted version of ezdp_atomic_write16_sum_addr().

Parameters:
[in] *addr* - summarized address
[in] *value* - write value

Returns:
void

static __always_inline void ezdp_atomic_write64_sum_addr (ezdp_sum_addr_t *addr*, uint64_t *value*) `[static]`

Atomically write a 64 bit value to a summarized address.

Parameters:
[in] *addr* - summarized address
[in] *value* - write value

Returns:
void

static __always_inline void ezdp_atomic_write64_sum_addr_async (ezdp_sum_addr_t *addr*, uint64_t *value*) [static]

Non-blocking/posted version of ezdp_atomic_write64_sum_addr().

Parameters:
[in] *addr* - summarized address
[in] *value* - write value

Returns:
void

static __always_inline uint32_t ezdp_atomic_xchg32_ext_addr (struct ezdp_ext_addr * *addr*, uint32_t *value*) [static]

Atomically exchange a 32 bit value in an extended address.

Parameters:
[in] *addr* - pointer to extended address
[in] *value* - value to exchange

Note:
Address must be 4-byte aligned.

Returns:
Original value.

static __always_inline uint32_t ezdp_atomic_xchg32_sum_addr (ezdp_sum_addr_t *addr*, uint32_t *value*) [static]

Atomically exchange a 32 bit value in a summarized address.

Parameters:
[in] *addr* - summarized address
[in] *value* - value to exchange

Returns:
Original value

static __always_inline uint32_t ezdp_atomic_cmpxchg8_ext_addr (uint8_t *compare_value*, struct ezdp_ext_addr * *addr*, uint8_t *value*) [static]

Atomically compare and exchange an 8 bit value in an extended address.

Compares the passed value with the existing value in memory, and exchanges the values only if they are equal.

Parameters:
[in] *compare_value* - value to compare before the change
[in] *addr* - pointer to extended address
[in] *value* - value to exchange

Returns:
Original value The return value is limited to 8 bits

static __always_inline uint32_t ezdp_atomic_cmpxchg16_ext_addr (uint16_t *compare_value*, struct ezdp_ext_addr * *addr*, uint16_t *value*) [static]

Atomically compare and exchange a 16 bit value in an extended address.

Compares the passed value with the existing value in memory, and exchanges the values only if they are equal.

Parameters:

[in] *compare_value* - value to compare before the change
[in] *addr* - pointer to extended address
[in] *value* - value to exchange

Note:

Address must be 2-byte aligned.

Returns:

Original value The return value is limited to 16 bits

static __always_inline uint32_t ezdp_atomic_cmpxchg32_ext_addr (uint32_t *compare_value*, struct ezdp_ext_addr * *addr*, uint32_t *value*) [static]

Atomically compare and exchange a 32 bit value in an extended address.

Compares the passed value with the existing value in memory, and exchanges the values only if they are equal.

Parameters:

[in] *compare_value* - value to compare before the change
[in] *addr* - pointer to extended address
[in] *value* - value to exchange

Note:

Address must be 4-byte aligned.

Returns:

Original value

static __always_inline uint32_t ezdp_atomic_cmpxchg32_sum_addr (uint32_t *compare_value*, ezdp_sum_addr_t *addr*, uint32_t *value*) [static]

Atomically compare and exchange a 32 bit value in a summarized address.

Compares the passed value with the existing value in memory, and exchanges the values only if they are equal.

Parameters:

[in] *compare_value* - value to compare before the change
[in] *addr* - summarized address
[in] *value* - value to exchange

Returns:

Original value

static __always_inline uint32_t ezdp_atomic_read_and_tst8_ext_addr (struct ezdp_ext_addr * *addr*, bool * *fail_flag*) [static]

Atomic read, test and set an 8 bit value in an extended address.

Parameters:

[in] *addr* - pointer to extended address
[out] *fail_flag* - will be true if failed in setting the bits

Returns:

Original/Read value. The value is limited to 8 bits

static __always_inline uint32_t ezdp_atomic_read_and_tst16_ext_addr (struct ezdp_ext_addr * *addr*, bool * *fail_flag*) `[static]`

Atomic read, test and set a 16 bit value in an extended address.

Parameters:

[in] *addr* - pointer to extended address
[out] *fail_flag* - will be true if failed in setting the bits

Note:

Address must be 2-byte aligned.

Returns:

Original/Read value. The value is limited to 16 bits

static __always_inline uint32_t ezdp_atomic_read_and_tst32_ext_addr (struct ezdp_ext_addr * *addr*, bool * *fail_flag*) `[static]`

Atomic read, test and set a 32 bit value in an extended address.

Parameters:

[in] *addr* - pointer to extended address
[out] *fail_flag* - will be true if failed in setting the bits

Note:

Address must be 4-byte aligned.

Returns:

Original/Read value.

static __always_inline uint32_t ezdp_atomic_read_and_tst32_sum_addr (ezdp_sum_addr_t *addr*, bool * *fail_flag*) `[static]`

Atomic read, test and set a 32 bit value in a summarized address.

Parameters:

[in] *addr* - summarized address
[out] *fail_flag* - will be true if failed in setting the bits

Returns:

Original/Read value.

static __always_inline uint32_t ezdp_atomic_read_and_clear8_ext_addr (struct ezdp_ext_addr * *addr*) `[static]`

Atomically read and clear an 8-bit value in an extended address.

Parameters:

[in] *addr* - pointer to extended address

Returns:

Original/Read value. The value is limited to 8 bits

static __always_inline uint32_t ezdp_atomic_read_and_clear16_ext_addr (struct ezdp_ext_addr * *addr*) `[static]`

Atomically read and clear a 16-bit value in an extended address.

Parameters:

[in] *addr* - pointer to extended address

Note:

Address must be 2-byte aligned.

Returns:

Original/Read value. The value is limited to 16 bits

static __always_inline uint32_t ezdp_atomic_read_and_clear32_ext_addr (struct ezdp_ext_addr * *addr*) `[static]`

Atomically read and clear a 32-bit value in an extended address.

Parameters:

[in] *addr* - pointer to extended address

Note:

Address must be 4-byte aligned.

Returns:

Original/Read value.

static __always_inline uint32_t ezdp_atomic_read_and_clear32_sum_addr (ezdp_sum_addr_t *addr*) `[static]`

Atomically read and clear a 32-bit value in a summarized address.

Parameters:

[in] *addr* - summarized address

Returns:

Original/Read value.

static __always_inline uint32_t ezdp_atomic_read_and_clear64_sum_addr (ezdp_sum_addr_t *addr*, uint64_t __cmem * *orig_value*) `[static]`

Atomically read and clear a 64-bit value in a summarized address.

Parameters:

[in] *addr* - summarized address

[out] *orig_value* - the address in CMEM to write the original value in

Returns:
The 32 MSB of the original/read value

static __always_inline void ezdp_atomic_add8_ext_addr (struct ezdp_ext_addr * *addr*, int8_t *value*) [static]

Atomically perform an 8 bit logical ADD operation on an extended address.

Performs a logical ADD operation between the existing value in memory and the value passed, and stores the result back into memory.

Parameters:
[in] *addr* - pointer to extended address
[in] *value* - value to add

Returns:
void

static __always_inline void ezdp_atomic_add8_ext_addr_async (struct ezdp_ext_addr * *addr*, int8_t *value*) [static]

Non-blocking/posted version of ezdp_atomic_add8_ext_addr().

Parameters:
[in] *addr* - pointer to extended address
[in] *value* - value to add

Note:
Call ezdp_sync() to wait for the operation to complete, confirming that the data was copied to the destination.

Returns:
none

static __always_inline int32_t ezdp_atomic_read_and_add8_ext_addr (struct ezdp_ext_addr * *addr*, int8_t *value*, bool * *overflow_flag*) [static]

Atomically read and perform an 8 bit logical ADD operation on an extended address.

Performs a logical ADD operation between the existing value in memory and the value passed, and stores the result back into memory.

Parameters:
[in] *addr* - pointer to extended address
[in] *value* - value to add
[out] *overflow_flag* - will be true if an overflow occurred An overflow on signed add is defined as changing originally positive memory contents to be negative when adding a positive number, or changing originally negative memory contents to be positive when adding a negative number

Returns:
Original/Read value. The value is limited to 8 bits

static __always_inline void ezdp_atomic_add16_ext_addr (struct ezdp_ext_addr * *addr*, int16_t *value*) [static]

Atomically perform a 16 bit logical ADD operation on an extended address.

Performs a logical ADD operation between the existing value in memory and the value passed, and stores the result back into memory.

Parameters:

[in] *addr* - pointer to extended address
[in] *value* - update value

Returns:

void

static __always_inline void ezdp_atomic_add16_ext_addr_async (struct ezdp_ext_addr * *addr*, int16_t *value*) `[static]`

Non-blocking/posted version of ezdp_atomic_add16_ext_addr().

Parameters:

[in] *addr* - pointer to extended address
[in] *value* - value to add

Note:

Call ezdp_sync() to wait for the operation to complete, confirming that the data was copied to the destination.

Returns:

void

static __always_inline int32_t ezdp_atomic_read_and_add16_ext_addr (struct ezdp_ext_addr * *addr*, int16_t *value*, bool * *overflow_flag*) `[static]`

Atomically read and perform a 16 bit logical ADD operation on an extended address.

Performs a logical ADD operation between the existing value in memory and the value passed, and stores the result back into memory.

Parameters:

[in] *addr* - pointer to extended address
[in] *value* - update value
[out] *overflow_flag* - will be true if an overflow occurred An overflow on signed add is defined as changing originally positive memory contents to be negative when adding a positive number, or changing originally negative memory contents to be positive when adding a negative number

Returns:

Original/Read value. The value is limited to 16 bits

static __always_inline void ezdp_atomic_add32_ext_addr (struct ezdp_ext_addr * *addr*, int32_t *value*) `[static]`

Atomically perform a 32 bit logical ADD operation on an extended address.

Performs a logical ADD operation between the existing value in memory and the value passed, and stores the result back into memory.

Parameters:

[in] *addr* - pointer to extended address
[in] *value* - value to add

Returns:

void

static __always_inline void ezdp_atomic_add32_ext_addr_async (struct ezdp_ext_addr * *addr*, int32_t *value*) `[static]`

Non-blocking/posted version of ezdp_atomic_add32_ext_addr().

Parameters:

[in] *addr* - pointer to extended address
[in] *value* - value to add

Note:

Call ezdp_sync() to wait for the operation to complete, confirming that the data was copied to the destination.

Returns:

void

static __always_inline int32_t ezdp_atomic_read_and_add32_ext_addr (struct ezdp_ext_addr * *addr*, int32_t *value*, bool * *overflow_flag*) [static]

Atomically read and perform a 32 bit logical ADD operation on an extended address.

Performs a logical ADD operation between the existing value in memory and the value passed, and stores the result back into memory.

Parameters:

[in] *addr* - pointer to extended address
[in] *value* - value to add
[out] *overflow_flag* - will be true if an overflow occurred An overflow on signed add is defined as changing originally positive memory contents to be negative when adding a positive number, or changing originally negative memory contents to be positive when adding a negative number

Returns:

Original/Read value.

static __always_inline void ezdp_atomic_add32_sum_addr (ezdp_sum_addr_t *addr*, int32_t *value*) [static]

Atomically perform a 32 bit logical ADD operation on a summarized address.

Performs a logical ADD operation between the existing value in memory and the value passed, and stores the result back into memory.

Parameters:

[in] *addr* - summarized address
[in] *value* - value to add

Returns:

void

static __always_inline void ezdp_atomic_add32_sum_addr_async (ezdp_sum_addr_t *addr*, int32_t *value*) [static]

Non-blocking/posted version of ezdp_atomic_add32_sum_addr().

Parameters:

[in] *addr* - summarized address
[in] *value* - value to add

Note:

Call ezdp_sync() to wait for the operation to complete, confirming that the data was copied to the destination.

Returns:
void

static __always_inline int32_t ezdp_atomic_read_and_add32_sum_addr (ezdp_sum_addr_t *addr*, int32_t *value*, bool * *overflow_flag*) [static]

Atomically read and perform a 32 bit logical ADD operation on a summarized address.

Performs a logical ADD operation between the existing value in memory and the value passed, and stores the result back into memory.

Parameters:
[in] *addr* - summarized address
[in] *value* - value to add
[out] *overflow_flag* - will be true if an overflow occurred An overflow on signed add is defined as changing originally positive memory contents to be negative when adding a positive number, or changing originally negative memory contents to be positive when adding a negative number

Returns:
Original/Read value.

static __always_inline void ezdp_atomic_add64_sum_addr (ezdp_sum_addr_t *addr*, int32_t *value*) [static]

Atomically perform a 64 bit logical ADD operation on a summarized address.

Performs a logical ADD operation between the existing value in memory and the value passed, and stores the result back into memory.

Parameters:
[in] *addr* - summarized address
[in] *value* - value to add

Returns:
void

static __always_inline void ezdp_atomic_add64_sum_addr_async (ezdp_sum_addr_t *addr*, int32_t *value*) [static]

Non-blocking/posted version of ezdp_atomic_add64_sum_addr().

Parameters:
[in] *addr* - summarized address
[in] *value* - value to add

Note:
Call ezdp_sync() to wait for the operation to complete, confirming that the data was copied to the destination.

Returns:
void

static __always_inline int32_t ezdp_atomic_read_and_add64_sum_addr (ezdp_sum_addr_t *addr*, int32_t *value*, int64_t __cmem * *orig_value*, bool * *overflow_flag*) [static]

Atomically read and perform a 64 bit logical ADD operation on a summarized address.

Performs a logical ADD operation between the existing value in memory and the value passed, and stores the result back into memory.

Parameters:

[in] *addr* - summarized address
[in] *value* - value to add
[out] *orig_value* - the address in CMEM to write the original value in
[out] *overflow_flag* - will be true if an overflow occurred An overflow on signed add is defined as changing originally positive memory contents to be negative when adding a positive number, or changing originally negative memory contents to be positive when adding a negative number

Returns:

The 32 MSB of the original/read value

static __always_inline void ezdp_atomic_dual_add32_ext_addr (struct ezdp_ext_addr * *addr*, int16_t *value1*, int16_t *value2*) [static]

Atomically perform a dual ADD operation to the two 32-bit variables pointed to by the extended address.

Performs a dual ADD operation between the existing 2 values, 32 bits each, in memory and the 2 values passed, and stores the result back into memory.

Parameters:

[in] *addr* - pointer to extended address
[in] *value1* - value1 to add to variable 1
[in] *value2* - value2 to add to variable 2

Returns:

void

static __always_inline void ezdp_atomic_dual_add32_ext_addr_async (struct ezdp_ext_addr * *addr*, int16_t *value1*, int16_t *value2*) [static]

Non-blocking/posted version of ezdp_atomic_dual_add32_ext_addr().

Parameters:

[in] *addr* - pointer to extended address
[in] *value1* - value1 to add to variable 1
[in] *value2* - value2 to add to variable 2

Note:

Call ezdp_sync() to wait for the operation to complete, confirming that the data was copied to the destination.

Returns:

none

static __always_inline void ezdp_atomic_dual_add32_sum_addr (ezdp_sum_addr_t *addr*, int16_t *value1*, int16_t *value2*) [static]

Atomically perform a dual ADD operation to the two 32-bit variables pointed to by the summarized address.

Performs a dual ADD operation between the existing 2 values, 32 bits each, in memory and the 2 values passed, and stores the result back into memory.

Parameters:

[in] *addr* - summarized address
[in] *value1* - value1 to add to variable 1
[in] *value2* - value2 to add to variable 2

Returns:

void

static __always_inline void ezdp_atomic_dual_add32_sum_addr_async (ezdp_sum_addr_t *addr*, int16_t *value1*, int16_t *value2*) [static]

Non-blocking/posted version of ezdp_atomic_dual_add32_sum_addr().

Parameters:

[in] *addr* - summarized address
[in] *value1* - value1 to add to variable 1
[in] *value2* - value2 to add to variable 2

Note:

Call ezdp_sync() to wait for the operation to complete, confirming that the data was copied to the destination.

Returns:

none

static __always_inline int32_t ezdp_atomic_read_and_dual_add32_sum_addr (ezdp_sum_addr_t *addr*, int16_t *value1*, int16_t *value2*, struct ezdp_dual_add32_result __cmem * *orig_value*, bool * *overflow_flag*) [static]

Atomically read and perform a dual ADD operation to the two 32-bit variables pointed to by the summarized address.

Performs a dual ADD operation between the existing 2 values, 32 bits each, in memory and the 2 values passed, and stores the result back into memory.

Parameters:

[in] *addr* - summarized address
[in] *value1* - value1 to add to variable 1
[in] *value2* - value2 to add to variable 2
[out] *orig_value* - pointer in CMEM to write both original values to
[out] *overflow_flag* - will be true if an overflow occurred in at least one counter An overflow on signed add is defined as changing originally positive memory contents to be negative when adding a positive number, or changing originally negative memory contents to be positive when adding a negative number

Returns:

Original value of variable 1

static __always_inline void ezdp_atomic_dual_add64_sum_addr (ezdp_sum_addr_t *addr*, int16_t *value1*, int16_t *value2*) [static]

Atomically perform dual ADD operation to the two 64-bit variables pointed to by the summarized address.

Performs dual ADD operation between the existing 2 values, 64 bit each, in memory and the 2 values passed, and stores the result back into memory.

Parameters:

[in] *addr* - summarized address
[in] *value1* - value1 to add to variable 1
[in] *value2* - value2 to add to variable 1

Returns:

void

static __always_inline void ezdp_atomic_dual_add64_sum_addr_async (ezdp_sum_addr_t *addr*, int16_t *value1*, int16_t *value2*) [static]

Non-blocking/posted version of ezdp_atomic_dual_add64_sum_addr().

Parameters:

[in] *addr* - summarized address
[in] *value1* - value1 to add to variable 1
[in] *value2* - value2 to add to variable 2

Note:

Call ezdp_sync() to wait for the operation to complete, confirming that the data was copied to the destination.

Returns:

none

static __always_inline int32_t ezdp_atomic_read_and_dual_add64_sum_addr (ezdp_sum_addr_t *addr*, int16_t *value1*, int16_t *value2*, struct ezdp_dual_add64_result __cmem * *orig_value*, bool * *overflow_flag*) `[static]`

Atomically read and perform a dual ADD operation to the two 64-bit variables pointed to by the summarized address.

Performs a dual ADD operation between the existing 2 values, 64 bits each, in memory and the 2 values passed, and stores the result back into memory.

Parameters:

[in] *addr* - summarized address
[in] *value1* - value1 to add to variable 1
[in] *value2* - value2 to add to variable 1
[out] *orig_value* - pointer in CMEM to write both original values to
[out] *overflow_flag* - will be true if an overflow occurred in at least one counter An overflow on signed add is defined as changing originally positive memory contents to be negative when adding a positive number, or changing originally negative memory contents to be positive when adding a negative number

Returns:

32 MSB original/read value of variable 2

static __always_inline uint32_t ezdp_atomic_read_and_inc8_ext_addr (struct ezdp_ext_addr * *addr*, bool * *zero_flag*) `[static]`

Atomically read and increment an 8 bit value in an extended address.

Parameters:

[in] *addr* - pointer to extended address
[out] *zero_flag* - will be true if the memory content was rolled back to zero

Returns:

Original/Read value. The value is limited to 8 bits

static __always_inline uint32_t ezdp_atomic_read_and_inc16_ext_addr (struct ezdp_ext_addr * *addr*, bool * *zero_flag*) `[static]`

Atomically read and increment a 16 bit value in an extended address.

Parameters:

[in] *addr* - pointer to extended address
[out] *zero_flag* - will be true if the memory content was rolled back to zero

Note:
Address must be 2-byte aligned.

Returns:
Original/Read value. The value is limited to 16 bits

static __always_inline uint32_t ezdp_atomic_read_and_inc32_ext_addr (struct ezdp_ext_addr * *addr*, bool * *zero_flag*) [static]

Atomically read and increment a 32 bit value in an extended address.

Parameters:
[in] *addr* - pointer to extended address
[out] *zero_flag* - will be true if the memory content was rolled back to zero

Note:
Address must be 4-byte aligned.

Returns:
Original/Read value.

static __always_inline uint32_t ezdp_atomic_read_and_inc32_sum_addr (ezdp_sum_addr_t *addr*, bool * *zero_flag*) [static]

Atomically read and increment a 32 bit value in a summarized address.

Parameters:
[in] *addr* - summarized address
[out] *zero_flag* - will be true if the memory content was rolled back to zero

Returns:
Original/Read value.

static __always_inline uint32_t ezdp_atomic_read_and_inc64_sum_addr (ezdp_sum_addr_t *addr*, uint64_t __cmem * *orig_value*, bool * *zero_flag*) [static]

Atomically read and increment a 64 bit value in a summarized address.

Parameters:
[in] *addr* - summarized address
[out] *orig_value* - the address in CMEM to write the original value in
[out] *zero_flag* - will be true if the memory content was rolled back to zero

Returns:
The 32 MSB of the original/read value

static __always_inline uint32_t ezdp_atomic_read_and_dec8_ext_addr (struct ezdp_ext_addr * *addr*, bool * *unaffected_flag*) [static]

Atomically read and decrement conditionally by one an 8-bit value in an extended address (zero value does not underflow).

Parameters:
[in] *addr* - pointer to extended address

[out] *unaffected_flag* - will be true if the memory content was not modified (original selected data value was zero)

Returns:

Original/Read value. The value is limited to 8 bits

static __always_inline uint32_t ezdp_atomic_read_and_dec16_ext_addr (struct ezdp_ext_addr * *addr*, bool * *unaffected_flag*) `[static]`

Atomically read and decrement conditionally by one a 16-bit value in an extended address (zero value does not underflow).

Parameters:

[in] *addr* - pointer to extended address
[out] *unaffected_flag* - will be true if the memory content was not modified (original selected data value was zero)

Note:

Address must be 2-byte aligned.

Returns:

Original/Read value. The value is limited to 16 bits

static __always_inline uint32_t ezdp_atomic_read_and_dec32_ext_addr (struct ezdp_ext_addr * *addr*, bool * *unaffected_flag*) `[static]`

Atomically read and decrement conditionally by one a 32-bit value in an extended address (zero value does not underflow).

Parameters:

[in] *addr* - pointer to extended address
[out] *unaffected_flag* - will be true if the memory content was not modified (original selected data value was zero)

Note:

Address must be 4-byte aligned.

Returns:

Original/Read value.

static __always_inline uint32_t ezdp_atomic_read_and_dec32_sum_addr (ezdp_sum_addr_t *addr*, bool * *unaffected_flag*) `[static]`

Atomically read and decrement conditionally by one a 32-bit value in a summarized address (zero value does not underflow).

Parameters:

[in] *addr* - summarized address
[out] *unaffected_flag* - will be true if the memory content was not modified (original selected data value was zero)

Returns:

Original/Read value.

static __always_inline uint32_t ezdp_atomic_read_and_dec64_sum_addr (ezdp_sum_addr_t *addr*, uint64_t __cmem * *orig_value*, bool * *unaffected_flag*) `[static]`

Atomically read and decrement conditionally by one a 64-bit value in a summarized address (zero value does not underflow).

Parameters:

[in] *addr* - summarized address
[out] *orig_value* - the address in CMEM to write the original value in
[out] *unaffected_flag* - will be true if the memory content was not modified (original selected data value was zero)

Returns:

The 32 MSB of the original/read value

static __always_inline uint32_t ezdp_atomic_read_and_inc32_cond_ext_addr (struct ezdp_ext_addr * *addr*, bool *res_mode*, bool * *unaffected_flag*) `[static]`

Atomically read and increment conditionally a 32-bit value in an extended address.

Parameters:

[in] *addr* - pointer to extended address
[in] *res_mode* - the reservation mode
[out] *unaffected_flag* - will be true if the memory content was not modified

Note:

Address must be 4-byte aligned.

Returns:

The original accessed counter value

static __always_inline uint32_t ezdp_atomic_read_and_inc32_cond_sum_addr (ezdp_sum_addr_t *addr*, bool *res_mode*, bool * *unaffected_flag*) `[static]`

Atomically read and increment conditionally a 32-bit value in a summarized address.

Parameters:

[in] *addr* - summarized address
[in] *res_mode* - the reservation mode
[out] *unaffected_flag* - will be true if the memory content was not modified

Returns:

The original accessed counter value

static __always_inline void ezdp_atomic_and8_ext_addr (struct ezdp_ext_addr * *addr*, uint8_t *value*) `[static]`

Atomically perform an 8 bit logical AND operation on an extended address.

Performs a logical AND operation between the existing value in memory and the value passed, and stores the result back into memory.

Parameters:

[in] *addr* - pointer to extended address
[in] *value* - update value

Returns:
void

static __always_inline void ezdp_atomic_and8_ext_addr_async (struct ezdp_ext_addr * *addr*, uint8_t *value*) [static]

Non-blocking/posted version of ezdp_atomic_and8_ext_addr().

Parameters:
[in] *addr* - pointer to extended address
[in] *value* - update value

Note:
Call ezdp_sync() to wait for the operation to complete, confirming that the data was copied to the destination.

Returns:
void

static __always_inline uint32_t ezdp_atomic_read_and_and8_ext_addr (struct ezdp_ext_addr * *addr*, uint8_t *value*, bool * *no_chng_flag*) [static]

Atomically read and perform an 8 bit logical AND operation on an extended address.

Performs a logical AND operation between the existing value in memory and the value passed, and stores the result back into memory.

Parameters:
[in] *addr* - pointer to extended address
[in] *value* - update value
[out] *no_chng_flag* - True when AND operation resulted in no change to original value

Returns:
Original/Read value. The value is limited to 8 bits

static __always_inline void ezdp_atomic_and16_ext_addr (struct ezdp_ext_addr * *addr*, uint16_t *value*) [static]

Atomically perform a 16 bit logical AND operation on an extended address.

Performs a logical AND operation between the existing value in memory and the value passed, and stores the result back into memory.

Parameters:
[in] *addr* - pointer to extended address
[in] *value* - update value

Note:
Address must be 2-byte aligned.

Returns:
void

static __always_inline void ezdp_atomic_and16_ext_addr_async (struct ezdp_ext_addr * *addr*, uint16_t *value*) [static]

Non-blocking/posted version of ezdp_atomic_and16_ext_addr().

Parameters:

[in] *addr* - pointer to extended address
[in] *value* - value to and with

Note:

Address must be 2-byte aligned. Call ezdp_sync() to wait for the operation to complete, confirming that the data was copied to the destination.

Returns:

void

static __always_inline uint32_t ezdp_atomic_read_and_and16_ext_addr (struct ezdp_ext_addr * *addr*, uint16_t *value*, bool * *no_chng_flag*) [static]

Atomically read and perform a 16 bit logical AND operation on an extended address.

Performs a logical AND operation between the existing value in memory and the value passed, and stores the result back into memory.

Parameters:

[in] *addr* - pointer to extended address
[in] *value* - update value
[out] *no_chng_flag* - True when AND operation resulted in no change to original value

Note:

Address must be 2-byte aligned.

Returns:

Original/Read value. The value is limited to 16 bits

static __always_inline void ezdp_atomic_and32_ext_addr (struct ezdp_ext_addr * *addr*, uint32_t *value*) [static]

Atomically perform a 32 bit logical AND operation on an extended address.

Performs a logical AND operation between the existing value in memory and the value passed, and stores the result back into memory.

Parameters:

[in] *addr* - pointer to extended address
[in] *value* - update value

Note:

Address must be 4-byte aligned.

Returns:

void

static __always_inline void ezdp_atomic_and32_ext_addr_async (struct ezdp_ext_addr * *addr*, uint32_t *value*) [static]

Non-blocking/posted version of ezdp_atomic_and32_ext_addr_async().

Parameters:

[in] *addr* - pointer to extended address
[in] *value* - update value

Note:

Address must be 4-byte aligned. Call ezdp_sync() to wait for the operation to complete, confirming that the data was copied to the destination.

Returns:
void

static __always_inline uint32_t ezdp_atomic_read_and_and32_ext_addr (struct ezdp_ext_addr * *addr*, uint32_t *value*, bool * *no_chng_flag*) [static]

Atomically read and perform a 32 bit logical AND operation on an extended address.

Performs a logical AND operation between the existing value in memory and the value passed, and stores the result back into memory.

Parameters:
[in] *addr* - pointer to extended address
[in] *value* - update value
[out] *no_chng_flag* - True when AND operation resulted in no change to original value

Note:
Address must be 4-byte aligned.

Returns:
Original/Read value

static __always_inline void ezdp_atomic_and32_sum_addr (ezdp_sum_addr_t *addr*, uint32_t *value*) [static]

Atomically perform a 32 bit logical AND operation on a summarized address.

Performs a logical AND operation between the existing value in memory and the value passed, and stores the result back into memory.

Parameters:
[in] *addr* - summarized address
[in] *value* - update value

Returns:
void

static __always_inline void ezdp_atomic_and32_sum_addr_async (ezdp_sum_addr_t *addr*, uint32_t *value*) [static]

Non-blocking/posted version of ezdp_atomic_and32_sum_addr_async().

Parameters:
[in] *addr* - summarized address
[in] *value* - update value

Note:
Address must be 4-byte aligned. Call ezdp_sync() to wait for the operation to complete, confirming that the data was copied to the destination.

Returns:
void

static __always_inline uint32_t ezdp_atomic_read_and_and32_sum_addr (ezdp_sum_addr_t *addr*, uint32_t *value*, bool * *no_chng_flag*) [static]

Atomically read and perform a 32 bit logical AND operation on a summarized address.

Performs a logical AND operation between the existing value in memory and the value passed, and stores the result back into memory.

Parameters:

[in] *addr* - summarized address
[in] *value* - update value
[out] *no_chng_flag* - True when AND operation resulted in no change to original value

Returns:

Original/Read value

static __always_inline void ezdp_atomic_or8_ext_addr (struct ezdp_ext_addr * *addr*, uint8_t *value*) [static]

Atomically perform an 8 bit logical OR operation on an extended address.

Performs a logical OR operation between the existing value in memory and the value passed, and stores the result back into memory.

Parameters:

[in] *addr* - pointer to extended address
[in] *value* - update value to OR with

Returns:

void

static __always_inline void ezdp_atomic_or8_ext_addr_async (struct ezdp_ext_addr * *addr*, uint8_t *value*) [static]

Non-blocking/posted version of ezdp_atomic_or8_ext_addr().

Parameters:

[in] *addr* - pointer to extended address
[in] *value* - update value to OR with

Note:

Call ezdp_sync() to wait for the operation to complete, confirming that the data was copied to the destination.

Returns:

void

static __always_inline uint32_t ezdp_atomic_read_and_or8_ext_addr (struct ezdp_ext_addr * *addr*, uint8_t *value*, bool * *no_chng_flag*) [static]

Atomically read and perform an 8 bit logical OR operation on an extended address.

Performs a logical OR operation between the existing value in memory and the value passed, and stores the result back into memory.

Parameters:

[in] *addr* - pointer to extended address
[in] *value* - update value to OR with
[out] *no_chng_flag* - True when OR operation resulted in no change to original value

Returns:

Original/Read value. The value is limited to 8 bits

static __always_inline void ezdp_atomic_or8_sum_addr (ezdp_sum_addr_t *addr*, uint8_t *value*) [static]

Atomically perform an 8 bit logical OR operation on a summarized address.

Performs a logical OR operation between the existing value in memory and the value passed, and stores the result back into memory.

Parameters:

[in] *addr* - summarized address
[in] *value* - update value to OR with

Returns:

void

static __always_inline void ezdp_atomic_or8_sum_addr_async (ezdp_sum_addr_t *addr*, uint8_t *value*) [static]

Non-blocking/posted version of ezdp_atomic_or8_sum_addr().

Parameters:

[in] *addr* - summarized address
[in] *value* - update value to OR with

Note:

Call ezdp_sync() to wait for the operation to complete, confirming that the data was copied to the destination.

Returns:

void

static __always_inline void ezdp_atomic_or16_ext_addr (struct ezdp_ext_addr * *addr*, uint16_t *value*) [static]

Atomically perform a 16 bit logical OR operation on an extended address.

Performs a logical OR operation between the existing value in memory and the value passed, and stores the result back into memory.

Parameters:

[in] *addr* - pointer to extended address
[in] *value* - update value to OR with

Note:

Address must be 2-byte aligned.

Returns:

void

static __always_inline void ezdp_atomic_or16_ext_addr_async (struct ezdp_ext_addr * *addr*, uint16_t *value*) [static]

Non-blocking/posted version of ezdp_atomic_or16_ext_addr().

Parameters:

[in] *addr* - pointer to extended address
[in] *value* - value to OR with

Note:

Address must be 2-byte aligned. Call ezdp_sync() to wait for the operation to complete, confirming that the data was copied to the destination.

Returns:

void

static __always_inline uint32_t ezdp_atomic_read_and_or16_ext_addr (struct ezdp_ext_addr * *addr*, uint16_t *value*, bool * *no_chng_flag*) [static]

Atomically read and perform a 16 bit logical OR operation on an extended address.

Performs a logical OR operation between the existing value in memory and the value passed, and stores the result back into memory.

Parameters:

[in] *addr* - pointer to extended address
[in] *value* - update value to OR with
[out] *no_chng_flag* - True when OR operation resulted in no change to original value

Note:

Address must be 2-byte aligned.

Returns:

Original/Read value. The value is limited to 16 bits

static __always_inline void ezdp_atomic_or16_sum_addr (ezdp_sum_addr_t *addr*, uint16_t *value*) [static]

Atomically perform a 16 bit logical OR operation on a summarized address.

Performs a logical OR operation between the existing value in memory and the value passed, and stores the result back into memory.

Parameters:

[in] *addr* - summarized address
[in] *value* - update value to OR with

Returns:

void

static __always_inline void ezdp_atomic_or16_sum_addr_async (ezdp_sum_addr_t *addr*, uint16_t *value*) [static]

Non-blocking/posted version of ezdp_atomic_or16_sum_addr().

Parameters:

[in] *addr* - summarized address
[in] *value* - value to OR with

Note:

Address must be 2-byte aligned. Call ezdp_sync() to wait for the operation to complete, confirming that the data was copied to the destination.

Returns:

void

static __always_inline void ezdp_atomic_or32_ext_addr (struct ezdp_ext_addr * *addr*, uint32_t *value*) [static]

Atomically perform a 32 bit logical OR operation on an extended address.

Performs a logical OR operation between the existing value in memory and the value passed, and stores the result back into memory.

Parameters:

[in] *addr* - pointer to extended address
[in] *value* - update value to OR with

Note:

Address must be 4-byte aligned.

Returns:

void

static __always_inline void ezdp_atomic_or32_ext_addr_async (struct ezdp_ext_addr * *addr*, uint32_t *value*) `[static]`

Non-blocking/posted version of ezdp_atomic_or32_ext_addr().

Parameters:

[in] *addr* - pointer to extended address
[in] *value* - value to OR with

Note:

Address must be 4-byte aligned. Call ezdp_sync() to wait for the operation to complete, confirming that the data was copied to the destination.

Returns:

void

static __always_inline uint32_t ezdp_atomic_read_and_or32_ext_addr (struct ezdp_ext_addr * *addr*, uint32_t *value*, bool * *no_chng_flag*) `[static]`

Atomically read and perform a 32 bit logical OR operation on an extended address.

Performs a logical OR operation between the existing value in memory and the value passed, and stores the result back into memory.

Parameters:

[in] *addr* - pointer to extended address
[in] *value* - update value to OR with
[out] *no_chng_flag* - True when OR operation resulted in no change to original value

Note:

Address must be 4-byte aligned.

Returns:

Original/Read value

static __always_inline void ezdp_atomic_or32_sum_addr (ezdp_sum_addr_t *addr*, uint32_t *value*) `[static]`

Atomically perform a 32 bit logical OR operation on a summarized address.

Performs a logical OR operation between the existing value in memory and the value passed, and stores the result back into memory.

Parameters:

[in] *addr* - summarized address
[in] *value* - update value to OR with

Returns:
void

static __always_inline void ezdp_atomic_or32_sum_addr_async (ezdp_sum_addr_t *addr*, uint32_t *value*) [static]

Non-blocking/posted version of ezdp_atomic_or32_sum_addr().

Parameters:
[in] *addr* - summarized address
[in] *value* - value to OR with

Note:
Address must be 4-byte aligned. Call ezdp_sync() to wait for the operation to complete, confirming that the data was copied to the destination.

Returns:
void

static __always_inline uint32_t ezdp_atomic_read_and_or32_sum_addr (ezdp_sum_addr_t *addr*, uint32_t *value*, bool * *no_chng_flag*) [static]

Atomically read and perform a 32 bit logical OR operation on a summarized address.

Performs a logical OR operation between the existing value in memory and the value passed, and stores the result back into memory.

Parameters:
[in] *addr* - summarized address
[in] *value* - update value to OR with
[out] *no_chng_flag* - True when OR operation resulted in no change to original value

Returns:
Original/Read value

static __always_inline void ezdp_atomic_xor8_ext_addr (struct ezdp_ext_addr * *addr*, uint8_t *value*) [static]

Atomically perform an 8 bit logical XOR operation on an extended address.

Performs a logical XOR operation between the existing value in memory and the value passed, and stores the result back into memory.

Parameters:
[in] *addr* - pointer to extended address
[in] *value* - update value

Returns:
void

static __always_inline void ezdp_atomic_xor8_ext_addr_async (struct ezdp_ext_addr * *addr*, uint8_t *value*) [static]

Non-blocking/posted version of ezdp_atomic_xor8_ext_addr().

Parameters:
[in] *addr* - pointer to extended address

[in] *value* - value to XOR with

Note:

Call ezdp_sync() to wait for the operation to complete, confirming that the data was copied to the destination.

Returns:

void

static __always_inline uint32_t ezdp_atomic_read_and_xor8_ext_addr (struct ezdp_ext_addr * *addr*, uint8_t *value*, bool * *no_id_flag*) [static]

Atomically read and perform an 8 bit logical XOR operation on an extended address.

Performs a logical XOR operation between the existing value in memory and the value passed, and stores the result back into memory.

Parameters:

[in] *addr* - pointer to extended address
[in] *value* - update value
[out] *no_id_flag* - True when the value passed to function and the value in memory are not identical.

Returns:

Original/Read value. The value is limited to 8 bits

static __always_inline void ezdp_atomic_xor16_ext_addr (struct ezdp_ext_addr * *addr*, uint16_t *value*) [static]

Atomically perform a 16 bit logical XOR operation on an extended address.

Performs a logical XOR operation between the existing value in memory and the value passed, and stores the result back into memory.

Parameters:

[in] *addr* - pointer to extended address
[in] *value* - update value

Note:

Address must be 2-byte aligned.

Returns:

void

static __always_inline void ezdp_atomic_xor16_ext_addr_async (struct ezdp_ext_addr * *addr*, uint16_t *value*) [static]

Non-blocking/posted version of ezdp_atomic_xor8_ext_addr().

Parameters:

[in] *addr* - pointer to extended address
[in] *value* - update value

Note:

Address must be 2-byte aligned. Call ezdp_sync() to wait for the operation to complete, confirming that the data was copied to the destination.

Returns:

void

static __always_inline uint32_t ezdp_atomic_read_and_xor16_ext_addr (struct ezdp_ext_addr * *addr*, uint16_t *value*, bool * *no_id_flag*) `[static]`

Atomically read and perform a 16 bit logical XOR operation on an extended address.

Performs a logical XOR operation between the existing value in memory and the value passed, and stores the result back into memory.

Parameters:

[in] *addr* - pointer to extended address
[in] *value* - update value
[out] *no_id_flag* - True when the value passed to function and the value in memory are not identical.

Note:

Address must be 2-byte aligned.

Returns:

Original/Read value. The value is limited to 16 bits

static __always_inline void ezdp_atomic_xor32_ext_addr (struct ezdp_ext_addr * *addr*, uint32_t *value*) `[static]`

Atomically perform a 32 bit logical XOR operation on an extended address.

Performs a logical XOR operation between the existing value in memory and the value passed, and stores the result back into memory.

Parameters:

[in] *addr* - pointer to extended address
[in] *value* - update value

Note:

Address must be 4-byte aligned.

Returns:

void

static __always_inline void ezdp_atomic_xor32_ext_addr_async (struct ezdp_ext_addr * *addr*, uint32_t *value*) `[static]`

Non-blocking/posted version of ezdp_atomic_xor32_ext_addr().

Parameters:

[in] *addr* - pointer to extended address
[in] *value* - update value

Note:

Address must be 4-byte aligned. Call ezdp_sync() to wait for the operation to complete, confirming that the data was copied to the destination.

Returns:

void

static __always_inline uint32_t ezdp_atomic_read_and_xor32_ext_addr (struct ezdp_ext_addr * *addr*, uint32_t *value*, bool * *no_id_flag*) `[static]`

Atomically read and perform a 32 bit logical XOR operation on an extended address.

Performs a logical XOR operation between the existing value in memory and the value passed, and stores the result back into memory.

Parameters:

[in] *addr* - pointer to extended address
[in] *value* - update value
[out] *no_id_flag* - True when the value passed to function and the value in memory are not identical.

Note:

Address must be 4-byte aligned.

Returns:

Original/Read value.

static __always_inline void ezdp_atomic_xor32_sum_addr (ezdp_sum_addr_t *addr*, uint32_t *value*) [static]

Atomically perform a 32 bit logical XOR operation on a summarized address.

Performs a logical XOR operation between the existing value in memory and the value passed, and stores the result back into memory.

Parameters:

[in] *addr* - summarized address
[in] *value* - update value

Returns:

void

static __always_inline void ezdp_atomic_xor32_sum_addr_async (ezdp_sum_addr_t *addr*, uint32_t *value*) [static]

Non-blocking/posted version of ezdp_atomic_xor32_sum_addr().

Parameters:

[in] *addr* - summarized address
[in] *value* - update value

Note:

Address must be 4-byte aligned. Call ezdp_sync() to wait for the operation to complete, confirming that the data was copied to the destination.

Returns:

void

static __always_inline uint32_t ezdp_atomic_read_and_xor32_sum_addr (ezdp_sum_addr_t *addr*, uint32_t *value*, bool * *no_id_flag*) [static]

Atomically read and perform a 32 bit logical XOR operation on a summarized address.

Performs a logical XOR operation between the existing value in memory and the value passed, and stores the result back into memory.

Parameters:

[in] *addr* - summarized address
[in] *value* - update value
[out] *no_id_flag* - True when the value passed to function and the value in memory are not identical.

Returns:

Original/Read value.

dpe/dp/include/ezdp_counter.h File Reference

Functions

- static __always_inline void ezdp_write_single_ctr_cfg (ezdp_sum_addr_t addr, struct ezdp_single_ctr_cfg *counter)
- *Configure single counter and its initial value.* static __always_inline void ezdp_write_single_ctr_cfg_async (ezdp_sum_addr_t addr, struct ezdp_single_ctr_cfg *counter)
- *Non blocking version of ezdp_write_single_ctr_cfg.* static __always_inline void ezdp_read_single_ctr_cfg (ezdp_sum_addr_t addr, struct ezdp_single_ctr_cfg __cmem *counter)
- *Read single counter configuration.* static __always_inline void ezdp_read_single_ctr_cfg_async (ezdp_sum_addr_t addr, struct ezdp_single_ctr_cfg __cmem *counter)
- *Non blocking version of ezdp_read_single_ctr_cfg.* static __always_inline void ezdp_write_single_ctr (ezdp_sum_addr_t addr, uint64_t value)
- *Initialize single counter with the value specified.* static __always_inline void ezdp_write_single_ctr_async (ezdp_sum_addr_t addr, uint64_t value)
- *Non blocking version of ezdp_write_single_ctr.* static __always_inline uint32_t ezdp_xchg_single_ctr (ezdp_sum_addr_t addr, uint64_t value, uint64_t __cmem *orig_value)
- *Write single counter with the value specified and read previous counter value.* static __always_inline uint32_t ezdp_read_single_ctr (ezdp_sum_addr_t addr, uint64_t __cmem *value, bool *overflow)
- *Read single counter value.* static __always_inline void ezdp_inc_single_ctr (ezdp_sum_addr_t addr, uint16_t increment_value)
- *Increment single counter by the value specified.* static __always_inline void ezdp_inc_single_ctr_async (ezdp_sum_addr_t addr, uint16_t increment_value)
- *Non blocking version of ezdp_inc_single_ctr.* static __always_inline uint32_t ezdp_read_and_inc_single_ctr (ezdp_sum_addr_t addr, uint16_t increment_value, uint64_t __cmem *orig_value, bool *overflow)
- *Increment single counter by the value specified and read previous counter value.* static __always_inline void ezdp_dec_single_ctr (ezdp_sum_addr_t addr, uint16_t decrement_value)
- *Decrement single counter by the value specified.* static __always_inline void ezdp_dec_single_ctr_async (ezdp_sum_addr_t addr, uint16_t decrement_value)
- *Non blocking version of ezdp_dec_single_ctr.* static __always_inline uint32_t ezdp_read_and_dec_single_ctr (ezdp_sum_addr_t addr, uint16_t decrement_value, uint64_t __cmem *orig_value, bool *overflow)
- *Decrement single counter by the value specified and read previous counter value.* static __always_inline void ezdp_reset_single_ctr (ezdp_sum_addr_t addr)
- *Reset single counter to zero.* static __always_inline void ezdp_reset_single_ctr_async (ezdp_sum_addr_t addr)
- *Non blocking version of ezdp_reset_single_ctr.* static __always_inline uint32_t ezdp_read_and_reset_single_ctr (ezdp_sum_addr_t addr, uint64_t __cmem *orig_value)
- *Reset single counter to zero and read previous counter value.* static __always_inline void ezdp_cond_dec_single_ctr (ezdp_sum_addr_t addr, uint16_t decrement_value)
- *Conditionally decrement single counter by the value specified.* static __always_inline void ezdp_cond_dec_single_ctr_async (ezdp_sum_addr_t addr, uint16_t decrement_value)
- *Non blocking version of ezdp_cond_dec_single_ctr.* static __always_inline uint32_t ezdp_read_and_cond_dec_single_ctr (ezdp_sum_addr_t addr, uint16_t decrement_value, uint64_t __cmem *orig_value, bool *failure_ind)
- *Conditionally decrement single counter by the value specified and read previous counter value.* static __always_inline void ezdp_prefetch_single_ctr (ezdp_sum_addr_t addr)
- *Prefetch single counter into the local cache.* static __always_inline void ezdp_prefetch_single_ctr_async (ezdp_sum_addr_t addr)
- *Non blocking version of ezdp_prefetch_single_ctr.* static __always_inline void ezdp_write_dual_ctr_cfg (ezdp_sum_addr_t addr, struct ezdp_dual_ctr_cfg *counter)
- *Configure dual counter and its initial values (byte and event).* static __always_inline void ezdp_write_dual_ctr_cfg_async (ezdp_sum_addr_t addr, struct ezdp_dual_ctr_cfg *counter)
- *Non blocking version of ezdp_write_dual_ctr_cfg.* static __always_inline void ezdp_read_dual_ctr_cfg (ezdp_sum_addr_t addr, struct ezdp_dual_ctr_cfg __cmem *counter)
- *Read dual counter configuration.* static __always_inline uint32_t ezdp_read_dual_ctr (ezdp_sum_addr_t addr, struct ezdp_dual_ctr_result __cmem *value, bool *overflow)
- *Read dual counter values (byte and event).* static __always_inline void ezdp_inc_dual_ctr (ezdp_sum_addr_t addr, uint16_t inc_byte_value, uint16_t inc_event_value)

- *Increment dual counter with the values specified (byte and event).* static __always_inline void ezdp_inc_dual_ctr_async (ezdp_sum_addr_t addr, uint16_t inc_byte_value, uint16_t inc_event_value)
- *Non blocking version of ezdp_inc_dual_ctr.* static __always_inline uint32_t ezdp_read_and_inc_dual_ctr (ezdp_sum_addr_t addr, uint16_t inc_byte_value, uint16_t inc_event_value, struct ezdp_dual_ctr_result __cmem *orig_value, bool *overflow)
- *Increment dual counter with the values specified (byte and event) and read previous counter values.* static __always_inline void ezdp_dec_dual_ctr (ezdp_sum_addr_t addr, uint16_t dec_byte_value)
- *Decrement dual counter's byte value by the value specified and event value by 1.* static __always_inline void ezdp_dec_dual_ctr_async (ezdp_sum_addr_t addr, uint16_t dec_byte_value)
- *Non blocking version of ezdp_dec_dual_ctr.* static __always_inline uint32_t ezdp_read_and_dec_dual_ctr (ezdp_sum_addr_t addr, uint16_t dec_byte_value, struct ezdp_dual_ctr_result __cmem *orig_value, bool *overflow)
- *Decrement dual counter's byte value by the value specified and event value by 1, and read previous counter values.* static __always_inline void ezdp_reset_dual_ctr (ezdp_sum_addr_t addr)
- *Reset dual counter values (byte and event) to zero.* static __always_inline void ezdp_reset_dual_ctr_async (ezdp_sum_addr_t addr)
- *Non blocking version of ezdp_reset_dual_ctr.* static __always_inline uint32_t ezdp_read_and_reset_dual_ctr (ezdp_sum_addr_t addr, struct ezdp_dual_ctr_result __cmem *orig_value)
- *Reset dual counter values (byte and event) to zero and read previous counter values.* static __always_inline void ezdp_prefetch_dual_ctr (ezdp_sum_addr_t addr)
- *Prefetch dual counter into the local cache.* static __always_inline void ezdp_prefetch_dual_ctr_async (ezdp_sum_addr_t addr)
- *Non blocking version of ezdp_prefetch_dual_ctr.* static __always_inline void ezdp_write_bitwise_ctr_cfg (ezdp_sum_addr_t addr, uint64_t value)
- *Configure bitwise counter and its initial value.* static __always_inline void ezdp_write_bitwise_ctr_cfg_async (ezdp_sum_addr_t addr, uint64_t value)
- *Non blocking version of ezdp_write_bitwise_ctr_cfg.* static __always_inline void ezdp_read_bitwise_ctr_cfg (ezdp_sum_addr_t addr, struct ezdp_bitwise_ctr_cfg __cmem *counter)
- *Read bitwise counter configuration.* static __always_inline void ezdp_read_bitwise_ctr_cfg_async (ezdp_sum_addr_t addr, struct ezdp_bitwise_ctr_cfg __cmem *counter)
- *Non blocking version of ezdp_read_bitwise_ctr_cfg.* static __always_inline void ezdp_write_bits_bitwise_ctr (ezdp_sum_addr_t addr, uint16_t value, enum ezdp_bitwise_size size, uint8_t offset)
- *Write the value to the selected bits in the bitwise counter.* static __always_inline void ezdp_write_bits_bitwise_ctr_async (ezdp_sum_addr_t addr, uint16_t value, enum ezdp_bitwise_size size, uint8_t offset)
- *Non blocking version of ezdp_write_bits_bitwise_ctr.* static __always_inline void ezdp_xchg_bits_bitwise_ctr (ezdp_sum_addr_t addr, uint16_t value, enum ezdp_bitwise_size size, uint8_t offset, uint64_t __cmem *orig_value)
- *Write the value to the selected bits in the bitwise counter and read previous counter value.* static __always_inline uint32_t ezdp_read_bitwise_ctr (ezdp_sum_addr_t addr, uint64_t __cmem *value)
- *Read bitwise counter value.* static __always_inline void ezdp_read_bits_bitwise_ctr (ezdp_sum_addr_t addr, enum ezdp_bitwise_size size, uint8_t offset, uint64_t __cmem *value)
- *Read the selected bits from the bitwise counter.* static __always_inline void ezdp_inc_bits_bitwise_ctr (ezdp_sum_addr_t addr, uint16_t increment_value, enum ezdp_bitwise_size size, uint8_t offset)
- *Increment the selected bits in the bitwise counter by the value specified.* static __always_inline void ezdp_inc_bits_bitwise_ctr_async (ezdp_sum_addr_t addr, uint16_t increment_value, enum ezdp_bitwise_size size, uint8_t offset)
- *Non blocking version of ezdp_inc_bits_bitwise_ctr.* static __always_inline void ezdp_read_and_inc_bits_bitwise_ctr (ezdp_sum_addr_t addr, uint16_t increment_value, enum ezdp_bitwise_size size, uint8_t offset, uint64_t __cmem *orig_value)
- *Increment the selected bits in the bitwise counter by the value specified and read previous counter value.* static __always_inline void ezdp_dec_bits_bitwise_ctr (ezdp_sum_addr_t addr, uint16_t decrement_value, enum ezdp_bitwise_size size, uint8_t offset)
- *Decrement the selected bits in the bitwise counter by the value specified.* static __always_inline void ezdp_dec_bits_bitwise_ctr_async (ezdp_sum_addr_t addr, uint16_t decrement_value, enum ezdp_bitwise_size size, uint8_t offset)
- *Non blocking version of ezdp_dec_bits_bitwise_ctr.* static __always_inline void ezdp_read_and_dec_bits_bitwise_ctr (ezdp_sum_addr_t addr, uint16_t decrement_value, enum ezdp_bitwise_size size, uint8_t offset, uint64_t __cmem *orig_value)

- *Decrement the selected bits in the bitwise counter by the value specified and read previous counter value.* static __always_inline void ezdp_reset_bitwise_ctr (ezdp_sum_addr_t addr)
- *Reset bitwise counter value to zero.* static __always_inline void ezdp_reset_bitwise_ctr_async (ezdp_sum_addr_t addr)
- *Non blocking version of ezdp_reset_bitwise_ctr.* static __always_inline void ezdp_read_and_reset_bitwise_ctr (ezdp_sum_addr_t addr, uint64_t __cmem *orig_value)
- *Reset bitwise counter value to zero and read previous counter value.* static __always_inline void ezdp_set_bits_bitwise_ctr (ezdp_sum_addr_t addr, uint16_t value, enum ezdp_bitwise_size size, uint8_t offset)
- *Set the selected bits in the bitwise counter according to the value specified.* static __always_inline void ezdp_set_bits_bitwise_ctr_async (ezdp_sum_addr_t addr, uint16_t value, enum ezdp_bitwise_size size, uint8_t offset)
- *Non blocking version of ezdp_set_bits_bitwise_ctr.* static __always_inline void ezdp_read_and_set_bits_bitwise_ctr (ezdp_sum_addr_t addr, uint16_t value, enum ezdp_bitwise_size size, uint8_t offset, uint64_t __cmem *orig_value)
- *Set the selected bits in the bitwise counter according to the value specified and read previous counter value.* static __always_inline void ezdp_clear_bits_bitwise_ctr (ezdp_sum_addr_t addr, uint16_t value, enum ezdp_bitwise_size size, uint8_t offset)
- *Clear the selected bits in the bitwise counter according to the value specified.* static __always_inline void ezdp_clear_bits_bitwise_ctr_async (ezdp_sum_addr_t addr, uint16_t value, enum ezdp_bitwise_size size, uint8_t offset)
- *Non blocking version of ezdp_clear_bits_bitwise_ctr.* static __always_inline void ezdp_read_and_clear_bits_bitwise_ctr (ezdp_sum_addr_t addr, uint16_t value, enum ezdp_bitwise_size size, uint8_t offset, uint64_t __cmem *orig_value)
- *Clear the selected bits in the bitwise counter according to the value specified and read previous counter value.* static __always_inline void ezdp_read_and_cond_write_bits_bitwise_ctr (ezdp_sum_addr_t addr, uint8_t value, enum ezdp_bitwise_size size, uint8_t offset, uint8_t cmp_value, uint64_t __cmem *orig_value, bool *success_ind)
- *Read, compare and conditionally set the specified bits in the bitwise counter with the value specified.* static __always_inline void ezdp_prefetch_bitwise_ctr (ezdp_sum_addr_t addr)
- *Prefetch bitwise counter into the local cache.* static __always_inline void ezdp_prefetch_bitwise_ctr_async (ezdp_sum_addr_t addr)
- *Non blocking version of ezdp_prefetch_bitwise_ctr.* static __always_inline void ezdp_write_tb_ctr_cfg (ezdp_sum_addr_t addr, struct ezdp_tb_ctr_cfg *counter)
- *Configure token bucket counter.* static __always_inline void ezdp_write_tb_ctr_cfg_async (ezdp_sum_addr_t addr, struct ezdp_tb_ctr_cfg *counter)
- *Non blocking version of ezdp_write_tb_ctr_cfg.* static __always_inline void ezdp_read_tb_ctr_cfg (ezdp_sum_addr_t addr, struct ezdp_tb_ctr_cfg __cmem *counter)
- *Read token bucket counter configuration.* static __always_inline void ezdp_update_tb_ctr (ezdp_sum_addr_t addr, uint16_t value, enum ezdp_tb_color pre_color)
- *Update a token bucket counter with the specified value (e.g.* static __always_inline void ezdp_update_tb_ctr_async (ezdp_sum_addr_t addr, uint16_t value, enum ezdp_tb_color pre_color)
- *Non blocking version of ezdp_update_tb_ctr.* static __always_inline void ezdp_read_tb_ctr (ezdp_sum_addr_t addr, uint16_t value, enum ezdp_tb_color pre_color, struct ezdp_tb_ctr_result __cmem *result_color)
- *Get the resulting color after updating a token bucket counter with the specified value (e.g.* static __always_inline void ezdp_read_tb_ctr_async (ezdp_sum_addr_t addr, uint16_t value, enum ezdp_tb_color pre_color, struct ezdp_tb_ctr_result __cmem *result_color)
- *Non blocking version of ezdp_read_tb_color.* static __always_inline void ezdp_check_tb_ctr (ezdp_sum_addr_t addr, uint16_t value, enum ezdp_tb_color pre_color, struct ezdp_tb_ctr_result __cmem *result_value)
- *Get the resulting color after updating a token bucket with the specified value (e.g.* static __always_inline void ezdp_check_tb_ctr_async (ezdp_sum_addr_t addr, uint16_t value, enum ezdp_tb_color pre_color, struct ezdp_tb_ctr_result __cmem *result_value)
- *Non blocking version of ezdp_check_tb_ctr.* static __always_inline void ezdp_inc_tb_ctr (ezdp_sum_addr_t addr, uint16_t value, enum ezdp_tb_color pre_color, bool inc_commit_bucket, bool inc_excess_bucket)
- *Force increment token buckets with the specified value (e.g.* static __always_inline void ezdp_inc_tb_ctr_async (ezdp_sum_addr_t addr, uint16_t value, enum ezdp_tb_color pre_color, bool inc_commit_bucket, bool inc_excess_bucket)
- *Non blocking version of ezdp_inc_tb_ctr.* static __always_inline void ezdp_read_and_inc_tb_ctr (ezdp_sum_addr_t addr, uint16_t value, enum ezdp_tb_color pre_color, bool inc_commit_bucket, bool inc_excess_bucket, struct ezdp_tb_ctr_result __cmem *result_value)

- *Force increment token buckets with the specified value (e.g.* static __always_inline void ezdp_dec_tb_ctr (ezdp_sum_addr_t addr, uint16_t value, enum ezdp_tb_color pre_color, bool dec_commit_bucket, bool dec_excess_bucket)
- *Force decrement token buckets with the specified value (e.g.* static __always_inline void ezdp_dec_tb_ctr_async (ezdp_sum_addr_t addr, uint16_t value, enum ezdp_tb_color pre_color, bool dec_commit_bucket, bool dec_excess_bucket)
- *Non blocking version of ezdp_dec_tb_ctr.* static __always_inline void ezdp_read_and_dec_tb_ctr (ezdp_sum_addr_t addr, uint16_t value, enum ezdp_tb_color pre_color, bool dec_commit_bucket, bool dec_excess_bucket, struct ezdp_tb_ctr_result __cmem *result_value)
- *Force decrement token buckets with the specified value (e.g.* static __always_inline void ezdp_prefetch_tb_ctr (ezdp_sum_addr_t addr)
- *Prefetch token bucket counter into the local cache.* static __always_inline void ezdp_prefetch_tb_ctr_async (ezdp_sum_addr_t addr)
- *Non blocking version of ezdp_prefetch_tb_ctr.* static __always_inline void ezdp_write_hier_tb_ctr_cfg (ezdp_sum_addr_t addr, struct ezdp_hier_tb_ctr_cfg *counter)
- *Configure hierarchical token bucket counter.* static __always_inline void ezdp_write_hier_tb_ctr_cfg_async (ezdp_sum_addr_t addr, struct ezdp_hier_tb_ctr_cfg *counter)
- *Non blocking version of ezdp_write_hier_tb_ctr_cfg.* static __always_inline void ezdp_read_hier_tb_ctr_cfg (ezdp_sum_addr_t addr, struct ezdp_hier_tb_ctr_cfg __cmem *counter)
- *Read hierarchical token bucket counter configuration.* static __always_inline void ezdp_inc_hier_tb_ctr (ezdp_sum_addr_t addr, uint16_t value, bool update_ctr0, bool update_acc1)
- *Increment hierarchical token bucket counter accumulator(s) by the value specified.* static __always_inline void ezdp_inc_hier_tb_ctr_async (ezdp_sum_addr_t addr, uint16_t value, bool update_ctr0, bool update_acc1)
- *Non blocking version of ezdp_inc_hier_tb_ctr.* static __always_inline void ezdp_read_and_inc_hier_tb_ctr (ezdp_sum_addr_t addr, uint16_t value, bool update_ctr0, bool update_acc1, struct ezdp_hier_tb_result __cmem *result)
- *Increment hierarchical token bucket counter accumulator(s) by the value specified and read previous counter value.* static __always_inline void ezdp_update_hier_tb_ctr (ezdp_sum_addr_t addr, struct ezdp_hier_tb_update *ctr_update)
- *Update hierarchical token bucket counter state, app bits or clear accumulators.* static __always_inline void ezdp_update_hier_tb_ctr_async (ezdp_sum_addr_t addr, struct ezdp_hier_tb_update *ctr_update)
- *Non blocking version of ezdp_inc_hier_tb_ctr.* static __always_inline void ezdp_read_and_update_hier_tb_ctr (ezdp_sum_addr_t addr, struct ezdp_hier_tb_update *ctr_update, struct ezdp_hier_tb_result __cmem *result)
- *Update hierarchical token bucket counter state, app bits or clear accumulators and read previous counter value.* static __always_inline void ezdp_change_state_hier_tb_ctr (ezdp_sum_addr_t addr, struct ezdp_hier_tb_result __cmem *result)
- *Change hierarchical token bucket state to Ph1.* static __always_inline void ezdp_write_watchdog_ctr_cfg (ezdp_sum_addr_t addr, struct ezdp_watchdog_ctr_cfg *counter)
- *Configure watchdog counter.* static __always_inline void ezdp_write_watchdog_ctr_cfg_async (ezdp_sum_addr_t addr, struct ezdp_watchdog_ctr_cfg *counter)
- *Non blocking version of ezdp_write_watchdog_ctr_cfg.* static __always_inline void ezdp_read_watchdog_ctr_cfg (ezdp_sum_addr_t addr, struct ezdp_watchdog_ctr_cfg __cmem *counter)
- *Read watchdog counter configuration.* static __always_inline void ezdp_start_watchdog_ctr (ezdp_sum_addr_t counter_address)
- *Start the watchdog counter.* static __always_inline void ezdp_start_watchdog_ctr_async (ezdp_sum_addr_t counter_address)
- *Non blocking version of ezdp_start_watchdog_ctr.* static __always_inline void ezdp_inc_watchdog_ctr (ezdp_sum_addr_t addr)
- *Increment the watchdog counter events by one.* static __always_inline void ezdp_inc_watchdog_ctr_async (ezdp_sum_addr_t addr)
- *Non blocking version of ezdp_inc_watchdog_ctr.* static __always_inline void ezdp_check_watchdog_ctr (ezdp_sum_addr_t counter_address, struct ezdp_watchdog_ctr_check_result __cmem *check_result)
- *Check the number of events in the watchdog counter.* static __always_inline void ezdp_check_watchdog_ctr_async (ezdp_sum_addr_t counter_address, struct ezdp_watchdog_ctr_check_result __cmem *check_result)
- *Non blocking version of ezdp_check_watchdog_ctr.* static __always_inline void ezdp_prefetch_watchdog_ctr (ezdp_sum_addr_t addr)
- *Prefetch watchdog_counter into the local cache.* static __always_inline void ezdp_prefetch_watchdog_ctr_async (ezdp_sum_addr_t addr)

- *Non blocking version of ezdp_prefetch_watchdog_ctr.* static __always_inline bool ezdp_init_ctr_msg_queue_desc (uint32_t partition_id, uint32_t queue_mask, ezdp_ctr_msg_queue_desc_t *ctr_queue_desc, char __cmem *work_area_ptr, uint32_t work_area_size)
- *Initialize counter message queue descriptor.* static __always_inline bool ezdp_read_ctr_msg (ezdp_ctr_msg_queue_desc_t *ctr_queue_desc, struct ezdp_ctr_msg *msg, char __cmem *work_area_ptr, uint32_t work_area_size)
- *Read counter message from message queue.* static __always_inline void ezdp_write_posted_ctr (ezdp_sum_addr_t addr, uint64_t value)
- *Initialize posted counter with the value specified.* static __always_inline void ezdp_write_posted_ctr_async (ezdp_sum_addr_t addr, uint64_t value)
- *Non blocking version of ezdp_write_posted_ctr.* static __always_inline void ezdp_dual_write_posted_ctr (ezdp_sum_addr_t addr, uint64_t counter1, uint64_t counter2)
- *Initialize two successive posted counter with the value specified.* static __always_inline void ezdp_dual_write_posted_ctr_async (ezdp_sum_addr_t addr, uint64_t counter1, uint64_t counter2)
- *Non blocking version of ezdp_dual_write_posted_ctr.* static __always_inline void ezdp_add_posted_ctr (ezdp_sum_addr_t addr, int32_t value)
- *Add signed value to posted counter.* static __always_inline void ezdp_add_posted_ctr_async (ezdp_sum_addr_t addr, int32_t value)
- *Non blocking version of ezdp_add_posted_ctr.* static __always_inline void ezdp_dual_add_posted_ctr (ezdp_sum_addr_t addr, int16_t counter1, int16_t counter2)
- *Add signed values to two successive posted counters.* static __always_inline void ezdp_dual_add_posted_ctr_async (ezdp_sum_addr_t addr, int16_t counter1, int16_t counter2)
- *Non blocking version of ezdp_add_posted_ctr_dual.* static __always_inline void ezdp_report_posted_ctr (ezdp_sum_addr_t sum_addr, bool flush)
- *Generate posted counter value report.* static __always_inline void ezdp_report_and_clear_posted_ctr (ezdp_sum_addr_t sum_addr, bool flush)
- *Generate posted counter value report and reset the counter to zero.* static __always_inline void ezdp_dual_report_posted_ctr (ezdp_sum_addr_t addr, bool flush)
- *Generate posted counter value report for two successive counters.* static __always_inline void ezdp_dual_report_and_clear_posted_ctr (ezdp_sum_addr_t addr, bool flush)
- *Generate posted counter value report for two successive counters and reset both counter values to zero.* static __always_inline void ezdp_reset_posted_ctr (ezdp_sum_addr_t addr)
- *Reset posted counter.* static __always_inline void ezdp_reset_posted_ctr_async (ezdp_sum_addr_t addr)
- *Non blocking version of ezdp_reset_posted_ctr.* static __always_inline void ezdp_dual_reset_posted_ctr (ezdp_sum_addr_t addr)
- *Reset two successive posted counter.* static __always_inline void ezdp_dual_reset_posted_ctr_async (ezdp_sum_addr_t addr)
- *Non blocking version of ezdp_dual_reset_posted_ctr.* static __always_inline bool ezdp_init_posted_ctr_msg_queue_desc (uint32_t partition_id, uint32_t queue_mask, ezdp_posted_ctr_msg_queue_desc_t *posted_ctr_queue_desc, char __cmem *work_area_ptr, uint32_t work_area_size)
- *Initialize posted counter message queue descriptor.* static __always_inline bool ezdp_read_posted_ctr_msg (ezdp_posted_ctr_msg_queue_desc_t *ctr_queue_desc, struct ezdp_posted_ctr_msg *msg, char __cmem *work_area_ptr, uint32_t work_area_size)

Read posted counter message from message queue.

Function Documentation

static __always_inline void ezdp_write_single_ctr_cfg (ezdp_sum_addr_t *addr*, struct ezdp_single_ctr_cfg * *counter*) [static]

Configure single counter and its initial value.

Parameters:

[in] *addr* - Address of counter
[in] *counter* - counter configuration + value

Returns:
void

static __always_inline void ezdp_write_single_ctr_cfg_async (ezdp_sum_addr_t *addr*, struct ezdp_single_ctr_cfg * *counter*) [static]

Non blocking version of ezdp_write_single_ctr_cfg.

Parameters:
[in] *addr* - Address of counter
[in] *counter* - counter configuration + value

Note:
Call ezdp_sync() to wait for the operation to complete, confirming that the data was copied to the destination.

Returns:
void

static __always_inline void ezdp_read_single_ctr_cfg (ezdp_sum_addr_t *addr*, struct ezdp_single_ctr_cfg __cmem * *counter*) [static]

Read single counter configuration.

Parameters:
[in] *addr* - Address of counter
[out] *counter* - Address at CMEM to write counter config

Returns:
void

static __always_inline void ezdp_read_single_ctr_cfg_async (ezdp_sum_addr_t *addr*, struct ezdp_single_ctr_cfg __cmem * *counter*) [static]

Non blocking version of ezdp_read_single_ctr_cfg.

Parameters:
[in] *addr* - Address of counter
[out] *counter* - Address at CMEM to write counter config

Note:
Call ezdp_sync() to wait for the operation to complete, confirming that the data was copied to the destination.

Returns:
void

static __always_inline void ezdp_write_single_ctr (ezdp_sum_addr_t *addr*, uint64_t *value*) [static]

Initialize single counter with the value specified.

Parameters:
[in] *addr* - Address of counter

[in] *value* - Counter value

Returns:

void

static __always_inline void ezdp_write_single_ctr_async (ezdp_sum_addr_t *addr*, uint64_t *value*) [static]

Non blocking version of ezdp_write_single_ctr.

Parameters:

[in] *addr* - Address of counter
[in] *value* - Counter value

Note:

Call ezdp_sync() to wait for the operation to complete, confirming that the data was copied to the destination.

Returns:

void

static __always_inline uint32_t ezdp_xchg_single_ctr (ezdp_sum_addr_t *addr*, uint64_t *value*, uint64_t __cmem * *orig_value*) [static]

Write single counter with the value specified and read previous counter value.

Parameters:

[in] *addr* - Address of counter
[in] *value* - Counter value
[out] *orig_value* - Address at CMEM to write old counter value

Returns:

uint32_t - old 32 MSB value

static __always_inline uint32_t ezdp_read_single_ctr (ezdp_sum_addr_t *addr*, uint64_t __cmem * *value*, bool * *overflow*) [static]

Read single counter value.

Parameters:

[in] *addr* - Address of counter
[out] *value* - Address at CMEM to write counter value
[out] *overflow* - over flow flag of previous operation

Returns:

uint32_t - 32 MSB value

static __always_inline void ezdp_inc_single_ctr (ezdp_sum_addr_t *addr*, uint16_t *increment_value*) [static]

Increment single counter by the value specified.

Parameters:

[in] *addr* - Address of counter

[in] *increment_value* - value to enlarge

Returns:

void

static __always_inline void ezdp_inc_single_ctr_async (ezdp_sum_addr_t *addr*, uint16_t *increment_value*) [static]

Non blocking version of ezdp_inc_single_ctr.

Parameters:

[in] *addr* - Address of counter
[in] *increment_value* - value to enlarge

Note:

Call ezdp_sync() to wait for the operation to complete, confirming that the data was copied to the destination.

Returns:

void

static __always_inline uint32_t ezdp_read_and_inc_single_ctr (ezdp_sum_addr_t *addr*, uint16_t *increment_value*, uint64_t __cmem * *orig_value*, bool * *overflow*) [static]

Increment single counter by the value specified and read previous counter value.

Parameters:

[in] *addr* - Address of counter
[in] *increment_value* - value to enlarge
[out] *orig_value* - Address at CMEM to write old counter value
[out] *overflow* - over flow flag of previous operation

Returns:

uint32_t - old 32 MSB value

static __always_inline void ezdp_dec_single_ctr (ezdp_sum_addr_t *addr*, uint16_t *decrement_value*) [static]

Decrement single counter by the value specified.

Parameters:

[in] *addr* - Address of counter
[in] *decrement_value* - value to reduce

Returns:

void

static __always_inline void ezdp_dec_single_ctr_async (ezdp_sum_addr_t *addr*, uint16_t *decrement_value*) [static]

Non blocking version of ezdp_dec_single_ctr.

Parameters:

[in] *addr* - Address of counter

[in] *decrement_value* - value to reduce

Note:

Call ezdp_sync() to wait for the operation to complete, confirming that the data was copied to the destination.

Returns:

void

static __always_inline uint32_t ezdp_read_and_dec_single_ctr (ezdp_sum_addr_t *addr*, uint16_t *decrement_value*, uint64_t __cmem * *orig_value*, bool * *overflow*) [static]

Decrement single counter by the value specified and read previous counter value.

Parameters:

[in] *addr* - Address of counter
[in] *decrement_value* - value to reduce
[out] *orig_value* - Address at CMEM to write old counter value
[out] *overflow* - over flow flag of previous operation

Returns:

uint32_t - old 32 MSB value

static __always_inline void ezdp_reset_single_ctr (ezdp_sum_addr_t *addr*) [static]

Reset single counter to zero.

Parameters:

[in] *addr* - Address of counter

Returns:

void

static __always_inline void ezdp_reset_single_ctr_async (ezdp_sum_addr_t *addr*) [static]

Non blocking version of ezdp_reset_single_ctr.

Parameters:

[in] *addr* - Address of counter

Note:

Call ezdp_sync() to wait for the operation to complete, confirming that the data was copied to the destination.

Returns:

void

static __always_inline uint32_t ezdp_read_and_reset_single_ctr (ezdp_sum_addr_t *addr*, uint64_t __cmem * *orig_value*) [static]

Reset single counter to zero and read previous counter value.

Parameters:

[in] *addr* - Address of counter
[out] *orig_value* - Address at CMEM to write old counter value

Returns:
uint32_t - old 32 MSB value

static __always_inline void ezdp_cond_dec_single_ctr (ezdp_sum_addr_t *addr*, uint16_t *decrement_value*) [static]

Conditionally decrement single counter by the value specified.

If the decremented value is greater than zero, performs decrement operation, otherwise does not perform the operation.

Parameters:
[in] *addr* - Address of counter
[in] *decrement_value* - value to reduce

Returns:
void

static __always_inline void ezdp_cond_dec_single_ctr_async (ezdp_sum_addr_t *addr*, uint16_t *decrement_value*) [static]

Non blocking version of ezdp_cond_dec_single_ctr.

Parameters:
[in] *addr* - Address of counter
[in] *decrement_value* - value to reduce

Note:
Call ezdp_sync() to wait for the operation to complete, confirming that the data was copied to the destination.

Returns:
void

static __always_inline uint32_t ezdp_read_and_cond_dec_single_ctr (ezdp_sum_addr_t *addr*, uint16_t *decrement_value*, uint64_t __cmem * *orig_value*, bool * *failure_ind*) [static]

Conditionally decrement single counter by the value specified and read previous counter value.

If the decremented value is greater than zero, performs decrement operation, otherwise does not perform the operation.

Parameters:
[in] *addr* - Address of counter
[in] *decrement_value* - value to reduce
[out] *orig_value* - Address at CMEM to write old counter value
[out] *failure_ind* - failure indication

Returns:
uint32_t - old 32 MSB value

static __always_inline void ezdp_prefetch_single_ctr (ezdp_sum_addr_t *addr*) [static]

Prefetch single counter into the local cache.

Load counter into the cache to hide latency of the accessing this counter.

Parameters:
[in] *addr* - Address of counter

Returns:
none

static __always_inline void ezdp_prefetch_single_ctr_async (ezdp_sum_addr_t *addr*) [static]

Non blocking version of ezdp_prefetch_single_ctr.

Parameters:
[in] *addr* - Address of counter

Note:
Call ezdp_sync() to wait for the operation to complete, confirming that the data was copied to the destination.

Returns:
void

static __always_inline void ezdp_write_dual_ctr_cfg (ezdp_sum_addr_t *addr*, struct ezdp_dual_ctr_cfg * *counter*) [static]

Configure dual counter and its initial values (byte and event).

Parameters:
[in] *addr* - Address of counter
[in] *counter* - counter configuration + value

Returns:
void

static __always_inline void ezdp_write_dual_ctr_cfg_async (ezdp_sum_addr_t *addr*, struct ezdp_dual_ctr_cfg * *counter*) [static]

Non blocking version of ezdp_write_dual_ctr_cfg.

Parameters:
[in] *addr* - Address of counter
[in] *counter* - counter configuration + value

Note:
Call ezdp_sync() to wait for the operation to complete, confirming that the data was copied to the destination.

Returns:
void

static __always_inline void ezdp_read_dual_ctr_cfg (ezdp_sum_addr_t *addr*, struct ezdp_dual_ctr_cfg __cmem * *counter*) [static]

Read dual counter configuration.

Parameters:
[in] *addr* - Address of counter
[out] *counter* - Address at CMEM to write counter config

Returns:
void

static __always_inline uint32_t ezdp_read_dual_ctr (ezdp_sum_addr_t *addr*, struct ezdp_dual_ctr_result __cmem * *value*, bool * *overflow*) [static]

Read dual counter values (byte and event).

Parameters:
[in] *addr* - Address of counter
[out] *value* - Address at CMEM to write counter value
[out] *overflow* - over flow flag of previous operation

Returns:
uint32_t - 32 MSB value

static __always_inline void ezdp_inc_dual_ctr (ezdp_sum_addr_t *addr*, uint16_t *inc_byte_value*, uint16_t *inc_event_value*) [static]

Increment dual counter with the values specified (byte and event).

Parameters:
[in] *addr* - Address of counter
[in] *inc_byte_value* - byte value to increase
[in] *inc_event_value* - event value to increase

Returns:
void

static __always_inline void ezdp_inc_dual_ctr_async (ezdp_sum_addr_t *addr*, uint16_t *inc_byte_value*, uint16_t *inc_event_value*) [static]

Non blocking version of ezdp_inc_dual_ctr.

Parameters:
[in] *addr* - Address of counter
[in] *inc_byte_value* - byte value to increase
[in] *inc_event_value* - event value to increase

Note:
Call ezdp_sync() to wait for the operation to complete, confirming that the data was copied to the destination.

Returns:
void

static __always_inline uint32_t ezdp_read_and_inc_dual_ctr (ezdp_sum_addr_t *addr*, uint16_t *inc_byte_value*, uint16_t *inc_event_value*, struct ezdp_dual_ctr_result __cmem * *orig_value*, bool * *overflow*) [static]

Increment dual counter with the values specified (byte and event) and read previous counter values.

Parameters:

[in] *addr* - Address of counter
[in] *inc_byte_value* - byte value to increase
[in] *inc_event_value* - event value to increase
[out] *orig_value* - Address at CMEM to write old counter value
[out] *overflow* - over flow flag of previous operation

Returns:

uint32_t - old 32 MSB value

static __always_inline void ezdp_dec_dual_ctr (ezdp_sum_addr_t *addr*, uint16_t *dec_byte_value*) [static]

Decrement dual counter's byte value by the value specified and event value by 1.

Parameters:

[in] *addr* - Address of counter
[in] *dec_byte_value* - value to reduce byte counter

Returns:

void

static __always_inline void ezdp_dec_dual_ctr_async (ezdp_sum_addr_t *addr*, uint16_t *dec_byte_value*) [static]

Non blocking version of ezdp_dec_dual_ctr.

Parameters:

[in] *addr* - Address of counter
[in] *dec_byte_value* - value to reduce byte counter

Note:

1. Call ezdp_sync() to wait for the operation to complete, confirming that the data was copied to the destination. 2. event will reduce by 1

Returns:

void

static __always_inline uint32_t ezdp_read_and_dec_dual_ctr (ezdp_sum_addr_t *addr*, uint16_t *dec_byte_value*, struct ezdp_dual_ctr_result __cmem * *orig_value*, bool * *overflow*) [static]

Decrement dual counter's byte value by the value specified and event value by 1, and read previous counter values.

Parameters:

[in] *addr* - Address of counter
[in] *dec_byte_value* - value to reduce byte counter
[out] *orig_value* - Address at CMEM to write old counter value
[out] *overflow* - over flow flag of previous operation

Returns:

uint32_t - old 32 MSB value

static __always_inline void ezdp_reset_dual_ctr (ezdp_sum_addr_t *addr*) [static]

Reset dual counter values (byte and event) to zero.

Parameters:

[in] *addr* - Address of counter

Returns:

void

static __always_inline void ezdp_reset_dual_ctr_async (ezdp_sum_addr_t *addr*) [static]

Non blocking version of ezdp_reset_dual_ctr.

Parameters:

[in] *addr* - Address of counter

Note:

Call ezdp_sync() to wait for the operation to complete, confirming that the data was copied to the destination.

Returns:

void

static __always_inline uint32_t ezdp_read_and_reset_dual_ctr (ezdp_sum_addr_t *addr*, struct ezdp_dual_ctr_result __cmem * *orig_value*) [static]

Reset dual counter values (byte and event) to zero and read previous counter values.

Parameters:

[in] *addr* - Address of counter

[out] *orig_value* - Address at CMEM to write old counter value

Returns:

uint32_t - old 32 MSB value

static __always_inline void ezdp_prefetch_dual_ctr (ezdp_sum_addr_t *addr*) [static]

Prefetch dual counter into the local cache.

Load counter into the cache to hide latency of the accessing this counter.

Parameters:

[in] *addr* - Address of counter

Returns:

none

static __always_inline void ezdp_prefetch_dual_ctr_async (ezdp_sum_addr_t *addr*) [static]

Non blocking version of ezdp_prefetch_dual_ctr.

Parameters:

[in] *addr* - Address of counter

Note:

Call ezdp_sync() to wait for the operation to complete, confirming that the data was copied to the destination.

Returns:

void

static __always_inline void ezdp_write_bitwise_ctr_cfg (ezdp_sum_addr_t *addr*, uint64_t *value*) [static]

Configure bitwise counter and its initial value.

Parameters:

[in] *addr* - Address of counter
[in] *value* - counter value

Returns:

void

static __always_inline void ezdp_write_bitwise_ctr_cfg_async (ezdp_sum_addr_t *addr*, uint64_t *value*) [static]

Non blocking version of ezdp_write_bitwise_ctr_cfg.

Parameters:

[in] *addr* - Address of counter
[in] *value* - counter value

Note:

Call ezdp_sync() to wait for the operation to complete, confirming that the data was copied to the destination.

Returns:

void

static __always_inline void ezdp_read_bitwise_ctr_cfg (ezdp_sum_addr_t *addr*, struct ezdp_bitwise_ctr_cfg __cmem * *counter*) [static]

Read bitwise counter configuration.

Parameters:

[in] *addr* - Address of counter
[out] *counter* - Address at CMEM to write counter config

Returns:

void

static __always_inline void ezdp_read_bitwise_ctr_cfg_async (ezdp_sum_addr_t *addr*, struct ezdp_bitwise_ctr_cfg __cmem * *counter*) [static]

Non blocking version of ezdp_read_bitwise_ctr_cfg.

Parameters:

[in] *addr* - Address of counter

[out] *counter* - Address at CMEM to write counter config

Note:

Call ezdp_sync() to wait for the operation to complete, confirming that the data was copied to the destination.

Returns:

void

static __always_inline void ezdp_write_bits_bitwise_ctr (ezdp_sum_addr_t *addr*, uint16_t *value*, enum ezdp_bitwise_size *size*, uint8_t *offset*) [static]

Write the value to the selected bits in the bitwise counter.

Parameters:

[in] *addr* - Address of counter
[in] *value* - bit value
[in] *size* - The size of the operation
[in] *offset* - Target bit offset. Must be multiply of the size

Note:

This command with the Read option may be used to implement a swap

Returns:

void

static __always_inline void ezdp_write_bits_bitwise_ctr_async (ezdp_sum_addr_t *addr*, uint16_t *value*, enum ezdp_bitwise_size *size*, uint8_t *offset*) [static]

Non blocking version of ezdp_write_bits_bitwise_ctr.

Parameters:

[in] *addr* - Address of counter
[in] *value* - bit value
[in] *size* - The size of the operation
[in] *offset* - Target bit offset. Must be multiply of the size

Note:

Call ezdp_sync() to wait for the operation to complete, confirming that the data was copied to the destination.

Returns:

void

static __always_inline void ezdp_xchg_bits_bitwise_ctr (ezdp_sum_addr_t *addr*, uint16_t *value*, enum ezdp_bitwise_size *size*, uint8_t *offset*, uint64_t __cmem * *orig_value*) [static]

Write the value to the selected bits in the bitwise counter and read previous counter value.

Parameters:

[in] *addr* - Address of counter
[in] *value* - bits value
[in] *size* - The size of the operation
[in] *offset* - Target bit offset. Must be multiply of the size
[out] *orig_value* - Address at CMEM to write old counter value

Returns:
void

static __always_inline uint32_t ezdp_read_bitwise_ctr (ezdp_sum_addr_t *addr*, uint64_t __cmem * *value*) [static]

Read bitwise counter value.

Parameters:
[in] *addr* - Address of counter
[out] *value* - Address at CMEM to write counter value

Returns:
uint32_t - 32 MSB value

static __always_inline void ezdp_read_bits_bitwise_ctr (ezdp_sum_addr_t *addr*, enum ezdp_bitwise_size *size*, uint8_t *offset*, uint64_t __cmem * *value*) [static]

Read the selected bits from the bitwise counter.

Parameters:
[in] *addr* - Address of counter
[in] *size* - The size of the operation
[in] *offset* - Target bit offset. Must be multiply of the size
[out] *value* - Address at CMEM to write counter value

Returns:
void

static __always_inline void ezdp_inc_bits_bitwise_ctr (ezdp_sum_addr_t *addr*, uint16_t *increment_value*, enum ezdp_bitwise_size *size*, uint8_t *offset*) [static]

Increment the selected bits in the bitwise counter by the value specified.

Parameters:
[in] *addr* - Address of counter
[in] *increment_value* - value to enlarge
[in] *size* - The size of the operation
[in] *offset* - Target bit offset. Must be multiply of the size

Returns:
void

static __always_inline void ezdp_inc_bits_bitwise_ctr_async (ezdp_sum_addr_t *addr*, uint16_t *increment_value*, enum ezdp_bitwise_size *size*, uint8_t *offset*) [static]

Non blocking version of ezdp_inc_bits_bitwise_ctr.

Parameters:
[in] *addr* - Address of counter
[in] *increment_value* - value to enlarge
[in] *size* - The size of the operation
[in] *offset* - Target bit offset. Must be multiply of the size

Note:

Call ezdp_sync() to wait for the operation to complete, confirming that the data was copied to the destination.

Returns:

void

static __always_inline void ezdp_read_and_inc_bits_bitwise_ctr (ezdp_sum_addr_t *addr*, uint16_t *increment_value*, enum ezdp_bitwise_size *size*, uint8_t *offset*, uint64_t __cmem * *orig_value*) [static]

Increment the selected bits in the bitwise counter by the value specified and read previous counter value.

Parameters:

[in] *addr* - Address of counter
[in] *increment_value* - value to enlarge
[in] *size* - The size of the operation
[in] *offset* - Target bit offset. Must be multiply of the size
[out] *orig_value* - Address at CMEM to write old counter value (16 bits)

Returns:

void

static __always_inline void ezdp_dec_bits_bitwise_ctr (ezdp_sum_addr_t *addr*, uint16_t *decrement_value*, enum ezdp_bitwise_size *size*, uint8_t *offset*) [static]

Decrement the selected bits in the bitwise counter by the value specified.

Parameters:

[in] *addr* - Address of counter
[in] *decrement_value* - value to reduce
[in] *size* - The size of the operation
[in] *offset* - Target bit offset. Must be multiply of the size

Returns:

void

static __always_inline void ezdp_dec_bits_bitwise_ctr_async (ezdp_sum_addr_t *addr*, uint16_t *decrement_value*, enum ezdp_bitwise_size *size*, uint8_t *offset*) [static]

Non blocking version of ezdp_dec_bits_bitwise_ctr.

Parameters:

[in] *addr* - Address of counter
[in] *decrement_value* - value to reduce
[in] *size* - The size of the operation
[in] *offset* - Target bit offset. Must be multiply of the size

Note:

Call ezdp_sync() to wait for the operation to complete, confirming that the data was copied to the destination.

Returns:

void

static __always_inline void ezdp_read_and_dec_bits_bitwise_ctr (ezdp_sum_addr_t *addr*, uint16_t *decrement_value*, enum ezdp_bitwise_size *size*, uint8_t *offset*, uint64_t __cmem * *orig_value*) [static]

Decrement the selected bits in the bitwise counter by the value specified and read previous counter value.

Parameters:

[in] *addr* - Address of counter
[in] *decrement_value* - value to reduce
[in] *size* - The size of the operation
[in] *offset* - Target bit offset. Must be multiply of the size
[out] *orig_value* - Address at CMEM to write old counter value (16 bits)

Returns:

void

static __always_inline void ezdp_reset_bitwise_ctr (ezdp_sum_addr_t *addr*) [static]

Reset bitwise counter value to zero.

Parameters:

[in] *addr* - Address of counter

Returns:

void

static __always_inline void ezdp_reset_bitwise_ctr_async (ezdp_sum_addr_t *addr*) [static]

Non blocking version of ezdp_reset_bitwise_ctr.

Parameters:

[in] *addr* - Address of counter

Note:

Call ezdp_sync() to wait for the operation to complete, confirming that the data was copied to the destination.

Returns:

void

static __always_inline void ezdp_read_and_reset_bitwise_ctr (ezdp_sum_addr_t *addr*, uint64_t __cmem * *orig_value*) [static]

Reset bitwise counter value to zero and read previous counter value.

Parameters:

[in] *addr* - Address of counter
[out] *orig_value* - Address at CMEM to write old counter value

Returns:

void

static __always_inline void ezdp_set_bits_bitwise_ctr (ezdp_sum_addr_t *addr*, uint16_t *value*, enum ezdp_bitwise_size *size*, uint8_t *offset*) [static]

Set the selected bits in the bitwise counter according to the value specified.

Parameters:

[in] *addr* - Address of counter
[in] *value* - event value to increase
[in] *size* - The size of the operation
[in] *offset* - Target bit offset. Must be multiply of the size

Returns:

void

static __always_inline void ezdp_set_bits_bitwise_ctr_async (ezdp_sum_addr_t *addr*, uint16_t *value*, enum ezdp_bitwise_size *size*, uint8_t *offset*) [static]

Non blocking version of ezdp_set_bits_bitwise_ctr.

Parameters:

[in] *addr* - Address of counter
[in] *value* - event value to increase
[in] *size* - The size of the operation
[in] *offset* - Target bit offset. Must be multiply of the size

Note:

Call ezdp_sync() to wait for the operation to complete, confirming that the data was copied to the destination.

Returns:

void

static __always_inline void ezdp_read_and_set_bits_bitwise_ctr (ezdp_sum_addr_t *addr*, uint16_t *value*, enum ezdp_bitwise_size *size*, uint8_t *offset*, uint64_t __cmem * *orig_value*) [static]

Set the selected bits in the bitwise counter according to the value specified and read previous counter value.

Parameters:

[in] *addr* - Address of counter
[in] *value* - event value to increase
[in] *size* - The size of the operation
[in] *offset* - Target bit offset. Must be multiply of the size
[out] *orig_value* - Address at CMEM to write old counter value (16 bits)

Returns:

void

static __always_inline void ezdp_clear_bits_bitwise_ctr (ezdp_sum_addr_t *addr*, uint16_t *value*, enum ezdp_bitwise_size *size*, uint8_t *offset*) [static]

Clear the selected bits in the bitwise counter according to the value specified.

Parameters:

[in] *addr* - Address of counter
[in] *value* - bits value
[in] *size* - The size of the operation
[in] *offset* - Target bit offset. Must be multiply of the size

Returns:

void

static __always_inline void ezdp_clear_bits_bitwise_ctr_async (ezdp_sum_addr_t *addr*, uint16_t *value*, enum ezdp_bitwise_size *size*, uint8_t *offset*) [static]

Non blocking version of ezdp_clear_bits_bitwise_ctr.

Parameters:

[in] *addr* - Address of counter
[in] *value* - bits value
[in] *size* - The size of the operation
[in] *offset* - Target bit offset. Must be multiply of the size

Note:

Call ezdp_sync() to wait for the operation to complete, confirming that the data was copied to the destination.

Returns:

void

static __always_inline void ezdp_read_and_clear_bits_bitwise_ctr (ezdp_sum_addr_t *addr*, uint16_t *value*, enum ezdp_bitwise_size *size*, uint8_t *offset*, uint64_t __cmem * *orig_value*) [static]

Clear the selected bits in the bitwise counter according to the value specified and read previous counter value.

Parameters:

[in] *addr* - Address of counter
[in] *value* - bits value
[in] *size* - The size of the operation
[in] *offset* - Target bit offset. Must be multiply of the size
[out] *orig_value* - Address at CMEM to write old counter value (16 bits)

Returns:

void

static __always_inline void ezdp_read_and_cond_write_bits_bitwise_ctr (ezdp_sum_addr_t *addr*, uint8_t *value*, enum ezdp_bitwise_size *size*, uint8_t *offset*, uint8_t *cmp_value*, uint64_t __cmem * *orig_value*, bool * *success_ind*) [static]

Read, compare and conditionally set the specified bits in the bitwise counter with the value specified.

If value of the selected bits, based on size (1, 2 or 4 bits), is match cmp_value, it replaces the bits with the new value Old value and success/failure indication is returned.

Parameters:

[in] *addr* - Address of counter
[in] *value* - bits value
[in] *size* - The size of the operation Applicable values are 1,2,4 Encoded as log
[in] *offset* - Target bit offset. Must be multiply of the size
[in] *cmp_value* - compare value

[out] *orig_value* - Address at CMEM to write old counter value (16 bits)
[out] *success_ind* - success indication

Note:
This command may be used for a state machine implementation

Returns:
void

static __always_inline void ezdp_prefetch_bitwise_ctr (ezdp_sum_addr_t *addr*) [static]

Prefetch bitwise counter into the local cache.

Load counter into the cache to hide latency of the accessing this counter.

Parameters:
[in] *addr* - Address of counter

Returns:
none

static __always_inline void ezdp_prefetch_bitwise_ctr_async (ezdp_sum_addr_t *addr*) [static]

Non blocking version of ezdp_prefetch_bitwise_ctr.

Parameters:
[in] *addr* - Address of counter

Note:
Call ezdp_sync() to wait for the operation to complete, confirming that the data was copied to the destination.

Returns:
void

static __always_inline void ezdp_write_tb_ctr_cfg (ezdp_sum_addr_t *addr*, struct ezdp_tb_ctr_cfg * *counter*) [static]

Configure token bucket counter.

Parameters:
[in] *addr* - Address of counter
[in] *counter* - counter configuration

Returns:
void

static __always_inline void ezdp_write_tb_ctr_cfg_async (ezdp_sum_addr_t *addr*, struct ezdp_tb_ctr_cfg * *counter*) [static]

Non blocking version of ezdp_write_tb_ctr_cfg.

Parameters:
[in] *addr* - Address of counter
[in] *counter* - counter configuration

Note:

Call ezdp_sync() to wait for the operation to complete, confirming that the data was copied to the destination.

Returns:

void

static __always_inline void ezdp_read_tb_ctr_cfg (ezdp_sum_addr_t *addr*, struct ezdp_tb_ctr_cfg __cmem * *counter*) [static]

Read token bucket counter configuration.

Parameters:

[in] *addr* - Address of counter
[out] *counter* - Address at CMEM to write counter config

Returns:

void

static __always_inline void ezdp_update_tb_ctr (ezdp_sum_addr_t *addr*, uint16_t *value*, enum ezdp_tb_color *pre_color*) [static]

Update a token bucket counter with the specified value (e.g.

packet length).

Parameters:

[in] *addr* - Address of counter
[in] *value* - Value to add to TB counter, e.g. packet length in bytes
[in] *pre_color* - Pre-Color (e.g. packet original color)

Returns:

void

static __always_inline void ezdp_update_tb_ctr_async (ezdp_sum_addr_t *addr*, uint16_t *value*, enum ezdp_tb_color *pre_color*) [static]

Non blocking version of ezdp_update_tb_ctr.

Parameters:

[in] *addr* - Address of counter
[in] *value* - Value to add to TB counter, e.g. packet length in bytes
[in] *pre_color* - Pre-Color (e.g. packet original color)

Returns:

void

static __always_inline void ezdp_read_tb_ctr (ezdp_sum_addr_t *addr*, uint16_t *value*, enum ezdp_tb_color *pre_color*, struct ezdp_tb_ctr_result __cmem * *result_color*) [static]

Get the resulting color after updating a token bucket counter with the specified value (e.g.

packet length).

Parameters:

[in] *addr* - Address of counter
[in] *value* - Value to add to TB counter, e.g. packet length in bytes

[in] *pre_color* - Pre-Color (e.g. packet original color)
[out] *result_color* - Address at CMEM to write result color

Returns:
void

static __always_inline void ezdp_read_tb_ctr_async (ezdp_sum_addr_t *addr*, uint16_t *value*, enum ezdp_tb_color *pre_color*, struct ezdp_tb_ctr_result __cmem * *result_color*) `[static]`

Non blocking version of ezdp_read_tb_color.

Parameters:
[in] *addr* - Address of counter
[in] *value* - Value to add to TB counter, e.g. packet length in bytes
[in] *pre_color* - Pre-Color (e.g. packet original color)
[out] *result_color* - Address at CMEM to write result color

Returns:
void

static __always_inline void ezdp_check_tb_ctr (ezdp_sum_addr_t *addr*, uint16_t *value*, enum ezdp_tb_color *pre_color*, struct ezdp_tb_ctr_result __cmem * *result_value*) `[static]`

Get the resulting color after updating a token bucket with the specified value (e.g.
packet length), without updating it.

Parameters:
[in] *addr* - Address of counter
[in] *value* - Value to add to TB counter, e.g. packet length in bytes
[in] *pre_color* - Pre-Color (e.g. packet original color)
[out] *result_value* - Address at CMEM to write result color

Returns:
ezdp_tb_color - resulting color

static __always_inline void ezdp_check_tb_ctr_async (ezdp_sum_addr_t *addr*, uint16_t *value*, enum ezdp_tb_color *pre_color*, struct ezdp_tb_ctr_result __cmem * *result_value*) `[static]`

Non blocking version of ezdp_check_tb_ctr.

Parameters:
[in] *addr* - Address of counter
[in] *value* - Value to add to TB counter, e.g. packet length in bytes
[in] *pre_color* - Pre-Color (e.g. packet original color)
[out] *result_value* - Address at CMEM to write result color

Returns:
void

static __always_inline void ezdp_inc_tb_ctr (ezdp_sum_addr_t *addr*, uint16_t *value*, enum ezdp_tb_color *pre_color*, bool *inc_commit_bucket*, bool *inc_excess_bucket*) `[static]`

Force increment token buckets with the specified value (e.g.
packet length).

Parameters:

[in] *addr* - Address of counter
[in] *value* - Value to add to TB counter, e.g. packet length in bytes
[in] *pre_color* - Pre-Color (e.g. packet original color)
[in] *inc_commit_bucket* - true if you want to increment commit bucket
[in] *inc_excess_bucket* - true if you want to increment excess bucket

Note:

Can be used to implement hierarchical token bucket algorithms

Returns:

void

static __always_inline void ezdp_inc_tb_ctr_async (ezdp_sum_addr_t *addr*, uint16_t *value*, enum ezdp_tb_color *pre_color*, bool *inc_commit_bucket*, bool *inc_excess_bucket*) `[static]`

Non blocking version of ezdp_inc_tb_ctr.

Parameters:

[in] *addr* - Address of counter
[in] *value* - Value to add to TB counter, e.g. packet length in bytes
[in] *pre_color* - Pre-Color (e.g. packet original color)
[in] *inc_commit_bucket* - true if you want to increment commit bucket
[in] *inc_excess_bucket* - true if you want to increment excess bucket

Note:

Call ezdp_sync() to wait for the operation to complete.

Returns:

void

static __always_inline void ezdp_read_and_inc_tb_ctr (ezdp_sum_addr_t *addr*, uint16_t *value*, enum ezdp_tb_color *pre_color*, bool *inc_commit_bucket*, bool *inc_excess_bucket*, struct ezdp_tb_ctr_result __cmem * *result_value*) `[static]`

Force increment token buckets with the specified value (e.g.

packet length) and get resulting color and bucket states.

Parameters:

[in] *addr* - Address of counter
[in] *value* - Value to add to TB counter, e.g. packet length in bytes
[in] *pre_color* - Pre-Color (e.g. packet original color)
[in] *inc_commit_bucket* - true if you want to increment commit bucket
[in] *inc_excess_bucket* - true if you want to increment excess bucket
[out] *result_value* - Address at CMEM to write result color

Note:

Can be used to implement hierarchical token bucket algorithms

Returns:

none

static __always_inline void ezdp_dec_tb_ctr (ezdp_sum_addr_t *addr*, uint16_t *value*, enum ezdp_tb_color *pre_color*, bool *dec_commit_bucket*, bool *dec_excess_bucket*) `[static]`

Force decrement token buckets with the specified value (e.g.

packet length).

Parameters:

[in] *addr* - Address of counter
[in] *value* - Value to decrement to TB counter, e.g. packet length in bytes
[in] *pre_color* - Pre-Color (e.g. packet original color)
[in] *dec_commit_bucket* - true if you want to increment commit bucket
[in] *dec_excess_bucket* - true if you want to increment excess bucket

Note:

Can be used to implement hierarchical token bucket algorithms

Returns:

void

static __always_inline void ezdp_dec_tb_ctr_async (ezdp_sum_addr_t *addr*, uint16_t *value*, enum ezdp_tb_color *pre_color*, bool *dec_commit_bucket*, bool *dec_excess_bucket*) `[static]`

Non blocking version of ezdp_dec_tb_ctr.

Parameters:

[in] *addr* - Address of counter
[in] *value* - Value to decrement to TB counter, e.g. packet length in bytes
[in] *pre_color* - Pre-Color (e.g. packet original color)
[in] *dec_commit_bucket* - true if you want to increment commit bucket
[in] *dec_excess_bucket* - true if you want to increment excess bucket

Note:

Call ezdp_sync() to wait for the operation to complete, confirming that the data was copied to the destination.

Returns:

void

static __always_inline void ezdp_read_and_dec_tb_ctr (ezdp_sum_addr_t *addr*, uint16_t *value*, enum ezdp_tb_color *pre_color*, bool *dec_commit_bucket*, bool *dec_excess_bucket*, struct ezdp_tb_ctr_result __cmem * *result_value*) `[static]`

Force decrement token buckets with the specified value (e.g.

packet length) and get resulting color and bucket states.

Parameters:

[in] *addr* - Address of counter
[in] *value* - Value to add to TB counter, e.g. packet length in bytes
[in] *pre_color* - Pre-Color (e.g. packet original color)
[in] *dec_commit_bucket* - true if you want to increment commit bucket
[in] *dec_excess_bucket* - true if you want to increment excess bucket
[out] *result_value* - Address at CMEM to write result color

Note:

Can be used to implement hierarchical token bucket algorithms

Returns:

none

static __always_inline void ezdp_prefetch_tb_ctr (ezdp_sum_addr_t *addr*) `[static]`

Prefetch token bucket counter into the local cache.

Load counter into the cache to hide latency of the accessing this counter.

Parameters:

[in] *addr* - Address of counter

Returns:

none

static __always_inline void ezdp_prefetch_tb_ctr_async (ezdp_sum_addr_t *addr*) [static]

Non blocking version of ezdp_prefetch_tb_ctr.

Parameters:

[in] *addr* - Address of counter

Note:

Call ezdp_sync() to wait for the operation to complete, confirming that the data was copied to the destination.

Returns:

void

static __always_inline void ezdp_write_hier_tb_ctr_cfg (ezdp_sum_addr_t *addr*, struct ezdp_hier_tb_ctr_cfg * *counter*) [static]

Configure hierarchical token bucket counter.

Parameters:

[in] *addr* - Address of counter
[in] *counter* - counter configuration

Returns:

void

static __always_inline void ezdp_write_hier_tb_ctr_cfg_async (ezdp_sum_addr_t *addr*, struct ezdp_hier_tb_ctr_cfg * *counter*) [static]

Non blocking version of ezdp_write_hier_tb_ctr_cfg.

Parameters:

[in] *addr* - Address of counter
[out] *counter* - Address at CMEM to write counter config

Note:

Call ezdp_sync() to wait for the operation to complete, confirming that the data was copied to the destination.

Returns:

void

static __always_inline void ezdp_read_hier_tb_ctr_cfg (ezdp_sum_addr_t *addr*, struct ezdp_hier_tb_ctr_cfg __cmem * *counter*) [static]

Read hierarchical token bucket counter configuration.

Parameters:

[in] *addr* - Address of counter
[out] *counter* - Address at CMEM to write counter config

Returns:

void

static __always_inline void ezdp_inc_hier_tb_ctr (ezdp_sum_addr_t *addr*, uint16_t *value*, bool *update_ctr0*, bool *update_acc1*) [static]

Increment hierarchical token bucket counter accumulator(s) by the value specified.

Parameters:

[in] *addr* - Address of counter
[in] *value* - 14 bits value to add to TB counter, e.g. packet length in bytes
[in] *update_ctr0* - true if you want to increment accumulator 0
[in] *update_acc1* - true if you want to increment accumulator 1

Returns:

void

static __always_inline void ezdp_inc_hier_tb_ctr_async (ezdp_sum_addr_t *addr*, uint16_t *value*, bool *update_ctr0*, bool *update_acc1*) [static]

Non blocking version of ezdp_inc_hier_tb_ctr.

Parameters:

[in] *addr* - Address of counter
[in] *value* - 14 bits value to add to TB counter, e.g. packet length in bytes
[in] *update_ctr0* - true if you want to increment accumulator 0
[in] *update_acc1* - true if you want to increment accumulator 1

Returns:

void

static __always_inline void ezdp_read_and_inc_hier_tb_ctr (ezdp_sum_addr_t *addr*, uint16_t *value*, bool *update_ctr0*, bool *update_acc1*, struct ezdp_hier_tb_result __cmem * *result*) [static]

Increment hierarchical token bucket counter accumulator(s) by the value specified and read previous counter value.

Parameters:

[in] *addr* - Address of counter
[in] *value* - 14 bits value to add to TB counter, e.g. packet length in bytes
[in] *update_ctr0* - true if you want to increment accumulator 0
[in] *update_acc1* - true if you want to increment accumulator 1
[in] *result* - Address at CMEM to write result

Returns:

void

static __always_inline void ezdp_update_hier_tb_ctr (ezdp_sum_addr_t *addr*, struct ezdp_hier_tb_update * *ctr_update*) [static]

Update hierarchical token bucket counter state, app bits or clear accumulators.

Parameters:

[in] *addr* - Address of counter
[in] *ctr_update* - counter update request structure

Returns:

void

static __always_inline void ezdp_update_hier_tb_ctr_async (ezdp_sum_addr_t *addr*, struct ezdp_hier_tb_update * *ctr_update*) [static]

Non blocking version of ezdp_inc_hier_tb_ctr.

Parameters:

[in] *addr* - Address of counter
[in] *ctr_update* - counter update request structure

Returns:

void

static __always_inline void ezdp_read_and_update_hier_tb_ctr (ezdp_sum_addr_t *addr*, struct ezdp_hier_tb_update * *ctr_update*, struct ezdp_hier_tb_result __cmem * *result*) [static]

Update hierarchical token bucket counter state, app bits or clear accumulators and read previous counter value.

Parameters:

[in] *addr* - Address of counter
[in] *ctr_update* - counter update request structure
[in] *result* - Address at CMEM to write result

Returns:

void

static __always_inline void ezdp_change_state_hier_tb_ctr (ezdp_sum_addr_t *addr*, struct ezdp_hier_tb_result __cmem * *result*) [static]

Change hierarchical token bucket state to Ph1.

Parameters:

[in] *addr* - Address of counter
[in] *result* - Address at CMEM to write result

Note:

1. In case of success, must update BE TBs with the accumulators at the result

Returns:

void

static __always_inline void ezdp_write_watchdog_ctr_cfg (ezdp_sum_addr_t *addr*, struct ezdp_watchdog_ctr_cfg * *counter*) [static]

Configure watchdog counter.

Parameters:

[in] *addr* - Address of counter
[in] *counter* - counter configuration + value

Returns:

void

static __always_inline void ezdp_write_watchdog_ctr_cfg_async (ezdp_sum_addr_t *addr*, struct ezdp_watchdog_ctr_cfg * *counter*) [static]

Non blocking version of ezdp_write_watchdog_ctr_cfg.

Parameters:

[in] *addr* - Address of counter
[in] *counter* - counter configuration + value

Note:

Call ezdp_sync() to wait for the operation to complete, confirming that the data was copied to the destination.

Returns:

void

static __always_inline void ezdp_read_watchdog_ctr_cfg (ezdp_sum_addr_t *addr*, struct ezdp_watchdog_ctr_cfg __cmem * *counter*) [static]

Read watchdog counter configuration.

Parameters:

[in] *addr* - Address of counter
[out] *counter* - Address at CMEM to write counter config

Returns:

void

static __always_inline void ezdp_start_watchdog_ctr (ezdp_sum_addr_t *counter_address*) [static]

Start the watchdog counter.

Sets the valid bit of the counter to one

Parameters:

[in] *counter_address* - Address of counter

Returns:

void

static __always_inline void ezdp_start_watchdog_ctr_async (ezdp_sum_addr_t *counter_address*) [static]

Non blocking version of ezdp_start_watchdog_ctr.

Parameters:

[in] *counter_address* - Address of counter

Returns:
void

static __always_inline void ezdp_inc_watchdog_ctr (ezdp_sum_addr_t *addr*) [static]

Increment the watchdog counter events by one.

Parameters:
[in] *addr* - Address of counter

Returns:
void

static __always_inline void ezdp_inc_watchdog_ctr_async (ezdp_sum_addr_t *addr*) [static]

Non blocking version of ezdp_inc_watchdog_ctr.

Parameters:
[in] *addr* - Address of counter

Note:
Call ezdp_sync() to wait for the operation to complete, confirming that the data was copied to the destination.

Returns:
void

static __always_inline void ezdp_check_watchdog_ctr (ezdp_sum_addr_t *counter_address*, struct ezdp_watchdog_ctr_check_result __cmem * *check_result*) [static]

Check the number of events in the watchdog counter.

If the number of events is within the required range (min/max) the counter window is shifted. Otherwise, an alert is indicated and the counter operation is stopped.

Parameters:
[in] *counter_address* - Address of counter
[out] *check_result* - Address at CMEM to write result

Returns:
void

static __always_inline void ezdp_check_watchdog_ctr_async (ezdp_sum_addr_t *counter_address*, struct ezdp_watchdog_ctr_check_result __cmem * *check_result*) [static]

Non blocking version of ezdp_check_watchdog_ctr.

Parameters:
[in] *counter_address* - Address of counter
[out] *check_result* - Address at CMEM to write result

Returns:
void

static __always_inline void ezdp_prefetch_watchdog_ctr (ezdp_sum_addr_t *addr*) [static]

Prefetch watchdog_counter into the local cache.

Load counter into the cache to hide latency of the accessing this counter.

Parameters:

[in] *addr* - Address of counter

Returns:

none

static __always_inline void ezdp_prefetch_watchdog_ctr_async (ezdp_sum_addr_t *addr*) [static]

Non blocking version of ezdp_prefetch_watchdog_ctr.

Parameters:

[in] *addr* - Address of counter

Note:

Call ezdp_sync() to wait for the operation to complete, confirming that the data was copied to the destination.

Returns:

void

static __always_inline bool ezdp_init_ctr_msg_queue_desc (uint32_t *partition_id*, uint32_t *queue_mask*, ezdp_ctr_msg_queue_desc_t * *ctr_queue_desc*, char __cmem * *work_area_ptr*, uint32_t *work_area_size*) [static]

Initialize counter message queue descriptor.

Parameters:

[in] *partition_id* - partition id of queue
[in] *queue_mask* - relevant queue bit mask for stat_desc 0 mean all queues
[out] *ctr_queue_desc* - counter message queue description
[in] *work_area_ptr* - work area pointer (temporary memory on CMEM to be used by the function). The size of the temporary memory is determined by EZDP_CTR_MSG_QUEUE_WORK_AREA_SIZE
[in] *work_area_size* - size of work area pointer

Returns:

bool - true/false - success/fatal error

static __always_inline bool ezdp_read_ctr_msg (ezdp_ctr_msg_queue_desc_t * *ctr_queue_desc*, struct ezdp_ctr_msg * *msg*, char __cmem * *work_area_ptr*, uint32_t *work_area_size*) [static]

Read counter message from message queue.

Parameters:

[in] *ctr_queue_desc* - counter message queue description
[out] *msg* - message from queue
[in] *work_area_ptr* - pointer to work area (temporary memory in CMEM to be used by the function). The size of the temporary memory is determined by EZDP_CTR_MSG_WORK_AREA_SIZE
[in] *work_area_size* - size of work area pointer

Returns:

bool - true/false - success/fatal error

static __always_inline void ezdp_write_posted_ctr (ezdp_sum_addr_t *addr*, uint64_t *value*) [static]

Initialize posted counter with the value specified.

Parameters:

[in] *addr* - Address of counter
[in] *value* - value to write

Returns:

void

static __always_inline void ezdp_write_posted_ctr_async (ezdp_sum_addr_t *addr*, uint64_t *value*) [static]

Non blocking version of ezdp_write_posted_ctr.

Parameters:

[in] *addr* - Address of counter
[in] *value* - value to set

Returns:

void

static __always_inline void ezdp_dual_write_posted_ctr (ezdp_sum_addr_t *addr*, uint64_t *counter1*, uint64_t *counter2*) [static]

Initialize two successive posted counter with the value specified.

Parameters:

[in] *addr* - Address of counter
[in] *counter1* - value to set counter1
[in] *counter2* - value to set counter2

Note:

The operation is not atomic and implemented as two write to two successive counters

Returns:

void

static __always_inline void ezdp_dual_write_posted_ctr_async (ezdp_sum_addr_t *addr*, uint64_t *counter1*, uint64_t *counter2*) [static]

Non blocking version of ezdp_dual_write_posted_ctr.

Parameters:

[in] *addr* - Address of counter
[in] *counter1* - value to set counter1
[in] *counter2* - value to set counter2

Note:

The operation is not atomic and implemented as two write to two successive counters

Returns:

void

static __always_inline void ezdp_add_posted_ctr (ezdp_sum_addr_t *addr*, int32_t *value*) [static]

Add signed value to posted counter.

Parameters:

[in] *addr* - Address of counter
[in] *value* - signed value to add

Returns:

void

static __always_inline void ezdp_add_posted_ctr_async (ezdp_sum_addr_t *addr*, int32_t *value*) [static]

Non blocking version of ezdp_add_posted_ctr.

Parameters:

[in] *addr* - Address of counter
[in] *value* - signed value to add

Returns:

void

static __always_inline void ezdp_dual_add_posted_ctr (ezdp_sum_addr_t *addr*, int16_t *counter1*, int16_t *counter2*) [static]

Add signed values to two successive posted counters.

Parameters:

[in] *addr* - Address of counter
[in] *counter1* - signed value to add counter1
[in] *counter2* - signed value to add counter2

Returns:

void

static __always_inline void ezdp_dual_add_posted_ctr_async (ezdp_sum_addr_t *addr*, int16_t *counter1*, int16_t *counter2*) [static]

Non blocking version of ezdp_add_posted_ctr_dual.

Parameters:

[in] *addr* - Address of counter
[in] *counter1* - signed value to add counter1
[in] *counter2* - signed value to add counter2

Returns:
void

static __always_inline void ezdp_report_posted_ctr (ezdp_sum_addr_t *sum_addr*, bool *flush*) [static]

Generate posted counter value report.

Parameters:
[in] *sum_addr* - summarize address
[in] *flush* - flush before report

Returns:
void

static __always_inline void ezdp_report_and_clear_posted_ctr (ezdp_sum_addr_t *sum_addr*, bool *flush*) [static]

Generate posted counter value report and reset the counter to zero.

Parameters:
[in] *sum_addr* - summarize address
[in] *flush* - flush before report

Returns:
void

static __always_inline void ezdp_dual_report_posted_ctr (ezdp_sum_addr_t *addr*, bool *flush*) [static]

Generate posted counter value report for two successive counters.

Parameters:
[in] *addr* - Address of counter
[in] *flush* - flush before report

Returns:
void

static __always_inline void ezdp_dual_report_and_clear_posted_ctr (ezdp_sum_addr_t *addr*, bool *flush*) [static]

Generate posted counter value report for two successive counters and reset both counter values to zero.

Parameters:
[in] *addr* - Address of counter
[in] *flush* - flush before report

Returns:
void

static __always_inline void ezdp_reset_posted_ctr (ezdp_sum_addr_t *addr*) [static]

Reset posted counter.

Parameters:

[in] *addr* - Address of counter

Returns:

void

static __always_inline void ezdp_reset_posted_ctr_async (ezdp_sum_addr_t *addr*) [static]

Non blocking version of ezdp_reset_posted_ctr.

Parameters:

[in] *addr* - Address of counter

Returns:

void

static __always_inline void ezdp_dual_reset_posted_ctr (ezdp_sum_addr_t *addr*) [static]

Reset two successive posted counter.

Parameters:

[in] *addr* - Address of counter

Note:

The operation is not atomic and implemented as two write to two successive counters

Returns:

void

static __always_inline void ezdp_dual_reset_posted_ctr_async (ezdp_sum_addr_t *addr*) [static]

Non blocking version of ezdp_dual_reset_posted_ctr.

Parameters:

[in] *addr* - Address of counter

Note:

The operation is not atomic and implemented as two write to two successive counters

Returns:

void

static __always_inline bool ezdp_init_posted_ctr_msg_queue_desc (uint32_t *partition_id*, uint32_t *queue_mask*, ezdp_posted_ctr_msg_queue_desc_t * *posted_ctr_queue_desc*, char __cmem * *work_area_ptr*, uint32_t *work_area_size*) [static]

Initialize posted counter message queue descriptor.

Parameters:

[in] *partition_id* - partition id of queue

[in] *queue_mask* - relevant queue bit mask for posted_desc 0 mean all queues
[out] *posted_ctr_queue_desc* - counter message queue description
[in] *work_area_ptr* - work area pointer (temporary memory on CMEM to be used by the function). The size of the temporary memory is determined by EZDP_CTR_MSG_QUEUE_WORK_AREA_SIZE
[in] *work_area_size* - size of work area pointer

Returns:

bool - true/false - success/fatal error

static __always_inline bool ezdp_read_posted_ctr_msg (ezdp_posted_ctr_msg_queue_desc_t * *ctr_queue_desc*, struct ezdp_posted_ctr_msg * *msg*, char __cmem * *work_area_ptr*, uint32_t *work_area_size*) `[static]`

Read posted counter message from message queue.

Parameters:

[in] *ctr_queue_desc* - counter message queue description
[out] *msg* - message from queue
[in] *work_area_ptr* - pointer to work area (temporary memory in CMEM to be used by the function). The size of the temporary memory is determined by EZDP_POSTED_CTR_MSG_WORK_AREA_SIZE
[in] *work_area_size* - size of work area pointer

Returns:

- true/false - success/fatal error

dpe/dp/include/ezdp_counter_defs.h File Reference

Data Structures

- struct ezdp_single_ctr_cfg
- *On-demand single value counter configuration definition.* struct ezdp_dual_ctr_result
- *On-demand dual value counter result value.* struct ezdp_dual_ctr
- *On-demand dual counter value.* struct ezdp_dual_ctr_cfg
- *On-demand dual counter configuration definition.* struct ezdp_tb_ctr_result
- *Token bucket counter result value definition.* struct ezdp_tb_ctr_cfg
- *Token bucket counter configuration definition.* struct ezdp_hier_tb_ug_app_bits
- *Application bits of Hierarchical token bucket for ultra green feature.* struct ezdp_hier_tb_ctr_cfg
- *Statistic hierarchical token bucket counter config structure (write cfg usage).* struct ezdp_hier_tb_result
- *Hierarchical token bucket counter result value definition.* struct ezdp_hier_tb_update
- *Hierarchical token bucket update counter definition.* struct ezdp_bitwise_ctr_cfg
- *On-demand bitwise counter configuration definition.* struct ezdp_watchdog_accumulative_window_cfg
- *Watchdog accumulative window configuration definition.* struct ezdp_watchdog_sliding_window_cfg
- *Watchdog sliding window configuration definition.* struct ezdp_watchdog_ctr_cfg
- *Watchdog counter configuration definition.* struct ezdp_watchdog_ctr_check_result
- *Watchdog counter check result definition.* struct ezdp_watchdog_ctr_start_result
- *Watchdog counter check result definition.* struct ezdp_ctr_msg
- *Counter message queue definition.* struct ezdp_posted_ctr_msg

Posted counter message queue definition. Defines

- #define EZDP_CTR_MSG_QUEUE_WORK_AREA_SIZE sizeof(struct ezdp_init_msgq_desc_working_area)
- *Work area minimal required size definitions.* #define EZDP_COUNTER_VERSION_MAJOR 2
- #define EZDP_COUNTER_VERSION_MINOR 1
- #define EZDP_SINGLE_CTR_CFG_RESERVED0_10_SIZE 11
- #define EZDP_SINGLE_CTR_CFG_RESERVED0_10_OFFSET 0
- #define EZDP_SINGLE_CTR_CFG_ENABLE_EXCEED_MESSAGE_SIZE 1
- #define EZDP_SINGLE_CTR_CFG_ENABLE_EXCEED_MESSAGE_OFFSET 11
- #define EZDP_SINGLE_CTR_CFG_ENABLE_EXCEED_MESSAGE_WORD_SELECT 0
- #define EZDP_SINGLE_CTR_CFG_ENABLE_EXCEED_MESSAGE_WORD_OFFSET 11
- #define EZDP_SINGLE_CTR_CFG_ENABLE_EXCEED_MESSAGE_MASK (1 << EZDP_SINGLE_CTR_CFG_ENABLE_EXCEED_MESSAGE_WORD_OFFSET)
- #define EZDP_SINGLE_CTR_CFG_ZERO_SIZE 1
- #define EZDP_SINGLE_CTR_CFG_ZERO_OFFSET 12
- #define EZDP_SINGLE_CTR_CFG_REPORT_EXCEEDED_SIZE 6
- #define EZDP_SINGLE_CTR_CFG_REPORT_EXCEEDED_OFFSET 13
- #define EZDP_SINGLE_CTR_CFG_REPORT_EXCEEDED_WORD_SELECT 0
- #define EZDP_SINGLE_CTR_CFG_REPORT_EXCEEDED_WORD_OFFSET 13
- #define EZDP_SINGLE_CTR_CFG_SUB_TYPE_SIZE 5
- #define EZDP_SINGLE_CTR_CFG_SUB_TYPE_OFFSET 19
- #define EZDP_SINGLE_CTR_CFG_ECC_SIZE 8
- #define EZDP_SINGLE_CTR_CFG_ECC_OFFSET 24
- #define EZDP_SINGLE_CTR_CFG_RESERVED32_63_SIZE 32
- #define EZDP_SINGLE_CTR_CFG_RESERVED32_63_OFFSET 32
- #define EZDP_SINGLE_CTR_CFG_VALUE_SIZE 64
- #define EZDP_SINGLE_CTR_CFG_VALUE_OFFSET 64
- #define EZDP_SINGLE_CTR_CFG_VALUE_WORD_SELECT 2
- #define EZDP_SINGLE_CTR_CFG_VALUE_WORD_OFFSET 0
- #define EZDP_SINGLE_CTR_CFG_WORD_COUNT 4
- #define EZDP_DUAL_CTR_RESULT_BYTE_VALUE_MSB_SIZE 31
- #define EZDP_DUAL_CTR_RESULT_BYTE_VALUE_MSB_OFFSET 0
- #define EZDP_DUAL_CTR_RESULT_BYTE_VALUE_MSB_WORD_SELECT 0

- #define EZDP_DUAL_CTR_RESULT_BYTE_VALUE_MSB_WORD_OFFSET 0
- #define EZDP_DUAL_CTR_RESULT_RESERVED31_SIZE 1
- #define EZDP_DUAL_CTR_RESULT_RESERVED31_OFFSET 31
- #define EZDP_DUAL_CTR_RESULT_EVENT_VALUE_SIZE 28
- #define EZDP_DUAL_CTR_RESULT_EVENT_VALUE_OFFSET 32
- #define EZDP_DUAL_CTR_RESULT_EVENT_VALUE_WORD_SELECT 1
- #define EZDP_DUAL_CTR_RESULT_EVENT_VALUE_WORD_OFFSET 0
- #define EZDP_DUAL_CTR_RESULT_BYTE_VALUE_LSB_SIZE 4
- #define EZDP_DUAL_CTR_RESULT_BYTE_VALUE_LSB_OFFSET 60
- #define EZDP_DUAL_CTR_RESULT_BYTE_VALUE_LSB_WORD_SELECT 1
- #define EZDP_DUAL_CTR_RESULT_BYTE_VALUE_LSB_WORD_OFFSET 28
- #define EZDP_DUAL_CTR_RESULT_WORD_COUNT 2
- #define EZDP_DUAL_CTR_BYTE_SIZE 64
- #define EZDP_DUAL_CTR_BYTE_OFFSET 0
- #define EZDP_DUAL_CTR_BYTE_WORD_SELECT 0
- #define EZDP_DUAL_CTR_BYTE_WORD_OFFSET 0
- #define EZDP_DUAL_CTR_EVENT_SIZE 64
- #define EZDP_DUAL_CTR_EVENT_OFFSET 64
- #define EZDP_DUAL_CTR_EVENT_WORD_SELECT 2
- #define EZDP_DUAL_CTR_EVENT_WORD_OFFSET 0
- #define EZDP_DUAL_CTR_WORD_COUNT 4
- #define EZDP_DUAL_CTR_CFG_RESERVED0_SIZE 1
- #define EZDP_DUAL_CTR_CFG_RESERVED0_OFFSET 0
- #define EZDP_DUAL_CTR_CFG_BYTE_VALUE_SIZE_SIZE 4
- #define EZDP_DUAL_CTR_CFG_BYTE_VALUE_SIZE_OFFSET 1
- #define EZDP_DUAL_CTR_CFG_BYTE_VALUE_SIZE_WORD_SELECT 0
- #define EZDP_DUAL_CTR_CFG_BYTE_VALUE_SIZE_WORD_OFFSET 1
- #define EZDP_DUAL_CTR_CFG_EVENT_REPORT_EXCEEDED_SIZE 6
- #define EZDP_DUAL_CTR_CFG_EVENT_REPORT_EXCEEDED_OFFSET 5
- #define EZDP_DUAL_CTR_CFG_EVENT_REPORT_EXCEEDED_WORD_SELECT 0
- #define EZDP_DUAL_CTR_CFG_EVENT_REPORT_EXCEEDED_WORD_OFFSET 5
- #define EZDP_DUAL_CTR_CFG_ENABLE_EXCEED_MESSAGE_SIZE 1
- #define EZDP_DUAL_CTR_CFG_ENABLE_EXCEED_MESSAGE_OFFSET 11
- #define EZDP_DUAL_CTR_CFG_ENABLE_EXCEED_MESSAGE_WORD_SELECT 0
- #define EZDP_DUAL_CTR_CFG_ENABLE_EXCEED_MESSAGE_WORD_OFFSET 11
- #define EZDP_DUAL_CTR_CFG_ENABLE_EXCEED_MESSAGE_MASK (1 << EZDP_DUAL_CTR_CFG_ENABLE_EXCEED_MESSAGE_WORD_OFFSET)
- #define EZDP_DUAL_CTR_CFG_CLR_ON_GC_SIZE 1
- #define EZDP_DUAL_CTR_CFG_CLR_ON_GC_OFFSET 12
- #define EZDP_DUAL_CTR_CFG_BYTE_REPORT_EXCEEDED_SIZE 6
- #define EZDP_DUAL_CTR_CFG_BYTE_REPORT_EXCEEDED_OFFSET 13
- #define EZDP_DUAL_CTR_CFG_BYTE_REPORT_EXCEEDED_WORD_SELECT 0
- #define EZDP_DUAL_CTR_CFG_BYTE_REPORT_EXCEEDED_WORD_OFFSET 13
- #define EZDP_DUAL_CTR_CFG_RESERVED19_23_SIZE 5
- #define EZDP_DUAL_CTR_CFG_RESERVED19_23_OFFSET 19
- #define EZDP_DUAL_CTR_CFG_ECC_SIZE 8
- #define EZDP_DUAL_CTR_CFG_ECC_OFFSET 24
- #define EZDP_DUAL_CTR_CFG_VALUE_SIZE 128
- #define EZDP_DUAL_CTR_CFG_VALUE_OFFSET 32
- #define EZDP_DUAL_CTR_CFG_VALUE_WORD_SELECT 1
- #define EZDP_DUAL_CTR_CFG_VALUE_WORD_OFFSET 0
- #define EZDP_DUAL_CTR_CFG_WORD_COUNT 5
- #define EZDP_TB_CTR_RESULT_RESERVED0_31_SIZE 32
- #define EZDP_TB_CTR_RESULT_RESERVED0_31_OFFSET 0
- #define EZDP_TB_CTR_RESULT_COLOR_SIZE 2
- #define EZDP_TB_CTR_RESULT_COLOR_OFFSET 32
- #define EZDP_TB_CTR_RESULT_COLOR_WORD_SELECT 1

- #define EZDP_TB_CTR_RESULT_COLOR_WORD_OFFSET 0
- #define EZDP_TB_CTR_RESULT_RESERVED34_57_SIZE 22
- #define EZDP_TB_CTR_RESULT_RESERVED34_57_OFFSET 34
- #define EZDP_TB_CTR_RESULT_EMPTY_COMMIT_BUCKET_SIZE 1
- #define EZDP_TB_CTR_RESULT_EMPTY_COMMIT_BUCKET_OFFSET 56
- #define EZDP_TB_CTR_RESULT_EMPTY_COMMIT_BUCKET_WORD_SELECT 1
- #define EZDP_TB_CTR_RESULT_EMPTY_COMMIT_BUCKET_WORD_OFFSET 24
- #define EZDP_TB_CTR_RESULT_EMPTY_COMMIT_BUCKET_MASK (1 << EZDP_TB_CTR_RESULT_EMPTY_COMMIT_BUCKET_WORD_OFFSET)
- #define EZDP_TB_CTR_RESULT_EMPTY_EXCESS_BUCKET_SIZE 1
- #define EZDP_TB_CTR_RESULT_EMPTY_EXCESS_BUCKET_OFFSET 57
- #define EZDP_TB_CTR_RESULT_EMPTY_EXCESS_BUCKET_WORD_SELECT 1
- #define EZDP_TB_CTR_RESULT_EMPTY_EXCESS_BUCKET_WORD_OFFSET 25
- #define EZDP_TB_CTR_RESULT_EMPTY_EXCESS_BUCKET_MASK (1 << EZDP_TB_CTR_RESULT_EMPTY_EXCESS_BUCKET_WORD_OFFSET)
- #define EZDP_TB_CTR_RESULT_EMPTY_COMMIT_BUCKET_UG_SIZE 1
- #define EZDP_TB_CTR_RESULT_EMPTY_COMMIT_BUCKET_UG_OFFSET 58
- #define EZDP_TB_CTR_RESULT_EMPTY_COMMIT_BUCKET_UG_WORD_SELECT 1
- #define EZDP_TB_CTR_RESULT_EMPTY_COMMIT_BUCKET_UG_WORD_OFFSET 26
- #define EZDP_TB_CTR_RESULT_EMPTY_COMMIT_BUCKET_UG_MASK (1 << EZDP_TB_CTR_RESULT_EMPTY_COMMIT_BUCKET_UG_WORD_OFFSET)
- #define EZDP_TB_CTR_RESULT_EMPTY_EXCESS_BUCKET_UG_SIZE 1
- #define EZDP_TB_CTR_RESULT_EMPTY_EXCESS_BUCKET_UG_OFFSET 59
- #define EZDP_TB_CTR_RESULT_EMPTY_EXCESS_BUCKET_UG_WORD_SELECT 1
- #define EZDP_TB_CTR_RESULT_EMPTY_EXCESS_BUCKET_UG_WORD_OFFSET 27
- #define EZDP_TB_CTR_RESULT_EMPTY_EXCESS_BUCKET_UG_MASK (1 << EZDP_TB_CTR_RESULT_EMPTY_EXCESS_BUCKET_UG_WORD_OFFSET)
- #define EZDP_TB_CTR_RESULT_RESERVED60_63_SIZE 4
- #define EZDP_TB_CTR_RESULT_RESERVED60_63_OFFSET 60
- #define EZDP_TB_CTR_RESULT_WORD_COUNT 2
- #define EZDP_TB_CTR_CFG_COMMIT_PROFILE_ID_SIZE 10
- #define EZDP_TB_CTR_CFG_COMMIT_PROFILE_ID_OFFSET 0
- #define EZDP_TB_CTR_CFG_COMMIT_PROFILE_ID_WORD_SELECT 0
- #define EZDP_TB_CTR_CFG_COMMIT_PROFILE_ID_WORD_OFFSET 0
- #define EZDP_TB_CTR_CFG_EXCESS_PROFILE_ID_SIZE 10
- #define EZDP_TB_CTR_CFG_EXCESS_PROFILE_ID_OFFSET 10
- #define EZDP_TB_CTR_CFG_EXCESS_PROFILE_ID_WORD_SELECT 0
- #define EZDP_TB_CTR_CFG_EXCESS_PROFILE_ID_WORD_OFFSET 10
- #define EZDP_TB_CTR_CFG_ALGORITHM_TYPE_SIZE 4
- #define EZDP_TB_CTR_CFG_ALGORITHM_TYPE_OFFSET 20
- #define EZDP_TB_CTR_CFG_ALGORITHM_TYPE_WORD_SELECT 0
- #define EZDP_TB_CTR_CFG_ALGORITHM_TYPE_WORD_OFFSET 20
- #define EZDP_TB_CTR_CFG_COLOR_AWARE_SIZE 1
- #define EZDP_TB_CTR_CFG_COLOR_AWARE_OFFSET 24
- #define EZDP_TB_CTR_CFG_COLOR_AWARE_WORD_SELECT 0
- #define EZDP_TB_CTR_CFG_COLOR_AWARE_WORD_OFFSET 24
- #define EZDP_TB_CTR_CFG_COLOR_AWARE_MASK (1 << EZDP_TB_CTR_CFG_COLOR_AWARE_WORD_OFFSET)
- #define EZDP_TB_CTR_CFG_COUPLING_FLAG_SIZE 1
- #define EZDP_TB_CTR_CFG_COUPLING_FLAG_OFFSET 25
- #define EZDP_TB_CTR_CFG_COUPLING_FLAG_WORD_SELECT 0
- #define EZDP_TB_CTR_CFG_COUPLING_FLAG_WORD_OFFSET 25
- #define EZDP_TB_CTR_CFG_COUPLING_FLAG_MASK (1 << EZDP_TB_CTR_CFG_COUPLING_FLAG_WORD_OFFSET)
- #define EZDP_TB_CTR_CFG_RESERVED26_31_SIZE 6
- #define EZDP_TB_CTR_CFG_RESERVED26_31_OFFSET 26
- #define EZDP_TB_CTR_CFG_RESERVED32_63_SIZE 32

- #define EZDP_TB_CTR_CFG_RESERVED32_63_OFFSET 32
- #define EZDP_TB_CTR_CFG_RESERVED64_95_SIZE 32
- #define EZDP_TB_CTR_CFG_RESERVED64_95_OFFSET 64
- #define EZDP_TB_CTR_CFG_RESERVED96_127_SIZE 32
- #define EZDP_TB_CTR_CFG_RESERVED96_127_OFFSET 96
- #define EZDP_TB_CTR_CFG_WORD_COUNT 4
- #define EZDP_HIER_TB_UG_APP_BITS_COLOR_STATE_G_SIZE 2
- #define EZDP_HIER_TB_UG_APP_BITS_COLOR_STATE_G_OFFSET 0
- #define EZDP_HIER_TB_UG_APP_BITS_COLOR_STATE_Y_SIZE 2
- #define EZDP_HIER_TB_UG_APP_BITS_COLOR_STATE_Y_OFFSET 2
- #define EZDP_HIER_TB_UG_APP_BITS_EIGTH_MODE_RET_BITS_SIZE 4
- #define EZDP_HIER_TB_UG_APP_BITS_EIGTH_MODE_RET_BITS_OFFSET 4
- #define EZDP_HIER_TB_UG_APP_BITS_APP_BITS_SIZE 16
- #define EZDP_HIER_TB_UG_APP_BITS_APP_BITS_OFFSET 8
- #define EZDP_HIER_TB_UG_APP_BITS_RESERVED24_31_SIZE 8
- #define EZDP_HIER_TB_UG_APP_BITS_RESERVED24_31_OFFSET 24
- #define EZDP_HIER_TB_CTR_CFG_RESERVED0_1_SIZE 2
- #define EZDP_HIER_TB_CTR_CFG_RESERVED0_1_OFFSET 0
- #define EZDP_HIER_TB_CTR_CFG_CTR1_UPDT_THRESHOLD_SIZE 5
- #define EZDP_HIER_TB_CTR_CFG_CTR1_UPDT_THRESHOLD_OFFSET 2
- #define EZDP_HIER_TB_CTR_CFG_CTR1_UPDT_THRESHOLD_WORD_SELECT 0
- #define EZDP_HIER_TB_CTR_CFG_CTR1_UPDT_THRESHOLD_WORD_OFFSET 2
- #define EZDP_HIER_TB_CTR_CFG_CTR1_FAIL_THRESHOLD_SIZE 5
- #define EZDP_HIER_TB_CTR_CFG_CTR1_FAIL_THRESHOLD_OFFSET 7
- #define EZDP_HIER_TB_CTR_CFG_CTR1_FAIL_THRESHOLD_WORD_SELECT 0
- #define EZDP_HIER_TB_CTR_CFG_CTR1_FAIL_THRESHOLD_WORD_OFFSET 7
- #define EZDP_HIER_TB_CTR_CFG_CTR_SUM_UPDT_THRESHOLD_SIZE 5
- #define EZDP_HIER_TB_CTR_CFG_CTR_SUM_UPDT_THRESHOLD_OFFSET 12
- #define EZDP_HIER_TB_CTR_CFG_CTR_SUM_UPDT_THRESHOLD_WORD_SELECT 0
- #define EZDP_HIER_TB_CTR_CFG_CTR_SUM_UPDT_THRESHOLD_WORD_OFFSET 12
- #define EZDP_HIER_TB_CTR_CFG_CTR_SUM_FAIL_THRESHOLD_SIZE 5
- #define EZDP_HIER_TB_CTR_CFG_CTR_SUM_FAIL_THRESHOLD_OFFSET 17
- #define EZDP_HIER_TB_CTR_CFG_CTR_SUM_FAIL_THRESHOLD_WORD_SELECT 0
- #define EZDP_HIER_TB_CTR_CFG_CTR_SUM_FAIL_THRESHOLD_WORD_OFFSET 17
- #define EZDP_HIER_TB_CTR_CFG_RESERVED22_26_SIZE 5
- #define EZDP_HIER_TB_CTR_CFG_RESERVED22_26_OFFSET 22
- #define EZDP_HIER_TB_CTR_CFG_CTR0_FAIL_THRESHOLD_SIZE 5
- #define EZDP_HIER_TB_CTR_CFG_CTR0_FAIL_THRESHOLD_OFFSET 27
- #define EZDP_HIER_TB_CTR_CFG_CTR0_FAIL_THRESHOLD_WORD_SELECT 0
- #define EZDP_HIER_TB_CTR_CFG_CTR0_FAIL_THRESHOLD_WORD_OFFSET 27
- #define EZDP_HIER_TB_CTR_CFG_APP_BITS_SIZE 24
- #define EZDP_HIER_TB_CTR_CFG_APP_BITS_OFFSET 32
- #define EZDP_HIER_TB_CTR_CFG_APP_BITS_WORD_SELECT 1
- #define EZDP_HIER_TB_CTR_CFG_APP_BITS_WORD_OFFSET 0
- #define EZDP_HIER_TB_CTR_CFG_CTR0_UPDT_THRESHOLD_SIZE 5
- #define EZDP_HIER_TB_CTR_CFG_CTR0_UPDT_THRESHOLD_OFFSET 56
- #define EZDP_HIER_TB_CTR_CFG_CTR0_UPDT_THRESHOLD_WORD_SELECT 1
- #define EZDP_HIER_TB_CTR_CFG_CTR0_UPDT_THRESHOLD_WORD_OFFSET 24
- #define EZDP_HIER_TB_CTR_CFG_TIMESTAMP_THRESHOLD_SIZE 2
- #define EZDP_HIER_TB_CTR_CFG_TIMESTAMP_THRESHOLD_OFFSET 61
- #define EZDP_HIER_TB_CTR_CFG_TIMESTAMP_THRESHOLD_WORD_SELECT 1
- #define EZDP_HIER_TB_CTR_CFG_TIMESTAMP_THRESHOLD_WORD_OFFSET 29
- #define EZDP_HIER_TB_CTR_CFG_RESERVED63_SIZE 1
- #define EZDP_HIER_TB_CTR_CFG_RESERVED63_OFFSET 63
- #define EZDP_HIER_TB_CTR_CFG_WORD_COUNT 2
- #define EZDP_HIER_TB_RESULT_RESERVED0_9_SIZE 10
- #define EZDP_HIER_TB_RESULT_RESERVED0_9_OFFSET 0

- #define EZDP_HIER_TB_RESULT_CTR1_SIZE 18
- #define EZDP_HIER_TB_RESULT_CTR1_OFFSET 10
- #define EZDP_HIER_TB_RESULT_CTR1_WORD_SELECT 0
- #define EZDP_HIER_TB_RESULT_CTR1_WORD_OFFSET 10
- #define EZDP_HIER_TB_RESULT_STATE_SIZE 2
- #define EZDP_HIER_TB_RESULT_STATE_OFFSET 28
- #define EZDP_HIER_TB_RESULT_STATE_WORD_SELECT 0
- #define EZDP_HIER_TB_RESULT_STATE_WORD_OFFSET 28
- #define EZDP_HIER_TB_RESULT_UPDATE_TASK_SIZE 1
- #define EZDP_HIER_TB_RESULT_UPDATE_TASK_OFFSET 30
- #define EZDP_HIER_TB_RESULT_UPDATE_TASK_WORD_SELECT 0
- #define EZDP_HIER_TB_RESULT_UPDATE_TASK_WORD_OFFSET 30
- #define EZDP_HIER_TB_RESULT_UPDATE_TASK_MASK (1 << EZDP_HIER_TB_RESULT_UPDATE_TASK_WORD_OFFSET)
- #define EZDP_HIER_TB_RESULT_FAIL_SIZE 1
- #define EZDP_HIER_TB_RESULT_FAIL_OFFSET 31
- #define EZDP_HIER_TB_RESULT_FAIL_WORD_SELECT 0
- #define EZDP_HIER_TB_RESULT_FAIL_WORD_OFFSET 31
- #define EZDP_HIER_TB_RESULT_FAIL_MASK (1 << EZDP_HIER_TB_RESULT_FAIL_WORD_OFFSET)
- #define EZDP_HIER_TB_RESULT_APP_BITS_SIZE 24
- #define EZDP_HIER_TB_RESULT_APP_BITS_OFFSET 32
- #define EZDP_HIER_TB_RESULT_APP_BITS_WORD_SELECT 1
- #define EZDP_HIER_TB_RESULT_APP_BITS_WORD_OFFSET 0
- #define EZDP_HIER_TB_RESULT_RESERVED56_63_SIZE 8
- #define EZDP_HIER_TB_RESULT_RESERVED56_63_OFFSET 56
- #define EZDP_HIER_TB_RESULT_CTR0_SIZE 18
- #define EZDP_HIER_TB_RESULT_CTR0_OFFSET 64
- #define EZDP_HIER_TB_RESULT_CTR0_WORD_SELECT 2
- #define EZDP_HIER_TB_RESULT_CTR0_WORD_OFFSET 0
- #define EZDP_HIER_TB_RESULT_RESERVED82_95_SIZE 14
- #define EZDP_HIER_TB_RESULT_RESERVED82_95_OFFSET 82
- #define EZDP_HIER_TB_RESULT_WORD_COUNT 3
- #define EZDP_HIER_TB_UPDATE_APP_BITS_SIZE 24
- #define EZDP_HIER_TB_UPDATE_APP_BITS_OFFSET 0
- #define EZDP_HIER_TB_UPDATE_RESERVED24_27_SIZE 4
- #define EZDP_HIER_TB_UPDATE_RESERVED24_27_OFFSET 24
- #define EZDP_HIER_TB_UPDATE_COND_SET_ACTIVE_STATE_SIZE 1
- #define EZDP_HIER_TB_UPDATE_COND_SET_ACTIVE_STATE_OFFSET 28
- #define EZDP_HIER_TB_UPDATE_COND_SET_ACTIVE_STATE_MASK (1 << EZDP_HIER_TB_UPDATE_COND_SET_ACTIVE_STATE_OFFSET)
- #define EZDP_HIER_TB_UPDATE_SET_APP_BITS_SIZE 1
- #define EZDP_HIER_TB_UPDATE_SET_APP_BITS_OFFSET 29
- #define EZDP_HIER_TB_UPDATE_SET_APP_BITS_MASK (1 << EZDP_HIER_TB_UPDATE_SET_APP_BITS_OFFSET)
- #define EZDP_HIER_TB_UPDATE_CLR_CTR_SIZE 1
- #define EZDP_HIER_TB_UPDATE_CLR_CTR_OFFSET 30
- #define EZDP_HIER_TB_UPDATE_CLR_CTR_MASK (1 << EZDP_HIER_TB_UPDATE_CLR_CTR_OFFSET)
- #define EZDP_HIER_TB_UPDATE_SET_ACTIVE_STATE_SIZE 1
- #define EZDP_HIER_TB_UPDATE_SET_ACTIVE_STATE_OFFSET 31
- #define EZDP_HIER_TB_UPDATE_SET_ACTIVE_STATE_MASK (1 << EZDP_HIER_TB_UPDATE_SET_ACTIVE_STATE_OFFSET)
- #define EZDP_BITWISE_CTR_CFG_RESERVED0_18_SIZE 19
- #define EZDP_BITWISE_CTR_CFG_RESERVED0_18_OFFSET 0
- #define EZDP_BITWISE_CTR_CFG_SUB_TYPE_SIZE 5
- #define EZDP_BITWISE_CTR_CFG_SUB_TYPE_OFFSET 19

- #define EZDP_BITWISE_CTR_CFG_ECC_SIZE 8
- #define EZDP_BITWISE_CTR_CFG_ECC_OFFSET 24
- #define EZDP_BITWISE_CTR_CFG_RESERVED32_63_SIZE 32
- #define EZDP_BITWISE_CTR_CFG_RESERVED32_63_OFFSET 32
- #define EZDP_BITWISE_CTR_CFG_DATA_SIZE 64
- #define EZDP_BITWISE_CTR_CFG_DATA_OFFSET 64
- #define EZDP_BITWISE_CTR_CFG_DATA_WORD_SELECT 2
- #define EZDP_BITWISE_CTR_CFG_DATA_WORD_OFFSET 0
- #define EZDP_BITWISE_CTR_CFG_WORD_COUNT 4
- #define EZDP_WATCHDOG_ACCUMULATIVE_WINDOW_CFG_ACCUMULATIVE_EVENTS_SIZE 5
- #define EZDP_WATCHDOG_ACCUMULATIVE_WINDOW_CFG_ACCUMULATIVE_EVENTS_OFFSET 0
- #define EZDP_WATCHDOG_ACCUMULATIVE_WINDOW_CFG_ACCUMULATIVE_EVENTS_WORD_SELECT 0
- #define EZDP_WATCHDOG_ACCUMULATIVE_WINDOW_CFG_ACCUMULATIVE_EVENTS_WORD_OFFSET 0
- #define EZDP_WATCHDOG_ACCUMULATIVE_WINDOW_CFG_RESERVED5_28_SIZE 24
- #define EZDP_WATCHDOG_ACCUMULATIVE_WINDOW_CFG_RESERVED5_28_OFFSET 5
- #define EZDP_WATCHDOG_ACCUMULATIVE_WINDOW_CFG_MAX_THRESHOLD_ALERT_SIZE 1
- #define EZDP_WATCHDOG_ACCUMULATIVE_WINDOW_CFG_MAX_THRESHOLD_ALERT_OFFSET 29
- #define EZDP_WATCHDOG_ACCUMULATIVE_WINDOW_CFG_MIN_THRESHOLD_ALERT_SIZE 1
- #define EZDP_WATCHDOG_ACCUMULATIVE_WINDOW_CFG_MIN_THRESHOLD_ALERT_OFFSET 30
- #define EZDP_WATCHDOG_ACCUMULATIVE_WINDOW_CFG_ALERT_SIZE 1
- #define EZDP_WATCHDOG_ACCUMULATIVE_WINDOW_CFG_ALERT_OFFSET 31
- #define EZDP_WATCHDOG_ACCUMULATIVE_WINDOW_CFG_LAST_EVENTS_SIZE 12
- #define EZDP_WATCHDOG_ACCUMULATIVE_WINDOW_CFG_LAST_EVENTS_OFFSET 32
- #define EZDP_WATCHDOG_ACCUMULATIVE_WINDOW_CFG_LAST_EVENTS_WORD_SELECT 1
- #define EZDP_WATCHDOG_ACCUMULATIVE_WINDOW_CFG_LAST_EVENTS_WORD_OFFSET 0
- #define EZDP_WATCHDOG_ACCUMULATIVE_WINDOW_CFG_CURR_EVENTS_SIZE 12
- #define EZDP_WATCHDOG_ACCUMULATIVE_WINDOW_CFG_CURR_EVENTS_OFFSET 44
- #define EZDP_WATCHDOG_ACCUMULATIVE_WINDOW_CFG_CURR_EVENTS_WORD_SELECT 1
- #define EZDP_WATCHDOG_ACCUMULATIVE_WINDOW_CFG_CURR_EVENTS_WORD_OFFSET 12
- #define EZDP_WATCHDOG_ACCUMULATIVE_WINDOW_CFG_INIT_BIT_SIZE 1
- #define EZDP_WATCHDOG_ACCUMULATIVE_WINDOW_CFG_INIT_BIT_OFFSET 56
- #define EZDP_WATCHDOG_ACCUMULATIVE_WINDOW_CFG_VALID_SIZE 1
- #define EZDP_WATCHDOG_ACCUMULATIVE_WINDOW_CFG_VALID_OFFSET 57
- #define EZDP_WATCHDOG_ACCUMULATIVE_WINDOW_CFG_VALID_WORD_SELECT 1
- #define EZDP_WATCHDOG_ACCUMULATIVE_WINDOW_CFG_VALID_WORD_OFFSET 25
- #define EZDP_WATCHDOG_ACCUMULATIVE_WINDOW_CFG_VALID_MASK (1 << EZDP_WATCHDOG_ACCUMULATIVE_WINDOW_CFG_VALID_WORD_OFFSET)
- #define EZDP_WATCHDOG_ACCUMULATIVE_WINDOW_CFG_PROFILE_ID_SIZE 4
- #define EZDP_WATCHDOG_ACCUMULATIVE_WINDOW_CFG_PROFILE_ID_OFFSET 58
- #define EZDP_WATCHDOG_ACCUMULATIVE_WINDOW_CFG_PROFILE_ID_WORD_SELECT 1
- #define EZDP_WATCHDOG_ACCUMULATIVE_WINDOW_CFG_PROFILE_ID_WORD_OFFSET 26
- #define EZDP_WATCHDOG_ACCUMULATIVE_WINDOW_CFG_PARITY_SIZE 1
- #define EZDP_WATCHDOG_ACCUMULATIVE_WINDOW_CFG_PARITY_OFFSET 62
- #define EZDP_WATCHDOG_ACCUMULATIVE_WINDOW_CFG_RESERVED63_SIZE 1
- #define EZDP_WATCHDOG_ACCUMULATIVE_WINDOW_CFG_RESERVED63_OFFSET 63
- #define EZDP_WATCHDOG_ACCUMULATIVE_WINDOW_CFG_WORD_COUNT 2
- #define EZDP_WATCHDOG_SLIDING_WINDOW_CFG_VALID_WINDOWS_SIZE 5
- #define EZDP_WATCHDOG_SLIDING_WINDOW_CFG_VALID_WINDOWS_OFFSET 0
- #define EZDP_WATCHDOG_SLIDING_WINDOW_CFG_VALID_WINDOWS_WORD_SELECT 0
- #define EZDP_WATCHDOG_SLIDING_WINDOW_CFG_VALID_WINDOWS_WORD_OFFSET 0

- #define EZDP_WATCHDOG_SLIDING_WINDOW_CFG_RESERVED5_28_SIZE 24
- #define EZDP_WATCHDOG_SLIDING_WINDOW_CFG_RESERVED5_28_OFFSET 5
- #define EZDP_WATCHDOG_SLIDING_WINDOW_CFG_MAX_THRESHOLD_ALERT_SIZE 1
- #define EZDP_WATCHDOG_SLIDING_WINDOW_CFG_MAX_THRESHOLD_ALERT_OFFSET 29
- #define EZDP_WATCHDOG_SLIDING_WINDOW_CFG_MIN_THRESHOLD_ALERT_SIZE 1
- #define EZDP_WATCHDOG_SLIDING_WINDOW_CFG_MIN_THRESHOLD_ALERT_OFFSET 30
- #define EZDP_WATCHDOG_SLIDING_WINDOW_CFG_ALERT_SIZE 1
- #define EZDP_WATCHDOG_SLIDING_WINDOW_CFG_ALERT_OFFSET 31
- #define EZDP_WATCHDOG_SLIDING_WINDOW_CFG_COUNTERS_SIZE 24
- #define EZDP_WATCHDOG_SLIDING_WINDOW_CFG_COUNTERS_OFFSET 32
- #define EZDP_WATCHDOG_SLIDING_WINDOW_CFG_COUNTERS_WORD_SELECT 1
- #define EZDP_WATCHDOG_SLIDING_WINDOW_CFG_COUNTERS_WORD_OFFSET 0
- #define EZDP_WATCHDOG_SLIDING_WINDOW_CFG_RESERVED56_SIZE 1
- #define EZDP_WATCHDOG_SLIDING_WINDOW_CFG_RESERVED56_OFFSET 56
- #define EZDP_WATCHDOG_SLIDING_WINDOW_CFG_VALID_SIZE 1
- #define EZDP_WATCHDOG_SLIDING_WINDOW_CFG_VALID_OFFSET 57
- #define EZDP_WATCHDOG_SLIDING_WINDOW_CFG_VALID_WORD_SELECT 1
- #define EZDP_WATCHDOG_SLIDING_WINDOW_CFG_VALID_WORD_OFFSET 25
- #define EZDP_WATCHDOG_SLIDING_WINDOW_CFG_VALID_MASK (1 << EZDP_WATCHDOG_SLIDING_WINDOW_CFG_VALID_WORD_OFFSET)
- #define EZDP_WATCHDOG_SLIDING_WINDOW_CFG_PROFILE_ID_SIZE 4
- #define EZDP_WATCHDOG_SLIDING_WINDOW_CFG_PROFILE_ID_OFFSET 58
- #define EZDP_WATCHDOG_SLIDING_WINDOW_CFG_PROFILE_ID_WORD_SELECT 1
- #define EZDP_WATCHDOG_SLIDING_WINDOW_CFG_PROFILE_ID_WORD_OFFSET 26
- #define EZDP_WATCHDOG_SLIDING_WINDOW_CFG_PARITY_SIZE 1
- #define EZDP_WATCHDOG_SLIDING_WINDOW_CFG_PARITY_OFFSET 62
- #define EZDP_WATCHDOG_SLIDING_WINDOW_CFG_RESERVED63_SIZE 1
- #define EZDP_WATCHDOG_SLIDING_WINDOW_CFG_RESERVED63_OFFSET 63
- #define EZDP_WATCHDOG_SLIDING_WINDOW_CFG_WORD_COUNT 2
- #define EZDP_WATCHDOG_CTR_CFG_RESERVED0_18_SIZE 19
- #define EZDP_WATCHDOG_CTR_CFG_RESERVED0_18_OFFSET 0
- #define EZDP_WATCHDOG_CTR_CFG_SUB_TYPE_SIZE 5
- #define EZDP_WATCHDOG_CTR_CFG_SUB_TYPE_OFFSET 19
- #define EZDP_WATCHDOG_CTR_CFG_ECC_SIZE 8
- #define EZDP_WATCHDOG_CTR_CFG_ECC_OFFSET 24
- #define EZDP_WATCHDOG_CTR_CFG_RESERVED32_63_SIZE 32
- #define EZDP_WATCHDOG_CTR_CFG_RESERVED32_63_OFFSET 32
- #define EZDP_WATCHDOG_CTR_CFG_WORD_COUNT 4
- #define EZDP_WATCHDOG_CTR_CHECK_RESULT_ACCUMULATIVE_EVENTS_SIZE 4
- #define EZDP_WATCHDOG_CTR_CHECK_RESULT_ACCUMULATIVE_EVENTS_OFFSET 0
- #define EZDP_WATCHDOG_CTR_CHECK_RESULT_RESERVED4_28_SIZE 25
- #define EZDP_WATCHDOG_CTR_CHECK_RESULT_RESERVED4_28_OFFSET 4
- #define EZDP_WATCHDOG_CTR_CHECK_RESULT_MAX_THRESHOLD_ALERT_SIZE 1
- #define EZDP_WATCHDOG_CTR_CHECK_RESULT_MAX_THRESHOLD_ALERT_OFFSET 29
- #define EZDP_WATCHDOG_CTR_CHECK_RESULT_MAX_THRESHOLD_ALERT_WORD_SELECT 0
- #define EZDP_WATCHDOG_CTR_CHECK_RESULT_MAX_THRESHOLD_ALERT_WORD_OFFSET 29
- #define EZDP_WATCHDOG_CTR_CHECK_RESULT_MAX_THRESHOLD_ALERT_MASK (1 << EZDP_WATCHDOG_CTR_CHECK_RESULT_MAX_THRESHOLD_ALERT_WORD_OFFSET)
- #define EZDP_WATCHDOG_CTR_CHECK_RESULT_MIN_THRESHOLD_ALERT_SIZE 1
- #define EZDP_WATCHDOG_CTR_CHECK_RESULT_MIN_THRESHOLD_ALERT_OFFSET 30
- #define EZDP_WATCHDOG_CTR_CHECK_RESULT_MIN_THRESHOLD_ALERT_WORD_SELECT 0
- #define EZDP_WATCHDOG_CTR_CHECK_RESULT_MIN_THRESHOLD_ALERT_WORD_OFFSET 30
- #define EZDP_WATCHDOG_CTR_CHECK_RESULT_MIN_THRESHOLD_ALERT_MASK (1 << EZDP_WATCHDOG_CTR_CHECK_RESULT_MIN_THRESHOLD_ALERT_WORD_OFFSET)
- #define EZDP_WATCHDOG_CTR_CHECK_RESULT_ALERT_SIZE 1
- #define EZDP_WATCHDOG_CTR_CHECK_RESULT_ALERT_OFFSET 31

- #define EZDP_WATCHDOG_CTR_CHECK_RESULT_ALERT_WORD_SELECT 0
- #define EZDP_WATCHDOG_CTR_CHECK_RESULT_ALERT_WORD_OFFSET 31
- #define EZDP_WATCHDOG_CTR_CHECK_RESULT_ALERT_MASK (1 << EZDP_WATCHDOG_CTR_CHECK_RESULT_ALERT_WORD_OFFSET)
- #define EZDP_WATCHDOG_CTR_CHECK_RESULT_LAST_EVENTS_SIZE 12
- #define EZDP_WATCHDOG_CTR_CHECK_RESULT_LAST_EVENTS_OFFSET 32
- #define EZDP_WATCHDOG_CTR_CHECK_RESULT_CURR_EVENTS_SIZE 12
- #define EZDP_WATCHDOG_CTR_CHECK_RESULT_CURR_EVENTS_OFFSET 44
- #define EZDP_WATCHDOG_CTR_CHECK_RESULT_INIT_BIT_SIZE 1
- #define EZDP_WATCHDOG_CTR_CHECK_RESULT_INIT_BIT_OFFSET 56
- #define EZDP_WATCHDOG_CTR_CHECK_RESULT_VALID_SIZE 1
- #define EZDP_WATCHDOG_CTR_CHECK_RESULT_VALID_OFFSET 57
- #define EZDP_WATCHDOG_CTR_CHECK_RESULT_PROFILE_ID_SIZE 4
- #define EZDP_WATCHDOG_CTR_CHECK_RESULT_PROFILE_ID_OFFSET 58
- #define EZDP_WATCHDOG_CTR_CHECK_RESULT_RESERVED62_SIZE 1
- #define EZDP_WATCHDOG_CTR_CHECK_RESULT_RESERVED62_OFFSET 62
- #define EZDP_WATCHDOG_CTR_CHECK_RESULT_WINDOW_RELATED_SIZE 1
- #define EZDP_WATCHDOG_CTR_CHECK_RESULT_WINDOW_RELATED_OFFSET 63
- #define EZDP_WATCHDOG_CTR_CHECK_RESULT_WORD_COUNT 2
- #define EZDP_WATCHDOG_CTR_START_RESULT_MSB_SIZE 32
- #define EZDP_WATCHDOG_CTR_START_RESULT_MSB_OFFSET 0
- #define EZDP_WATCHDOG_CTR_START_RESULT_MSB_WORD_SELECT 0
- #define EZDP_WATCHDOG_CTR_START_RESULT_MSB_WORD_OFFSET 0
- #define EZDP_WATCHDOG_CTR_START_RESULT_LSB_SIZE 32
- #define EZDP_WATCHDOG_CTR_START_RESULT_LSB_OFFSET 32
- #define EZDP_WATCHDOG_CTR_START_RESULT_LSB_WORD_SELECT 1
- #define EZDP_WATCHDOG_CTR_START_RESULT_LSB_WORD_OFFSET 0
- #define EZDP_WATCHDOG_CTR_START_RESULT_WORD_COUNT 2
- #define EZDP_CTR_MSG_COUNTER_TYPE_SIZE 2
- #define EZDP_CTR_MSG_COUNTER_TYPE_OFFSET 0
- #define EZDP_CTR_MSG_COUNTER_TYPE_WORD_SELECT 0
- #define EZDP_CTR_MSG_COUNTER_TYPE_WORD_OFFSET 0
- #define EZDP_CTR_MSG_MSG_TYPE_SIZE 3
- #define EZDP_CTR_MSG_MSG_TYPE_OFFSET 2
- #define EZDP_CTR_MSG_MSG_TYPE_WORD_SELECT 0
- #define EZDP_CTR_MSG_MSG_TYPE_WORD_OFFSET 2
- #define EZDP_CTR_MSG_OVERFLOW_SIZE 1
- #define EZDP_CTR_MSG_OVERFLOW_OFFSET 5
- #define EZDP_CTR_MSG_OVERFLOW_WORD_SELECT 0
- #define EZDP_CTR_MSG_OVERFLOW_WORD_OFFSET 5
- #define EZDP_CTR_MSG_OVERFLOW_MASK (1 << EZDP_CTR_MSG_OVERFLOW_WORD_OFFSET)
- #define EZDP_CTR_MSG_RESERVED6_SIZE 1
- #define EZDP_CTR_MSG_RESERVED6_OFFSET 6
- #define EZDP_CTR_MSG_OVERRUN_ERROR_CONDITION_SIZE 1
- #define EZDP_CTR_MSG_OVERRUN_ERROR_CONDITION_OFFSET 7
- #define EZDP_CTR_MSG_OVERRUN_ERROR_CONDITION_WORD_SELECT 0
- #define EZDP_CTR_MSG_OVERRUN_ERROR_CONDITION_WORD_OFFSET 7
- #define EZDP_CTR_MSG_OVERRUN_ERROR_CONDITION_MASK (1 << EZDP_CTR_MSG_OVERRUN_ERROR_CONDITION_WORD_OFFSET)
- #define EZDP_CTR_MSG_RESERVED8_23_SIZE 16
- #define EZDP_CTR_MSG_RESERVED8_23_OFFSET 8
- #define EZDP_CTR_MSG_ECC_SIZE 8
- #define EZDP_CTR_MSG_ECC_OFFSET 24
- #define EZDP_CTR_MSG_SUM_ADDR_SIZE 32
- #define EZDP_CTR_MSG_SUM_ADDR_OFFSET 32
- #define EZDP_CTR_MSG_SUM_ADDR_WORD_SELECT 1

- #define EZDP_CTR_MSG_SUM_ADDR_WORD_OFFSET 0
- #define EZDP_CTR_MSG_WORD_COUNT 6
- #define EZDP_POSTED_CTR_MSG_MSG_TYPE_SIZE 3
- #define EZDP_POSTED_CTR_MSG_MSG_TYPE_OFFSET 0
- #define EZDP_POSTED_CTR_MSG_MSG_TYPE_WORD_SELECT 0
- #define EZDP_POSTED_CTR_MSG_MSG_TYPE_WORD_OFFSET 0
- #define EZDP_POSTED_CTR_MSG_FLUSH_SIZE 1
- #define EZDP_POSTED_CTR_MSG_FLUSH_OFFSET 3
- #define EZDP_POSTED_CTR_MSG_FLUSH_WORD_SELECT 0
- #define EZDP_POSTED_CTR_MSG_FLUSH_WORD_OFFSET 3
- #define EZDP_POSTED_CTR_MSG_FLUSH_MASK (1 << EZDP_POSTED_CTR_MSG_FLUSH_WORD_OFFSET)
- #define EZDP_POSTED_CTR_MSG_CLEAR_SIZE 1
- #define EZDP_POSTED_CTR_MSG_CLEAR_OFFSET 4
- #define EZDP_POSTED_CTR_MSG_CLEAR_WORD_SELECT 0
- #define EZDP_POSTED_CTR_MSG_CLEAR_WORD_OFFSET 4
- #define EZDP_POSTED_CTR_MSG_CLEAR_MASK (1 << EZDP_POSTED_CTR_MSG_CLEAR_WORD_OFFSET)
- #define EZDP_POSTED_CTR_MSG_RESERVED5_6_SIZE 2
- #define EZDP_POSTED_CTR_MSG_RESERVED5_6_OFFSET 5
- #define EZDP_POSTED_CTR_MSG_OVERRUN_ERROR_CONDITION_SIZE 1
- #define EZDP_POSTED_CTR_MSG_OVERRUN_ERROR_CONDITION_OFFSET 7
- #define EZDP_POSTED_CTR_MSG_OVERRUN_ERROR_CONDITION_WORD_SELECT 0
- #define EZDP_POSTED_CTR_MSG_OVERRUN_ERROR_CONDITION_WORD_OFFSET 7
- #define EZDP_POSTED_CTR_MSG_OVERRUN_ERROR_CONDITION_MASK (1 << EZDP_POSTED_CTR_MSG_OVERRUN_ERROR_CONDITION_WORD_OFFSET)
- #define EZDP_POSTED_CTR_MSG_RESERVED8_23_SIZE 16
- #define EZDP_POSTED_CTR_MSG_RESERVED8_23_OFFSET 8
- #define EZDP_POSTED_CTR_MSG_ECC_SIZE 8
- #define EZDP_POSTED_CTR_MSG_ECC_OFFSET 24
- #define EZDP_POSTED_CTR_MSG_SUM_ADDR_SIZE 32
- #define EZDP_POSTED_CTR_MSG_SUM_ADDR_OFFSET 32
- #define EZDP_POSTED_CTR_MSG_SUM_ADDR_WORD_SELECT 1
- #define EZDP_POSTED_CTR_MSG_SUM_ADDR_WORD_OFFSET 0
- #define EZDP_POSTED_CTR_MSG_VALUE_SIZE 64
- #define EZDP_POSTED_CTR_MSG_VALUE_OFFSET 64
- #define EZDP_POSTED_CTR_MSG_VALUE_WORD_SELECT 2
- #define EZDP_POSTED_CTR_MSG_VALUE_WORD_OFFSET 0
- #define EZDP_POSTED_CTR_MSG_WORD_COUNT 4
- #define EZDP_CTR_MSG_WORK_AREA_SIZE ((sizeof(struct ezdp_stat_msg_int) > sizeof(struct ezdp_double_ctr_int)/2) ? sizeof(struct ezdp_stat_msg_int) : sizeof(struct ezdp_double_ctr_int)/2)
- #define EZDP_POSTED_CTR_MSG_WORK_AREA_SIZE sizeof(struct ezdp_stat_msg_int)

Typedefs

- typedef uint32_t ezdp_hier_tb_ug_app_bits_t
- typedef uint32_t ezdp_hier_tb_update_t
- typedef struct ezdp_get_color_hier_tb_ctr_working_area ezdp_get_color_hier_tb_ctr_working_area_t
- typedef uint32_t(* EZDP_HIER_TB_UPDATE_BES)(uint32_t app_bits, uint32_t ctr_g, uint32_t ctr_y, ezdp_get_color_hier_tb_ctr_working_area_t __cmem *wa_ptr, void *user_data)
- *Type definition for the data plane application's updating BE TBs according to algorithm and user information.* typedef struct ezdp_msgq_desc ezdp_ctr_msg_queue_desc_t
- *Statistic message queue descriptor.* typedef struct ezdp_msgq_desc ezdp_posted_ctr_msg_queue_desc_t

Posted message queue descriptor. Enumerations

- enum ezdp_tb_color { EZDP_RED_TRAFFIC = 0x0, EZDP_YELLOW_TRAFFIC = 0x1, EZDP_GREEN_TRAFFIC = 0x2 }

tb color possible values.

- enum ezdp_tb_algo { EZDP_INACTIVE = 0x0, EZDP_SINGLE_BUCKET = 0x1, EZDP_SR_TCM = 0x4, EZDP_TR_TCM = 5, EZDP_TR_TCM_MEF = 0x8 }
  tb algo possible values.
- enum ezdp_hier_tb_state { EZDP_UG_FAST_PATH = 0x0, EZDP_UG_SLOW_PATH = 0x3 }
  hier tb state possible values.
- enum ezdp_ctr_type { EZDP_SINGLE_CTR = 0x0, EZDP_DUAL_CTR = 0x1 }
  ctr type possible values.
- enum ezdp_ctr_msg_type { EZDP_PERIODIC_CTR_MSG = 0x0, EZDP_THRESHOLD_CTR_MSG = 0x1, EZDP_NULL_CTR_MSG = 0x3 }
  ctr msg type possible values.
- enum ezdp_msg_posted_ctr_type { EZDP_PERIODIC_POSTED_CTR_MSG = 0x0, EZDP_THRESHOLD_POSTED_CTR_MSG = 0x1, EZDP_REPORT_POSTED_CTR_MSG = 0x2, EZDP_NULL_POSTED_CTR_MSG = 0x3 }
  msg posted ctr type possible values.
- enum ezdp_bitwise_size { EZDP_1_BITS = EZASM_BITWISE_SIZE_1_BIT, EZDP_2_BITS = EZASM_BITWISE_SIZE_2_BITS, EZDP_4_BITS = EZASM_BITWISE_SIZE_4_BITS, EZDP_8_BITS = EZASM_BITWISE_SIZE_8_BITS, EZDP_16_BITS = EZASM_BITWISE_SIZE_16_BITS }
  bitwise size possible values.

Variables

- enum ezdp_tb_color(* EZDP_HIER_TB_CALC_COLOR)(uint32_t app_bits, enum ezdp_tb_color pre_color, uint32_t *user_data)

User Call Back function calc color according to algorithm and user information.

Define Documentation

#define EZDP_CTR_MSG_QUEUE_WORK_AREA_SIZE sizeof(struct ezdp_init_msgq_desc_working_area)

Work area minimal required size definitions.

#define EZDP_COUNTER_VERSION_MAJOR 2

#define EZDP_COUNTER_VERSION_MINOR 1

#define EZDP_SINGLE_CTR_CFG_RESERVED0_10_SIZE 11

#define EZDP_SINGLE_CTR_CFG_RESERVED0_10_OFFSET 0

#define EZDP_SINGLE_CTR_CFG_ENABLE_EXCEED_MESSAGE_SIZE 1

#define EZDP_SINGLE_CTR_CFG_ENABLE_EXCEED_MESSAGE_OFFSET 11

#define EZDP_SINGLE_CTR_CFG_ENABLE_EXCEED_MESSAGE_WORD_SELECT 0

#define EZDP_SINGLE_CTR_CFG_ENABLE_EXCEED_MESSAGE_WORD_OFFSET 11

#define EZDP_SINGLE_CTR_CFG_ENABLE_EXCEED_MESSAGE_MASK (1 << EZDP_SINGLE_CTR_CFG_ENABLE_EXCEED_MESSAGE_WORD_OFFSET)

#define EZDP_SINGLE_CTR_CFG_ZERO_SIZE 1

#define EZDP_SINGLE_CTR_CFG_ZERO_OFFSET 12

#define EZDP_SINGLE_CTR_CFG_REPORT_EXCEEDED_SIZE 6

#define EZDP_SINGLE_CTR_CFG_REPORT_EXCEEDED_OFFSET 13

#define EZDP_SINGLE_CTR_CFG_REPORT_EXCEEDED_WORD_SELECT 0

#define EZDP_SINGLE_CTR_CFG_REPORT_EXCEEDED_WORD_OFFSET 13

#define EZDP_SINGLE_CTR_CFG_SUB_TYPE_SIZE 5

#define EZDP_SINGLE_CTR_CFG_SUB_TYPE_OFFSET 19

#define EZDP_SINGLE_CTR_CFG_ECC_SIZE 8

#define EZDP_SINGLE_CTR_CFG_ECC_OFFSET 24

#define EZDP_SINGLE_CTR_CFG_RESERVED32_63_SIZE 32

#define EZDP_SINGLE_CTR_CFG_RESERVED32_63_OFFSET 32

#define EZDP_SINGLE_CTR_CFG_VALUE_SIZE 64

#define EZDP_SINGLE_CTR_CFG_VALUE_OFFSET 64

#define EZDP_SINGLE_CTR_CFG_VALUE_WORD_SELECT 2

#define EZDP_SINGLE_CTR_CFG_VALUE_WORD_OFFSET 0

#define EZDP_SINGLE_CTR_CFG_WORD_COUNT 4

```
#define EZDP_DUAL_CTR_RESULT_BYTE_VALUE_MSB_SIZE 31

#define EZDP_DUAL_CTR_RESULT_BYTE_VALUE_MSB_OFFSET 0

#define EZDP_DUAL_CTR_RESULT_BYTE_VALUE_MSB_WORD_SELECT 0

#define EZDP_DUAL_CTR_RESULT_BYTE_VALUE_MSB_WORD_OFFSET 0

#define EZDP_DUAL_CTR_RESULT_RESERVED31_SIZE 1

#define EZDP_DUAL_CTR_RESULT_RESERVED31_OFFSET 31

#define EZDP_DUAL_CTR_RESULT_EVENT_VALUE_SIZE 28

#define EZDP_DUAL_CTR_RESULT_EVENT_VALUE_OFFSET 32

#define EZDP_DUAL_CTR_RESULT_EVENT_VALUE_WORD_SELECT 1

#define EZDP_DUAL_CTR_RESULT_EVENT_VALUE_WORD_OFFSET 0

#define EZDP_DUAL_CTR_RESULT_BYTE_VALUE_LSB_SIZE 4

#define EZDP_DUAL_CTR_RESULT_BYTE_VALUE_LSB_OFFSET 60

#define EZDP_DUAL_CTR_RESULT_BYTE_VALUE_LSB_WORD_SELECT 1

#define EZDP_DUAL_CTR_RESULT_BYTE_VALUE_LSB_WORD_OFFSET 28

#define EZDP_DUAL_CTR_RESULT_WORD_COUNT 2

#define EZDP_DUAL_CTR_BYTE_SIZE 64

#define EZDP_DUAL_CTR_BYTE_OFFSET 0

#define EZDP_DUAL_CTR_BYTE_WORD_SELECT 0

#define EZDP_DUAL_CTR_BYTE_WORD_OFFSET 0

#define EZDP_DUAL_CTR_EVENT_SIZE 64

#define EZDP_DUAL_CTR_EVENT_OFFSET 64

#define EZDP_DUAL_CTR_EVENT_WORD_SELECT 2

#define EZDP_DUAL_CTR_EVENT_WORD_OFFSET 0

#define EZDP_DUAL_CTR_WORD_COUNT 4

#define EZDP_DUAL_CTR_CFG_RESERVED0_SIZE 1

#define EZDP_DUAL_CTR_CFG_RESERVED0_OFFSET 0
```

```
#define EZDP_DUAL_CTR_CFG_BYTE_VALUE_SIZE_SIZE 4

#define EZDP_DUAL_CTR_CFG_BYTE_VALUE_SIZE_OFFSET 1

#define EZDP_DUAL_CTR_CFG_BYTE_VALUE_SIZE_WORD_SELECT 0

#define EZDP_DUAL_CTR_CFG_BYTE_VALUE_SIZE_WORD_OFFSET 1

#define EZDP_DUAL_CTR_CFG_EVENT_REPORT_EXCEEDED_SIZE 6

#define EZDP_DUAL_CTR_CFG_EVENT_REPORT_EXCEEDED_OFFSET 5

#define EZDP_DUAL_CTR_CFG_EVENT_REPORT_EXCEEDED_WORD_SELECT 0

#define EZDP_DUAL_CTR_CFG_EVENT_REPORT_EXCEEDED_WORD_OFFSET 5

#define EZDP_DUAL_CTR_CFG_ENABLE_EXCEED_MESSAGE_SIZE 1

#define EZDP_DUAL_CTR_CFG_ENABLE_EXCEED_MESSAGE_OFFSET 11

#define EZDP_DUAL_CTR_CFG_ENABLE_EXCEED_MESSAGE_WORD_SELECT 0

#define EZDP_DUAL_CTR_CFG_ENABLE_EXCEED_MESSAGE_WORD_OFFSET 11

#define EZDP_DUAL_CTR_CFG_ENABLE_EXCEED_MESSAGE_MASK (1 <<
EZDP_DUAL_CTR_CFG_ENABLE_EXCEED_MESSAGE_WORD_OFFSET)

#define EZDP_DUAL_CTR_CFG_CLR_ON_GC_SIZE 1

#define EZDP_DUAL_CTR_CFG_CLR_ON_GC_OFFSET 12

#define EZDP_DUAL_CTR_CFG_BYTE_REPORT_EXCEEDED_SIZE 6

#define EZDP_DUAL_CTR_CFG_BYTE_REPORT_EXCEEDED_OFFSET 13

#define EZDP_DUAL_CTR_CFG_BYTE_REPORT_EXCEEDED_WORD_SELECT 0

#define EZDP_DUAL_CTR_CFG_BYTE_REPORT_EXCEEDED_WORD_OFFSET 13

#define EZDP_DUAL_CTR_CFG_RESERVED19_23_SIZE 5

#define EZDP_DUAL_CTR_CFG_RESERVED19_23_OFFSET 19

#define EZDP_DUAL_CTR_CFG_ECC_SIZE 8

#define EZDP_DUAL_CTR_CFG_ECC_OFFSET 24

#define EZDP_DUAL_CTR_CFG_VALUE_SIZE 128

#define EZDP_DUAL_CTR_CFG_VALUE_OFFSET 32

#define EZDP_DUAL_CTR_CFG_VALUE_WORD_SELECT 1
```

#define EZDP_DUAL_CTR_CFG_VALUE_WORD_OFFSET 0

#define EZDP_DUAL_CTR_CFG_WORD_COUNT 5

#define EZDP_TB_CTR_RESULT_RESERVED0_31_SIZE 32

#define EZDP_TB_CTR_RESULT_RESERVED0_31_OFFSET 0

#define EZDP_TB_CTR_RESULT_COLOR_SIZE 2

#define EZDP_TB_CTR_RESULT_COLOR_OFFSET 32

#define EZDP_TB_CTR_RESULT_COLOR_WORD_SELECT 1

#define EZDP_TB_CTR_RESULT_COLOR_WORD_OFFSET 0

#define EZDP_TB_CTR_RESULT_RESERVED34_57_SIZE 22

#define EZDP_TB_CTR_RESULT_RESERVED34_57_OFFSET 34

#define EZDP_TB_CTR_RESULT_EMPTY_COMMIT_BUCKET_SIZE 1

#define EZDP_TB_CTR_RESULT_EMPTY_COMMIT_BUCKET_OFFSET 56

#define EZDP_TB_CTR_RESULT_EMPTY_COMMIT_BUCKET_WORD_SELECT 1

#define EZDP_TB_CTR_RESULT_EMPTY_COMMIT_BUCKET_WORD_OFFSET 24

#define EZDP_TB_CTR_RESULT_EMPTY_COMMIT_BUCKET_MASK (1 << EZDP_TB_CTR_RESULT_EMPTY_COMMIT_BUCKET_WORD_OFFSET)

#define EZDP_TB_CTR_RESULT_EMPTY_EXCESS_BUCKET_SIZE 1

#define EZDP_TB_CTR_RESULT_EMPTY_EXCESS_BUCKET_OFFSET 57

#define EZDP_TB_CTR_RESULT_EMPTY_EXCESS_BUCKET_WORD_SELECT 1

#define EZDP_TB_CTR_RESULT_EMPTY_EXCESS_BUCKET_WORD_OFFSET 25

#define EZDP_TB_CTR_RESULT_EMPTY_EXCESS_BUCKET_MASK (1 << EZDP_TB_CTR_RESULT_EMPTY_EXCESS_BUCKET_WORD_OFFSET)

#define EZDP_TB_CTR_RESULT_EMPTY_COMMIT_BUCKET_UG_SIZE 1

#define EZDP_TB_CTR_RESULT_EMPTY_COMMIT_BUCKET_UG_OFFSET 58

#define EZDP_TB_CTR_RESULT_EMPTY_COMMIT_BUCKET_UG_WORD_SELECT 1

#define EZDP_TB_CTR_RESULT_EMPTY_COMMIT_BUCKET_UG_WORD_OFFSET 26

#define EZDP_TB_CTR_RESULT_EMPTY_COMMIT_BUCKET_UG_MASK (1 << EZDP_TB_CTR_RESULT_EMPTY_COMMIT_BUCKET_UG_WORD_OFFSET)

#define EZDP_TB_CTR_RESULT_EMPTY_EXCESS_BUCKET_UG_SIZE 1

#define EZDP_TB_CTR_RESULT_EMPTY_EXCESS_BUCKET_UG_OFFSET 59

#define EZDP_TB_CTR_RESULT_EMPTY_EXCESS_BUCKET_UG_WORD_SELECT 1

#define EZDP_TB_CTR_RESULT_EMPTY_EXCESS_BUCKET_UG_WORD_OFFSET 27

#define EZDP_TB_CTR_RESULT_EMPTY_EXCESS_BUCKET_UG_MASK (1 << EZDP_TB_CTR_RESULT_EMPTY_EXCESS_BUCKET_UG_WORD_OFFSET)

#define EZDP_TB_CTR_RESULT_RESERVED60_63_SIZE 4

#define EZDP_TB_CTR_RESULT_RESERVED60_63_OFFSET 60

#define EZDP_TB_CTR_RESULT_WORD_COUNT 2

#define EZDP_TB_CTR_CFG_COMMIT_PROFILE_ID_SIZE 10

#define EZDP_TB_CTR_CFG_COMMIT_PROFILE_ID_OFFSET 0

#define EZDP_TB_CTR_CFG_COMMIT_PROFILE_ID_WORD_SELECT 0

#define EZDP_TB_CTR_CFG_COMMIT_PROFILE_ID_WORD_OFFSET 0

#define EZDP_TB_CTR_CFG_EXCESS_PROFILE_ID_SIZE 10

#define EZDP_TB_CTR_CFG_EXCESS_PROFILE_ID_OFFSET 10

#define EZDP_TB_CTR_CFG_EXCESS_PROFILE_ID_WORD_SELECT 0

#define EZDP_TB_CTR_CFG_EXCESS_PROFILE_ID_WORD_OFFSET 10

#define EZDP_TB_CTR_CFG_ALGORITHM_TYPE_SIZE 4

#define EZDP_TB_CTR_CFG_ALGORITHM_TYPE_OFFSET 20

#define EZDP_TB_CTR_CFG_ALGORITHM_TYPE_WORD_SELECT 0

#define EZDP_TB_CTR_CFG_ALGORITHM_TYPE_WORD_OFFSET 20

#define EZDP_TB_CTR_CFG_COLOR_AWARE_SIZE 1

#define EZDP_TB_CTR_CFG_COLOR_AWARE_OFFSET 24

#define EZDP_TB_CTR_CFG_COLOR_AWARE_WORD_SELECT 0

#define EZDP_TB_CTR_CFG_COLOR_AWARE_WORD_OFFSET 24

#define EZDP_TB_CTR_CFG_COLOR_AWARE_MASK (1 << EZDP_TB_CTR_CFG_COLOR_AWARE_WORD_OFFSET)

#define EZDP_TB_CTR_CFG_COUPLING_FLAG_SIZE 1

```
#define EZDP_TB_CTR_CFG_COUPLING_FLAG_OFFSET 25

#define EZDP_TB_CTR_CFG_COUPLING_FLAG_WORD_SELECT 0

#define EZDP_TB_CTR_CFG_COUPLING_FLAG_WORD_OFFSET 25

#define EZDP_TB_CTR_CFG_COUPLING_FLAG_MASK (1 <<
EZDP_TB_CTR_CFG_COUPLING_FLAG_WORD_OFFSET)

#define EZDP_TB_CTR_CFG_RESERVED26_31_SIZE 6

#define EZDP_TB_CTR_CFG_RESERVED26_31_OFFSET 26

#define EZDP_TB_CTR_CFG_RESERVED32_63_SIZE 32

#define EZDP_TB_CTR_CFG_RESERVED32_63_OFFSET 32

#define EZDP_TB_CTR_CFG_RESERVED64_95_SIZE 32

#define EZDP_TB_CTR_CFG_RESERVED64_95_OFFSET 64

#define EZDP_TB_CTR_CFG_RESERVED96_127_SIZE 32

#define EZDP_TB_CTR_CFG_RESERVED96_127_OFFSET 96

#define EZDP_TB_CTR_CFG_WORD_COUNT 4

#define EZDP_HIER_TB_UG_APP_BITS_COLOR_STATE_G_SIZE 2

#define EZDP_HIER_TB_UG_APP_BITS_COLOR_STATE_G_OFFSET 0

#define EZDP_HIER_TB_UG_APP_BITS_COLOR_STATE_Y_SIZE 2

#define EZDP_HIER_TB_UG_APP_BITS_COLOR_STATE_Y_OFFSET 2

#define EZDP_HIER_TB_UG_APP_BITS_EIGTH_MODE_RET_BITS_SIZE 4

#define EZDP_HIER_TB_UG_APP_BITS_EIGTH_MODE_RET_BITS_OFFSET 4

#define EZDP_HIER_TB_UG_APP_BITS_APP_BITS_SIZE 16

#define EZDP_HIER_TB_UG_APP_BITS_APP_BITS_OFFSET 8

#define EZDP_HIER_TB_UG_APP_BITS_RESERVED24_31_SIZE 8

#define EZDP_HIER_TB_UG_APP_BITS_RESERVED24_31_OFFSET 24

#define EZDP_HIER_TB_CTR_CFG_RESERVED0_1_SIZE 2

#define EZDP_HIER_TB_CTR_CFG_RESERVED0_1_OFFSET 0

#define EZDP_HIER_TB_CTR_CFG_CTR1_UPDT_THRESHOLD_SIZE 5
```

```
#define EZDP_HIER_TB_CTR_CFG_CTR1_UPDT_THRESHOLD_OFFSET 2

#define EZDP_HIER_TB_CTR_CFG_CTR1_UPDT_THRESHOLD_WORD_SELECT 0

#define EZDP_HIER_TB_CTR_CFG_CTR1_UPDT_THRESHOLD_WORD_OFFSET 2

#define EZDP_HIER_TB_CTR_CFG_CTR1_FAIL_THRESHOLD_SIZE 5

#define EZDP_HIER_TB_CTR_CFG_CTR1_FAIL_THRESHOLD_OFFSET 7

#define EZDP_HIER_TB_CTR_CFG_CTR1_FAIL_THRESHOLD_WORD_SELECT 0

#define EZDP_HIER_TB_CTR_CFG_CTR1_FAIL_THRESHOLD_WORD_OFFSET 7

#define EZDP_HIER_TB_CTR_CFG_CTR_SUM_UPDT_THRESHOLD_SIZE 5

#define EZDP_HIER_TB_CTR_CFG_CTR_SUM_UPDT_THRESHOLD_OFFSET 12

#define EZDP_HIER_TB_CTR_CFG_CTR_SUM_UPDT_THRESHOLD_WORD_SELECT 0

#define EZDP_HIER_TB_CTR_CFG_CTR_SUM_UPDT_THRESHOLD_WORD_OFFSET 12

#define EZDP_HIER_TB_CTR_CFG_CTR_SUM_FAIL_THRESHOLD_SIZE 5

#define EZDP_HIER_TB_CTR_CFG_CTR_SUM_FAIL_THRESHOLD_OFFSET 17

#define EZDP_HIER_TB_CTR_CFG_CTR_SUM_FAIL_THRESHOLD_WORD_SELECT 0

#define EZDP_HIER_TB_CTR_CFG_CTR_SUM_FAIL_THRESHOLD_WORD_OFFSET 17

#define EZDP_HIER_TB_CTR_CFG_RESERVED22_26_SIZE 5

#define EZDP_HIER_TB_CTR_CFG_RESERVED22_26_OFFSET 22

#define EZDP_HIER_TB_CTR_CFG_CTR0_FAIL_THRESHOLD_SIZE 5

#define EZDP_HIER_TB_CTR_CFG_CTR0_FAIL_THRESHOLD_OFFSET 27

#define EZDP_HIER_TB_CTR_CFG_CTR0_FAIL_THRESHOLD_WORD_SELECT 0

#define EZDP_HIER_TB_CTR_CFG_CTR0_FAIL_THRESHOLD_WORD_OFFSET 27

#define EZDP_HIER_TB_CTR_CFG_APP_BITS_SIZE 24

#define EZDP_HIER_TB_CTR_CFG_APP_BITS_OFFSET 32

#define EZDP_HIER_TB_CTR_CFG_APP_BITS_WORD_SELECT 1

#define EZDP_HIER_TB_CTR_CFG_APP_BITS_WORD_OFFSET 0

#define EZDP_HIER_TB_CTR_CFG_CTR0_UPDT_THRESHOLD_SIZE 5
```

```
#define EZDP_HIER_TB_CTR_CFG_CTR0_UPDT_THRESHOLD_OFFSET 56

#define EZDP_HIER_TB_CTR_CFG_CTR0_UPDT_THRESHOLD_WORD_SELECT 1

#define EZDP_HIER_TB_CTR_CFG_CTR0_UPDT_THRESHOLD_WORD_OFFSET 24

#define EZDP_HIER_TB_CTR_CFG_TIMESTAMP_THRESHOLD_SIZE 2

#define EZDP_HIER_TB_CTR_CFG_TIMESTAMP_THRESHOLD_OFFSET 61

#define EZDP_HIER_TB_CTR_CFG_TIMESTAMP_THRESHOLD_WORD_SELECT 1

#define EZDP_HIER_TB_CTR_CFG_TIMESTAMP_THRESHOLD_WORD_OFFSET 29

#define EZDP_HIER_TB_CTR_CFG_RESERVED63_SIZE 1

#define EZDP_HIER_TB_CTR_CFG_RESERVED63_OFFSET 63

#define EZDP_HIER_TB_CTR_CFG_WORD_COUNT 2

#define EZDP_HIER_TB_RESULT_RESERVED0_9_SIZE 10

#define EZDP_HIER_TB_RESULT_RESERVED0_9_OFFSET 0

#define EZDP_HIER_TB_RESULT_CTR1_SIZE 18

#define EZDP_HIER_TB_RESULT_CTR1_OFFSET 10

#define EZDP_HIER_TB_RESULT_CTR1_WORD_SELECT 0

#define EZDP_HIER_TB_RESULT_CTR1_WORD_OFFSET 10

#define EZDP_HIER_TB_RESULT_STATE_SIZE 2

#define EZDP_HIER_TB_RESULT_STATE_OFFSET 28

#define EZDP_HIER_TB_RESULT_STATE_WORD_SELECT 0

#define EZDP_HIER_TB_RESULT_STATE_WORD_OFFSET 28

#define EZDP_HIER_TB_RESULT_UPDATE_TASK_SIZE 1

#define EZDP_HIER_TB_RESULT_UPDATE_TASK_OFFSET 30

#define EZDP_HIER_TB_RESULT_UPDATE_TASK_WORD_SELECT 0

#define EZDP_HIER_TB_RESULT_UPDATE_TASK_WORD_OFFSET 30

#define EZDP_HIER_TB_RESULT_UPDATE_TASK_MASK (1 <<
EZDP_HIER_TB_RESULT_UPDATE_TASK_WORD_OFFSET)

#define EZDP_HIER_TB_RESULT_FAIL_SIZE 1
```

```
#define EZDP_HIER_TB_RESULT_FAIL_OFFSET 31

#define EZDP_HIER_TB_RESULT_FAIL_WORD_SELECT 0

#define EZDP_HIER_TB_RESULT_FAIL_WORD_OFFSET 31

#define EZDP_HIER_TB_RESULT_FAIL_MASK (1 <<
EZDP_HIER_TB_RESULT_FAIL_WORD_OFFSET)

#define EZDP_HIER_TB_RESULT_APP_BITS_SIZE 24

#define EZDP_HIER_TB_RESULT_APP_BITS_OFFSET 32

#define EZDP_HIER_TB_RESULT_APP_BITS_WORD_SELECT 1

#define EZDP_HIER_TB_RESULT_APP_BITS_WORD_OFFSET 0

#define EZDP_HIER_TB_RESULT_RESERVED56_63_SIZE 8

#define EZDP_HIER_TB_RESULT_RESERVED56_63_OFFSET 56

#define EZDP_HIER_TB_RESULT_CTR0_SIZE 18

#define EZDP_HIER_TB_RESULT_CTR0_OFFSET 64

#define EZDP_HIER_TB_RESULT_CTR0_WORD_SELECT 2

#define EZDP_HIER_TB_RESULT_CTR0_WORD_OFFSET 0

#define EZDP_HIER_TB_RESULT_RESERVED82_95_SIZE 14

#define EZDP_HIER_TB_RESULT_RESERVED82_95_OFFSET 82

#define EZDP_HIER_TB_RESULT_WORD_COUNT 3

#define EZDP_HIER_TB_UPDATE_APP_BITS_SIZE 24

#define EZDP_HIER_TB_UPDATE_APP_BITS_OFFSET 0

#define EZDP_HIER_TB_UPDATE_RESERVED24_27_SIZE 4

#define EZDP_HIER_TB_UPDATE_RESERVED24_27_OFFSET 24

#define EZDP_HIER_TB_UPDATE_COND_SET_ACTIVE_STATE_SIZE 1

#define EZDP_HIER_TB_UPDATE_COND_SET_ACTIVE_STATE_OFFSET 28

#define EZDP_HIER_TB_UPDATE_COND_SET_ACTIVE_STATE_MASK (1 <<
EZDP_HIER_TB_UPDATE_COND_SET_ACTIVE_STATE_OFFSET)

#define EZDP_HIER_TB_UPDATE_SET_APP_BITS_SIZE 1

#define EZDP_HIER_TB_UPDATE_SET_APP_BITS_OFFSET 29
```

#define EZDP_HIER_TB_UPDATE_SET_APP_BITS_MASK (1 << EZDP_HIER_TB_UPDATE_SET_APP_BITS_OFFSET)

#define EZDP_HIER_TB_UPDATE_CLR_CTR_SIZE 1

#define EZDP_HIER_TB_UPDATE_CLR_CTR_OFFSET 30

#define EZDP_HIER_TB_UPDATE_CLR_CTR_MASK (1 << EZDP_HIER_TB_UPDATE_CLR_CTR_OFFSET)

#define EZDP_HIER_TB_UPDATE_SET_ACTIVE_STATE_SIZE 1

#define EZDP_HIER_TB_UPDATE_SET_ACTIVE_STATE_OFFSET 31

#define EZDP_HIER_TB_UPDATE_SET_ACTIVE_STATE_MASK (1 << EZDP_HIER_TB_UPDATE_SET_ACTIVE_STATE_OFFSET)

#define EZDP_BITWISE_CTR_CFG_RESERVED0_18_SIZE 19

#define EZDP_BITWISE_CTR_CFG_RESERVED0_18_OFFSET 0

#define EZDP_BITWISE_CTR_CFG_SUB_TYPE_SIZE 5

#define EZDP_BITWISE_CTR_CFG_SUB_TYPE_OFFSET 19

#define EZDP_BITWISE_CTR_CFG_ECC_SIZE 8

#define EZDP_BITWISE_CTR_CFG_ECC_OFFSET 24

#define EZDP_BITWISE_CTR_CFG_RESERVED32_63_SIZE 32

#define EZDP_BITWISE_CTR_CFG_RESERVED32_63_OFFSET 32

#define EZDP_BITWISE_CTR_CFG_DATA_SIZE 64

#define EZDP_BITWISE_CTR_CFG_DATA_OFFSET 64

#define EZDP_BITWISE_CTR_CFG_DATA_WORD_SELECT 2

#define EZDP_BITWISE_CTR_CFG_DATA_WORD_OFFSET 0

#define EZDP_BITWISE_CTR_CFG_WORD_COUNT 4

#define EZDP_WATCHDOG_ACCUMULATIVE_WINDOW_CFG_ACCUMULATIVE_EVENTS_SIZE 5

#define EZDP_WATCHDOG_ACCUMULATIVE_WINDOW_CFG_ACCUMULATIVE_EVENTS_OFFSET 0

#define EZDP_WATCHDOG_ACCUMULATIVE_WINDOW_CFG_ACCUMULATIVE_EVENTS_WORD_SELECT 0

#define EZDP_WATCHDOG_ACCUMULATIVE_WINDOW_CFG_ACCUMULATIVE_EVENTS_WORD_OFFSET 0

#define EZDP_WATCHDOG_ACCUMULATIVE_WINDOW_CFG_RESERVED5_28_SIZE 24

#define EZDP_WATCHDOG_ACCUMULATIVE_WINDOW_CFG_RESERVED5_28_OFFSET 5

#define EZDP_WATCHDOG_ACCUMULATIVE_WINDOW_CFG_MAX_THRESHOLD_ALERT_SIZE 1

#define EZDP_WATCHDOG_ACCUMULATIVE_WINDOW_CFG_MAX_THRESHOLD_ALERT_OFFSET 29

#define EZDP_WATCHDOG_ACCUMULATIVE_WINDOW_CFG_MIN_THRESHOLD_ALERT_SIZE 1

#define EZDP_WATCHDOG_ACCUMULATIVE_WINDOW_CFG_MIN_THRESHOLD_ALERT_OFFSET 30

#define EZDP_WATCHDOG_ACCUMULATIVE_WINDOW_CFG_ALERT_SIZE 1

#define EZDP_WATCHDOG_ACCUMULATIVE_WINDOW_CFG_ALERT_OFFSET 31

#define EZDP_WATCHDOG_ACCUMULATIVE_WINDOW_CFG_LAST_EVENTS_SIZE 12

#define EZDP_WATCHDOG_ACCUMULATIVE_WINDOW_CFG_LAST_EVENTS_OFFSET 32

#define EZDP_WATCHDOG_ACCUMULATIVE_WINDOW_CFG_LAST_EVENTS_WORD_SELECT 1

#define EZDP_WATCHDOG_ACCUMULATIVE_WINDOW_CFG_LAST_EVENTS_WORD_OFFSET 0

#define EZDP_WATCHDOG_ACCUMULATIVE_WINDOW_CFG_CURR_EVENTS_SIZE 12

#define EZDP_WATCHDOG_ACCUMULATIVE_WINDOW_CFG_CURR_EVENTS_OFFSET 44

#define EZDP_WATCHDOG_ACCUMULATIVE_WINDOW_CFG_CURR_EVENTS_WORD_SELECT 1

#define EZDP_WATCHDOG_ACCUMULATIVE_WINDOW_CFG_CURR_EVENTS_WORD_OFFSET 12

#define EZDP_WATCHDOG_ACCUMULATIVE_WINDOW_CFG_INIT_BIT_SIZE 1

#define EZDP_WATCHDOG_ACCUMULATIVE_WINDOW_CFG_INIT_BIT_OFFSET 56

#define EZDP_WATCHDOG_ACCUMULATIVE_WINDOW_CFG_VALID_SIZE 1

#define EZDP_WATCHDOG_ACCUMULATIVE_WINDOW_CFG_VALID_OFFSET 57

#define EZDP_WATCHDOG_ACCUMULATIVE_WINDOW_CFG_VALID_WORD_SELECT 1

#define EZDP_WATCHDOG_ACCUMULATIVE_WINDOW_CFG_VALID_WORD_OFFSET 25

#define EZDP_WATCHDOG_ACCUMULATIVE_WINDOW_CFG_VALID_MASK (1 << EZDP_WATCHDOG_ACCUMULATIVE_WINDOW_CFG_VALID_WORD_OFFSET)

```
#define EZDP_WATCHDOG_ACCUMULATIVE_WINDOW_CFG_PROFILE_ID_SIZE 4

#define EZDP_WATCHDOG_ACCUMULATIVE_WINDOW_CFG_PROFILE_ID_OFFSET 58

#define EZDP_WATCHDOG_ACCUMULATIVE_WINDOW_CFG_PROFILE_ID_WORD_SELECT 1

#define EZDP_WATCHDOG_ACCUMULATIVE_WINDOW_CFG_PROFILE_ID_WORD_OFFSET 26

#define EZDP_WATCHDOG_ACCUMULATIVE_WINDOW_CFG_PARITY_SIZE 1

#define EZDP_WATCHDOG_ACCUMULATIVE_WINDOW_CFG_PARITY_OFFSET 62

#define EZDP_WATCHDOG_ACCUMULATIVE_WINDOW_CFG_RESERVED63_SIZE 1

#define EZDP_WATCHDOG_ACCUMULATIVE_WINDOW_CFG_RESERVED63_OFFSET 63

#define EZDP_WATCHDOG_ACCUMULATIVE_WINDOW_CFG_WORD_COUNT 2

#define EZDP_WATCHDOG_SLIDING_WINDOW_CFG_VALID_WINDOWS_SIZE 5

#define EZDP_WATCHDOG_SLIDING_WINDOW_CFG_VALID_WINDOWS_OFFSET 0

#define EZDP_WATCHDOG_SLIDING_WINDOW_CFG_VALID_WINDOWS_WORD_SELECT 0

#define EZDP_WATCHDOG_SLIDING_WINDOW_CFG_VALID_WINDOWS_WORD_OFFSET 0

#define EZDP_WATCHDOG_SLIDING_WINDOW_CFG_RESERVED5_28_SIZE 24

#define EZDP_WATCHDOG_SLIDING_WINDOW_CFG_RESERVED5_28_OFFSET 5

#define EZDP_WATCHDOG_SLIDING_WINDOW_CFG_MAX_THRESHOLD_ALERT_SIZE 1

#define EZDP_WATCHDOG_SLIDING_WINDOW_CFG_MAX_THRESHOLD_ALERT_OFFSET 29

#define EZDP_WATCHDOG_SLIDING_WINDOW_CFG_MIN_THRESHOLD_ALERT_SIZE 1

#define EZDP_WATCHDOG_SLIDING_WINDOW_CFG_MIN_THRESHOLD_ALERT_OFFSET 30

#define EZDP_WATCHDOG_SLIDING_WINDOW_CFG_ALERT_SIZE 1

#define EZDP_WATCHDOG_SLIDING_WINDOW_CFG_ALERT_OFFSET 31

#define EZDP_WATCHDOG_SLIDING_WINDOW_CFG_COUNTERS_SIZE 24

#define EZDP_WATCHDOG_SLIDING_WINDOW_CFG_COUNTERS_OFFSET 32

#define EZDP_WATCHDOG_SLIDING_WINDOW_CFG_COUNTERS_WORD_SELECT 1

#define EZDP_WATCHDOG_SLIDING_WINDOW_CFG_COUNTERS_WORD_OFFSET 0

#define EZDP_WATCHDOG_SLIDING_WINDOW_CFG_RESERVED56_SIZE 1
```

```
#define EZDP_WATCHDOG_SLIDING_WINDOW_CFG_RESERVED56_OFFSET 56

#define EZDP_WATCHDOG_SLIDING_WINDOW_CFG_VALID_SIZE 1

#define EZDP_WATCHDOG_SLIDING_WINDOW_CFG_VALID_OFFSET 57

#define EZDP_WATCHDOG_SLIDING_WINDOW_CFG_VALID_WORD_SELECT 1

#define EZDP_WATCHDOG_SLIDING_WINDOW_CFG_VALID_WORD_OFFSET 25

#define EZDP_WATCHDOG_SLIDING_WINDOW_CFG_VALID_MASK (1 <<
EZDP_WATCHDOG_SLIDING_WINDOW_CFG_VALID_WORD_OFFSET)

#define EZDP_WATCHDOG_SLIDING_WINDOW_CFG_PROFILE_ID_SIZE 4

#define EZDP_WATCHDOG_SLIDING_WINDOW_CFG_PROFILE_ID_OFFSET 58

#define EZDP_WATCHDOG_SLIDING_WINDOW_CFG_PROFILE_ID_WORD_SELECT 1

#define EZDP_WATCHDOG_SLIDING_WINDOW_CFG_PROFILE_ID_WORD_OFFSET 26

#define EZDP_WATCHDOG_SLIDING_WINDOW_CFG_PARITY_SIZE 1

#define EZDP_WATCHDOG_SLIDING_WINDOW_CFG_PARITY_OFFSET 62

#define EZDP_WATCHDOG_SLIDING_WINDOW_CFG_RESERVED63_SIZE 1

#define EZDP_WATCHDOG_SLIDING_WINDOW_CFG_RESERVED63_OFFSET 63

#define EZDP_WATCHDOG_SLIDING_WINDOW_CFG_WORD_COUNT 2

#define EZDP_WATCHDOG_CTR_CFG_RESERVED0_18_SIZE 19

#define EZDP_WATCHDOG_CTR_CFG_RESERVED0_18_OFFSET 0

#define EZDP_WATCHDOG_CTR_CFG_SUB_TYPE_SIZE 5

#define EZDP_WATCHDOG_CTR_CFG_SUB_TYPE_OFFSET 19

#define EZDP_WATCHDOG_CTR_CFG_ECC_SIZE 8

#define EZDP_WATCHDOG_CTR_CFG_ECC_OFFSET 24

#define EZDP_WATCHDOG_CTR_CFG_RESERVED32_63_SIZE 32

#define EZDP_WATCHDOG_CTR_CFG_RESERVED32_63_OFFSET 32

#define EZDP_WATCHDOG_CTR_CFG_WORD_COUNT 4

#define EZDP_WATCHDOG_CTR_CHECK_RESULT_ACCUMULATIVE_EVENTS_SIZE 4

#define EZDP_WATCHDOG_CTR_CHECK_RESULT_ACCUMULATIVE_EVENTS_OFFSET 0
```

```
#define EZDP_WATCHDOG_CTR_CHECK_RESULT_RESERVED4_28_SIZE 25

#define EZDP_WATCHDOG_CTR_CHECK_RESULT_RESERVED4_28_OFFSET 4

#define EZDP_WATCHDOG_CTR_CHECK_RESULT_MAX_THRESHOLD_ALERT_SIZE 1

#define EZDP_WATCHDOG_CTR_CHECK_RESULT_MAX_THRESHOLD_ALERT_OFFSET 29

#define
EZDP_WATCHDOG_CTR_CHECK_RESULT_MAX_THRESHOLD_ALERT_WORD_SELECT 0

#define
EZDP_WATCHDOG_CTR_CHECK_RESULT_MAX_THRESHOLD_ALERT_WORD_OFFSET 29

#define EZDP_WATCHDOG_CTR_CHECK_RESULT_MAX_THRESHOLD_ALERT_MASK (1 <<
EZDP_WATCHDOG_CTR_CHECK_RESULT_MAX_THRESHOLD_ALERT_WORD_OFFSET)

#define EZDP_WATCHDOG_CTR_CHECK_RESULT_MIN_THRESHOLD_ALERT_SIZE 1

#define EZDP_WATCHDOG_CTR_CHECK_RESULT_MIN_THRESHOLD_ALERT_OFFSET 30

#define
EZDP_WATCHDOG_CTR_CHECK_RESULT_MIN_THRESHOLD_ALERT_WORD_SELECT 0

#define
EZDP_WATCHDOG_CTR_CHECK_RESULT_MIN_THRESHOLD_ALERT_WORD_OFFSET 30

#define EZDP_WATCHDOG_CTR_CHECK_RESULT_MIN_THRESHOLD_ALERT_MASK (1 <<
EZDP_WATCHDOG_CTR_CHECK_RESULT_MIN_THRESHOLD_ALERT_WORD_OFFSET)

#define EZDP_WATCHDOG_CTR_CHECK_RESULT_ALERT_SIZE 1

#define EZDP_WATCHDOG_CTR_CHECK_RESULT_ALERT_OFFSET 31

#define EZDP_WATCHDOG_CTR_CHECK_RESULT_ALERT_WORD_SELECT 0

#define EZDP_WATCHDOG_CTR_CHECK_RESULT_ALERT_WORD_OFFSET 31

#define EZDP_WATCHDOG_CTR_CHECK_RESULT_ALERT_MASK (1 <<
EZDP_WATCHDOG_CTR_CHECK_RESULT_ALERT_WORD_OFFSET)

#define EZDP_WATCHDOG_CTR_CHECK_RESULT_LAST_EVENTS_SIZE 12

#define EZDP_WATCHDOG_CTR_CHECK_RESULT_LAST_EVENTS_OFFSET 32

#define EZDP_WATCHDOG_CTR_CHECK_RESULT_CURR_EVENTS_SIZE 12

#define EZDP_WATCHDOG_CTR_CHECK_RESULT_CURR_EVENTS_OFFSET 44

#define EZDP_WATCHDOG_CTR_CHECK_RESULT_INIT_BIT_SIZE 1

#define EZDP_WATCHDOG_CTR_CHECK_RESULT_INIT_BIT_OFFSET 56
```

```
#define EZDP_WATCHDOG_CTR_CHECK_RESULT_VALID_SIZE 1

#define EZDP_WATCHDOG_CTR_CHECK_RESULT_VALID_OFFSET 57

#define EZDP_WATCHDOG_CTR_CHECK_RESULT_PROFILE_ID_SIZE 4

#define EZDP_WATCHDOG_CTR_CHECK_RESULT_PROFILE_ID_OFFSET 58

#define EZDP_WATCHDOG_CTR_CHECK_RESULT_RESERVED62_SIZE 1

#define EZDP_WATCHDOG_CTR_CHECK_RESULT_RESERVED62_OFFSET 62

#define EZDP_WATCHDOG_CTR_CHECK_RESULT_WINDOW_RELATED_SIZE 1

#define EZDP_WATCHDOG_CTR_CHECK_RESULT_WINDOW_RELATED_OFFSET 63

#define EZDP_WATCHDOG_CTR_CHECK_RESULT_WORD_COUNT 2

#define EZDP_WATCHDOG_CTR_START_RESULT_MSB_SIZE 32

#define EZDP_WATCHDOG_CTR_START_RESULT_MSB_OFFSET 0

#define EZDP_WATCHDOG_CTR_START_RESULT_MSB_WORD_SELECT 0

#define EZDP_WATCHDOG_CTR_START_RESULT_MSB_WORD_OFFSET 0

#define EZDP_WATCHDOG_CTR_START_RESULT_LSB_SIZE 32

#define EZDP_WATCHDOG_CTR_START_RESULT_LSB_OFFSET 32

#define EZDP_WATCHDOG_CTR_START_RESULT_LSB_WORD_SELECT 1

#define EZDP_WATCHDOG_CTR_START_RESULT_LSB_WORD_OFFSET 0

#define EZDP_WATCHDOG_CTR_START_RESULT_WORD_COUNT 2

#define EZDP_CTR_MSG_COUNTER_TYPE_SIZE 2

#define EZDP_CTR_MSG_COUNTER_TYPE_OFFSET 0

#define EZDP_CTR_MSG_COUNTER_TYPE_WORD_SELECT 0

#define EZDP_CTR_MSG_COUNTER_TYPE_WORD_OFFSET 0

#define EZDP_CTR_MSG_MSG_TYPE_SIZE 3

#define EZDP_CTR_MSG_MSG_TYPE_OFFSET 2

#define EZDP_CTR_MSG_MSG_TYPE_WORD_SELECT 0

#define EZDP_CTR_MSG_MSG_TYPE_WORD_OFFSET 2
```

```
#define EZDP_CTR_MSG_OVERFLOW_SIZE 1

#define EZDP_CTR_MSG_OVERFLOW_OFFSET 5

#define EZDP_CTR_MSG_OVERFLOW_WORD_SELECT 0

#define EZDP_CTR_MSG_OVERFLOW_WORD_OFFSET 5

#define EZDP_CTR_MSG_OVERFLOW_MASK (1 <<
EZDP_CTR_MSG_OVERFLOW_WORD_OFFSET)

#define EZDP_CTR_MSG_RESERVED6_SIZE 1

#define EZDP_CTR_MSG_RESERVED6_OFFSET 6

#define EZDP_CTR_MSG_OVERRUN_ERROR_CONDITION_SIZE 1

#define EZDP_CTR_MSG_OVERRUN_ERROR_CONDITION_OFFSET 7

#define EZDP_CTR_MSG_OVERRUN_ERROR_CONDITION_WORD_SELECT 0

#define EZDP_CTR_MSG_OVERRUN_ERROR_CONDITION_WORD_OFFSET 7

#define EZDP_CTR_MSG_OVERRUN_ERROR_CONDITION_MASK (1 <<
EZDP_CTR_MSG_OVERRUN_ERROR_CONDITION_WORD_OFFSET)

#define EZDP_CTR_MSG_RESERVED8_23_SIZE 16

#define EZDP_CTR_MSG_RESERVED8_23_OFFSET 8

#define EZDP_CTR_MSG_ECC_SIZE 8

#define EZDP_CTR_MSG_ECC_OFFSET 24

#define EZDP_CTR_MSG_SUM_ADDR_SIZE 32

#define EZDP_CTR_MSG_SUM_ADDR_OFFSET 32

#define EZDP_CTR_MSG_SUM_ADDR_WORD_SELECT 1

#define EZDP_CTR_MSG_SUM_ADDR_WORD_OFFSET 0

#define EZDP_CTR_MSG_WORD_COUNT 6

#define EZDP_POSTED_CTR_MSG_MSG_TYPE_SIZE 3

#define EZDP_POSTED_CTR_MSG_MSG_TYPE_OFFSET 0

#define EZDP_POSTED_CTR_MSG_MSG_TYPE_WORD_SELECT 0

#define EZDP_POSTED_CTR_MSG_MSG_TYPE_WORD_OFFSET 0

#define EZDP_POSTED_CTR_MSG_FLUSH_SIZE 1
```

```
#define EZDP_POSTED_CTR_MSG_FLUSH_OFFSET 3

#define EZDP_POSTED_CTR_MSG_FLUSH_WORD_SELECT 0

#define EZDP_POSTED_CTR_MSG_FLUSH_WORD_OFFSET 3

#define EZDP_POSTED_CTR_MSG_FLUSH_MASK (1 <<
EZDP_POSTED_CTR_MSG_FLUSH_WORD_OFFSET)

#define EZDP_POSTED_CTR_MSG_CLEAR_SIZE 1

#define EZDP_POSTED_CTR_MSG_CLEAR_OFFSET 4

#define EZDP_POSTED_CTR_MSG_CLEAR_WORD_SELECT 0

#define EZDP_POSTED_CTR_MSG_CLEAR_WORD_OFFSET 4

#define EZDP_POSTED_CTR_MSG_CLEAR_MASK (1 <<
EZDP_POSTED_CTR_MSG_CLEAR_WORD_OFFSET)

#define EZDP_POSTED_CTR_MSG_RESERVED5_6_SIZE 2

#define EZDP_POSTED_CTR_MSG_RESERVED5_6_OFFSET 5

#define EZDP_POSTED_CTR_MSG_OVERRUN_ERROR_CONDITION_SIZE 1

#define EZDP_POSTED_CTR_MSG_OVERRUN_ERROR_CONDITION_OFFSET 7

#define EZDP_POSTED_CTR_MSG_OVERRUN_ERROR_CONDITION_WORD_SELECT 0

#define EZDP_POSTED_CTR_MSG_OVERRUN_ERROR_CONDITION_WORD_OFFSET 7

#define EZDP_POSTED_CTR_MSG_OVERRUN_ERROR_CONDITION_MASK (1 <<
EZDP_POSTED_CTR_MSG_OVERRUN_ERROR_CONDITION_WORD_OFFSET)

#define EZDP_POSTED_CTR_MSG_RESERVED8_23_SIZE 16

#define EZDP_POSTED_CTR_MSG_RESERVED8_23_OFFSET 8

#define EZDP_POSTED_CTR_MSG_ECC_SIZE 8

#define EZDP_POSTED_CTR_MSG_ECC_OFFSET 24

#define EZDP_POSTED_CTR_MSG_SUM_ADDR_SIZE 32

#define EZDP_POSTED_CTR_MSG_SUM_ADDR_OFFSET 32

#define EZDP_POSTED_CTR_MSG_SUM_ADDR_WORD_SELECT 1

#define EZDP_POSTED_CTR_MSG_SUM_ADDR_WORD_OFFSET 0

#define EZDP_POSTED_CTR_MSG_VALUE_SIZE 64
```

#define EZDP_POSTED_CTR_MSG_VALUE_OFFSET 64

#define EZDP_POSTED_CTR_MSG_VALUE_WORD_SELECT 2

#define EZDP_POSTED_CTR_MSG_VALUE_WORD_OFFSET 0

#define EZDP_POSTED_CTR_MSG_WORD_COUNT 4

#define EZDP_CTR_MSG_WORK_AREA_SIZE ((sizeof(struct ezdp_stat_msg_int) > sizeof(struct ezdp_double_ctr_int)/2) ? sizeof(struct ezdp_stat_msg_int) : sizeof(struct ezdp_double_ctr_int)/2)

#define EZDP_POSTED_CTR_MSG_WORK_AREA_SIZE sizeof(struct ezdp_stat_msg_int)

Typedef Documentation

typedef uint32_t ezdp_hier_tb_ug_app_bits_t

typedef uint32_t ezdp_hier_tb_update_t

typedef struct ezdp_get_color_hier_tb_ctr_working_area ezdp_get_color_hier_tb_ctr_working_area_t

typedef uint32_t(* EZDP_HIER_TB_UPDATE_BES)(uint32_t app_bits, uint32_t ctr_g, uint32_t ctr_y, ezdp_get_color_hier_tb_ctr_working_area_t __cmem *wa_ptr, void *user_data)

Type definition for the data plane application's updating BE TBs according to algorithm and user information.

Parameters:

[in] *app_bits* - 24b user data at HIER_TB [in] ctr_g - counter/accumulator of pre-color green packets [in] ctr_y - counter/accumulator of pre-color yellow packets [in] wa_ptr - working area pointer (temporary memory on CMEM to be used by the function) [in] user_data - data for passing to user call back function

Returns:

updated application bits

typedef struct ezdp_msgq_desc ezdp_ctr_msg_queue_desc_t

Statistic message queue descriptor.

typedef struct ezdp_msgq_desc ezdp_posted_ctr_msg_queue_desc_t

Posted message queue descriptor.

Enumeration Type Documentation

enum ezdp_tb_color

tb color possible values.

Enumerator:

EZDP_RED_TRAFFIC Red color marking traffic.

EZDP_YELLOW_TRAFFIC Yellow color marking traffic.

EZDP_GREEN_TRAFFIC Green color marking traffic.

enum ezdp_tb_algo

tb algo possible values.

Enumerator:

EZDP_INACTIVE Inactive.

EZDP_SINGLE_BUCKET Single bucket.

EZDP_SR_TCM Single Rate Three Color Marker (srTCM).

EZDP_TR_TCM Two Rate Three Color Marker (trTCM).

EZDP_TR_TCM_MEF Two Rate Three Color Marker - MEF standard (trTCM MEF).

enum ezdp_hier_tb_state

hier tb state possible values.

Enumerator:

EZDP_UG_FAST_PATH Fast path result in case of ultra green feature.

EZDP_UG_SLOW_PATH Slow path result in case of ultra green feature.

enum ezdp_ctr_type

ctr type possible values.

Enumerator:

EZDP_SINGLE_CTR On-demand counter.

EZDP_DUAL_CTR On-demand dual counter.

enum ezdp_ctr_msg_type

ctr msg type possible values.

Enumerator:

EZDP_PERIODIC_CTR_MSG Periodical update message.

EZDP_THRESHOLD_CTR_MSG Counter exceeded threshold message.

EZDP_NULL_CTR_MSG Null message - there is no message.

enum ezdp_msg_posted_ctr_type

msg posted ctr type possible values.

Enumerator:

EZDP_PERIODIC_POSTED_CTR_MSG Periodical update message.

EZDP_THRESHOLD_POSTED_CTR_MSG Counter exceeded threshold message.

EZDP_REPORT_POSTED_CTR_MSG Report message.
Applicable only for posted counters.

EZDP_NULL_POSTED_CTR_MSG Null message - there is no message.

enum ezdp_bitwise_size

bitwise size possible values.

Enumerator:

EZDP_1_BITS Size of operation is 1 bit.

EZDP_2_BITS Size of operation is 2 bits.

EZDP_4_BITS Size of operation is 4 bits.

EZDP_8_BITS Size of operation is 8 bits.

EZDP_16_BITS Size of operation is 16 bits.

Variable Documentation

enum ezdp_tb_color(* EZDP_HIER_TB_CALC_COLOR)(uint32_t app_bits, enum ezdp_tb_color pre_color, uint32_t *user_data)

User Call Back function calc color according to algorithm and user information.

Parameters:

[in] *app_bits* - 24b user data at HIER_TB [in] pre_color - Packet Original Color [in] user_data - data for passing to user call back function

Returns:

color

dpe/dp/include/ezdp_decode.h File Reference

Functions

- static __always_inline ezdp_decode_mac_retval_t ezdp_decode_mac (uint8_t __cmem *frame_ptr, uint32_t size, struct ezdp_decode_mac_result __cmem *decode_result)
- *Parse and decode an Ethernet header.* static __always_inline void ezdp_decode_mac_async (uint8_t __cmem *frame_ptr, uint32_t size, struct ezdp_decode_mac_result __cmem *decode_result)
- *Non blocking version of ezdp_decode_mac().* static __always_inline ezdp_decode_ipv4_retval_t ezdp_decode_ipv4 (uint8_t __cmem *frame_ptr, uint32_t size, uint32_t frame_size, struct ezdp_decode_ipv4_result __cmem *decode_result)
- *Parse and decode an IPv4 header.* static __always_inline void ezdp_decode_ipv4_async (uint8_t __cmem *frame_ptr, uint32_t size, uint32_t frame_size, struct ezdp_decode_ipv4_result __cmem *decode_result)
- *Non blocking version of ezdp_decode_ipv4().* static __always_inline ezdp_decode_ipv6_retval_t ezdp_decode_ipv6 (uint8_t __cmem *frame_ptr, uint32_t size, uint32_t frame_size, struct ezdp_decode_ipv6_result __cmem *decode_result)
- *Parse and decode an IPv6 header.* static __always_inline void ezdp_decode_ipv6_async (uint8_t __cmem *frame_ptr, uint32_t size, uint32_t frame_size, struct ezdp_decode_ipv6_result __cmem *decode_result)
- *Non blocking version of ezdp_decode_ipv6().* static __always_inline ezdp_decode_mpls_result_t ezdp_decode_mpls (uint8_t __cmem *frame_ptr, uint32_t size, struct ezdp_decode_mpls_result __cmem *decode_result)
- *Parse and decode an MPLS header.* static __always_inline void ezdp_decode_mpls_async (uint8_t __cmem *frame_ptr, uint32_t size, struct ezdp_decode_mpls_result __cmem *decode_result)
- *Non blocking version of ezdp_decode_mpls().* static __always_inline ezdp_decode_mpls_label_retval_t ezdp_decode_mpls_label (uint8_t __cmem *frame_ptr, uint32_t size, struct ezdp_decode_mpls_label_result __cmem *decode_result)
- *Parse and decode an MPLS label.* static __always_inline void ezdp_decode_mpls_label_async (uint8_t __cmem *frame_ptr, uint32_t size, struct ezdp_decode_mpls_label_result __cmem *decode_result)
- *Non blocking version of ezdp_decode_mpls_label().* static __always_inline ezdp_decode_tcp_retval_t ezdp_decode_tcp (uint32_t __cmem *frame_ptr)
- *Parse and decode a TCP header.* static __always_inline ezdp_decode_ip_protocol_retval_t ezdp_decode_ip_protocol (uint32_t ip_type, uint32_t def_ip_prot_0, uint32_t def_ip_prot_1, uint32_t def_ip_prot_2, uint32_t def_ip_prot_3)
- *Decode an IP protocol value.* static __always_inline ezdp_decode_eth_type_retval_t ezdp_decode_eth_type (uint32_t eth_type, uint32_t def_eth_type_0, uint32_t def_eth_type_1)

Decode an Ethernet type value.

Function Documentation

static __always_inline ezdp_decode_mac_retval_t ezdp_decode_mac (uint8_t __cmem * *frame_ptr*, uint32_t *size*, struct ezdp_decode_mac_result __cmem * *decode_result*) [static]

Parse and decode an Ethernet header.

The decode result is written to CMEM. In addition, the first 4 bytes of the decode result are returned.

Parameters:

[in] *frame_ptr* - pointer to frame header data (in CMEM) to be decoded
[in] *size* - number of bytes to decode
[out] *decode_result* - decode result

Returns:

uint32_t - according to ezdp_decode_dcmac_retval

static __always_inline void ezdp_decode_mac_async (uint8_t __cmem * *frame_ptr*, uint32_t *size*, struct ezdp_decode_mac_result __cmem * *decode_result*) [static]

Non blocking version of ezdp_decode_mac().

Parameters:

[in] *frame_ptr* - pointer to frame header data (in CMEM) to be decoded
[in] *size* - number of bytes to decode
[out] *decode_result* - decode result

Note:

Call ezdp_sync() to wait for the operation to complete, confirming that the decode result is ready in CMEM.

Returns:

none

static __always_inline ezdp_decode_ipv4_retval_t ezdp_decode_ipv4 (uint8_t __cmem * *frame_ptr*, uint32_t *size*, uint32_t *frame_size*, struct ezdp_decode_ipv4_result __cmem * *decode_result*) `[static]`

Parse and decode an IPv4 header.

The decode result is written to CMEM. In addition, the first 4 bytes of the decode result are returned.

Parameters:

[in] *frame_ptr* - pointer to frame header data (in CMEM) to be decoded
[in] *size* - number of bytes to decode
[in] *frame_size* - frame data size
[out] *decode_result* - decode result

Returns:

uint32_t - according to ezdp_decode_dcipv4_retval

static __always_inline void ezdp_decode_ipv4_async (uint8_t __cmem * *frame_ptr*, uint32_t *size*, uint32_t *frame_size*, struct ezdp_decode_ipv4_result __cmem * *decode_result*) `[static]`

Non blocking version of ezdp_decode_ipv4().

Parameters:

[in] *frame_ptr* - pointer to frame header data (in CMEM) to be decoded
[in] *size* - number of bytes to decode
[in] *frame_size* - frame data size
[out] *decode_result* - decode result

Note:

Call ezdp_sync() to wait for the operation to complete, confirming that the decode result is ready in CMEM.

Returns:

none

static __always_inline ezdp_decode_ipv6_retval_t ezdp_decode_ipv6 (uint8_t __cmem * *frame_ptr*, uint32_t *size*, uint32_t *frame_size*, struct ezdp_decode_ipv6_result __cmem * *decode_result*) `[static]`

Parse and decode an IPv6 header.

The decode result is written to CMEM. In addition, the first 4 bytes of the decode result are returned.

Parameters:

[in] *frame_ptr* - pointer to frame header data (in CMEM) to be decoded
[in] *size* - number of bytes to decode
[in] *frame_size* - frame data size
[out] *decode_result* - decode result

Returns:

uint32_t - according to ezdp_decode_dcipv6_retval

static __always_inline void ezdp_decode_ipv6_async (uint8_t __cmem * *frame_ptr*, uint32_t *size*, uint32_t *frame_size*, struct ezdp_decode_ipv6_result __cmem * *decode_result*) [static]

Non blocking version of ezdp_decode_ipv6().

Parameters:

[in] *frame_ptr* - pointer to frame header data (in CMEM) to be decoded
[in] *size* - number of bytes to decode
[in] *frame_size* - frame data size
[out] *decode_result* - decode result

Note:

Call ezdp_sync() to wait for the operation to complete, confirming that the decode result is ready in CMEM.

Returns:

none

static __always_inline ezdp_decode_mpls_result_t ezdp_decode_mpls (uint8_t __cmem * *frame_ptr*, uint32_t *size*, struct ezdp_decode_mpls_result __cmem * *decode_result*) [static]

Parse and decode an MPLS header.

The decode result is written to CMEM. In addition, the first 4 bytes of the decode result are returned.

Parameters:

[in] *frame_ptr* - pointer to frame header data (in CMEM) to be decoded
[in] *size* - number of bytes to decode
[out] *decode_result* - decode result

Returns:

uint32_t - according to ezdp_decode_mpls_result

static __always_inline void ezdp_decode_mpls_async (uint8_t __cmem * *frame_ptr*, uint32_t *size*, struct ezdp_decode_mpls_result __cmem * *decode_result*) [static]

Non blocking version of ezdp_decode_mpls().

Parameters:

[in] *frame_ptr* - pointer to frame header data (in CMEM) to be decoded
[in] *size* - number of bytes to decode
[out] *decode_result* - decode result

Note:

Call ezdp_sync() to wait for the operation to complete, confirming that the decode result is ready in CMEM.

Returns:
none

static __always_inline ezdp_decode_mpls_label_retval_t ezdp_decode_mpls_label (uint8_t __cmem * *frame_ptr*, uint32_t *size*, struct ezdp_decode_mpls_label_result __cmem * *decode_result*) [static]

Parse and decode an MPLS label.

The decode result is written to CMEM. In addition, the first 4 bytes of the decode result are returned.

Parameters:
[in] *frame_ptr* - pointer to frame header data (in CMEM) to be decoded
[in] *size* - number of bytes to decode
[out] *decode_result* - decode result

Returns:
uint32_t - according to ezdp_decode_mpls_label_retval_t

static __always_inline void ezdp_decode_mpls_label_async (uint8_t __cmem * *frame_ptr*, uint32_t *size*, struct ezdp_decode_mpls_label_result __cmem * *decode_result*) [static]

Non blocking version of ezdp_decode_mpls_label().

Parameters:
[in] *frame_ptr* - pointer to frame header data (in CMEM) to be decoded
[in] *size* - number of bytes to decode
[out] *decode_result* - decode result

Note:
Call ezdp_sync() to wait for the operation to complete, confirming that the decode result is ready in CMEM.

Returns:
none

static __always_inline ezdp_decode_tcp_retval_t ezdp_decode_tcp (uint32_t __cmem * *frame_ptr*) [static]

Parse and decode a TCP header.

Parameters:
[in] *frame_ptr* - pointer to frame header data (in CMEM) to be decoded

Returns:
uint32_t - according to ezdp_decode_tcp_retval

static __always_inline ezdp_decode_ip_protocol_retval_t ezdp_decode_ip_protocol (uint32_t *ip_type*, uint32_t *def_ip_prot_0*, uint32_t *def_ip_prot_1*, uint32_t *def_ip_prot_2*, uint32_t *def_ip_prot_3*) [static]

Decode an IP protocol value.

Parameters:
[in] *ip_type* - IP protocol type

[in] *def_ip_prot_0* - Default/config ip protocol type 0 immediate value to be used by decode_ip_protocol instruction to turn on def_ip_prot_0 flag.
[in] *def_ip_prot_1* - Default/config ip protocol type 1 immediate value to be used by decode_ip_protocol instruction to turn on def_ip_prot_1 flag.
[in] *def_ip_prot_2* - Default/config ip protocol type 2 immediate value to be used by decode_ip_protocol instruction to turn on def_ip_prot_2 flag.
[in] *def_ip_prot_3* - Default/config ip protocol type 3 immediate value to be used by decode_ip_protocol instruction to turn on def_ip_prot_3 flag.

Returns:

uint32_t - according to ezdp_decode_ip_protocol_retval

static __always_inline ezdp_decode_eth_type_retval_t ezdp_decode_eth_type (uint32_t *eth_type*, uint32_t *def_eth_type_0*, uint32_t *def_eth_type_1*) [static]

Decode an Ethernet type value.

Parameters:

[in] *eth_type* - Ethernet type
[in] *def_eth_type_0* - Default/config Ethernet type 0 immediate value to be used by decode_eth_type instruction to turn on user_def0 flag.
[in] *def_eth_type_1* - Default/config Ethernet type 1 immediate value to be used by decode_eth_type instruction to turn on user_def1 flag.

Note:

When both type0 and type1 are 0, default/conf values are ignored and a smaller/optimized instruction is used.

Returns:

uint32_t - according to decode_eth_type_retval

dpe/dp/include/ezdp_decode_defs.h File Reference

Data Structures

- struct ezdp_decode_ipv4_control
- *IPv4 addresses decoding result.* struct ezdp_decode_ipv4_errors
- *IPv4 header decode error flags.* struct ezdp_decode_ip_next_protocol
- *IP protocol type flags.* struct ezdp_decode_ipv6_control
- *IPv6 addresses decoding result.* struct ezdp_decode_ipv6_errors
- *IPv6 header decode error flags.* struct ezdp_decode_mac_control
- *MAC addresses decoding result.* struct ezdp_decode_mac_errors
- *MAC header decode error flags.* struct ezdp_decode_mac_protocol_type
- *Ethernet type definition.* struct ezdp_decode_tcp_errors
- *TCP header decode error flags.* struct ezdp_decode_tcp_retval
- *Decode TCP return value struct definition.* struct ezdp_decode_ip_protocol_retval
- *Decode ip protocol return value struct definition.* struct ezdp_decode_eth_type_retval
- *Decode ip protocol return value struct definition.* struct ezdp_decode_ipv4_retval
- *Decode IPv4 return value struct definition.* struct ezdp_decode_ipv4_result
- *Decode IPv4 result.* struct ezdp_decode_ipv6_retval
- *Decode IPv4 return value struct definition.* struct ezdp_decode_ipv6_result
- *Decode IPv6 result.* struct ezdp_decode_mac_retval
- *Decode MAC return value struct definition.* struct ezdp_decode_mac_result
- *Decode MAC result.* struct ezdp_decode_mpls_retval
- *Decode MPLS return value struct definition.* struct ezdp_decode_mpls_result
- *ezdp_decode_mpls_result struct for ezdp* struct ezdp_decode_mpls_label_retval
- *Decode MPLS label return value struct definition.* struct ezdp_decode_mpls_label_result

ezdp_decode_mpls_label_result struct for ezdp Defines

- #define EZDP_DECODE_VERSION_MAJOR 2
- #define EZDP_DECODE_VERSION_MINOR 1
- #define EZDP_DECODE_IPV4_CONTROL_LINK_LOCAL_MULTICAST_RANGE_SIZE 1
- #define EZDP_DECODE_IPV4_CONTROL_LINK_LOCAL_MULTICAST_RANGE_OFFSET 0
- #define EZDP_DECODE_IPV4_CONTROL_LINK_LOCAL_MULTICAST_RANGE_MASK (1 << EZDP_DECODE_IPV4_CONTROL_LINK_LOCAL_MULTICAST_RANGE_OFFSET)
- #define EZDP_DECODE_IPV4_CONTROL_INTERNETWORK_MULTICAST_RANGE_SIZE 1
- #define EZDP_DECODE_IPV4_CONTROL_INTERNETWORK_MULTICAST_RANGE_OFFSET 1
- #define EZDP_DECODE_IPV4_CONTROL_INTERNETWORK_MULTICAST_RANGE_MASK (1 << EZDP_DECODE_IPV4_CONTROL_INTERNETWORK_MULTICAST_RANGE_OFFSET)
- #define EZDP_DECODE_IPV4_CONTROL_RESERVED_2_SIZE 1
- #define EZDP_DECODE_IPV4_CONTROL_RESERVED_2_OFFSET 2
- #define EZDP_DECODE_IPV4_CONTROL_ICMP_SIZE 1
- #define EZDP_DECODE_IPV4_CONTROL_ICMP_OFFSET 3
- #define EZDP_DECODE_IPV4_CONTROL_ICMP_MASK (1 << EZDP_DECODE_IPV4_CONTROL_ICMP_OFFSET)
- #define EZDP_DECODE_IPV4_CONTROL_IGMP_SIZE 1
- #define EZDP_DECODE_IPV4_CONTROL_IGMP_OFFSET 4
- #define EZDP_DECODE_IPV4_CONTROL_IGMP_MASK (1 << EZDP_DECODE_IPV4_CONTROL_IGMP_OFFSET)
- #define EZDP_DECODE_IPV4_CONTROL_USER_CONFIG0_SIZE 1
- #define EZDP_DECODE_IPV4_CONTROL_USER_CONFIG0_OFFSET 5
- #define EZDP_DECODE_IPV4_CONTROL_USER_CONFIG0_MASK (1 << EZDP_DECODE_IPV4_CONTROL_USER_CONFIG0_OFFSET)
- #define EZDP_DECODE_IPV4_CONTROL_USER_CONFIG1_SIZE 1
- #define EZDP_DECODE_IPV4_CONTROL_USER_CONFIG1_OFFSET 6
- #define EZDP_DECODE_IPV4_CONTROL_USER_CONFIG1_MASK (1 << EZDP_DECODE_IPV4_CONTROL_USER_CONFIG1_OFFSET)

- #define EZDP_DECODE_IPV4_CONTROL_USER_CONFIG2_SIZE 1
- #define EZDP_DECODE_IPV4_CONTROL_USER_CONFIG2_OFFSET 7
- #define EZDP_DECODE_IPV4_CONTROL_USER_CONFIG2_MASK (1 << EZDP_DECODE_IPV4_CONTROL_USER_CONFIG2_OFFSET)
- #define EZDP_DECODE_IPV4_ERRORS_SIP_IS_MULTICAST_SIZE 1
- #define EZDP_DECODE_IPV4_ERRORS_SIP_IS_MULTICAST_OFFSET 0
- #define EZDP_DECODE_IPV4_ERRORS_SIP_IS_MULTICAST_MASK (1 << EZDP_DECODE_IPV4_ERRORS_SIP_IS_MULTICAST_OFFSET)
- #define EZDP_DECODE_IPV4_ERRORS_SIP_IS_ZERO_SIZE 1
- #define EZDP_DECODE_IPV4_ERRORS_SIP_IS_ZERO_OFFSET 1
- #define EZDP_DECODE_IPV4_ERRORS_SIP_IS_ZERO_MASK (1 << EZDP_DECODE_IPV4_ERRORS_SIP_IS_ZERO_OFFSET)
- #define EZDP_DECODE_IPV4_ERRORS_HEADER_LENGTH_LT_5_SIZE 1
- #define EZDP_DECODE_IPV4_ERRORS_HEADER_LENGTH_LT_5_OFFSET 2
- #define EZDP_DECODE_IPV4_ERRORS_HEADER_LENGTH_LT_5_MASK (1 << EZDP_DECODE_IPV4_ERRORS_HEADER_LENGTH_LT_5_OFFSET)
- #define EZDP_DECODE_IPV4_ERRORS_TOTAL_LENGTH_GT_FRAME_LENGTH_SIZE 1
- #define EZDP_DECODE_IPV4_ERRORS_TOTAL_LENGTH_GT_FRAME_LENGTH_OFFSET 3
- #define EZDP_DECODE_IPV4_ERRORS_TOTAL_LENGTH_GT_FRAME_LENGTH_MASK (1 << EZDP_DECODE_IPV4_ERRORS_TOTAL_LENGTH_GT_FRAME_LENGTH_OFFSET)
- #define EZDP_DECODE_IPV4_ERRORS_HEADER_LENGTH_GT_FRAME_LENGTH_SIZE 1
- #define EZDP_DECODE_IPV4_ERRORS_HEADER_LENGTH_GT_FRAME_LENGTH_OFFSET 4
- #define EZDP_DECODE_IPV4_ERRORS_HEADER_LENGTH_GT_FRAME_LENGTH_MASK (1 << EZDP_DECODE_IPV4_ERRORS_HEADER_LENGTH_GT_FRAME_LENGTH_OFFSET)
- #define EZDP_DECODE_IPV4_ERRORS_NOT_IPV4_VERSION_SIZE 1
- #define EZDP_DECODE_IPV4_ERRORS_NOT_IPV4_VERSION_OFFSET 5
- #define EZDP_DECODE_IPV4_ERRORS_NOT_IPV4_VERSION_MASK (1 << EZDP_DECODE_IPV4_ERRORS_NOT_IPV4_VERSION_OFFSET)
- #define EZDP_DECODE_IPV4_ERRORS_CHECKSUM_ERROR_SIZE 1
- #define EZDP_DECODE_IPV4_ERRORS_CHECKSUM_ERROR_OFFSET 6
- #define EZDP_DECODE_IPV4_ERRORS_CHECKSUM_ERROR_MASK (1 << EZDP_DECODE_IPV4_ERRORS_CHECKSUM_ERROR_OFFSET)
- #define EZDP_DECODE_IPV4_ERRORS_SIP_EQUAL_DIP_SIZE 1
- #define EZDP_DECODE_IPV4_ERRORS_SIP_EQUAL_DIP_OFFSET 7
- #define EZDP_DECODE_IPV4_ERRORS_SIP_EQUAL_DIP_MASK (1 << EZDP_DECODE_IPV4_ERRORS_SIP_EQUAL_DIP_OFFSET)
- #define EZDP_DECODE_IPV4_ERRORS_DECODE_ERROR_SIZE 1
- #define EZDP_DECODE_IPV4_ERRORS_DECODE_ERROR_OFFSET 8
- #define EZDP_DECODE_IPV4_ERRORS_DECODE_ERROR_MASK (1 << EZDP_DECODE_IPV4_ERRORS_DECODE_ERROR_OFFSET)
- #define EZDP_DECODE_IPV4_ERRORS_RESERVED9_15_SIZE 7
- #define EZDP_DECODE_IPV4_ERRORS_RESERVED9_15_OFFSET 9
- #define EZDP_DECODE_IP_NEXT_PROTOCOL_TCP_SIZE 1
- #define EZDP_DECODE_IP_NEXT_PROTOCOL_TCP_OFFSET 0
- #define EZDP_DECODE_IP_NEXT_PROTOCOL_TCP_MASK (1 << EZDP_DECODE_IP_NEXT_PROTOCOL_TCP_OFFSET)
- #define EZDP_DECODE_IP_NEXT_PROTOCOL_UDP_SIZE 1
- #define EZDP_DECODE_IP_NEXT_PROTOCOL_UDP_OFFSET 1
- #define EZDP_DECODE_IP_NEXT_PROTOCOL_UDP_MASK (1 << EZDP_DECODE_IP_NEXT_PROTOCOL_UDP_OFFSET)
- #define EZDP_DECODE_IP_NEXT_PROTOCOL_MPLS_SIZE 1
- #define EZDP_DECODE_IP_NEXT_PROTOCOL_MPLS_OFFSET 2
- #define EZDP_DECODE_IP_NEXT_PROTOCOL_MPLS_MASK (1 << EZDP_DECODE_IP_NEXT_PROTOCOL_MPLS_OFFSET)
- #define EZDP_DECODE_IP_NEXT_PROTOCOL_GRE_SIZE 1
- #define EZDP_DECODE_IP_NEXT_PROTOCOL_GRE_OFFSET 3
- #define EZDP_DECODE_IP_NEXT_PROTOCOL_GRE_MASK (1 << EZDP_DECODE_IP_NEXT_PROTOCOL_GRE_OFFSET)

- #define EZDP_DECODE_IP_NEXT_PROTOCOL_IPV4_SIZE 1
- #define EZDP_DECODE_IP_NEXT_PROTOCOL_IPV4_OFFSET 4
- #define EZDP_DECODE_IP_NEXT_PROTOCOL_IPV4_MASK (1 << EZDP_DECODE_IP_NEXT_PROTOCOL_IPV4_OFFSET)
- #define EZDP_DECODE_IP_NEXT_PROTOCOL_IPV6_SIZE 1
- #define EZDP_DECODE_IP_NEXT_PROTOCOL_IPV6_OFFSET 5
- #define EZDP_DECODE_IP_NEXT_PROTOCOL_IPV6_MASK (1 << EZDP_DECODE_IP_NEXT_PROTOCOL_IPV6_OFFSET)
- #define EZDP_DECODE_IP_NEXT_PROTOCOL_ICMP_IGMP_SIZE 1
- #define EZDP_DECODE_IP_NEXT_PROTOCOL_ICMP_IGMP_OFFSET 6
- #define EZDP_DECODE_IP_NEXT_PROTOCOL_ICMP_IGMP_MASK (1 << EZDP_DECODE_IP_NEXT_PROTOCOL_ICMP_IGMP_OFFSET)
- #define EZDP_DECODE_IP_NEXT_PROTOCOL_OTHER_SIZE 1
- #define EZDP_DECODE_IP_NEXT_PROTOCOL_OTHER_OFFSET 7
- #define EZDP_DECODE_IP_NEXT_PROTOCOL_OTHER_MASK (1 << EZDP_DECODE_IP_NEXT_PROTOCOL_OTHER_OFFSET)
- #define EZDP_DECODE_IPV6_CONTROL_LINK_LOCAL_MULTICAST_RANGE_SIZE 1
- #define EZDP_DECODE_IPV6_CONTROL_LINK_LOCAL_MULTICAST_RANGE_OFFSET 0
- #define EZDP_DECODE_IPV6_CONTROL_LINK_LOCAL_MULTICAST_RANGE_MASK (1 << EZDP_DECODE_IPV6_CONTROL_LINK_LOCAL_MULTICAST_RANGE_OFFSET)
- #define EZDP_DECODE_IPV6_CONTROL_INTERNETWORK_MULTICAST_RANGE_SIZE 1
- #define EZDP_DECODE_IPV6_CONTROL_INTERNETWORK_MULTICAST_RANGE_OFFSET 1
- #define EZDP_DECODE_IPV6_CONTROL_INTERNETWORK_MULTICAST_RANGE_MASK (1 << EZDP_DECODE_IPV6_CONTROL_INTERNETWORK_MULTICAST_RANGE_OFFSET)
- #define EZDP_DECODE_IPV6_CONTROL_SOLICITED_NODE_MULTICAST_RANGE_SIZE 1
- #define EZDP_DECODE_IPV6_CONTROL_SOLICITED_NODE_MULTICAST_RANGE_OFFSET 2
- #define EZDP_DECODE_IPV6_CONTROL_SOLICITED_NODE_MULTICAST_RANGE_MASK (1 << EZDP_DECODE_IPV6_CONTROL_SOLICITED_NODE_MULTICAST_RANGE_OFFSET)
- #define EZDP_DECODE_IPV6_CONTROL_RESERVED_3_SIZE 1
- #define EZDP_DECODE_IPV6_CONTROL_RESERVED_3_OFFSET 3
- #define EZDP_DECODE_IPV6_CONTROL_DIP_IS_MULTICAST_SIZE 1
- #define EZDP_DECODE_IPV6_CONTROL_DIP_IS_MULTICAST_OFFSET 4
- #define EZDP_DECODE_IPV6_CONTROL_DIP_IS_MULTICAST_MASK (1 << EZDP_DECODE_IPV6_CONTROL_DIP_IS_MULTICAST_OFFSET)
- #define EZDP_DECODE_IPV6_CONTROL_DIP_IS_WELLKNOWN_MULTICAST_SIZE 1
- #define EZDP_DECODE_IPV6_CONTROL_DIP_IS_WELLKNOWN_MULTICAST_OFFSET 5
- #define EZDP_DECODE_IPV6_CONTROL_DIP_IS_WELLKNOWN_MULTICAST_MASK (1 << EZDP_DECODE_IPV6_CONTROL_DIP_IS_WELLKNOWN_MULTICAST_OFFSET)
- #define EZDP_DECODE_IPV6_CONTROL_RESERVED7_8_SIZE 2
- #define EZDP_DECODE_IPV6_CONTROL_RESERVED7_8_OFFSET 6
- #define EZDP_DECODE_IPV6_ERRORS_PAYLOAD_GT_FRAME_LENGTH_SIZE 1
- #define EZDP_DECODE_IPV6_ERRORS_PAYLOAD_GT_FRAME_LENGTH_OFFSET 0
- #define EZDP_DECODE_IPV6_ERRORS_PAYLOAD_GT_FRAME_LENGTH_MASK (1 << EZDP_DECODE_IPV6_ERRORS_PAYLOAD_GT_FRAME_LENGTH_OFFSET)
- #define EZDP_DECODE_IPV6_ERRORS_NOT_IPV6_VERSION_SIZE 1
- #define EZDP_DECODE_IPV6_ERRORS_NOT_IPV6_VERSION_OFFSET 1
- #define EZDP_DECODE_IPV6_ERRORS_NOT_IPV6_VERSION_MASK (1 << EZDP_DECODE_IPV6_ERRORS_NOT_IPV6_VERSION_OFFSET)
- #define EZDP_DECODE_IPV6_ERRORS_SIP_IS_ZERO_SIZE 1
- #define EZDP_DECODE_IPV6_ERRORS_SIP_IS_ZERO_OFFSET 2
- #define EZDP_DECODE_IPV6_ERRORS_SIP_IS_ZERO_MASK (1 << EZDP_DECODE_IPV6_ERRORS_SIP_IS_ZERO_OFFSET)
- #define EZDP_DECODE_IPV6_ERRORS_SIP_IS_ONE_SIZE 1
- #define EZDP_DECODE_IPV6_ERRORS_SIP_IS_ONE_OFFSET 3
- #define EZDP_DECODE_IPV6_ERRORS_SIP_IS_ONE_MASK (1 << EZDP_DECODE_IPV6_ERRORS_SIP_IS_ONE_OFFSET)
- #define EZDP_DECODE_IPV6_ERRORS_DIP_IS_ZERO_SIZE 1
- #define EZDP_DECODE_IPV6_ERRORS_DIP_IS_ZERO_OFFSET 4

- #define EZDP_DECODE_IPV6_ERRORS_DIP_IS_ZERO_MASK (1 << EZDP_DECODE_IPV6_ERRORS_DIP_IS_ZERO_OFFSET)
- #define EZDP_DECODE_IPV6_ERRORS_DIP_IS_ONE_SIZE 1
- #define EZDP_DECODE_IPV6_ERRORS_DIP_IS_ONE_OFFSET 5
- #define EZDP_DECODE_IPV6_ERRORS_DIP_IS_ONE_MASK (1 << EZDP_DECODE_IPV6_ERRORS_DIP_IS_ONE_OFFSET)
- #define EZDP_DECODE_IPV6_ERRORS_SIP_EQUAL_DIP_SIZE 1
- #define EZDP_DECODE_IPV6_ERRORS_SIP_EQUAL_DIP_OFFSET 6
- #define EZDP_DECODE_IPV6_ERRORS_SIP_EQUAL_DIP_MASK (1 << EZDP_DECODE_IPV6_ERRORS_SIP_EQUAL_DIP_OFFSET)
- #define EZDP_DECODE_IPV6_ERRORS_DECODE_ERROR_SIZE 1
- #define EZDP_DECODE_IPV6_ERRORS_DECODE_ERROR_OFFSET 7
- #define EZDP_DECODE_IPV6_ERRORS_DECODE_ERROR_MASK (1 << EZDP_DECODE_IPV6_ERRORS_DECODE_ERROR_OFFSET)
- #define EZDP_DECODE_IPV6_ERRORS_PAYLOAD_MISSING_SIZE 1
- #define EZDP_DECODE_IPV6_ERRORS_PAYLOAD_MISSING_OFFSET 8
- #define EZDP_DECODE_IPV6_ERRORS_PAYLOAD_MISSING_MASK (1 << EZDP_DECODE_IPV6_ERRORS_PAYLOAD_MISSING_OFFSET)
- #define EZDP_DECODE_IPV6_ERRORS_SIP_IS_MULTICAST_SIZE 1
- #define EZDP_DECODE_IPV6_ERRORS_SIP_IS_MULTICAST_OFFSET 9
- #define EZDP_DECODE_IPV6_ERRORS_SIP_IS_MULTICAST_MASK (1 << EZDP_DECODE_IPV6_ERRORS_SIP_IS_MULTICAST_OFFSET)
- #define EZDP_DECODE_IPV6_ERRORS_RESERVED10_15_SIZE 6
- #define EZDP_DECODE_IPV6_ERRORS_RESERVED10_15_OFFSET 10
- #define EZDP_DECODE_MAC_CONTROL_MY_MAC_SIZE 1
- #define EZDP_DECODE_MAC_CONTROL_MY_MAC_OFFSET 0
- #define EZDP_DECODE_MAC_CONTROL_MY_MAC_MASK (1 << EZDP_DECODE_MAC_CONTROL_MY_MAC_OFFSET)
- #define EZDP_DECODE_MAC_CONTROL_MAC_CONTROL_LSB_0X_SIZE 1
- #define EZDP_DECODE_MAC_CONTROL_MAC_CONTROL_LSB_0X_OFFSET 1
- #define EZDP_DECODE_MAC_CONTROL_MAC_CONTROL_LSB_0X_MASK (1 << EZDP_DECODE_MAC_CONTROL_MAC_CONTROL_LSB_0X_OFFSET)
- #define EZDP_DECODE_MAC_CONTROL_MAC_CONTROL_LSB_1X_SIZE 1
- #define EZDP_DECODE_MAC_CONTROL_MAC_CONTROL_LSB_1X_OFFSET 2
- #define EZDP_DECODE_MAC_CONTROL_MAC_CONTROL_LSB_1X_MASK (1 << EZDP_DECODE_MAC_CONTROL_MAC_CONTROL_LSB_1X_OFFSET)
- #define EZDP_DECODE_MAC_CONTROL_MAC_CONTROL_LSB_2X_SIZE 1
- #define EZDP_DECODE_MAC_CONTROL_MAC_CONTROL_LSB_2X_OFFSET 3
- #define EZDP_DECODE_MAC_CONTROL_MAC_CONTROL_LSB_2X_MASK (1 << EZDP_DECODE_MAC_CONTROL_MAC_CONTROL_LSB_2X_OFFSET)
- #define EZDP_DECODE_MAC_CONTROL_MAC_CONTROL_OTHER_SIZE 1
- #define EZDP_DECODE_MAC_CONTROL_MAC_CONTROL_OTHER_OFFSET 4
- #define EZDP_DECODE_MAC_CONTROL_MAC_CONTROL_OTHER_MASK (1 << EZDP_DECODE_MAC_CONTROL_MAC_CONTROL_OTHER_OFFSET)
- #define EZDP_DECODE_MAC_CONTROL_VRRP_MAC_SIZE 1
- #define EZDP_DECODE_MAC_CONTROL_VRRP_MAC_OFFSET 5
- #define EZDP_DECODE_MAC_CONTROL_VRRP_MAC_MASK (1 << EZDP_DECODE_MAC_CONTROL_VRRP_MAC_OFFSET)
- #define EZDP_DECODE_MAC_CONTROL_IPV4_MULTICAST_SIZE 1
- #define EZDP_DECODE_MAC_CONTROL_IPV4_MULTICAST_OFFSET 6
- #define EZDP_DECODE_MAC_CONTROL_IPV4_MULTICAST_MASK (1 << EZDP_DECODE_MAC_CONTROL_IPV4_MULTICAST_OFFSET)
- #define EZDP_DECODE_MAC_CONTROL_IPV6_MULTICAST_SIZE 1
- #define EZDP_DECODE_MAC_CONTROL_IPV6_MULTICAST_OFFSET 7
- #define EZDP_DECODE_MAC_CONTROL_IPV6_MULTICAST_MASK (1 << EZDP_DECODE_MAC_CONTROL_IPV6_MULTICAST_OFFSET)
- #define EZDP_DECODE_MAC_CONTROL_USER_CONFIG0_SIZE 1
- #define EZDP_DECODE_MAC_CONTROL_USER_CONFIG0_OFFSET 8

- #define EZDP_DECODE_MAC_CONTROL_USER_CONFIG0_MASK (1 << EZDP_DECODE_MAC_CONTROL_USER_CONFIG0_OFFSET)
- #define EZDP_DECODE_MAC_CONTROL_USER_CONFIG1_SIZE 1
- #define EZDP_DECODE_MAC_CONTROL_USER_CONFIG1_OFFSET 9
- #define EZDP_DECODE_MAC_CONTROL_USER_CONFIG1_MASK (1 << EZDP_DECODE_MAC_CONTROL_USER_CONFIG1_OFFSET)
- #define EZDP_DECODE_MAC_CONTROL_USER_CONFIG2_SIZE 1
- #define EZDP_DECODE_MAC_CONTROL_USER_CONFIG2_OFFSET 10
- #define EZDP_DECODE_MAC_CONTROL_USER_CONFIG2_MASK (1 << EZDP_DECODE_MAC_CONTROL_USER_CONFIG2_OFFSET)
- #define EZDP_DECODE_MAC_CONTROL_USER_CONFIG3_SIZE 1
- #define EZDP_DECODE_MAC_CONTROL_USER_CONFIG3_OFFSET 11
- #define EZDP_DECODE_MAC_CONTROL_USER_CONFIG3_MASK (1 << EZDP_DECODE_MAC_CONTROL_USER_CONFIG3_OFFSET)
- #define EZDP_DECODE_MAC_CONTROL_SMAC_EQUALS_DMAC_SIZE 1
- #define EZDP_DECODE_MAC_CONTROL_SMAC_EQUALS_DMAC_OFFSET 12
- #define EZDP_DECODE_MAC_CONTROL_SMAC_EQUALS_DMAC_MASK (1 << EZDP_DECODE_MAC_CONTROL_SMAC_EQUALS_DMAC_OFFSET)
- #define EZDP_DECODE_MAC_CONTROL_RESERVED13_15_SIZE 3
- #define EZDP_DECODE_MAC_CONTROL_RESERVED13_15_OFFSET 13
- #define EZDP_DECODE_MAC_ERRORS_SMAC_IS_NOT_UNICAST_SIZE 1
- #define EZDP_DECODE_MAC_ERRORS_SMAC_IS_NOT_UNICAST_OFFSET 0
- #define EZDP_DECODE_MAC_ERRORS_SMAC_IS_NOT_UNICAST_MASK (1 << EZDP_DECODE_MAC_ERRORS_SMAC_IS_NOT_UNICAST_OFFSET)
- #define EZDP_DECODE_MAC_ERRORS_SMAC_IS_ZERO_SIZE 1
- #define EZDP_DECODE_MAC_ERRORS_SMAC_IS_ZERO_OFFSET 1
- #define EZDP_DECODE_MAC_ERRORS_SMAC_IS_ZERO_MASK (1 << EZDP_DECODE_MAC_ERRORS_SMAC_IS_ZERO_OFFSET)
- #define EZDP_DECODE_MAC_ERRORS_DMAC_IS_ZERO_SIZE 1
- #define EZDP_DECODE_MAC_ERRORS_DMAC_IS_ZERO_OFFSET 2
- #define EZDP_DECODE_MAC_ERRORS_DMAC_IS_ZERO_MASK (1 << EZDP_DECODE_MAC_ERRORS_DMAC_IS_ZERO_OFFSET)
- #define EZDP_DECODE_MAC_ERRORS_IP_VERSION_MISMATCH_IN_PPPOE_SIZE 1
- #define EZDP_DECODE_MAC_ERRORS_IP_VERSION_MISMATCH_IN_PPPOE_OFFSET 3
- #define EZDP_DECODE_MAC_ERRORS_IP_VERSION_MISMATCH_IN_PPPOE_MASK (1 << EZDP_DECODE_MAC_ERRORS_IP_VERSION_MISMATCH_IN_PPPOE_OFFSET)
- #define EZDP_DECODE_MAC_ERRORS_DECODE_ERROR_SIZE 1
- #define EZDP_DECODE_MAC_ERRORS_DECODE_ERROR_OFFSET 4
- #define EZDP_DECODE_MAC_ERRORS_DECODE_ERROR_MASK (1 << EZDP_DECODE_MAC_ERRORS_DECODE_ERROR_OFFSET)
- #define EZDP_DECODE_MAC_ERRORS_RESERVED5_7_SIZE 3
- #define EZDP_DECODE_MAC_ERRORS_RESERVED5_7_OFFSET 5
- #define EZDP_DECODE_MAC_PROTOCOL_TYPE_IPV4_SIZE 1
- #define EZDP_DECODE_MAC_PROTOCOL_TYPE_IPV4_OFFSET 0
- #define EZDP_DECODE_MAC_PROTOCOL_TYPE_IPV4_MASK (1 << EZDP_DECODE_MAC_PROTOCOL_TYPE_IPV4_OFFSET)
- #define EZDP_DECODE_MAC_PROTOCOL_TYPE_USER_CONFIG_VLAN0_SIZE 1
- #define EZDP_DECODE_MAC_PROTOCOL_TYPE_USER_CONFIG_VLAN0_OFFSET 1
- #define EZDP_DECODE_MAC_PROTOCOL_TYPE_USER_CONFIG_VLAN0_MASK (1 << EZDP_DECODE_MAC_PROTOCOL_TYPE_USER_CONFIG_VLAN0_OFFSET)
- #define EZDP_DECODE_MAC_PROTOCOL_TYPE_USER_CONFIG_VLAN1_SIZE 1
- #define EZDP_DECODE_MAC_PROTOCOL_TYPE_USER_CONFIG_VLAN1_OFFSET 2
- #define EZDP_DECODE_MAC_PROTOCOL_TYPE_USER_CONFIG_VLAN1_MASK (1 << EZDP_DECODE_MAC_PROTOCOL_TYPE_USER_CONFIG_VLAN1_OFFSET)
- #define EZDP_DECODE_MAC_PROTOCOL_TYPE_ARP_SIZE 1
- #define EZDP_DECODE_MAC_PROTOCOL_TYPE_ARP_OFFSET 3
- #define EZDP_DECODE_MAC_PROTOCOL_TYPE_ARP_MASK (1 << EZDP_DECODE_MAC_PROTOCOL_TYPE_ARP_OFFSET)

- #define EZDP_DECODE_MAC_PROTOCOL_TYPE_MPLS_UNICAST_SIZE 1
- #define EZDP_DECODE_MAC_PROTOCOL_TYPE_MPLS_UNICAST_OFFSET 4
- #define EZDP_DECODE_MAC_PROTOCOL_TYPE_MPLS_UNICAST_MASK (1 << EZDP_DECODE_MAC_PROTOCOL_TYPE_MPLS_UNICAST_OFFSET)
- #define EZDP_DECODE_MAC_PROTOCOL_TYPE_MPLS_MULTICAST_SIZE 1
- #define EZDP_DECODE_MAC_PROTOCOL_TYPE_MPLS_MULTICAST_OFFSET 5
- #define EZDP_DECODE_MAC_PROTOCOL_TYPE_MPLS_MULTICAST_MASK (1 << EZDP_DECODE_MAC_PROTOCOL_TYPE_MPLS_MULTICAST_OFFSET)
- #define EZDP_DECODE_MAC_PROTOCOL_TYPE_IPV6_SIZE 1
- #define EZDP_DECODE_MAC_PROTOCOL_TYPE_IPV6_OFFSET 6
- #define EZDP_DECODE_MAC_PROTOCOL_TYPE_IPV6_MASK (1 << EZDP_DECODE_MAC_PROTOCOL_TYPE_IPV6_OFFSET)
- #define EZDP_DECODE_MAC_PROTOCOL_TYPE_LENGTH_SIZE 1
- #define EZDP_DECODE_MAC_PROTOCOL_TYPE_LENGTH_OFFSET 7
- #define EZDP_DECODE_MAC_PROTOCOL_TYPE_LENGTH_MASK (1 << EZDP_DECODE_MAC_PROTOCOL_TYPE_LENGTH_OFFSET)
- #define EZDP_DECODE_MAC_PROTOCOL_TYPE_USER_CONFIG0_SIZE 1
- #define EZDP_DECODE_MAC_PROTOCOL_TYPE_USER_CONFIG0_OFFSET 8
- #define EZDP_DECODE_MAC_PROTOCOL_TYPE_USER_CONFIG0_MASK (1 << EZDP_DECODE_MAC_PROTOCOL_TYPE_USER_CONFIG0_OFFSET)
- #define EZDP_DECODE_MAC_PROTOCOL_TYPE_USER_CONFIG1_SIZE 1
- #define EZDP_DECODE_MAC_PROTOCOL_TYPE_USER_CONFIG1_OFFSET 9
- #define EZDP_DECODE_MAC_PROTOCOL_TYPE_USER_CONFIG1_MASK (1 << EZDP_DECODE_MAC_PROTOCOL_TYPE_USER_CONFIG1_OFFSET)
- #define EZDP_DECODE_MAC_PROTOCOL_TYPE_USER_CONFIG2_SIZE 1
- #define EZDP_DECODE_MAC_PROTOCOL_TYPE_USER_CONFIG2_OFFSET 10
- #define EZDP_DECODE_MAC_PROTOCOL_TYPE_USER_CONFIG2_MASK (1 << EZDP_DECODE_MAC_PROTOCOL_TYPE_USER_CONFIG2_OFFSET)
- #define EZDP_DECODE_MAC_PROTOCOL_TYPE_USER_CONFIG3_SIZE 1
- #define EZDP_DECODE_MAC_PROTOCOL_TYPE_USER_CONFIG3_OFFSET 11
- #define EZDP_DECODE_MAC_PROTOCOL_TYPE_USER_CONFIG3_MASK (1 << EZDP_DECODE_MAC_PROTOCOL_TYPE_USER_CONFIG3_OFFSET)
- #define EZDP_DECODE_MAC_PROTOCOL_TYPE_PPPOE_SESSION_SIZE 1
- #define EZDP_DECODE_MAC_PROTOCOL_TYPE_PPPOE_SESSION_OFFSET 12
- #define EZDP_DECODE_MAC_PROTOCOL_TYPE_PPPOE_SESSION_MASK (1 << EZDP_DECODE_MAC_PROTOCOL_TYPE_PPPOE_SESSION_OFFSET)
- #define EZDP_DECODE_MAC_PROTOCOL_TYPE_PPPOE_DISCOVERY_SIZE 1
- #define EZDP_DECODE_MAC_PROTOCOL_TYPE_PPPOE_DISCOVERY_OFFSET 13
- #define EZDP_DECODE_MAC_PROTOCOL_TYPE_PPPOE_DISCOVERY_MASK (1 << EZDP_DECODE_MAC_PROTOCOL_TYPE_PPPOE_DISCOVERY_OFFSET)
- #define EZDP_DECODE_MAC_PROTOCOL_TYPE_USER_CONFIG_VLAN2_SIZE 1
- #define EZDP_DECODE_MAC_PROTOCOL_TYPE_USER_CONFIG_VLAN2_OFFSET 14
- #define EZDP_DECODE_MAC_PROTOCOL_TYPE_USER_CONFIG_VLAN2_MASK (1 << EZDP_DECODE_MAC_PROTOCOL_TYPE_USER_CONFIG_VLAN2_OFFSET)
- #define EZDP_DECODE_MAC_PROTOCOL_TYPE_RESERVED_15_SIZE 1
- #define EZDP_DECODE_MAC_PROTOCOL_TYPE_RESERVED_15_OFFSET 15
- #define EZDP_DECODE_TCP_ERRORS_DATA_OFFSET_LT_5_SIZE 1
- #define EZDP_DECODE_TCP_ERRORS_DATA_OFFSET_LT_5_OFFSET 0
- #define EZDP_DECODE_TCP_ERRORS_DATA_OFFSET_LT_5_MASK (1 << EZDP_DECODE_TCP_ERRORS_DATA_OFFSET_LT_5_OFFSET)
- #define EZDP_DECODE_TCP_ERRORS_SYN_AND_FIN_EQ_1_SIZE 1
- #define EZDP_DECODE_TCP_ERRORS_SYN_AND_FIN_EQ_1_OFFSET 1
- #define EZDP_DECODE_TCP_ERRORS_SYN_AND_FIN_EQ_1_MASK (1 << EZDP_DECODE_TCP_ERRORS_SYN_AND_FIN_EQ_1_OFFSET)
- #define EZDP_DECODE_TCP_ERRORS_DECODE_ERROR_SIZE 1
- #define EZDP_DECODE_TCP_ERRORS_DECODE_ERROR_OFFSET 2
- #define EZDP_DECODE_TCP_ERRORS_DECODE_ERROR_MASK (1 << EZDP_DECODE_TCP_ERRORS_DECODE_ERROR_OFFSET)

- #define EZDP_DECODE_TCP_ERRORS_RESERVED3_7_SIZE 5
- #define EZDP_DECODE_TCP_ERRORS_RESERVED3_7_OFFSET 3
- #define EZDP_DECODE_TCP_RETVAL_ERROR_CODES_SIZE 8
- #define EZDP_DECODE_TCP_RETVAL_ERROR_CODES_OFFSET 0
- #define EZDP_DECODE_TCP_RETVAL_OPTIONS_EXIST_SIZE 1
- #define EZDP_DECODE_TCP_RETVAL_OPTIONS_EXIST_OFFSET 8
- #define EZDP_DECODE_TCP_RETVAL_OPTIONS_EXIST_MASK (1 << EZDP_DECODE_TCP_RETVAL_OPTIONS_EXIST_OFFSET)
- #define EZDP_DECODE_TCP_RETVAL_RESERVED9_15_SIZE 7
- #define EZDP_DECODE_TCP_RETVAL_RESERVED9_15_OFFSET 9
- #define EZDP_DECODE_TCP_RETVAL_DATA_OFFSET_SIZE 6
- #define EZDP_DECODE_TCP_RETVAL_DATA_OFFSET_OFFSET 16
- #define EZDP_DECODE_TCP_RETVAL_RESERVED22_23_SIZE 2
- #define EZDP_DECODE_TCP_RETVAL_RESERVED22_23_OFFSET 22
- #define EZDP_DECODE_TCP_RETVAL_RESERVED24_31_SIZE 8
- #define EZDP_DECODE_TCP_RETVAL_RESERVED24_31_OFFSET 24
- #define EZDP_DECODE_IP_PROTOCOL_RETVAL_TCP_SIZE 1
- #define EZDP_DECODE_IP_PROTOCOL_RETVAL_TCP_OFFSET 0
- #define EZDP_DECODE_IP_PROTOCOL_RETVAL_TCP_MASK (1 << EZDP_DECODE_IP_PROTOCOL_RETVAL_TCP_OFFSET)
- #define EZDP_DECODE_IP_PROTOCOL_RETVAL_UDP_SIZE 1
- #define EZDP_DECODE_IP_PROTOCOL_RETVAL_UDP_OFFSET 1
- #define EZDP_DECODE_IP_PROTOCOL_RETVAL_UDP_MASK (1 << EZDP_DECODE_IP_PROTOCOL_RETVAL_UDP_OFFSET)
- #define EZDP_DECODE_IP_PROTOCOL_RETVAL_MPLS_SIZE 1
- #define EZDP_DECODE_IP_PROTOCOL_RETVAL_MPLS_OFFSET 2
- #define EZDP_DECODE_IP_PROTOCOL_RETVAL_MPLS_MASK (1 << EZDP_DECODE_IP_PROTOCOL_RETVAL_MPLS_OFFSET)
- #define EZDP_DECODE_IP_PROTOCOL_RETVAL_GRE_SIZE 1
- #define EZDP_DECODE_IP_PROTOCOL_RETVAL_GRE_OFFSET 3
- #define EZDP_DECODE_IP_PROTOCOL_RETVAL_GRE_MASK (1 << EZDP_DECODE_IP_PROTOCOL_RETVAL_GRE_OFFSET)
- #define EZDP_DECODE_IP_PROTOCOL_RETVAL_IPV4_SIZE 1
- #define EZDP_DECODE_IP_PROTOCOL_RETVAL_IPV4_OFFSET 4
- #define EZDP_DECODE_IP_PROTOCOL_RETVAL_IPV4_MASK (1 << EZDP_DECODE_IP_PROTOCOL_RETVAL_IPV4_OFFSET)
- #define EZDP_DECODE_IP_PROTOCOL_RETVAL_IPV6_SIZE 1
- #define EZDP_DECODE_IP_PROTOCOL_RETVAL_IPV6_OFFSET 5
- #define EZDP_DECODE_IP_PROTOCOL_RETVAL_IPV6_MASK (1 << EZDP_DECODE_IP_PROTOCOL_RETVAL_IPV6_OFFSET)
- #define EZDP_DECODE_IP_PROTOCOL_RETVAL_ICMP_IGMP_SIZE 1
- #define EZDP_DECODE_IP_PROTOCOL_RETVAL_ICMP_IGMP_OFFSET 6
- #define EZDP_DECODE_IP_PROTOCOL_RETVAL_ICMP_IGMP_MASK (1 << EZDP_DECODE_IP_PROTOCOL_RETVAL_ICMP_IGMP_OFFSET)
- #define EZDP_DECODE_IP_PROTOCOL_RETVAL_DEF_IP_PROT_0_SIZE 1
- #define EZDP_DECODE_IP_PROTOCOL_RETVAL_DEF_IP_PROT_0_OFFSET 7
- #define EZDP_DECODE_IP_PROTOCOL_RETVAL_DEF_IP_PROT_0_MASK (1 << EZDP_DECODE_IP_PROTOCOL_RETVAL_DEF_IP_PROT_0_OFFSET)
- #define EZDP_DECODE_IP_PROTOCOL_RETVAL_DEF_IP_PROT_1_SIZE 1
- #define EZDP_DECODE_IP_PROTOCOL_RETVAL_DEF_IP_PROT_1_OFFSET 8
- #define EZDP_DECODE_IP_PROTOCOL_RETVAL_DEF_IP_PROT_1_MASK (1 << EZDP_DECODE_IP_PROTOCOL_RETVAL_DEF_IP_PROT_1_OFFSET)
- #define EZDP_DECODE_IP_PROTOCOL_RETVAL_DEF_IP_PROT_2_SIZE 1
- #define EZDP_DECODE_IP_PROTOCOL_RETVAL_DEF_IP_PROT_2_OFFSET 9
- #define EZDP_DECODE_IP_PROTOCOL_RETVAL_DEF_IP_PROT_2_MASK (1 << EZDP_DECODE_IP_PROTOCOL_RETVAL_DEF_IP_PROT_2_OFFSET)
- #define EZDP_DECODE_IP_PROTOCOL_RETVAL_DEF_IP_PROT_3_SIZE 1

- #define EZDP_DECODE_IP_PROTOCOL_RETVAL_DEF_IP_PROT_3_OFFSET 10
- #define EZDP_DECODE_IP_PROTOCOL_RETVAL_DEF_IP_PROT_3_MASK (1 << EZDP_DECODE_IP_PROTOCOL_RETVAL_DEF_IP_PROT_3_OFFSET)
- #define EZDP_DECODE_IP_PROTOCOL_RETVAL_ESP_PROT_SIZE 1
- #define EZDP_DECODE_IP_PROTOCOL_RETVAL_ESP_PROT_OFFSET 11
- #define EZDP_DECODE_IP_PROTOCOL_RETVAL_ESP_PROT_MASK (1 << EZDP_DECODE_IP_PROTOCOL_RETVAL_ESP_PROT_OFFSET)
- #define EZDP_DECODE_IP_PROTOCOL_RETVAL_AH_PROT_SIZE 1
- #define EZDP_DECODE_IP_PROTOCOL_RETVAL_AH_PROT_OFFSET 12
- #define EZDP_DECODE_IP_PROTOCOL_RETVAL_AH_PROT_MASK (1 << EZDP_DECODE_IP_PROTOCOL_RETVAL_AH_PROT_OFFSET)
- #define EZDP_DECODE_IP_PROTOCOL_RETVAL_OTHER_SIZE 1
- #define EZDP_DECODE_IP_PROTOCOL_RETVAL_OTHER_OFFSET 13
- #define EZDP_DECODE_IP_PROTOCOL_RETVAL_OTHER_MASK (1 << EZDP_DECODE_IP_PROTOCOL_RETVAL_OTHER_OFFSET)
- #define EZDP_DECODE_IP_PROTOCOL_RETVAL_RESERVED14_31_SIZE 18
- #define EZDP_DECODE_IP_PROTOCOL_RETVAL_RESERVED14_31_OFFSET 14
- #define EZDP_DECODE_ETH_TYPE_RETVAL_IPV4_SIZE 1
- #define EZDP_DECODE_ETH_TYPE_RETVAL_IPV4_OFFSET 0
- #define EZDP_DECODE_ETH_TYPE_RETVAL_IPV4_MASK (1 << EZDP_DECODE_ETH_TYPE_RETVAL_IPV4_OFFSET)
- #define EZDP_DECODE_ETH_TYPE_RETVAL_ETH_8100_SIZE 1
- #define EZDP_DECODE_ETH_TYPE_RETVAL_ETH_8100_OFFSET 1
- #define EZDP_DECODE_ETH_TYPE_RETVAL_ETH_8100_MASK (1 << EZDP_DECODE_ETH_TYPE_RETVAL_ETH_8100_OFFSET)
- #define EZDP_DECODE_ETH_TYPE_RETVAL_ETH_88A8_SIZE 1
- #define EZDP_DECODE_ETH_TYPE_RETVAL_ETH_88A8_OFFSET 2
- #define EZDP_DECODE_ETH_TYPE_RETVAL_ETH_88A8_MASK (1 << EZDP_DECODE_ETH_TYPE_RETVAL_ETH_88A8_OFFSET)
- #define EZDP_DECODE_ETH_TYPE_RETVAL_ARP_SIZE 1
- #define EZDP_DECODE_ETH_TYPE_RETVAL_ARP_OFFSET 3
- #define EZDP_DECODE_ETH_TYPE_RETVAL_ARP_MASK (1 << EZDP_DECODE_ETH_TYPE_RETVAL_ARP_OFFSET)
- #define EZDP_DECODE_ETH_TYPE_RETVAL_MPLS_UNICAST_SIZE 1
- #define EZDP_DECODE_ETH_TYPE_RETVAL_MPLS_UNICAST_OFFSET 4
- #define EZDP_DECODE_ETH_TYPE_RETVAL_MPLS_UNICAST_MASK (1 << EZDP_DECODE_ETH_TYPE_RETVAL_MPLS_UNICAST_OFFSET)
- #define EZDP_DECODE_ETH_TYPE_RETVAL_MPLS_MULTICAST_SIZE 1
- #define EZDP_DECODE_ETH_TYPE_RETVAL_MPLS_MULTICAST_OFFSET 5
- #define EZDP_DECODE_ETH_TYPE_RETVAL_MPLS_MULTICAST_MASK (1 << EZDP_DECODE_ETH_TYPE_RETVAL_MPLS_MULTICAST_OFFSET)
- #define EZDP_DECODE_ETH_TYPE_RETVAL_IPV6_SIZE 1
- #define EZDP_DECODE_ETH_TYPE_RETVAL_IPV6_OFFSET 6
- #define EZDP_DECODE_ETH_TYPE_RETVAL_IPV6_MASK (1 << EZDP_DECODE_ETH_TYPE_RETVAL_IPV6_OFFSET)
- #define EZDP_DECODE_ETH_TYPE_RETVAL_LENGTH_SIZE 1
- #define EZDP_DECODE_ETH_TYPE_RETVAL_LENGTH_OFFSET 7
- #define EZDP_DECODE_ETH_TYPE_RETVAL_LENGTH_MASK (1 << EZDP_DECODE_ETH_TYPE_RETVAL_LENGTH_OFFSET)
- #define EZDP_DECODE_ETH_TYPE_RETVAL_USER_DEF0_SIZE 1
- #define EZDP_DECODE_ETH_TYPE_RETVAL_USER_DEF0_OFFSET 8
- #define EZDP_DECODE_ETH_TYPE_RETVAL_USER_DEF0_MASK (1 << EZDP_DECODE_ETH_TYPE_RETVAL_USER_DEF0_OFFSET)
- #define EZDP_DECODE_ETH_TYPE_RETVAL_USER_DEF1_SIZE 1
- #define EZDP_DECODE_ETH_TYPE_RETVAL_USER_DEF1_OFFSET 9
- #define EZDP_DECODE_ETH_TYPE_RETVAL_USER_DEF1_MASK (1 << EZDP_DECODE_ETH_TYPE_RETVAL_USER_DEF1_OFFSET)
- #define EZDP_DECODE_ETH_TYPE_RETVAL_PPPOE_SESSION_SIZE 1

- #define EZDP_DECODE_ETH_TYPE_RETVAL_PPPOE_SESSION_OFFSET 10
- #define EZDP_DECODE_ETH_TYPE_RETVAL_PPPOE_SESSION_MASK (1 << EZDP_DECODE_ETH_TYPE_RETVAL_PPPOE_SESSION_OFFSET)
- #define EZDP_DECODE_ETH_TYPE_RETVAL_PPPOE_DISCOVERY_SIZE 1
- #define EZDP_DECODE_ETH_TYPE_RETVAL_PPPOE_DISCOVERY_OFFSET 11
- #define EZDP_DECODE_ETH_TYPE_RETVAL_PPPOE_DISCOVERY_MASK (1 << EZDP_DECODE_ETH_TYPE_RETVAL_PPPOE_DISCOVERY_OFFSET)
- #define EZDP_DECODE_ETH_TYPE_RETVAL_OTHER_SIZE 1
- #define EZDP_DECODE_ETH_TYPE_RETVAL_OTHER_OFFSET 12
- #define EZDP_DECODE_ETH_TYPE_RETVAL_OTHER_MASK (1 << EZDP_DECODE_ETH_TYPE_RETVAL_OTHER_OFFSET)
- #define EZDP_DECODE_ETH_TYPE_RETVAL_RESERVED13_31_SIZE 19
- #define EZDP_DECODE_ETH_TYPE_RETVAL_RESERVED13_31_OFFSET 13
- #define EZDP_DECODE_IPV4_RETVAL_OPTION_EXIST_SIZE 1
- #define EZDP_DECODE_IPV4_RETVAL_OPTION_EXIST_OFFSET 0
- #define EZDP_DECODE_IPV4_RETVAL_OPTION_EXIST_MASK (1 << EZDP_DECODE_IPV4_RETVAL_OPTION_EXIST_OFFSET)
- #define EZDP_DECODE_IPV4_RETVAL_USER_CONFIG_SIP_SIZE 1
- #define EZDP_DECODE_IPV4_RETVAL_USER_CONFIG_SIP_OFFSET 1
- #define EZDP_DECODE_IPV4_RETVAL_USER_CONFIG_SIP_MASK (1 << EZDP_DECODE_IPV4_RETVAL_USER_CONFIG_SIP_OFFSET)
- #define EZDP_DECODE_IPV4_RETVAL_RESERVED_2_6_SIZE 5
- #define EZDP_DECODE_IPV4_RETVAL_RESERVED_2_6_OFFSET 2
- #define EZDP_DECODE_IPV4_RETVAL_FIRST_FRAGMENT_SIZE 1
- #define EZDP_DECODE_IPV4_RETVAL_FIRST_FRAGMENT_OFFSET 7
- #define EZDP_DECODE_IPV4_RETVAL_FIRST_FRAGMENT_MASK (1 << EZDP_DECODE_IPV4_RETVAL_FIRST_FRAGMENT_OFFSET)
- #define EZDP_DECODE_IPV4_RETVAL_ERROR_CODES_SIZE 16
- #define EZDP_DECODE_IPV4_RETVAL_ERROR_CODES_OFFSET 8
- #define EZDP_DECODE_IPV4_RETVAL_CONTROL_SIZE 8
- #define EZDP_DECODE_IPV4_RETVAL_CONTROL_OFFSET 24
- #define EZDP_DECODE_IPV4_RESULT_OPTION_EXIST_SIZE 1
- #define EZDP_DECODE_IPV4_RESULT_OPTION_EXIST_OFFSET 0
- #define EZDP_DECODE_IPV4_RESULT_OPTION_EXIST_WORD_SELECT 0
- #define EZDP_DECODE_IPV4_RESULT_OPTION_EXIST_WORD_OFFSET 0
- #define EZDP_DECODE_IPV4_RESULT_OPTION_EXIST_MASK (1 << EZDP_DECODE_IPV4_RESULT_OPTION_EXIST_WORD_OFFSET)
- #define EZDP_DECODE_IPV4_RESULT_USER_CONFIG_SIP_SIZE 1
- #define EZDP_DECODE_IPV4_RESULT_USER_CONFIG_SIP_OFFSET 1
- #define EZDP_DECODE_IPV4_RESULT_USER_CONFIG_SIP_WORD_SELECT 0
- #define EZDP_DECODE_IPV4_RESULT_USER_CONFIG_SIP_WORD_OFFSET 1
- #define EZDP_DECODE_IPV4_RESULT_USER_CONFIG_SIP_MASK (1 << EZDP_DECODE_IPV4_RESULT_USER_CONFIG_SIP_WORD_OFFSET)
- #define EZDP_DECODE_IPV4_RESULT_RESERVED_2_6_SIZE 5
- #define EZDP_DECODE_IPV4_RESULT_RESERVED_2_6_OFFSET 2
- #define EZDP_DECODE_IPV4_RESULT_FIRST_FRAGMENT_SIZE 1
- #define EZDP_DECODE_IPV4_RESULT_FIRST_FRAGMENT_OFFSET 7
- #define EZDP_DECODE_IPV4_RESULT_FIRST_FRAGMENT_WORD_SELECT 0
- #define EZDP_DECODE_IPV4_RESULT_FIRST_FRAGMENT_WORD_OFFSET 7
- #define EZDP_DECODE_IPV4_RESULT_FIRST_FRAGMENT_MASK (1 << EZDP_DECODE_IPV4_RESULT_FIRST_FRAGMENT_WORD_OFFSET)
- #define EZDP_DECODE_IPV4_RESULT_ERROR_CODES_SIZE 16
- #define EZDP_DECODE_IPV4_RESULT_ERROR_CODES_OFFSET 8
- #define EZDP_DECODE_IPV4_RESULT_ERROR_CODES_WORD_SELECT 0
- #define EZDP_DECODE_IPV4_RESULT_ERROR_CODES_WORD_OFFSET 8
- #define EZDP_DECODE_IPV4_RESULT_CONTROL_SIZE 8
- #define EZDP_DECODE_IPV4_RESULT_CONTROL_OFFSET 24

- #define EZDP_DECODE_IPV4_RESULT_CONTROL_WORD_SELECT 0
- #define EZDP_DECODE_IPV4_RESULT_CONTROL_WORD_OFFSET 24
- #define EZDP_DECODE_IPV4_RESULT_SIP_DIP_HASH_SIZE 16
- #define EZDP_DECODE_IPV4_RESULT_SIP_DIP_HASH_OFFSET 32
- #define EZDP_DECODE_IPV4_RESULT_SIP_DIP_HASH_WORD_SELECT 1
- #define EZDP_DECODE_IPV4_RESULT_SIP_DIP_HASH_WORD_OFFSET 0
- #define EZDP_DECODE_IPV4_RESULT_NEXT_PROTOCOL_SIZE 8
- #define EZDP_DECODE_IPV4_RESULT_NEXT_PROTOCOL_OFFSET 48
- #define EZDP_DECODE_IPV4_RESULT_NEXT_PROTOCOL_WORD_SELECT 1
- #define EZDP_DECODE_IPV4_RESULT_NEXT_PROTOCOL_WORD_OFFSET 16
- #define EZDP_DECODE_IPV4_RESULT_RESERVED_56_63_SIZE 8
- #define EZDP_DECODE_IPV4_RESULT_RESERVED_56_63_OFFSET 56
- #define EZDP_DECODE_IPV4_RESULT_WORD_COUNT 2
- #define EZDP_DECODE_IPV6_RETVAL_CONTROL_SIZE 8
- #define EZDP_DECODE_IPV6_RETVAL_CONTROL_OFFSET 0
- #define EZDP_DECODE_IPV6_RETVAL_OPTIONS_EXIST_SIZE 1
- #define EZDP_DECODE_IPV6_RETVAL_OPTIONS_EXIST_OFFSET 8
- #define EZDP_DECODE_IPV6_RETVAL_OPTIONS_EXIST_MASK (1 << EZDP_DECODE_IPV6_RETVAL_OPTIONS_EXIST_OFFSET)
- #define EZDP_DECODE_IPV6_RETVAL_RESERVED9_11_SIZE 3
- #define EZDP_DECODE_IPV6_RETVAL_RESERVED9_11_OFFSET 9
- #define EZDP_DECODE_IPV6_RETVAL_LINK_LOCAL_ADDRESS_SIZE 1
- #define EZDP_DECODE_IPV6_RETVAL_LINK_LOCAL_ADDRESS_OFFSET 12
- #define EZDP_DECODE_IPV6_RETVAL_LINK_LOCAL_ADDRESS_MASK (1 << EZDP_DECODE_IPV6_RETVAL_LINK_LOCAL_ADDRESS_OFFSET)
- #define EZDP_DECODE_IPV6_RETVAL_SITE_LOCAL_ADDRESS_SIZE 1
- #define EZDP_DECODE_IPV6_RETVAL_SITE_LOCAL_ADDRESS_OFFSET 13
- #define EZDP_DECODE_IPV6_RETVAL_SITE_LOCAL_ADDRESS_MASK (1 << EZDP_DECODE_IPV6_RETVAL_SITE_LOCAL_ADDRESS_OFFSET)
- #define EZDP_DECODE_IPV6_RETVAL_GLOBAL_ADDRESSES_SIZE 1
- #define EZDP_DECODE_IPV6_RETVAL_GLOBAL_ADDRESSES_OFFSET 14
- #define EZDP_DECODE_IPV6_RETVAL_GLOBAL_ADDRESSES_MASK (1 << EZDP_DECODE_IPV6_RETVAL_GLOBAL_ADDRESSES_OFFSET)
- #define EZDP_DECODE_IPV6_RETVAL_RESERVED_15_SIZE 1
- #define EZDP_DECODE_IPV6_RETVAL_RESERVED_15_OFFSET 15
- #define EZDP_DECODE_IPV6_RETVAL_ERROR_CODES_SIZE 16
- #define EZDP_DECODE_IPV6_RETVAL_ERROR_CODES_OFFSET 16
- #define EZDP_DECODE_IPV6_RESULT_CONTROL_SIZE 8
- #define EZDP_DECODE_IPV6_RESULT_CONTROL_OFFSET 0
- #define EZDP_DECODE_IPV6_RESULT_CONTROL_WORD_SELECT 0
- #define EZDP_DECODE_IPV6_RESULT_CONTROL_WORD_OFFSET 0
- #define EZDP_DECODE_IPV6_RESULT_OPTIONS_EXIST_SIZE 1
- #define EZDP_DECODE_IPV6_RESULT_OPTIONS_EXIST_OFFSET 8
- #define EZDP_DECODE_IPV6_RESULT_OPTIONS_EXIST_WORD_SELECT 0
- #define EZDP_DECODE_IPV6_RESULT_OPTIONS_EXIST_WORD_OFFSET 8
- #define EZDP_DECODE_IPV6_RESULT_OPTIONS_EXIST_MASK (1 << EZDP_DECODE_IPV6_RESULT_OPTIONS_EXIST_WORD_OFFSET)
- #define EZDP_DECODE_IPV6_RESULT_RESERVED9_11_SIZE 3
- #define EZDP_DECODE_IPV6_RESULT_RESERVED9_11_OFFSET 9
- #define EZDP_DECODE_IPV6_RESULT_LINK_LOCAL_ADDRESS_SIZE 1
- #define EZDP_DECODE_IPV6_RESULT_LINK_LOCAL_ADDRESS_OFFSET 12
- #define EZDP_DECODE_IPV6_RESULT_LINK_LOCAL_ADDRESS_WORD_SELECT 0
- #define EZDP_DECODE_IPV6_RESULT_LINK_LOCAL_ADDRESS_WORD_OFFSET 12
- #define EZDP_DECODE_IPV6_RESULT_LINK_LOCAL_ADDRESS_MASK (1 << EZDP_DECODE_IPV6_RESULT_LINK_LOCAL_ADDRESS_WORD_OFFSET)
- #define EZDP_DECODE_IPV6_RESULT_SITE_LOCAL_ADDRESS_SIZE 1
- #define EZDP_DECODE_IPV6_RESULT_SITE_LOCAL_ADDRESS_OFFSET 13

- #define EZDP_DECODE_IPV6_RESULT_SITE_LOCAL_ADDRESS_WORD_SELECT 0
- #define EZDP_DECODE_IPV6_RESULT_SITE_LOCAL_ADDRESS_WORD_OFFSET 13
- #define EZDP_DECODE_IPV6_RESULT_SITE_LOCAL_ADDRESS_MASK (1 << EZDP_DECODE_IPV6_RESULT_SITE_LOCAL_ADDRESS_WORD_OFFSET)
- #define EZDP_DECODE_IPV6_RESULT_GLOBAL_ADDRESSES_SIZE 1
- #define EZDP_DECODE_IPV6_RESULT_GLOBAL_ADDRESSES_OFFSET 14
- #define EZDP_DECODE_IPV6_RESULT_GLOBAL_ADDRESSES_WORD_SELECT 0
- #define EZDP_DECODE_IPV6_RESULT_GLOBAL_ADDRESSES_WORD_OFFSET 14
- #define EZDP_DECODE_IPV6_RESULT_GLOBAL_ADDRESSES_MASK (1 << EZDP_DECODE_IPV6_RESULT_GLOBAL_ADDRESSES_WORD_OFFSET)
- #define EZDP_DECODE_IPV6_RESULT_RESERVED_15_SIZE 1
- #define EZDP_DECODE_IPV6_RESULT_RESERVED_15_OFFSET 15
- #define EZDP_DECODE_IPV6_RESULT_ERROR_CODES_SIZE 16
- #define EZDP_DECODE_IPV6_RESULT_ERROR_CODES_OFFSET 16
- #define EZDP_DECODE_IPV6_RESULT_ERROR_CODES_WORD_SELECT 0
- #define EZDP_DECODE_IPV6_RESULT_ERROR_CODES_WORD_OFFSET 16
- #define EZDP_DECODE_IPV6_RESULT_SIP_DIP_HASH_SIZE 16
- #define EZDP_DECODE_IPV6_RESULT_SIP_DIP_HASH_OFFSET 32
- #define EZDP_DECODE_IPV6_RESULT_SIP_DIP_HASH_WORD_SELECT 1
- #define EZDP_DECODE_IPV6_RESULT_SIP_DIP_HASH_WORD_OFFSET 0
- #define EZDP_DECODE_IPV6_RESULT_NEXT_PROTOCOL_SIZE 8
- #define EZDP_DECODE_IPV6_RESULT_NEXT_PROTOCOL_OFFSET 48
- #define EZDP_DECODE_IPV6_RESULT_NEXT_PROTOCOL_WORD_SELECT 1
- #define EZDP_DECODE_IPV6_RESULT_NEXT_PROTOCOL_WORD_OFFSET 16
- #define EZDP_DECODE_IPV6_RESULT_RESERVED_56_63_SIZE 8
- #define EZDP_DECODE_IPV6_RESULT_RESERVED_56_63_OFFSET 56
- #define EZDP_DECODE_IPV6_RESULT_WORD_COUNT 2
- #define EZDP_DECODE_MAC_RETVAL_CONTROL_SIZE 16
- #define EZDP_DECODE_MAC_RETVAL_CONTROL_OFFSET 0
- #define EZDP_DECODE_MAC_RETVAL_ERROR_CODES_SIZE 8
- #define EZDP_DECODE_MAC_RETVAL_ERROR_CODES_OFFSET 16
- #define EZDP_DECODE_MAC_RETVAL_IPV4_IN_PPPOE_SIZE 1
- #define EZDP_DECODE_MAC_RETVAL_IPV4_IN_PPPOE_OFFSET 24
- #define EZDP_DECODE_MAC_RETVAL_IPV4_IN_PPPOE_MASK (1 << EZDP_DECODE_MAC_RETVAL_IPV4_IN_PPPOE_OFFSET)
- #define EZDP_DECODE_MAC_RETVAL_IPV6_IN_PPPOE_SIZE 1
- #define EZDP_DECODE_MAC_RETVAL_IPV6_IN_PPPOE_OFFSET 25
- #define EZDP_DECODE_MAC_RETVAL_IPV6_IN_PPPOE_MASK (1 << EZDP_DECODE_MAC_RETVAL_IPV6_IN_PPPOE_OFFSET)
- #define EZDP_DECODE_MAC_RETVAL_RESERVED26_27_SIZE 2
- #define EZDP_DECODE_MAC_RETVAL_RESERVED26_27_OFFSET 26
- #define EZDP_DECODE_MAC_RETVAL_NUMBER_OF_TAGS_SIZE 3
- #define EZDP_DECODE_MAC_RETVAL_NUMBER_OF_TAGS_OFFSET 28
- #define EZDP_DECODE_MAC_RETVAL_RESERVED_31_SIZE 1
- #define EZDP_DECODE_MAC_RETVAL_RESERVED_31_OFFSET 31
- #define EZDP_DECODE_MAC_RESULT_CONTROL_SIZE 16
- #define EZDP_DECODE_MAC_RESULT_CONTROL_OFFSET 0
- #define EZDP_DECODE_MAC_RESULT_CONTROL_WORD_SELECT 0
- #define EZDP_DECODE_MAC_RESULT_CONTROL_WORD_OFFSET 0
- #define EZDP_DECODE_MAC_RESULT_ERROR_CODES_SIZE 8
- #define EZDP_DECODE_MAC_RESULT_ERROR_CODES_OFFSET 16
- #define EZDP_DECODE_MAC_RESULT_ERROR_CODES_WORD_SELECT 0
- #define EZDP_DECODE_MAC_RESULT_ERROR_CODES_WORD_OFFSET 16
- #define EZDP_DECODE_MAC_RESULT_IPV4_IN_PPPOE_SIZE 1
- #define EZDP_DECODE_MAC_RESULT_IPV4_IN_PPPOE_OFFSET 24
- #define EZDP_DECODE_MAC_RESULT_IPV4_IN_PPPOE_WORD_SELECT 0
- #define EZDP_DECODE_MAC_RESULT_IPV4_IN_PPPOE_WORD_OFFSET 24

- #define EZDP_DECODE_MAC_RESULT_IPV4_IN_PPPOE_MASK (1 << EZDP_DECODE_MAC_RESULT_IPV4_IN_PPPOE_WORD_OFFSET)
- #define EZDP_DECODE_MAC_RESULT_IPV6_IN_PPPOE_SIZE 1
- #define EZDP_DECODE_MAC_RESULT_IPV6_IN_PPPOE_OFFSET 25
- #define EZDP_DECODE_MAC_RESULT_IPV6_IN_PPPOE_WORD_SELECT 0
- #define EZDP_DECODE_MAC_RESULT_IPV6_IN_PPPOE_WORD_OFFSET 25
- #define EZDP_DECODE_MAC_RESULT_IPV6_IN_PPPOE_MASK (1 << EZDP_DECODE_MAC_RESULT_IPV6_IN_PPPOE_WORD_OFFSET)
- #define EZDP_DECODE_MAC_RESULT_RESERVED26_27_SIZE 2
- #define EZDP_DECODE_MAC_RESULT_RESERVED26_27_OFFSET 26
- #define EZDP_DECODE_MAC_RESULT_NUMBER_OF_TAGS_SIZE 3
- #define EZDP_DECODE_MAC_RESULT_NUMBER_OF_TAGS_OFFSET 28
- #define EZDP_DECODE_MAC_RESULT_NUMBER_OF_TAGS_WORD_SELECT 0
- #define EZDP_DECODE_MAC_RESULT_NUMBER_OF_TAGS_WORD_OFFSET 28
- #define EZDP_DECODE_MAC_RESULT_RESERVED_31_SIZE 1
- #define EZDP_DECODE_MAC_RESULT_RESERVED_31_OFFSET 31
- #define EZDP_DECODE_MAC_RESULT_TAG1_PROTOCOL_TYPE_SIZE 16
- #define EZDP_DECODE_MAC_RESULT_TAG1_PROTOCOL_TYPE_OFFSET 32
- #define EZDP_DECODE_MAC_RESULT_TAG1_PROTOCOL_TYPE_WORD_SELECT 1
- #define EZDP_DECODE_MAC_RESULT_TAG1_PROTOCOL_TYPE_WORD_OFFSET 0
- #define EZDP_DECODE_MAC_RESULT_TAG2_PROTOCOL_TYPE_SIZE 16
- #define EZDP_DECODE_MAC_RESULT_TAG2_PROTOCOL_TYPE_OFFSET 48
- #define EZDP_DECODE_MAC_RESULT_TAG2_PROTOCOL_TYPE_WORD_SELECT 1
- #define EZDP_DECODE_MAC_RESULT_TAG2_PROTOCOL_TYPE_WORD_OFFSET 16
- #define EZDP_DECODE_MAC_RESULT_TAG3_PROTOCOL_TYPE_SIZE 16
- #define EZDP_DECODE_MAC_RESULT_TAG3_PROTOCOL_TYPE_OFFSET 64
- #define EZDP_DECODE_MAC_RESULT_TAG3_PROTOCOL_TYPE_WORD_SELECT 2
- #define EZDP_DECODE_MAC_RESULT_TAG3_PROTOCOL_TYPE_WORD_OFFSET 0
- #define EZDP_DECODE_MAC_RESULT_LAST_TAG_PROTOCOL_TYPE_SIZE 16
- #define EZDP_DECODE_MAC_RESULT_LAST_TAG_PROTOCOL_TYPE_OFFSET 80
- #define EZDP_DECODE_MAC_RESULT_LAST_TAG_PROTOCOL_TYPE_WORD_SELECT 2
- #define EZDP_DECODE_MAC_RESULT_LAST_TAG_PROTOCOL_TYPE_WORD_OFFSET 16
- #define EZDP_DECODE_MAC_RESULT_DA_SA_HASH_SIZE 16
- #define EZDP_DECODE_MAC_RESULT_DA_SA_HASH_OFFSET 96
- #define EZDP_DECODE_MAC_RESULT_DA_SA_HASH_WORD_SELECT 3
- #define EZDP_DECODE_MAC_RESULT_DA_SA_HASH_WORD_OFFSET 0
- #define EZDP_DECODE_MAC_RESULT_LAYER2_SIZE_SIZE 8
- #define EZDP_DECODE_MAC_RESULT_LAYER2_SIZE_OFFSET 112
- #define EZDP_DECODE_MAC_RESULT_LAYER2_SIZE_WORD_SELECT 3
- #define EZDP_DECODE_MAC_RESULT_LAYER2_SIZE_WORD_OFFSET 16
- #define EZDP_DECODE_MAC_RESULT_RESERVED120_127_SIZE 8
- #define EZDP_DECODE_MAC_RESULT_RESERVED120_127_OFFSET 120
- #define EZDP_DECODE_MAC_RESULT_WORD_COUNT 4
- #define EZDP_DECODE_MPLS_RETVAL_DECODE_ERROR_SIZE 1
- #define EZDP_DECODE_MPLS_RETVAL_DECODE_ERROR_OFFSET 0
- #define EZDP_DECODE_MPLS_RETVAL_DECODE_ERROR_MASK (1 << EZDP_DECODE_MPLS_RETVAL_DECODE_ERROR_OFFSET)
- #define EZDP_DECODE_MPLS_RETVAL_RESERVED1_7_SIZE 7
- #define EZDP_DECODE_MPLS_RETVAL_RESERVED1_7_OFFSET 1
- #define EZDP_DECODE_MPLS_RETVAL_LAST_ENTRY_IN_STACK_SIZE 2
- #define EZDP_DECODE_MPLS_RETVAL_LAST_ENTRY_IN_STACK_OFFSET 8
- #define EZDP_DECODE_MPLS_RETVAL_RESERVED10_15_SIZE 6
- #define EZDP_DECODE_MPLS_RETVAL_RESERVED10_15_OFFSET 10
- #define EZDP_DECODE_MPLS_RETVAL_LABEL1_TTL_IS_ZERO_SIZE 1
- #define EZDP_DECODE_MPLS_RETVAL_LABEL1_TTL_IS_ZERO_OFFSET 16
- #define EZDP_DECODE_MPLS_RETVAL_LABEL1_TTL_IS_ZERO_MASK (1 << EZDP_DECODE_MPLS_RETVAL_LABEL1_TTL_IS_ZERO_OFFSET)

- #define EZDP_DECODE_MPLS_RETVAL_LABEL2_TTL_IS_ZERO_SIZE 1
- #define EZDP_DECODE_MPLS_RETVAL_LABEL2_TTL_IS_ZERO_OFFSET 17
- #define EZDP_DECODE_MPLS_RETVAL_LABEL2_TTL_IS_ZERO_MASK (1 << EZDP_DECODE_MPLS_RETVAL_LABEL2_TTL_IS_ZERO_OFFSET)
- #define EZDP_DECODE_MPLS_RETVAL_LABEL3_TTL_IS_ZERO_SIZE 1
- #define EZDP_DECODE_MPLS_RETVAL_LABEL3_TTL_IS_ZERO_OFFSET 18
- #define EZDP_DECODE_MPLS_RETVAL_LABEL3_TTL_IS_ZERO_MASK (1 << EZDP_DECODE_MPLS_RETVAL_LABEL3_TTL_IS_ZERO_OFFSET)
- #define EZDP_DECODE_MPLS_RETVAL_LABEL4_TTL_IS_ZERO_SIZE 1
- #define EZDP_DECODE_MPLS_RETVAL_LABEL4_TTL_IS_ZERO_OFFSET 19
- #define EZDP_DECODE_MPLS_RETVAL_LABEL4_TTL_IS_ZERO_MASK (1 << EZDP_DECODE_MPLS_RETVAL_LABEL4_TTL_IS_ZERO_OFFSET)
- #define EZDP_DECODE_MPLS_RETVAL_RESERVED20_23_SIZE 4
- #define EZDP_DECODE_MPLS_RETVAL_RESERVED20_23_OFFSET 20
- #define EZDP_DECODE_MPLS_RETVAL_LABEL1_TTL_IS_ONE_SIZE 1
- #define EZDP_DECODE_MPLS_RETVAL_LABEL1_TTL_IS_ONE_OFFSET 24
- #define EZDP_DECODE_MPLS_RETVAL_LABEL1_TTL_IS_ONE_MASK (1 << EZDP_DECODE_MPLS_RETVAL_LABEL1_TTL_IS_ONE_OFFSET)
- #define EZDP_DECODE_MPLS_RETVAL_LABEL2_TTL_IS_ONE_SIZE 1
- #define EZDP_DECODE_MPLS_RETVAL_LABEL2_TTL_IS_ONE_OFFSET 25
- #define EZDP_DECODE_MPLS_RETVAL_LABEL2_TTL_IS_ONE_MASK (1 << EZDP_DECODE_MPLS_RETVAL_LABEL2_TTL_IS_ONE_OFFSET)
- #define EZDP_DECODE_MPLS_RETVAL_LABEL3_TTL_IS_ONE_SIZE 1
- #define EZDP_DECODE_MPLS_RETVAL_LABEL3_TTL_IS_ONE_OFFSET 26
- #define EZDP_DECODE_MPLS_RETVAL_LABEL3_TTL_IS_ONE_MASK (1 << EZDP_DECODE_MPLS_RETVAL_LABEL3_TTL_IS_ONE_OFFSET)
- #define EZDP_DECODE_MPLS_RETVAL_LABEL4_TTL_IS_ONE_SIZE 1
- #define EZDP_DECODE_MPLS_RETVAL_LABEL4_TTL_IS_ONE_OFFSET 27
- #define EZDP_DECODE_MPLS_RETVAL_LABEL4_TTL_IS_ONE_MASK (1 << EZDP_DECODE_MPLS_RETVAL_LABEL4_TTL_IS_ONE_OFFSET)
- #define EZDP_DECODE_MPLS_RETVAL_RESERVED28_31_SIZE 4
- #define EZDP_DECODE_MPLS_RETVAL_RESERVED28_31_OFFSET 28
- #define EZDP_DECODE_MPLS_RESULT_DECODE_ERROR_SIZE 1
- #define EZDP_DECODE_MPLS_RESULT_DECODE_ERROR_OFFSET 0
- #define EZDP_DECODE_MPLS_RESULT_DECODE_ERROR_MASK (1 << EZDP_DECODE_MPLS_RESULT_DECODE_ERROR_OFFSET)
- #define EZDP_DECODE_MPLS_RESULT_RESERVED1_7_SIZE 7
- #define EZDP_DECODE_MPLS_RESULT_RESERVED1_7_OFFSET 1
- #define EZDP_DECODE_MPLS_RESULT_LAST_ENTRY_IN_STACK_SIZE 2
- #define EZDP_DECODE_MPLS_RESULT_LAST_ENTRY_IN_STACK_OFFSET 8
- #define EZDP_DECODE_MPLS_RESULT_RESERVED10_15_SIZE 6
- #define EZDP_DECODE_MPLS_RESULT_RESERVED10_15_OFFSET 10
- #define EZDP_DECODE_MPLS_RESULT_LABEL1_TTL_IS_ZERO_SIZE 1
- #define EZDP_DECODE_MPLS_RESULT_LABEL1_TTL_IS_ZERO_OFFSET 16
- #define EZDP_DECODE_MPLS_RESULT_LABEL1_TTL_IS_ZERO_MASK (1 << EZDP_DECODE_MPLS_RESULT_LABEL1_TTL_IS_ZERO_OFFSET)
- #define EZDP_DECODE_MPLS_RESULT_LABEL2_TTL_IS_ZERO_SIZE 1
- #define EZDP_DECODE_MPLS_RESULT_LABEL2_TTL_IS_ZERO_OFFSET 17
- #define EZDP_DECODE_MPLS_RESULT_LABEL2_TTL_IS_ZERO_MASK (1 << EZDP_DECODE_MPLS_RESULT_LABEL2_TTL_IS_ZERO_OFFSET)
- #define EZDP_DECODE_MPLS_RESULT_LABEL3_TTL_IS_ZERO_SIZE 1
- #define EZDP_DECODE_MPLS_RESULT_LABEL3_TTL_IS_ZERO_OFFSET 18
- #define EZDP_DECODE_MPLS_RESULT_LABEL3_TTL_IS_ZERO_MASK (1 << EZDP_DECODE_MPLS_RESULT_LABEL3_TTL_IS_ZERO_OFFSET)
- #define EZDP_DECODE_MPLS_RESULT_LABEL4_TTL_IS_ZERO_SIZE 1
- #define EZDP_DECODE_MPLS_RESULT_LABEL4_TTL_IS_ZERO_OFFSET 19
- #define EZDP_DECODE_MPLS_RESULT_LABEL4_TTL_IS_ZERO_MASK (1 << EZDP_DECODE_MPLS_RESULT_LABEL4_TTL_IS_ZERO_OFFSET)

- #define EZDP_DECODE_MPLS_RESULT_RESERVED20_23_SIZE 4
- #define EZDP_DECODE_MPLS_RESULT_RESERVED20_23_OFFSET 20
- #define EZDP_DECODE_MPLS_RESULT_LABEL1_TTL_IS_ONE_SIZE 1
- #define EZDP_DECODE_MPLS_RESULT_LABEL1_TTL_IS_ONE_OFFSET 24
- #define EZDP_DECODE_MPLS_RESULT_LABEL1_TTL_IS_ONE_MASK (1 << EZDP_DECODE_MPLS_RESULT_LABEL1_TTL_IS_ONE_OFFSET)
- #define EZDP_DECODE_MPLS_RESULT_LABEL2_TTL_IS_ONE_SIZE 1
- #define EZDP_DECODE_MPLS_RESULT_LABEL2_TTL_IS_ONE_OFFSET 25
- #define EZDP_DECODE_MPLS_RESULT_LABEL2_TTL_IS_ONE_MASK (1 << EZDP_DECODE_MPLS_RESULT_LABEL2_TTL_IS_ONE_OFFSET)
- #define EZDP_DECODE_MPLS_RESULT_LABEL3_TTL_IS_ONE_SIZE 1
- #define EZDP_DECODE_MPLS_RESULT_LABEL3_TTL_IS_ONE_OFFSET 26
- #define EZDP_DECODE_MPLS_RESULT_LABEL3_TTL_IS_ONE_MASK (1 << EZDP_DECODE_MPLS_RESULT_LABEL3_TTL_IS_ONE_OFFSET)
- #define EZDP_DECODE_MPLS_RESULT_LABEL4_TTL_IS_ONE_SIZE 1
- #define EZDP_DECODE_MPLS_RESULT_LABEL4_TTL_IS_ONE_OFFSET 27
- #define EZDP_DECODE_MPLS_RESULT_LABEL4_TTL_IS_ONE_MASK (1 << EZDP_DECODE_MPLS_RESULT_LABEL4_TTL_IS_ONE_OFFSET)
- #define EZDP_DECODE_MPLS_RESULT_RESERVED28_31_SIZE 4
- #define EZDP_DECODE_MPLS_RESULT_RESERVED28_31_OFFSET 28
- #define EZDP_DECODE_MPLS_LABEL_RETVAL_END_OF_STACK_SIZE 1
- #define EZDP_DECODE_MPLS_LABEL_RETVAL_END_OF_STACK_OFFSET 0
- #define EZDP_DECODE_MPLS_LABEL_RETVAL_END_OF_STACK_MASK (1 << EZDP_DECODE_MPLS_LABEL_RETVAL_END_OF_STACK_OFFSET)
- #define EZDP_DECODE_MPLS_LABEL_RETVAL_RESERVED_LABEL_SIZE 1
- #define EZDP_DECODE_MPLS_LABEL_RETVAL_RESERVED_LABEL_OFFSET 1
- #define EZDP_DECODE_MPLS_LABEL_RETVAL_RESERVED_LABEL_MASK (1 << EZDP_DECODE_MPLS_LABEL_RETVAL_RESERVED_LABEL_OFFSET)
- #define EZDP_DECODE_MPLS_LABEL_RETVAL_TTL_IS_ZERO_SIZE 1
- #define EZDP_DECODE_MPLS_LABEL_RETVAL_TTL_IS_ZERO_OFFSET 2
- #define EZDP_DECODE_MPLS_LABEL_RETVAL_TTL_IS_ZERO_MASK (1 << EZDP_DECODE_MPLS_LABEL_RETVAL_TTL_IS_ZERO_OFFSET)
- #define EZDP_DECODE_MPLS_LABEL_RETVAL_TTL_IS_ONE_SIZE 1
- #define EZDP_DECODE_MPLS_LABEL_RETVAL_TTL_IS_ONE_OFFSET 3
- #define EZDP_DECODE_MPLS_LABEL_RETVAL_TTL_IS_ONE_MASK (1 << EZDP_DECODE_MPLS_LABEL_RETVAL_TTL_IS_ONE_OFFSET)
- #define EZDP_DECODE_MPLS_LABEL_RETVAL_USER_CONFIG0_SIZE 1
- #define EZDP_DECODE_MPLS_LABEL_RETVAL_USER_CONFIG0_OFFSET 4
- #define EZDP_DECODE_MPLS_LABEL_RETVAL_USER_CONFIG0_MASK (1 << EZDP_DECODE_MPLS_LABEL_RETVAL_USER_CONFIG0_OFFSET)
- #define EZDP_DECODE_MPLS_LABEL_RETVAL_USER_CONFIG1_SIZE 1
- #define EZDP_DECODE_MPLS_LABEL_RETVAL_USER_CONFIG1_OFFSET 5
- #define EZDP_DECODE_MPLS_LABEL_RETVAL_USER_CONFIG1_MASK (1 << EZDP_DECODE_MPLS_LABEL_RETVAL_USER_CONFIG1_OFFSET)
- #define EZDP_DECODE_MPLS_LABEL_RETVAL_USER_CONFIG2_SIZE 1
- #define EZDP_DECODE_MPLS_LABEL_RETVAL_USER_CONFIG2_OFFSET 6
- #define EZDP_DECODE_MPLS_LABEL_RETVAL_USER_CONFIG2_MASK (1 << EZDP_DECODE_MPLS_LABEL_RETVAL_USER_CONFIG2_OFFSET)
- #define EZDP_DECODE_MPLS_LABEL_RETVAL_USER_CONFIG3_SIZE 1
- #define EZDP_DECODE_MPLS_LABEL_RETVAL_USER_CONFIG3_OFFSET 7
- #define EZDP_DECODE_MPLS_LABEL_RETVAL_USER_CONFIG3_MASK (1 << EZDP_DECODE_MPLS_LABEL_RETVAL_USER_CONFIG3_OFFSET)
- #define EZDP_DECODE_MPLS_LABEL_RETVAL_EXCEPTION_BIT_SIZE 1
- #define EZDP_DECODE_MPLS_LABEL_RETVAL_EXCEPTION_BIT_OFFSET 8
- #define EZDP_DECODE_MPLS_LABEL_RETVAL_EXCEPTION_BIT_MASK (1 << EZDP_DECODE_MPLS_LABEL_RETVAL_EXCEPTION_BIT_OFFSET)
- #define EZDP_DECODE_MPLS_LABEL_RETVAL_STOP_BIT_SIZE 1
- #define EZDP_DECODE_MPLS_LABEL_RETVAL_STOP_BIT_OFFSET 9

- #define EZDP_DECODE_MPLS_LABEL_RETVAL_STOP_BIT_MASK (1 << EZDP_DECODE_MPLS_LABEL_RETVAL_STOP_BIT_OFFSET)
- #define EZDP_DECODE_MPLS_LABEL_RETVAL_RESERVED10_31_SIZE 22
- #define EZDP_DECODE_MPLS_LABEL_RETVAL_RESERVED10_31_OFFSET 10
- #define EZDP_DECODE_MPLS_LABEL_RESULT_END_OF_STACK_SIZE 1
- #define EZDP_DECODE_MPLS_LABEL_RESULT_END_OF_STACK_OFFSET 0
- #define EZDP_DECODE_MPLS_LABEL_RESULT_END_OF_STACK_MASK (1 << EZDP_DECODE_MPLS_LABEL_RESULT_END_OF_STACK_OFFSET)
- #define EZDP_DECODE_MPLS_LABEL_RESULT_RESERVED_LABEL_SIZE 1
- #define EZDP_DECODE_MPLS_LABEL_RESULT_RESERVED_LABEL_OFFSET 1
- #define EZDP_DECODE_MPLS_LABEL_RESULT_RESERVED_LABEL_MASK (1 << EZDP_DECODE_MPLS_LABEL_RESULT_RESERVED_LABEL_OFFSET)
- #define EZDP_DECODE_MPLS_LABEL_RESULT_TTL_IS_ZERO_SIZE 1
- #define EZDP_DECODE_MPLS_LABEL_RESULT_TTL_IS_ZERO_OFFSET 2
- #define EZDP_DECODE_MPLS_LABEL_RESULT_TTL_IS_ZERO_MASK (1 << EZDP_DECODE_MPLS_LABEL_RESULT_TTL_IS_ZERO_OFFSET)
- #define EZDP_DECODE_MPLS_LABEL_RESULT_TTL_IS_ONE_SIZE 1
- #define EZDP_DECODE_MPLS_LABEL_RESULT_TTL_IS_ONE_OFFSET 3
- #define EZDP_DECODE_MPLS_LABEL_RESULT_TTL_IS_ONE_MASK (1 << EZDP_DECODE_MPLS_LABEL_RESULT_TTL_IS_ONE_OFFSET)
- #define EZDP_DECODE_MPLS_LABEL_RESULT_USER_CONFIG0_SIZE 1
- #define EZDP_DECODE_MPLS_LABEL_RESULT_USER_CONFIG0_OFFSET 4
- #define EZDP_DECODE_MPLS_LABEL_RESULT_USER_CONFIG0_MASK (1 << EZDP_DECODE_MPLS_LABEL_RESULT_USER_CONFIG0_OFFSET)
- #define EZDP_DECODE_MPLS_LABEL_RESULT_USER_CONFIG1_SIZE 1
- #define EZDP_DECODE_MPLS_LABEL_RESULT_USER_CONFIG1_OFFSET 5
- #define EZDP_DECODE_MPLS_LABEL_RESULT_USER_CONFIG1_MASK (1 << EZDP_DECODE_MPLS_LABEL_RESULT_USER_CONFIG1_OFFSET)
- #define EZDP_DECODE_MPLS_LABEL_RESULT_USER_CONFIG2_SIZE 1
- #define EZDP_DECODE_MPLS_LABEL_RESULT_USER_CONFIG2_OFFSET 6
- #define EZDP_DECODE_MPLS_LABEL_RESULT_USER_CONFIG2_MASK (1 << EZDP_DECODE_MPLS_LABEL_RESULT_USER_CONFIG2_OFFSET)
- #define EZDP_DECODE_MPLS_LABEL_RESULT_USER_CONFIG3_SIZE 1
- #define EZDP_DECODE_MPLS_LABEL_RESULT_USER_CONFIG3_OFFSET 7
- #define EZDP_DECODE_MPLS_LABEL_RESULT_USER_CONFIG3_MASK (1 << EZDP_DECODE_MPLS_LABEL_RESULT_USER_CONFIG3_OFFSET)
- #define EZDP_DECODE_MPLS_LABEL_RESULT_EXCEPTION_BIT_SIZE 1
- #define EZDP_DECODE_MPLS_LABEL_RESULT_EXCEPTION_BIT_OFFSET 8
- #define EZDP_DECODE_MPLS_LABEL_RESULT_EXCEPTION_BIT_MASK (1 << EZDP_DECODE_MPLS_LABEL_RESULT_EXCEPTION_BIT_OFFSET)
- #define EZDP_DECODE_MPLS_LABEL_RESULT_STOP_BIT_SIZE 1
- #define EZDP_DECODE_MPLS_LABEL_RESULT_STOP_BIT_OFFSET 9
- #define EZDP_DECODE_MPLS_LABEL_RESULT_STOP_BIT_MASK (1 << EZDP_DECODE_MPLS_LABEL_RESULT_STOP_BIT_OFFSET)
- #define EZDP_DECODE_MPLS_LABEL_RESULT_RESERVED10_31_SIZE 22
- #define EZDP_DECODE_MPLS_LABEL_RESULT_RESERVED10_31_OFFSET 10

Typedefs

- typedef uint8_t ezdp_decode_ipv4_control_t
- typedef uint16_t ezdp_decode_ipv4_errors_t
- typedef uint8_t ezdp_decode_ip_next_protocol_t
- typedef uint8_t ezdp_decode_ipv6_control_t
- typedef uint16_t ezdp_decode_ipv6_errors_t
- typedef uint16_t ezdp_decode_mac_control_t
- typedef uint8_t ezdp_decode_mac_errors_t
- typedef uint16_t ezdp_decode_mac_protocol_type_t
- typedef uint8_t ezdp_decode_tcp_errors_t

- typedef uint32_t ezdp_decode_tcp_retval_t
- typedef uint32_t ezdp_decode_ip_protocol_retval_t
- typedef uint32_t ezdp_decode_eth_type_retval_t
- typedef uint32_t ezdp_decode_ipv4_retval_t
- typedef uint32_t ezdp_decode_ipv6_retval_t
- typedef uint32_t ezdp_decode_mac_retval_t
- typedef uint32_t ezdp_decode_mpls_retval_t
- typedef uint32_t ezdp_decode_mpls_result_t
- typedef uint32_t ezdp_decode_mpls_label_retval_t
- typedef uint32_t ezdp_decode_mpls_label_result_t

Define Documentation

#define EZDP_DECODE_VERSION_MAJOR 2

#define EZDP_DECODE_VERSION_MINOR 1

#define EZDP_DECODE_IPV4_CONTROL_LINK_LOCAL_MULTICAST_RANGE_SIZE 1

#define EZDP_DECODE_IPV4_CONTROL_LINK_LOCAL_MULTICAST_RANGE_OFFSET 0

#define EZDP_DECODE_IPV4_CONTROL_LINK_LOCAL_MULTICAST_RANGE_MASK (1 << EZDP_DECODE_IPV4_CONTROL_LINK_LOCAL_MULTICAST_RANGE_OFFSET)

#define EZDP_DECODE_IPV4_CONTROL_INTERNETWORK_MULTICAST_RANGE_SIZE 1

#define EZDP_DECODE_IPV4_CONTROL_INTERNETWORK_MULTICAST_RANGE_OFFSET 1

#define EZDP_DECODE_IPV4_CONTROL_INTERNETWORK_MULTICAST_RANGE_MASK (1 << EZDP_DECODE_IPV4_CONTROL_INTERNETWORK_MULTICAST_RANGE_OFFSET)

#define EZDP_DECODE_IPV4_CONTROL_RESERVED_2_SIZE 1

#define EZDP_DECODE_IPV4_CONTROL_RESERVED_2_OFFSET 2

#define EZDP_DECODE_IPV4_CONTROL_ICMP_SIZE 1

#define EZDP_DECODE_IPV4_CONTROL_ICMP_OFFSET 3

#define EZDP_DECODE_IPV4_CONTROL_ICMP_MASK (1 << EZDP_DECODE_IPV4_CONTROL_ICMP_OFFSET)

#define EZDP_DECODE_IPV4_CONTROL_IGMP_SIZE 1

#define EZDP_DECODE_IPV4_CONTROL_IGMP_OFFSET 4

#define EZDP_DECODE_IPV4_CONTROL_IGMP_MASK (1 << EZDP_DECODE_IPV4_CONTROL_IGMP_OFFSET)

#define EZDP_DECODE_IPV4_CONTROL_USER_CONFIG0_SIZE 1

#define EZDP_DECODE_IPV4_CONTROL_USER_CONFIG0_OFFSET 5

#define EZDP_DECODE_IPV4_CONTROL_USER_CONFIG0_MASK (1 << EZDP_DECODE_IPV4_CONTROL_USER_CONFIG0_OFFSET)

#define EZDP_DECODE_IPV4_CONTROL_USER_CONFIG1_SIZE 1

#define EZDP_DECODE_IPV4_CONTROL_USER_CONFIG1_OFFSET 6

#define EZDP_DECODE_IPV4_CONTROL_USER_CONFIG1_MASK (1 << EZDP_DECODE_IPV4_CONTROL_USER_CONFIG1_OFFSET)

#define EZDP_DECODE_IPV4_CONTROL_USER_CONFIG2_SIZE 1

#define EZDP_DECODE_IPV4_CONTROL_USER_CONFIG2_OFFSET 7

#define EZDP_DECODE_IPV4_CONTROL_USER_CONFIG2_MASK (1 << EZDP_DECODE_IPV4_CONTROL_USER_CONFIG2_OFFSET)

#define EZDP_DECODE_IPV4_ERRORS_SIP_IS_MULTICAST_SIZE 1

#define EZDP_DECODE_IPV4_ERRORS_SIP_IS_MULTICAST_OFFSET 0

#define EZDP_DECODE_IPV4_ERRORS_SIP_IS_MULTICAST_MASK (1 << EZDP_DECODE_IPV4_ERRORS_SIP_IS_MULTICAST_OFFSET)

#define EZDP_DECODE_IPV4_ERRORS_SIP_IS_ZERO_SIZE 1

#define EZDP_DECODE_IPV4_ERRORS_SIP_IS_ZERO_OFFSET 1

#define EZDP_DECODE_IPV4_ERRORS_SIP_IS_ZERO_MASK (1 << EZDP_DECODE_IPV4_ERRORS_SIP_IS_ZERO_OFFSET)

#define EZDP_DECODE_IPV4_ERRORS_HEADER_LENGTH_LT_5_SIZE 1

#define EZDP_DECODE_IPV4_ERRORS_HEADER_LENGTH_LT_5_OFFSET 2

#define EZDP_DECODE_IPV4_ERRORS_HEADER_LENGTH_LT_5_MASK (1 << EZDP_DECODE_IPV4_ERRORS_HEADER_LENGTH_LT_5_OFFSET)

#define EZDP_DECODE_IPV4_ERRORS_TOTAL_LENGTH_GT_FRAME_LENGTH_SIZE 1

#define EZDP_DECODE_IPV4_ERRORS_TOTAL_LENGTH_GT_FRAME_LENGTH_OFFSET 3

#define EZDP_DECODE_IPV4_ERRORS_TOTAL_LENGTH_GT_FRAME_LENGTH_MASK (1 << EZDP_DECODE_IPV4_ERRORS_TOTAL_LENGTH_GT_FRAME_LENGTH_OFFSET)

#define EZDP_DECODE_IPV4_ERRORS_HEADER_LENGTH_GT_FRAME_LENGTH_SIZE 1

#define EZDP_DECODE_IPV4_ERRORS_HEADER_LENGTH_GT_FRAME_LENGTH_OFFSET 4

#define EZDP_DECODE_IPV4_ERRORS_HEADER_LENGTH_GT_FRAME_LENGTH_MASK (1 << EZDP_DECODE_IPV4_ERRORS_HEADER_LENGTH_GT_FRAME_LENGTH_OFFSET)

#define EZDP_DECODE_IPV4_ERRORS_NOT_IPV4_VERSION_SIZE 1

#define EZDP_DECODE_IPV4_ERRORS_NOT_IPV4_VERSION_OFFSET 5

#define EZDP_DECODE_IPV4_ERRORS_NOT_IPV4_VERSION_MASK (1 << EZDP_DECODE_IPV4_ERRORS_NOT_IPV4_VERSION_OFFSET)

#define EZDP_DECODE_IPV4_ERRORS_CHECKSUM_ERROR_SIZE 1

#define EZDP_DECODE_IPV4_ERRORS_CHECKSUM_ERROR_OFFSET 6

#define EZDP_DECODE_IPV4_ERRORS_CHECKSUM_ERROR_MASK (1 << EZDP_DECODE_IPV4_ERRORS_CHECKSUM_ERROR_OFFSET)

```
#define EZDP_DECODE_IPV4_ERRORS_SIP_EQUAL_DIP_SIZE 1

#define EZDP_DECODE_IPV4_ERRORS_SIP_EQUAL_DIP_OFFSET 7

#define EZDP_DECODE_IPV4_ERRORS_SIP_EQUAL_DIP_MASK (1 <<
EZDP_DECODE_IPV4_ERRORS_SIP_EQUAL_DIP_OFFSET)

#define EZDP_DECODE_IPV4_ERRORS_DECODE_ERROR_SIZE 1

#define EZDP_DECODE_IPV4_ERRORS_DECODE_ERROR_OFFSET 8

#define EZDP_DECODE_IPV4_ERRORS_DECODE_ERROR_MASK (1 <<
EZDP_DECODE_IPV4_ERRORS_DECODE_ERROR_OFFSET)

#define EZDP_DECODE_IPV4_ERRORS_RESERVED9_15_SIZE 7

#define EZDP_DECODE_IPV4_ERRORS_RESERVED9_15_OFFSET 9

#define EZDP_DECODE_IP_NEXT_PROTOCOL_TCP_SIZE 1

#define EZDP_DECODE_IP_NEXT_PROTOCOL_TCP_OFFSET 0

#define EZDP_DECODE_IP_NEXT_PROTOCOL_TCP_MASK (1 <<
EZDP_DECODE_IP_NEXT_PROTOCOL_TCP_OFFSET)

#define EZDP_DECODE_IP_NEXT_PROTOCOL_UDP_SIZE 1

#define EZDP_DECODE_IP_NEXT_PROTOCOL_UDP_OFFSET 1

#define EZDP_DECODE_IP_NEXT_PROTOCOL_UDP_MASK (1 <<
EZDP_DECODE_IP_NEXT_PROTOCOL_UDP_OFFSET)

#define EZDP_DECODE_IP_NEXT_PROTOCOL_MPLS_SIZE 1

#define EZDP_DECODE_IP_NEXT_PROTOCOL_MPLS_OFFSET 2

#define EZDP_DECODE_IP_NEXT_PROTOCOL_MPLS_MASK (1 <<
EZDP_DECODE_IP_NEXT_PROTOCOL_MPLS_OFFSET)

#define EZDP_DECODE_IP_NEXT_PROTOCOL_GRE_SIZE 1

#define EZDP_DECODE_IP_NEXT_PROTOCOL_GRE_OFFSET 3

#define EZDP_DECODE_IP_NEXT_PROTOCOL_GRE_MASK (1 <<
EZDP_DECODE_IP_NEXT_PROTOCOL_GRE_OFFSET)

#define EZDP_DECODE_IP_NEXT_PROTOCOL_IPV4_SIZE 1

#define EZDP_DECODE_IP_NEXT_PROTOCOL_IPV4_OFFSET 4

#define EZDP_DECODE_IP_NEXT_PROTOCOL_IPV4_MASK (1 <<
EZDP_DECODE_IP_NEXT_PROTOCOL_IPV4_OFFSET)
```

```
#define EZDP_DECODE_IP_NEXT_PROTOCOL_IPV6_SIZE 1

#define EZDP_DECODE_IP_NEXT_PROTOCOL_IPV6_OFFSET 5

#define EZDP_DECODE_IP_NEXT_PROTOCOL_IPV6_MASK (1 <<
EZDP_DECODE_IP_NEXT_PROTOCOL_IPV6_OFFSET)

#define EZDP_DECODE_IP_NEXT_PROTOCOL_ICMP_IGMP_SIZE 1

#define EZDP_DECODE_IP_NEXT_PROTOCOL_ICMP_IGMP_OFFSET 6

#define EZDP_DECODE_IP_NEXT_PROTOCOL_ICMP_IGMP_MASK (1 <<
EZDP_DECODE_IP_NEXT_PROTOCOL_ICMP_IGMP_OFFSET)

#define EZDP_DECODE_IP_NEXT_PROTOCOL_OTHER_SIZE 1

#define EZDP_DECODE_IP_NEXT_PROTOCOL_OTHER_OFFSET 7

#define EZDP_DECODE_IP_NEXT_PROTOCOL_OTHER_MASK (1 <<
EZDP_DECODE_IP_NEXT_PROTOCOL_OTHER_OFFSET)

#define EZDP_DECODE_IPV6_CONTROL_LINK_LOCAL_MULTICAST_RANGE_SIZE 1

#define EZDP_DECODE_IPV6_CONTROL_LINK_LOCAL_MULTICAST_RANGE_OFFSET 0

#define EZDP_DECODE_IPV6_CONTROL_LINK_LOCAL_MULTICAST_RANGE_MASK (1 <<
EZDP_DECODE_IPV6_CONTROL_LINK_LOCAL_MULTICAST_RANGE_OFFSET)

#define EZDP_DECODE_IPV6_CONTROL_INTERNETWORK_MULTICAST_RANGE_SIZE 1

#define EZDP_DECODE_IPV6_CONTROL_INTERNETWORK_MULTICAST_RANGE_OFFSET 1

#define EZDP_DECODE_IPV6_CONTROL_INTERNETWORK_MULTICAST_RANGE_MASK (1 <<
EZDP_DECODE_IPV6_CONTROL_INTERNETWORK_MULTICAST_RANGE_OFFSET)

#define EZDP_DECODE_IPV6_CONTROL_SOLICITED_NODE_MULTICAST_RANGE_SIZE 1

#define EZDP_DECODE_IPV6_CONTROL_SOLICITED_NODE_MULTICAST_RANGE_OFFSET 2

#define EZDP_DECODE_IPV6_CONTROL_SOLICITED_NODE_MULTICAST_RANGE_MASK (1 <<
EZDP_DECODE_IPV6_CONTROL_SOLICITED_NODE_MULTICAST_RANGE_OFFSET)

#define EZDP_DECODE_IPV6_CONTROL_RESERVED_3_SIZE 1

#define EZDP_DECODE_IPV6_CONTROL_RESERVED_3_OFFSET 3

#define EZDP_DECODE_IPV6_CONTROL_DIP_IS_MULTICAST_SIZE 1

#define EZDP_DECODE_IPV6_CONTROL_DIP_IS_MULTICAST_OFFSET 4

#define EZDP_DECODE_IPV6_CONTROL_DIP_IS_MULTICAST_MASK (1 <<
EZDP_DECODE_IPV6_CONTROL_DIP_IS_MULTICAST_OFFSET)
```

```
#define EZDP_DECODE_IPV6_CONTROL_DIP_IS_WELLKNOWN_MULTICAST_SIZE 1

#define EZDP_DECODE_IPV6_CONTROL_DIP_IS_WELLKNOWN_MULTICAST_OFFSET 5

#define EZDP_DECODE_IPV6_CONTROL_DIP_IS_WELLKNOWN_MULTICAST_MASK (1 <<
EZDP_DECODE_IPV6_CONTROL_DIP_IS_WELLKNOWN_MULTICAST_OFFSET)

#define EZDP_DECODE_IPV6_CONTROL_RESERVED7_8_SIZE 2

#define EZDP_DECODE_IPV6_CONTROL_RESERVED7_8_OFFSET 6

#define EZDP_DECODE_IPV6_ERRORS_PAYLOAD_GT_FRAME_LENGTH_SIZE 1

#define EZDP_DECODE_IPV6_ERRORS_PAYLOAD_GT_FRAME_LENGTH_OFFSET 0

#define EZDP_DECODE_IPV6_ERRORS_PAYLOAD_GT_FRAME_LENGTH_MASK (1 <<
EZDP_DECODE_IPV6_ERRORS_PAYLOAD_GT_FRAME_LENGTH_OFFSET)

#define EZDP_DECODE_IPV6_ERRORS_NOT_IPV6_VERSION_SIZE 1

#define EZDP_DECODE_IPV6_ERRORS_NOT_IPV6_VERSION_OFFSET 1

#define EZDP_DECODE_IPV6_ERRORS_NOT_IPV6_VERSION_MASK (1 <<
EZDP_DECODE_IPV6_ERRORS_NOT_IPV6_VERSION_OFFSET)

#define EZDP_DECODE_IPV6_ERRORS_SIP_IS_ZERO_SIZE 1

#define EZDP_DECODE_IPV6_ERRORS_SIP_IS_ZERO_OFFSET 2

#define EZDP_DECODE_IPV6_ERRORS_SIP_IS_ZERO_MASK (1 <<
EZDP_DECODE_IPV6_ERRORS_SIP_IS_ZERO_OFFSET)

#define EZDP_DECODE_IPV6_ERRORS_SIP_IS_ONE_SIZE 1

#define EZDP_DECODE_IPV6_ERRORS_SIP_IS_ONE_OFFSET 3

#define EZDP_DECODE_IPV6_ERRORS_SIP_IS_ONE_MASK (1 <<
EZDP_DECODE_IPV6_ERRORS_SIP_IS_ONE_OFFSET)

#define EZDP_DECODE_IPV6_ERRORS_DIP_IS_ZERO_SIZE 1

#define EZDP_DECODE_IPV6_ERRORS_DIP_IS_ZERO_OFFSET 4

#define EZDP_DECODE_IPV6_ERRORS_DIP_IS_ZERO_MASK (1 <<
EZDP_DECODE_IPV6_ERRORS_DIP_IS_ZERO_OFFSET)

#define EZDP_DECODE_IPV6_ERRORS_DIP_IS_ONE_SIZE 1

#define EZDP_DECODE_IPV6_ERRORS_DIP_IS_ONE_OFFSET 5

#define EZDP_DECODE_IPV6_ERRORS_DIP_IS_ONE_MASK (1 <<
EZDP_DECODE_IPV6_ERRORS_DIP_IS_ONE_OFFSET)
```

```
#define EZDP_DECODE_IPV6_ERRORS_SIP_EQUAL_DIP_SIZE 1

#define EZDP_DECODE_IPV6_ERRORS_SIP_EQUAL_DIP_OFFSET 6

#define EZDP_DECODE_IPV6_ERRORS_SIP_EQUAL_DIP_MASK (1 <<
EZDP_DECODE_IPV6_ERRORS_SIP_EQUAL_DIP_OFFSET)

#define EZDP_DECODE_IPV6_ERRORS_DECODE_ERROR_SIZE 1

#define EZDP_DECODE_IPV6_ERRORS_DECODE_ERROR_OFFSET 7

#define EZDP_DECODE_IPV6_ERRORS_DECODE_ERROR_MASK (1 <<
EZDP_DECODE_IPV6_ERRORS_DECODE_ERROR_OFFSET)

#define EZDP_DECODE_IPV6_ERRORS_PAYLOAD_MISSING_SIZE 1

#define EZDP_DECODE_IPV6_ERRORS_PAYLOAD_MISSING_OFFSET 8

#define EZDP_DECODE_IPV6_ERRORS_PAYLOAD_MISSING_MASK (1 <<
EZDP_DECODE_IPV6_ERRORS_PAYLOAD_MISSING_OFFSET)

#define EZDP_DECODE_IPV6_ERRORS_SIP_IS_MULTICAST_SIZE 1

#define EZDP_DECODE_IPV6_ERRORS_SIP_IS_MULTICAST_OFFSET 9

#define EZDP_DECODE_IPV6_ERRORS_SIP_IS_MULTICAST_MASK (1 <<
EZDP_DECODE_IPV6_ERRORS_SIP_IS_MULTICAST_OFFSET)

#define EZDP_DECODE_IPV6_ERRORS_RESERVED10_15_SIZE 6

#define EZDP_DECODE_IPV6_ERRORS_RESERVED10_15_OFFSET 10

#define EZDP_DECODE_MAC_CONTROL_MY_MAC_SIZE 1

#define EZDP_DECODE_MAC_CONTROL_MY_MAC_OFFSET 0

#define EZDP_DECODE_MAC_CONTROL_MY_MAC_MASK (1 <<
EZDP_DECODE_MAC_CONTROL_MY_MAC_OFFSET)

#define EZDP_DECODE_MAC_CONTROL_MAC_CONTROL_LSB_0X_SIZE 1

#define EZDP_DECODE_MAC_CONTROL_MAC_CONTROL_LSB_0X_OFFSET 1

#define EZDP_DECODE_MAC_CONTROL_MAC_CONTROL_LSB_0X_MASK (1 <<
EZDP_DECODE_MAC_CONTROL_MAC_CONTROL_LSB_0X_OFFSET)

#define EZDP_DECODE_MAC_CONTROL_MAC_CONTROL_LSB_1X_SIZE 1

#define EZDP_DECODE_MAC_CONTROL_MAC_CONTROL_LSB_1X_OFFSET 2

#define EZDP_DECODE_MAC_CONTROL_MAC_CONTROL_LSB_1X_MASK (1 <<
EZDP_DECODE_MAC_CONTROL_MAC_CONTROL_LSB_1X_OFFSET)
```

```
#define EZDP_DECODE_MAC_CONTROL_MAC_CONTROL_LSB_2X_SIZE 1

#define EZDP_DECODE_MAC_CONTROL_MAC_CONTROL_LSB_2X_OFFSET 3

#define EZDP_DECODE_MAC_CONTROL_MAC_CONTROL_LSB_2X_MASK (1 <<
EZDP_DECODE_MAC_CONTROL_MAC_CONTROL_LSB_2X_OFFSET)

#define EZDP_DECODE_MAC_CONTROL_MAC_CONTROL_OTHER_SIZE 1

#define EZDP_DECODE_MAC_CONTROL_MAC_CONTROL_OTHER_OFFSET 4

#define EZDP_DECODE_MAC_CONTROL_MAC_CONTROL_OTHER_MASK (1 <<
EZDP_DECODE_MAC_CONTROL_MAC_CONTROL_OTHER_OFFSET)

#define EZDP_DECODE_MAC_CONTROL_VRRP_MAC_SIZE 1

#define EZDP_DECODE_MAC_CONTROL_VRRP_MAC_OFFSET 5

#define EZDP_DECODE_MAC_CONTROL_VRRP_MAC_MASK (1 <<
EZDP_DECODE_MAC_CONTROL_VRRP_MAC_OFFSET)

#define EZDP_DECODE_MAC_CONTROL_IPV4_MULTICAST_SIZE 1

#define EZDP_DECODE_MAC_CONTROL_IPV4_MULTICAST_OFFSET 6

#define EZDP_DECODE_MAC_CONTROL_IPV4_MULTICAST_MASK (1 <<
EZDP_DECODE_MAC_CONTROL_IPV4_MULTICAST_OFFSET)

#define EZDP_DECODE_MAC_CONTROL_IPV6_MULTICAST_SIZE 1

#define EZDP_DECODE_MAC_CONTROL_IPV6_MULTICAST_OFFSET 7

#define EZDP_DECODE_MAC_CONTROL_IPV6_MULTICAST_MASK (1 <<
EZDP_DECODE_MAC_CONTROL_IPV6_MULTICAST_OFFSET)

#define EZDP_DECODE_MAC_CONTROL_USER_CONFIG0_SIZE 1

#define EZDP_DECODE_MAC_CONTROL_USER_CONFIG0_OFFSET 8

#define EZDP_DECODE_MAC_CONTROL_USER_CONFIG0_MASK (1 <<
EZDP_DECODE_MAC_CONTROL_USER_CONFIG0_OFFSET)

#define EZDP_DECODE_MAC_CONTROL_USER_CONFIG1_SIZE 1

#define EZDP_DECODE_MAC_CONTROL_USER_CONFIG1_OFFSET 9

#define EZDP_DECODE_MAC_CONTROL_USER_CONFIG1_MASK (1 <<
EZDP_DECODE_MAC_CONTROL_USER_CONFIG1_OFFSET)

#define EZDP_DECODE_MAC_CONTROL_USER_CONFIG2_SIZE 1

#define EZDP_DECODE_MAC_CONTROL_USER_CONFIG2_OFFSET 10
```

#define EZDP_DECODE_MAC_CONTROL_USER_CONFIG2_MASK (1 << EZDP_DECODE_MAC_CONTROL_USER_CONFIG2_OFFSET)

#define EZDP_DECODE_MAC_CONTROL_USER_CONFIG3_SIZE 1

#define EZDP_DECODE_MAC_CONTROL_USER_CONFIG3_OFFSET 11

#define EZDP_DECODE_MAC_CONTROL_USER_CONFIG3_MASK (1 << EZDP_DECODE_MAC_CONTROL_USER_CONFIG3_OFFSET)

#define EZDP_DECODE_MAC_CONTROL_SMAC_EQUALS_DMAC_SIZE 1

#define EZDP_DECODE_MAC_CONTROL_SMAC_EQUALS_DMAC_OFFSET 12

#define EZDP_DECODE_MAC_CONTROL_SMAC_EQUALS_DMAC_MASK (1 << EZDP_DECODE_MAC_CONTROL_SMAC_EQUALS_DMAC_OFFSET)

#define EZDP_DECODE_MAC_CONTROL_RESERVED13_15_SIZE 3

#define EZDP_DECODE_MAC_CONTROL_RESERVED13_15_OFFSET 13

#define EZDP_DECODE_MAC_ERRORS_SMAC_IS_NOT_UNICAST_SIZE 1

#define EZDP_DECODE_MAC_ERRORS_SMAC_IS_NOT_UNICAST_OFFSET 0

#define EZDP_DECODE_MAC_ERRORS_SMAC_IS_NOT_UNICAST_MASK (1 << EZDP_DECODE_MAC_ERRORS_SMAC_IS_NOT_UNICAST_OFFSET)

#define EZDP_DECODE_MAC_ERRORS_SMAC_IS_ZERO_SIZE 1

#define EZDP_DECODE_MAC_ERRORS_SMAC_IS_ZERO_OFFSET 1

#define EZDP_DECODE_MAC_ERRORS_SMAC_IS_ZERO_MASK (1 << EZDP_DECODE_MAC_ERRORS_SMAC_IS_ZERO_OFFSET)

#define EZDP_DECODE_MAC_ERRORS_DMAC_IS_ZERO_SIZE 1

#define EZDP_DECODE_MAC_ERRORS_DMAC_IS_ZERO_OFFSET 2

#define EZDP_DECODE_MAC_ERRORS_DMAC_IS_ZERO_MASK (1 << EZDP_DECODE_MAC_ERRORS_DMAC_IS_ZERO_OFFSET)

#define EZDP_DECODE_MAC_ERRORS_IP_VERSION_MISMATCH_IN_PPPOE_SIZE 1

#define EZDP_DECODE_MAC_ERRORS_IP_VERSION_MISMATCH_IN_PPPOE_OFFSET 3

#define EZDP_DECODE_MAC_ERRORS_IP_VERSION_MISMATCH_IN_PPPOE_MASK (1 << EZDP_DECODE_MAC_ERRORS_IP_VERSION_MISMATCH_IN_PPPOE_OFFSET)

#define EZDP_DECODE_MAC_ERRORS_DECODE_ERROR_SIZE 1

#define EZDP_DECODE_MAC_ERRORS_DECODE_ERROR_OFFSET 4

#define EZDP_DECODE_MAC_ERRORS_DECODE_ERROR_MASK (1 << EZDP_DECODE_MAC_ERRORS_DECODE_ERROR_OFFSET)

#define EZDP_DECODE_MAC_ERRORS_RESERVED5_7_SIZE 3

#define EZDP_DECODE_MAC_ERRORS_RESERVED5_7_OFFSET 5

#define EZDP_DECODE_MAC_PROTOCOL_TYPE_IPV4_SIZE 1

#define EZDP_DECODE_MAC_PROTOCOL_TYPE_IPV4_OFFSET 0

#define EZDP_DECODE_MAC_PROTOCOL_TYPE_IPV4_MASK (1 << EZDP_DECODE_MAC_PROTOCOL_TYPE_IPV4_OFFSET)

#define EZDP_DECODE_MAC_PROTOCOL_TYPE_USER_CONFIG_VLAN0_SIZE 1

#define EZDP_DECODE_MAC_PROTOCOL_TYPE_USER_CONFIG_VLAN0_OFFSET 1

#define EZDP_DECODE_MAC_PROTOCOL_TYPE_USER_CONFIG_VLAN0_MASK (1 << EZDP_DECODE_MAC_PROTOCOL_TYPE_USER_CONFIG_VLAN0_OFFSET)

#define EZDP_DECODE_MAC_PROTOCOL_TYPE_USER_CONFIG_VLAN1_SIZE 1

#define EZDP_DECODE_MAC_PROTOCOL_TYPE_USER_CONFIG_VLAN1_OFFSET 2

#define EZDP_DECODE_MAC_PROTOCOL_TYPE_USER_CONFIG_VLAN1_MASK (1 << EZDP_DECODE_MAC_PROTOCOL_TYPE_USER_CONFIG_VLAN1_OFFSET)

#define EZDP_DECODE_MAC_PROTOCOL_TYPE_ARP_SIZE 1

#define EZDP_DECODE_MAC_PROTOCOL_TYPE_ARP_OFFSET 3

#define EZDP_DECODE_MAC_PROTOCOL_TYPE_ARP_MASK (1 << EZDP_DECODE_MAC_PROTOCOL_TYPE_ARP_OFFSET)

#define EZDP_DECODE_MAC_PROTOCOL_TYPE_MPLS_UNICAST_SIZE 1

#define EZDP_DECODE_MAC_PROTOCOL_TYPE_MPLS_UNICAST_OFFSET 4

#define EZDP_DECODE_MAC_PROTOCOL_TYPE_MPLS_UNICAST_MASK (1 << EZDP_DECODE_MAC_PROTOCOL_TYPE_MPLS_UNICAST_OFFSET)

#define EZDP_DECODE_MAC_PROTOCOL_TYPE_MPLS_MULTICAST_SIZE 1

#define EZDP_DECODE_MAC_PROTOCOL_TYPE_MPLS_MULTICAST_OFFSET 5

#define EZDP_DECODE_MAC_PROTOCOL_TYPE_MPLS_MULTICAST_MASK (1 << EZDP_DECODE_MAC_PROTOCOL_TYPE_MPLS_MULTICAST_OFFSET)

#define EZDP_DECODE_MAC_PROTOCOL_TYPE_IPV6_SIZE 1

#define EZDP_DECODE_MAC_PROTOCOL_TYPE_IPV6_OFFSET 6

#define EZDP_DECODE_MAC_PROTOCOL_TYPE_IPV6_MASK (1 << EZDP_DECODE_MAC_PROTOCOL_TYPE_IPV6_OFFSET)

#define EZDP_DECODE_MAC_PROTOCOL_TYPE_LENGTH_SIZE 1

#define EZDP_DECODE_MAC_PROTOCOL_TYPE_LENGTH_OFFSET 7

#define EZDP_DECODE_MAC_PROTOCOL_TYPE_LENGTH_MASK (1 << EZDP_DECODE_MAC_PROTOCOL_TYPE_LENGTH_OFFSET)

#define EZDP_DECODE_MAC_PROTOCOL_TYPE_USER_CONFIG0_SIZE 1

#define EZDP_DECODE_MAC_PROTOCOL_TYPE_USER_CONFIG0_OFFSET 8

#define EZDP_DECODE_MAC_PROTOCOL_TYPE_USER_CONFIG0_MASK (1 << EZDP_DECODE_MAC_PROTOCOL_TYPE_USER_CONFIG0_OFFSET)

#define EZDP_DECODE_MAC_PROTOCOL_TYPE_USER_CONFIG1_SIZE 1

#define EZDP_DECODE_MAC_PROTOCOL_TYPE_USER_CONFIG1_OFFSET 9

#define EZDP_DECODE_MAC_PROTOCOL_TYPE_USER_CONFIG1_MASK (1 << EZDP_DECODE_MAC_PROTOCOL_TYPE_USER_CONFIG1_OFFSET)

#define EZDP_DECODE_MAC_PROTOCOL_TYPE_USER_CONFIG2_SIZE 1

#define EZDP_DECODE_MAC_PROTOCOL_TYPE_USER_CONFIG2_OFFSET 10

#define EZDP_DECODE_MAC_PROTOCOL_TYPE_USER_CONFIG2_MASK (1 << EZDP_DECODE_MAC_PROTOCOL_TYPE_USER_CONFIG2_OFFSET)

#define EZDP_DECODE_MAC_PROTOCOL_TYPE_USER_CONFIG3_SIZE 1

#define EZDP_DECODE_MAC_PROTOCOL_TYPE_USER_CONFIG3_OFFSET 11

#define EZDP_DECODE_MAC_PROTOCOL_TYPE_USER_CONFIG3_MASK (1 << EZDP_DECODE_MAC_PROTOCOL_TYPE_USER_CONFIG3_OFFSET)

#define EZDP_DECODE_MAC_PROTOCOL_TYPE_PPPOE_SESSION_SIZE 1

#define EZDP_DECODE_MAC_PROTOCOL_TYPE_PPPOE_SESSION_OFFSET 12

#define EZDP_DECODE_MAC_PROTOCOL_TYPE_PPPOE_SESSION_MASK (1 << EZDP_DECODE_MAC_PROTOCOL_TYPE_PPPOE_SESSION_OFFSET)

#define EZDP_DECODE_MAC_PROTOCOL_TYPE_PPPOE_DISCOVERY_SIZE 1

#define EZDP_DECODE_MAC_PROTOCOL_TYPE_PPPOE_DISCOVERY_OFFSET 13

#define EZDP_DECODE_MAC_PROTOCOL_TYPE_PPPOE_DISCOVERY_MASK (1 << EZDP_DECODE_MAC_PROTOCOL_TYPE_PPPOE_DISCOVERY_OFFSET)

#define EZDP_DECODE_MAC_PROTOCOL_TYPE_USER_CONFIG_VLAN2_SIZE 1

#define EZDP_DECODE_MAC_PROTOCOL_TYPE_USER_CONFIG_VLAN2_OFFSET 14

#define EZDP_DECODE_MAC_PROTOCOL_TYPE_USER_CONFIG_VLAN2_MASK (1 << EZDP_DECODE_MAC_PROTOCOL_TYPE_USER_CONFIG_VLAN2_OFFSET)

#define EZDP_DECODE_MAC_PROTOCOL_TYPE_RESERVED_15_SIZE 1

#define EZDP_DECODE_MAC_PROTOCOL_TYPE_RESERVED_15_OFFSET 15

#define EZDP_DECODE_TCP_ERRORS_DATA_OFFSET_LT_5_SIZE 1

#define EZDP_DECODE_TCP_ERRORS_DATA_OFFSET_LT_5_OFFSET 0

#define EZDP_DECODE_TCP_ERRORS_DATA_OFFSET_LT_5_MASK (1 << EZDP_DECODE_TCP_ERRORS_DATA_OFFSET_LT_5_OFFSET)

#define EZDP_DECODE_TCP_ERRORS_SYN_AND_FIN_EQ_1_SIZE 1

#define EZDP_DECODE_TCP_ERRORS_SYN_AND_FIN_EQ_1_OFFSET 1

#define EZDP_DECODE_TCP_ERRORS_SYN_AND_FIN_EQ_1_MASK (1 << EZDP_DECODE_TCP_ERRORS_SYN_AND_FIN_EQ_1_OFFSET)

#define EZDP_DECODE_TCP_ERRORS_DECODE_ERROR_SIZE 1

#define EZDP_DECODE_TCP_ERRORS_DECODE_ERROR_OFFSET 2

#define EZDP_DECODE_TCP_ERRORS_DECODE_ERROR_MASK (1 << EZDP_DECODE_TCP_ERRORS_DECODE_ERROR_OFFSET)

#define EZDP_DECODE_TCP_ERRORS_RESERVED3_7_SIZE 5

#define EZDP_DECODE_TCP_ERRORS_RESERVED3_7_OFFSET 3

#define EZDP_DECODE_TCP_RETVAL_ERROR_CODES_SIZE 8

#define EZDP_DECODE_TCP_RETVAL_ERROR_CODES_OFFSET 0

#define EZDP_DECODE_TCP_RETVAL_OPTIONS_EXIST_SIZE 1

#define EZDP_DECODE_TCP_RETVAL_OPTIONS_EXIST_OFFSET 8

#define EZDP_DECODE_TCP_RETVAL_OPTIONS_EXIST_MASK (1 << EZDP_DECODE_TCP_RETVAL_OPTIONS_EXIST_OFFSET)

#define EZDP_DECODE_TCP_RETVAL_RESERVED9_15_SIZE 7

#define EZDP_DECODE_TCP_RETVAL_RESERVED9_15_OFFSET 9

#define EZDP_DECODE_TCP_RETVAL_DATA_OFFSET_SIZE 6

#define EZDP_DECODE_TCP_RETVAL_DATA_OFFSET_OFFSET 16

```
#define EZDP_DECODE_TCP_RETVAL_RESERVED22_23_SIZE 2

#define EZDP_DECODE_TCP_RETVAL_RESERVED22_23_OFFSET 22

#define EZDP_DECODE_TCP_RETVAL_RESERVED24_31_SIZE 8

#define EZDP_DECODE_TCP_RETVAL_RESERVED24_31_OFFSET 24

#define EZDP_DECODE_IP_PROTOCOL_RETVAL_TCP_SIZE 1

#define EZDP_DECODE_IP_PROTOCOL_RETVAL_TCP_OFFSET 0

#define EZDP_DECODE_IP_PROTOCOL_RETVAL_TCP_MASK (1 <<
EZDP_DECODE_IP_PROTOCOL_RETVAL_TCP_OFFSET)

#define EZDP_DECODE_IP_PROTOCOL_RETVAL_UDP_SIZE 1

#define EZDP_DECODE_IP_PROTOCOL_RETVAL_UDP_OFFSET 1

#define EZDP_DECODE_IP_PROTOCOL_RETVAL_UDP_MASK (1 <<
EZDP_DECODE_IP_PROTOCOL_RETVAL_UDP_OFFSET)

#define EZDP_DECODE_IP_PROTOCOL_RETVAL_MPLS_SIZE 1

#define EZDP_DECODE_IP_PROTOCOL_RETVAL_MPLS_OFFSET 2

#define EZDP_DECODE_IP_PROTOCOL_RETVAL_MPLS_MASK (1 <<
EZDP_DECODE_IP_PROTOCOL_RETVAL_MPLS_OFFSET)

#define EZDP_DECODE_IP_PROTOCOL_RETVAL_GRE_SIZE 1

#define EZDP_DECODE_IP_PROTOCOL_RETVAL_GRE_OFFSET 3

#define EZDP_DECODE_IP_PROTOCOL_RETVAL_GRE_MASK (1 <<
EZDP_DECODE_IP_PROTOCOL_RETVAL_GRE_OFFSET)

#define EZDP_DECODE_IP_PROTOCOL_RETVAL_IPV4_SIZE 1

#define EZDP_DECODE_IP_PROTOCOL_RETVAL_IPV4_OFFSET 4

#define EZDP_DECODE_IP_PROTOCOL_RETVAL_IPV4_MASK (1 <<
EZDP_DECODE_IP_PROTOCOL_RETVAL_IPV4_OFFSET)

#define EZDP_DECODE_IP_PROTOCOL_RETVAL_IPV6_SIZE 1

#define EZDP_DECODE_IP_PROTOCOL_RETVAL_IPV6_OFFSET 5

#define EZDP_DECODE_IP_PROTOCOL_RETVAL_IPV6_MASK (1 <<
EZDP_DECODE_IP_PROTOCOL_RETVAL_IPV6_OFFSET)

#define EZDP_DECODE_IP_PROTOCOL_RETVAL_ICMP_IGMP_SIZE 1

#define EZDP_DECODE_IP_PROTOCOL_RETVAL_ICMP_IGMP_OFFSET 6
```

#define EZDP_DECODE_IP_PROTOCOL_RETVAL_ICMP_IGMP_MASK (1 << EZDP_DECODE_IP_PROTOCOL_RETVAL_ICMP_IGMP_OFFSET)

#define EZDP_DECODE_IP_PROTOCOL_RETVAL_DEF_IP_PROT_0_SIZE 1

#define EZDP_DECODE_IP_PROTOCOL_RETVAL_DEF_IP_PROT_0_OFFSET 7

#define EZDP_DECODE_IP_PROTOCOL_RETVAL_DEF_IP_PROT_0_MASK (1 << EZDP_DECODE_IP_PROTOCOL_RETVAL_DEF_IP_PROT_0_OFFSET)

#define EZDP_DECODE_IP_PROTOCOL_RETVAL_DEF_IP_PROT_1_SIZE 1

#define EZDP_DECODE_IP_PROTOCOL_RETVAL_DEF_IP_PROT_1_OFFSET 8

#define EZDP_DECODE_IP_PROTOCOL_RETVAL_DEF_IP_PROT_1_MASK (1 << EZDP_DECODE_IP_PROTOCOL_RETVAL_DEF_IP_PROT_1_OFFSET)

#define EZDP_DECODE_IP_PROTOCOL_RETVAL_DEF_IP_PROT_2_SIZE 1

#define EZDP_DECODE_IP_PROTOCOL_RETVAL_DEF_IP_PROT_2_OFFSET 9

#define EZDP_DECODE_IP_PROTOCOL_RETVAL_DEF_IP_PROT_2_MASK (1 << EZDP_DECODE_IP_PROTOCOL_RETVAL_DEF_IP_PROT_2_OFFSET)

#define EZDP_DECODE_IP_PROTOCOL_RETVAL_DEF_IP_PROT_3_SIZE 1

#define EZDP_DECODE_IP_PROTOCOL_RETVAL_DEF_IP_PROT_3_OFFSET 10

#define EZDP_DECODE_IP_PROTOCOL_RETVAL_DEF_IP_PROT_3_MASK (1 << EZDP_DECODE_IP_PROTOCOL_RETVAL_DEF_IP_PROT_3_OFFSET)

#define EZDP_DECODE_IP_PROTOCOL_RETVAL_ESP_PROT_SIZE 1

#define EZDP_DECODE_IP_PROTOCOL_RETVAL_ESP_PROT_OFFSET 11

#define EZDP_DECODE_IP_PROTOCOL_RETVAL_ESP_PROT_MASK (1 << EZDP_DECODE_IP_PROTOCOL_RETVAL_ESP_PROT_OFFSET)

#define EZDP_DECODE_IP_PROTOCOL_RETVAL_AH_PROT_SIZE 1

#define EZDP_DECODE_IP_PROTOCOL_RETVAL_AH_PROT_OFFSET 12

#define EZDP_DECODE_IP_PROTOCOL_RETVAL_AH_PROT_MASK (1 << EZDP_DECODE_IP_PROTOCOL_RETVAL_AH_PROT_OFFSET)

#define EZDP_DECODE_IP_PROTOCOL_RETVAL_OTHER_SIZE 1

#define EZDP_DECODE_IP_PROTOCOL_RETVAL_OTHER_OFFSET 13

#define EZDP_DECODE_IP_PROTOCOL_RETVAL_OTHER_MASK (1 << EZDP_DECODE_IP_PROTOCOL_RETVAL_OTHER_OFFSET)

#define EZDP_DECODE_IP_PROTOCOL_RETVAL_RESERVED14_31_SIZE 18

```
#define EZDP_DECODE_IP_PROTOCOL_RETVAL_RESERVED14_31_OFFSET 14

#define EZDP_DECODE_ETH_TYPE_RETVAL_IPV4_SIZE 1

#define EZDP_DECODE_ETH_TYPE_RETVAL_IPV4_OFFSET 0

#define EZDP_DECODE_ETH_TYPE_RETVAL_IPV4_MASK (1 <<
EZDP_DECODE_ETH_TYPE_RETVAL_IPV4_OFFSET)

#define EZDP_DECODE_ETH_TYPE_RETVAL_ETH_8100_SIZE 1

#define EZDP_DECODE_ETH_TYPE_RETVAL_ETH_8100_OFFSET 1

#define EZDP_DECODE_ETH_TYPE_RETVAL_ETH_8100_MASK (1 <<
EZDP_DECODE_ETH_TYPE_RETVAL_ETH_8100_OFFSET)

#define EZDP_DECODE_ETH_TYPE_RETVAL_ETH_88A8_SIZE 1

#define EZDP_DECODE_ETH_TYPE_RETVAL_ETH_88A8_OFFSET 2

#define EZDP_DECODE_ETH_TYPE_RETVAL_ETH_88A8_MASK (1 <<
EZDP_DECODE_ETH_TYPE_RETVAL_ETH_88A8_OFFSET)

#define EZDP_DECODE_ETH_TYPE_RETVAL_ARP_SIZE 1

#define EZDP_DECODE_ETH_TYPE_RETVAL_ARP_OFFSET 3

#define EZDP_DECODE_ETH_TYPE_RETVAL_ARP_MASK (1 <<
EZDP_DECODE_ETH_TYPE_RETVAL_ARP_OFFSET)

#define EZDP_DECODE_ETH_TYPE_RETVAL_MPLS_UNICAST_SIZE 1

#define EZDP_DECODE_ETH_TYPE_RETVAL_MPLS_UNICAST_OFFSET 4

#define EZDP_DECODE_ETH_TYPE_RETVAL_MPLS_UNICAST_MASK (1 <<
EZDP_DECODE_ETH_TYPE_RETVAL_MPLS_UNICAST_OFFSET)

#define EZDP_DECODE_ETH_TYPE_RETVAL_MPLS_MULTICAST_SIZE 1

#define EZDP_DECODE_ETH_TYPE_RETVAL_MPLS_MULTICAST_OFFSET 5

#define EZDP_DECODE_ETH_TYPE_RETVAL_MPLS_MULTICAST_MASK (1 <<
EZDP_DECODE_ETH_TYPE_RETVAL_MPLS_MULTICAST_OFFSET)

#define EZDP_DECODE_ETH_TYPE_RETVAL_IPV6_SIZE 1

#define EZDP_DECODE_ETH_TYPE_RETVAL_IPV6_OFFSET 6

#define EZDP_DECODE_ETH_TYPE_RETVAL_IPV6_MASK (1 <<
EZDP_DECODE_ETH_TYPE_RETVAL_IPV6_OFFSET)

#define EZDP_DECODE_ETH_TYPE_RETVAL_LENGTH_SIZE 1
```

#define EZDP_DECODE_ETH_TYPE_RETVAL_LENGTH_OFFSET 7

#define EZDP_DECODE_ETH_TYPE_RETVAL_LENGTH_MASK (1 << EZDP_DECODE_ETH_TYPE_RETVAL_LENGTH_OFFSET)

#define EZDP_DECODE_ETH_TYPE_RETVAL_USER_DEF0_SIZE 1

#define EZDP_DECODE_ETH_TYPE_RETVAL_USER_DEF0_OFFSET 8

#define EZDP_DECODE_ETH_TYPE_RETVAL_USER_DEF0_MASK (1 << EZDP_DECODE_ETH_TYPE_RETVAL_USER_DEF0_OFFSET)

#define EZDP_DECODE_ETH_TYPE_RETVAL_USER_DEF1_SIZE 1

#define EZDP_DECODE_ETH_TYPE_RETVAL_USER_DEF1_OFFSET 9

#define EZDP_DECODE_ETH_TYPE_RETVAL_USER_DEF1_MASK (1 << EZDP_DECODE_ETH_TYPE_RETVAL_USER_DEF1_OFFSET)

#define EZDP_DECODE_ETH_TYPE_RETVAL_PPPOE_SESSION_SIZE 1

#define EZDP_DECODE_ETH_TYPE_RETVAL_PPPOE_SESSION_OFFSET 10

#define EZDP_DECODE_ETH_TYPE_RETVAL_PPPOE_SESSION_MASK (1 << EZDP_DECODE_ETH_TYPE_RETVAL_PPPOE_SESSION_OFFSET)

#define EZDP_DECODE_ETH_TYPE_RETVAL_PPPOE_DISCOVERY_SIZE 1

#define EZDP_DECODE_ETH_TYPE_RETVAL_PPPOE_DISCOVERY_OFFSET 11

#define EZDP_DECODE_ETH_TYPE_RETVAL_PPPOE_DISCOVERY_MASK (1 << EZDP_DECODE_ETH_TYPE_RETVAL_PPPOE_DISCOVERY_OFFSET)

#define EZDP_DECODE_ETH_TYPE_RETVAL_OTHER_SIZE 1

#define EZDP_DECODE_ETH_TYPE_RETVAL_OTHER_OFFSET 12

#define EZDP_DECODE_ETH_TYPE_RETVAL_OTHER_MASK (1 << EZDP_DECODE_ETH_TYPE_RETVAL_OTHER_OFFSET)

#define EZDP_DECODE_ETH_TYPE_RETVAL_RESERVED13_31_SIZE 19

#define EZDP_DECODE_ETH_TYPE_RETVAL_RESERVED13_31_OFFSET 13

#define EZDP_DECODE_IPV4_RETVAL_OPTION_EXIST_SIZE 1

#define EZDP_DECODE_IPV4_RETVAL_OPTION_EXIST_OFFSET 0

#define EZDP_DECODE_IPV4_RETVAL_OPTION_EXIST_MASK (1 << EZDP_DECODE_IPV4_RETVAL_OPTION_EXIST_OFFSET)

#define EZDP_DECODE_IPV4_RETVAL_USER_CONFIG_SIP_SIZE 1

#define EZDP_DECODE_IPV4_RETVAL_USER_CONFIG_SIP_OFFSET 1

#define EZDP_DECODE_IPV4_RETVAL_USER_CONFIG_SIP_MASK (1 << EZDP_DECODE_IPV4_RETVAL_USER_CONFIG_SIP_OFFSET)

#define EZDP_DECODE_IPV4_RETVAL_RESERVED_2_6_SIZE 5

#define EZDP_DECODE_IPV4_RETVAL_RESERVED_2_6_OFFSET 2

#define EZDP_DECODE_IPV4_RETVAL_FIRST_FRAGMENT_SIZE 1

#define EZDP_DECODE_IPV4_RETVAL_FIRST_FRAGMENT_OFFSET 7

#define EZDP_DECODE_IPV4_RETVAL_FIRST_FRAGMENT_MASK (1 << EZDP_DECODE_IPV4_RETVAL_FIRST_FRAGMENT_OFFSET)

#define EZDP_DECODE_IPV4_RETVAL_ERROR_CODES_SIZE 16

#define EZDP_DECODE_IPV4_RETVAL_ERROR_CODES_OFFSET 8

#define EZDP_DECODE_IPV4_RETVAL_CONTROL_SIZE 8

#define EZDP_DECODE_IPV4_RETVAL_CONTROL_OFFSET 24

#define EZDP_DECODE_IPV4_RESULT_OPTION_EXIST_SIZE 1

#define EZDP_DECODE_IPV4_RESULT_OPTION_EXIST_OFFSET 0

#define EZDP_DECODE_IPV4_RESULT_OPTION_EXIST_WORD_SELECT 0

#define EZDP_DECODE_IPV4_RESULT_OPTION_EXIST_WORD_OFFSET 0

#define EZDP_DECODE_IPV4_RESULT_OPTION_EXIST_MASK (1 << EZDP_DECODE_IPV4_RESULT_OPTION_EXIST_WORD_OFFSET)

#define EZDP_DECODE_IPV4_RESULT_USER_CONFIG_SIP_SIZE 1

#define EZDP_DECODE_IPV4_RESULT_USER_CONFIG_SIP_OFFSET 1

#define EZDP_DECODE_IPV4_RESULT_USER_CONFIG_SIP_WORD_SELECT 0

#define EZDP_DECODE_IPV4_RESULT_USER_CONFIG_SIP_WORD_OFFSET 1

#define EZDP_DECODE_IPV4_RESULT_USER_CONFIG_SIP_MASK (1 << EZDP_DECODE_IPV4_RESULT_USER_CONFIG_SIP_WORD_OFFSET)

#define EZDP_DECODE_IPV4_RESULT_RESERVED_2_6_SIZE 5

#define EZDP_DECODE_IPV4_RESULT_RESERVED_2_6_OFFSET 2

#define EZDP_DECODE_IPV4_RESULT_FIRST_FRAGMENT_SIZE 1

#define EZDP_DECODE_IPV4_RESULT_FIRST_FRAGMENT_OFFSET 7

```
#define EZDP_DECODE_IPV4_RESULT_FIRST_FRAGMENT_WORD_SELECT 0

#define EZDP_DECODE_IPV4_RESULT_FIRST_FRAGMENT_WORD_OFFSET 7

#define EZDP_DECODE_IPV4_RESULT_FIRST_FRAGMENT_MASK (1 <<
EZDP_DECODE_IPV4_RESULT_FIRST_FRAGMENT_WORD_OFFSET)

#define EZDP_DECODE_IPV4_RESULT_ERROR_CODES_SIZE 16

#define EZDP_DECODE_IPV4_RESULT_ERROR_CODES_OFFSET 8

#define EZDP_DECODE_IPV4_RESULT_ERROR_CODES_WORD_SELECT 0

#define EZDP_DECODE_IPV4_RESULT_ERROR_CODES_WORD_OFFSET 8

#define EZDP_DECODE_IPV4_RESULT_CONTROL_SIZE 8

#define EZDP_DECODE_IPV4_RESULT_CONTROL_OFFSET 24

#define EZDP_DECODE_IPV4_RESULT_CONTROL_WORD_SELECT 0

#define EZDP_DECODE_IPV4_RESULT_CONTROL_WORD_OFFSET 24

#define EZDP_DECODE_IPV4_RESULT_SIP_DIP_HASH_SIZE 16

#define EZDP_DECODE_IPV4_RESULT_SIP_DIP_HASH_OFFSET 32

#define EZDP_DECODE_IPV4_RESULT_SIP_DIP_HASH_WORD_SELECT 1

#define EZDP_DECODE_IPV4_RESULT_SIP_DIP_HASH_WORD_OFFSET 0

#define EZDP_DECODE_IPV4_RESULT_NEXT_PROTOCOL_SIZE 8

#define EZDP_DECODE_IPV4_RESULT_NEXT_PROTOCOL_OFFSET 48

#define EZDP_DECODE_IPV4_RESULT_NEXT_PROTOCOL_WORD_SELECT 1

#define EZDP_DECODE_IPV4_RESULT_NEXT_PROTOCOL_WORD_OFFSET 16

#define EZDP_DECODE_IPV4_RESULT_RESERVED_56_63_SIZE 8

#define EZDP_DECODE_IPV4_RESULT_RESERVED_56_63_OFFSET 56

#define EZDP_DECODE_IPV4_RESULT_WORD_COUNT 2

#define EZDP_DECODE_IPV6_RETVAL_CONTROL_SIZE 8

#define EZDP_DECODE_IPV6_RETVAL_CONTROL_OFFSET 0

#define EZDP_DECODE_IPV6_RETVAL_OPTIONS_EXIST_SIZE 1

#define EZDP_DECODE_IPV6_RETVAL_OPTIONS_EXIST_OFFSET 8
```

#define EZDP_DECODE_IPV6_RETVAL_OPTIONS_EXIST_MASK (1 << EZDP_DECODE_IPV6_RETVAL_OPTIONS_EXIST_OFFSET)

#define EZDP_DECODE_IPV6_RETVAL_RESERVED9_11_SIZE 3

#define EZDP_DECODE_IPV6_RETVAL_RESERVED9_11_OFFSET 9

#define EZDP_DECODE_IPV6_RETVAL_LINK_LOCAL_ADDRESS_SIZE 1

#define EZDP_DECODE_IPV6_RETVAL_LINK_LOCAL_ADDRESS_OFFSET 12

#define EZDP_DECODE_IPV6_RETVAL_LINK_LOCAL_ADDRESS_MASK (1 << EZDP_DECODE_IPV6_RETVAL_LINK_LOCAL_ADDRESS_OFFSET)

#define EZDP_DECODE_IPV6_RETVAL_SITE_LOCAL_ADDRESS_SIZE 1

#define EZDP_DECODE_IPV6_RETVAL_SITE_LOCAL_ADDRESS_OFFSET 13

#define EZDP_DECODE_IPV6_RETVAL_SITE_LOCAL_ADDRESS_MASK (1 << EZDP_DECODE_IPV6_RETVAL_SITE_LOCAL_ADDRESS_OFFSET)

#define EZDP_DECODE_IPV6_RETVAL_GLOBAL_ADDRESSES_SIZE 1

#define EZDP_DECODE_IPV6_RETVAL_GLOBAL_ADDRESSES_OFFSET 14

#define EZDP_DECODE_IPV6_RETVAL_GLOBAL_ADDRESSES_MASK (1 << EZDP_DECODE_IPV6_RETVAL_GLOBAL_ADDRESSES_OFFSET)

#define EZDP_DECODE_IPV6_RETVAL_RESERVED_15_SIZE 1

#define EZDP_DECODE_IPV6_RETVAL_RESERVED_15_OFFSET 15

#define EZDP_DECODE_IPV6_RETVAL_ERROR_CODES_SIZE 16

#define EZDP_DECODE_IPV6_RETVAL_ERROR_CODES_OFFSET 16

#define EZDP_DECODE_IPV6_RESULT_CONTROL_SIZE 8

#define EZDP_DECODE_IPV6_RESULT_CONTROL_OFFSET 0

#define EZDP_DECODE_IPV6_RESULT_CONTROL_WORD_SELECT 0

#define EZDP_DECODE_IPV6_RESULT_CONTROL_WORD_OFFSET 0

#define EZDP_DECODE_IPV6_RESULT_OPTIONS_EXIST_SIZE 1

#define EZDP_DECODE_IPV6_RESULT_OPTIONS_EXIST_OFFSET 8

#define EZDP_DECODE_IPV6_RESULT_OPTIONS_EXIST_WORD_SELECT 0

#define EZDP_DECODE_IPV6_RESULT_OPTIONS_EXIST_WORD_OFFSET 8

```
#define EZDP_DECODE_IPV6_RESULT_OPTIONS_EXIST_MASK (1 << EZDP_DECODE_IPV6_RESULT_OPTIONS_EXIST_WORD_OFFSET)

#define EZDP_DECODE_IPV6_RESULT_RESERVED9_11_SIZE 3

#define EZDP_DECODE_IPV6_RESULT_RESERVED9_11_OFFSET 9

#define EZDP_DECODE_IPV6_RESULT_LINK_LOCAL_ADDRESS_SIZE 1

#define EZDP_DECODE_IPV6_RESULT_LINK_LOCAL_ADDRESS_OFFSET 12

#define EZDP_DECODE_IPV6_RESULT_LINK_LOCAL_ADDRESS_WORD_SELECT 0

#define EZDP_DECODE_IPV6_RESULT_LINK_LOCAL_ADDRESS_WORD_OFFSET 12

#define EZDP_DECODE_IPV6_RESULT_LINK_LOCAL_ADDRESS_MASK (1 << EZDP_DECODE_IPV6_RESULT_LINK_LOCAL_ADDRESS_WORD_OFFSET)

#define EZDP_DECODE_IPV6_RESULT_SITE_LOCAL_ADDRESS_SIZE 1

#define EZDP_DECODE_IPV6_RESULT_SITE_LOCAL_ADDRESS_OFFSET 13

#define EZDP_DECODE_IPV6_RESULT_SITE_LOCAL_ADDRESS_WORD_SELECT 0

#define EZDP_DECODE_IPV6_RESULT_SITE_LOCAL_ADDRESS_WORD_OFFSET 13

#define EZDP_DECODE_IPV6_RESULT_SITE_LOCAL_ADDRESS_MASK (1 << EZDP_DECODE_IPV6_RESULT_SITE_LOCAL_ADDRESS_WORD_OFFSET)

#define EZDP_DECODE_IPV6_RESULT_GLOBAL_ADDRESSES_SIZE 1

#define EZDP_DECODE_IPV6_RESULT_GLOBAL_ADDRESSES_OFFSET 14

#define EZDP_DECODE_IPV6_RESULT_GLOBAL_ADDRESSES_WORD_SELECT 0

#define EZDP_DECODE_IPV6_RESULT_GLOBAL_ADDRESSES_WORD_OFFSET 14

#define EZDP_DECODE_IPV6_RESULT_GLOBAL_ADDRESSES_MASK (1 << EZDP_DECODE_IPV6_RESULT_GLOBAL_ADDRESSES_WORD_OFFSET)

#define EZDP_DECODE_IPV6_RESULT_RESERVED_15_SIZE 1

#define EZDP_DECODE_IPV6_RESULT_RESERVED_15_OFFSET 15

#define EZDP_DECODE_IPV6_RESULT_ERROR_CODES_SIZE 16

#define EZDP_DECODE_IPV6_RESULT_ERROR_CODES_OFFSET 16

#define EZDP_DECODE_IPV6_RESULT_ERROR_CODES_WORD_SELECT 0

#define EZDP_DECODE_IPV6_RESULT_ERROR_CODES_WORD_OFFSET 16

#define EZDP_DECODE_IPV6_RESULT_SIP_DIP_HASH_SIZE 16
```

```
#define EZDP_DECODE_IPV6_RESULT_SIP_DIP_HASH_OFFSET 32

#define EZDP_DECODE_IPV6_RESULT_SIP_DIP_HASH_WORD_SELECT 1

#define EZDP_DECODE_IPV6_RESULT_SIP_DIP_HASH_WORD_OFFSET 0

#define EZDP_DECODE_IPV6_RESULT_NEXT_PROTOCOL_SIZE 8

#define EZDP_DECODE_IPV6_RESULT_NEXT_PROTOCOL_OFFSET 48

#define EZDP_DECODE_IPV6_RESULT_NEXT_PROTOCOL_WORD_SELECT 1

#define EZDP_DECODE_IPV6_RESULT_NEXT_PROTOCOL_WORD_OFFSET 16

#define EZDP_DECODE_IPV6_RESULT_RESERVED_56_63_SIZE 8

#define EZDP_DECODE_IPV6_RESULT_RESERVED_56_63_OFFSET 56

#define EZDP_DECODE_IPV6_RESULT_WORD_COUNT 2

#define EZDP_DECODE_MAC_RETVAL_CONTROL_SIZE 16

#define EZDP_DECODE_MAC_RETVAL_CONTROL_OFFSET 0

#define EZDP_DECODE_MAC_RETVAL_ERROR_CODES_SIZE 8

#define EZDP_DECODE_MAC_RETVAL_ERROR_CODES_OFFSET 16

#define EZDP_DECODE_MAC_RETVAL_IPV4_IN_PPPOE_SIZE 1

#define EZDP_DECODE_MAC_RETVAL_IPV4_IN_PPPOE_OFFSET 24

#define EZDP_DECODE_MAC_RETVAL_IPV4_IN_PPPOE_MASK (1 <<
EZDP_DECODE_MAC_RETVAL_IPV4_IN_PPPOE_OFFSET)

#define EZDP_DECODE_MAC_RETVAL_IPV6_IN_PPPOE_SIZE 1

#define EZDP_DECODE_MAC_RETVAL_IPV6_IN_PPPOE_OFFSET 25

#define EZDP_DECODE_MAC_RETVAL_IPV6_IN_PPPOE_MASK (1 <<
EZDP_DECODE_MAC_RETVAL_IPV6_IN_PPPOE_OFFSET)

#define EZDP_DECODE_MAC_RETVAL_RESERVED26_27_SIZE 2

#define EZDP_DECODE_MAC_RETVAL_RESERVED26_27_OFFSET 26

#define EZDP_DECODE_MAC_RETVAL_NUMBER_OF_TAGS_SIZE 3

#define EZDP_DECODE_MAC_RETVAL_NUMBER_OF_TAGS_OFFSET 28

#define EZDP_DECODE_MAC_RETVAL_RESERVED_31_SIZE 1

#define EZDP_DECODE_MAC_RETVAL_RESERVED_31_OFFSET 31
```

```
#define EZDP_DECODE_MAC_RESULT_CONTROL_SIZE 16

#define EZDP_DECODE_MAC_RESULT_CONTROL_OFFSET 0

#define EZDP_DECODE_MAC_RESULT_CONTROL_WORD_SELECT 0

#define EZDP_DECODE_MAC_RESULT_CONTROL_WORD_OFFSET 0

#define EZDP_DECODE_MAC_RESULT_ERROR_CODES_SIZE 8

#define EZDP_DECODE_MAC_RESULT_ERROR_CODES_OFFSET 16

#define EZDP_DECODE_MAC_RESULT_ERROR_CODES_WORD_SELECT 0

#define EZDP_DECODE_MAC_RESULT_ERROR_CODES_WORD_OFFSET 16

#define EZDP_DECODE_MAC_RESULT_IPV4_IN_PPPOE_SIZE 1

#define EZDP_DECODE_MAC_RESULT_IPV4_IN_PPPOE_OFFSET 24

#define EZDP_DECODE_MAC_RESULT_IPV4_IN_PPPOE_WORD_SELECT 0

#define EZDP_DECODE_MAC_RESULT_IPV4_IN_PPPOE_WORD_OFFSET 24

#define EZDP_DECODE_MAC_RESULT_IPV4_IN_PPPOE_MASK (1 <<
EZDP_DECODE_MAC_RESULT_IPV4_IN_PPPOE_WORD_OFFSET)

#define EZDP_DECODE_MAC_RESULT_IPV6_IN_PPPOE_SIZE 1

#define EZDP_DECODE_MAC_RESULT_IPV6_IN_PPPOE_OFFSET 25

#define EZDP_DECODE_MAC_RESULT_IPV6_IN_PPPOE_WORD_SELECT 0

#define EZDP_DECODE_MAC_RESULT_IPV6_IN_PPPOE_WORD_OFFSET 25

#define EZDP_DECODE_MAC_RESULT_IPV6_IN_PPPOE_MASK (1 <<
EZDP_DECODE_MAC_RESULT_IPV6_IN_PPPOE_WORD_OFFSET)

#define EZDP_DECODE_MAC_RESULT_RESERVED26_27_SIZE 2

#define EZDP_DECODE_MAC_RESULT_RESERVED26_27_OFFSET 26

#define EZDP_DECODE_MAC_RESULT_NUMBER_OF_TAGS_SIZE 3

#define EZDP_DECODE_MAC_RESULT_NUMBER_OF_TAGS_OFFSET 28

#define EZDP_DECODE_MAC_RESULT_NUMBER_OF_TAGS_WORD_SELECT 0

#define EZDP_DECODE_MAC_RESULT_NUMBER_OF_TAGS_WORD_OFFSET 28

#define EZDP_DECODE_MAC_RESULT_RESERVED_31_SIZE 1

#define EZDP_DECODE_MAC_RESULT_RESERVED_31_OFFSET 31
```

```
#define EZDP_DECODE_MAC_RESULT_TAG1_PROTOCOL_TYPE_SIZE 16

#define EZDP_DECODE_MAC_RESULT_TAG1_PROTOCOL_TYPE_OFFSET 32

#define EZDP_DECODE_MAC_RESULT_TAG1_PROTOCOL_TYPE_WORD_SELECT 1

#define EZDP_DECODE_MAC_RESULT_TAG1_PROTOCOL_TYPE_WORD_OFFSET 0

#define EZDP_DECODE_MAC_RESULT_TAG2_PROTOCOL_TYPE_SIZE 16

#define EZDP_DECODE_MAC_RESULT_TAG2_PROTOCOL_TYPE_OFFSET 48

#define EZDP_DECODE_MAC_RESULT_TAG2_PROTOCOL_TYPE_WORD_SELECT 1

#define EZDP_DECODE_MAC_RESULT_TAG2_PROTOCOL_TYPE_WORD_OFFSET 16

#define EZDP_DECODE_MAC_RESULT_TAG3_PROTOCOL_TYPE_SIZE 16

#define EZDP_DECODE_MAC_RESULT_TAG3_PROTOCOL_TYPE_OFFSET 64

#define EZDP_DECODE_MAC_RESULT_TAG3_PROTOCOL_TYPE_WORD_SELECT 2

#define EZDP_DECODE_MAC_RESULT_TAG3_PROTOCOL_TYPE_WORD_OFFSET 0

#define EZDP_DECODE_MAC_RESULT_LAST_TAG_PROTOCOL_TYPE_SIZE 16

#define EZDP_DECODE_MAC_RESULT_LAST_TAG_PROTOCOL_TYPE_OFFSET 80

#define EZDP_DECODE_MAC_RESULT_LAST_TAG_PROTOCOL_TYPE_WORD_SELECT 2

#define EZDP_DECODE_MAC_RESULT_LAST_TAG_PROTOCOL_TYPE_WORD_OFFSET 16

#define EZDP_DECODE_MAC_RESULT_DA_SA_HASH_SIZE 16

#define EZDP_DECODE_MAC_RESULT_DA_SA_HASH_OFFSET 96

#define EZDP_DECODE_MAC_RESULT_DA_SA_HASH_WORD_SELECT 3

#define EZDP_DECODE_MAC_RESULT_DA_SA_HASH_WORD_OFFSET 0

#define EZDP_DECODE_MAC_RESULT_LAYER2_SIZE_SIZE 8

#define EZDP_DECODE_MAC_RESULT_LAYER2_SIZE_OFFSET 112

#define EZDP_DECODE_MAC_RESULT_LAYER2_SIZE_WORD_SELECT 3

#define EZDP_DECODE_MAC_RESULT_LAYER2_SIZE_WORD_OFFSET 16

#define EZDP_DECODE_MAC_RESULT_RESERVED120_127_SIZE 8

#define EZDP_DECODE_MAC_RESULT_RESERVED120_127_OFFSET 120
```

```
#define EZDP_DECODE_MAC_RESULT_WORD_COUNT 4

#define EZDP_DECODE_MPLS_RETVAL_DECODE_ERROR_SIZE 1

#define EZDP_DECODE_MPLS_RETVAL_DECODE_ERROR_OFFSET 0

#define EZDP_DECODE_MPLS_RETVAL_DECODE_ERROR_MASK (1 <<
EZDP_DECODE_MPLS_RETVAL_DECODE_ERROR_OFFSET)

#define EZDP_DECODE_MPLS_RETVAL_RESERVED1_7_SIZE 7

#define EZDP_DECODE_MPLS_RETVAL_RESERVED1_7_OFFSET 1

#define EZDP_DECODE_MPLS_RETVAL_LAST_ENTRY_IN_STACK_SIZE 2

#define EZDP_DECODE_MPLS_RETVAL_LAST_ENTRY_IN_STACK_OFFSET 8

#define EZDP_DECODE_MPLS_RETVAL_RESERVED10_15_SIZE 6

#define EZDP_DECODE_MPLS_RETVAL_RESERVED10_15_OFFSET 10

#define EZDP_DECODE_MPLS_RETVAL_LABEL1_TTL_IS_ZERO_SIZE 1

#define EZDP_DECODE_MPLS_RETVAL_LABEL1_TTL_IS_ZERO_OFFSET 16

#define EZDP_DECODE_MPLS_RETVAL_LABEL1_TTL_IS_ZERO_MASK (1 <<
EZDP_DECODE_MPLS_RETVAL_LABEL1_TTL_IS_ZERO_OFFSET)

#define EZDP_DECODE_MPLS_RETVAL_LABEL2_TTL_IS_ZERO_SIZE 1

#define EZDP_DECODE_MPLS_RETVAL_LABEL2_TTL_IS_ZERO_OFFSET 17

#define EZDP_DECODE_MPLS_RETVAL_LABEL2_TTL_IS_ZERO_MASK (1 <<
EZDP_DECODE_MPLS_RETVAL_LABEL2_TTL_IS_ZERO_OFFSET)

#define EZDP_DECODE_MPLS_RETVAL_LABEL3_TTL_IS_ZERO_SIZE 1

#define EZDP_DECODE_MPLS_RETVAL_LABEL3_TTL_IS_ZERO_OFFSET 18

#define EZDP_DECODE_MPLS_RETVAL_LABEL3_TTL_IS_ZERO_MASK (1 <<
EZDP_DECODE_MPLS_RETVAL_LABEL3_TTL_IS_ZERO_OFFSET)

#define EZDP_DECODE_MPLS_RETVAL_LABEL4_TTL_IS_ZERO_SIZE 1

#define EZDP_DECODE_MPLS_RETVAL_LABEL4_TTL_IS_ZERO_OFFSET 19

#define EZDP_DECODE_MPLS_RETVAL_LABEL4_TTL_IS_ZERO_MASK (1 <<
EZDP_DECODE_MPLS_RETVAL_LABEL4_TTL_IS_ZERO_OFFSET)

#define EZDP_DECODE_MPLS_RETVAL_RESERVED20_23_SIZE 4

#define EZDP_DECODE_MPLS_RETVAL_RESERVED20_23_OFFSET 20
```

```
#define EZDP_DECODE_MPLS_RETVAL_LABEL1_TTL_IS_ONE_SIZE 1

#define EZDP_DECODE_MPLS_RETVAL_LABEL1_TTL_IS_ONE_OFFSET 24

#define EZDP_DECODE_MPLS_RETVAL_LABEL1_TTL_IS_ONE_MASK (1 <<
EZDP_DECODE_MPLS_RETVAL_LABEL1_TTL_IS_ONE_OFFSET)

#define EZDP_DECODE_MPLS_RETVAL_LABEL2_TTL_IS_ONE_SIZE 1

#define EZDP_DECODE_MPLS_RETVAL_LABEL2_TTL_IS_ONE_OFFSET 25

#define EZDP_DECODE_MPLS_RETVAL_LABEL2_TTL_IS_ONE_MASK (1 <<
EZDP_DECODE_MPLS_RETVAL_LABEL2_TTL_IS_ONE_OFFSET)

#define EZDP_DECODE_MPLS_RETVAL_LABEL3_TTL_IS_ONE_SIZE 1

#define EZDP_DECODE_MPLS_RETVAL_LABEL3_TTL_IS_ONE_OFFSET 26

#define EZDP_DECODE_MPLS_RETVAL_LABEL3_TTL_IS_ONE_MASK (1 <<
EZDP_DECODE_MPLS_RETVAL_LABEL3_TTL_IS_ONE_OFFSET)

#define EZDP_DECODE_MPLS_RETVAL_LABEL4_TTL_IS_ONE_SIZE 1

#define EZDP_DECODE_MPLS_RETVAL_LABEL4_TTL_IS_ONE_OFFSET 27

#define EZDP_DECODE_MPLS_RETVAL_LABEL4_TTL_IS_ONE_MASK (1 <<
EZDP_DECODE_MPLS_RETVAL_LABEL4_TTL_IS_ONE_OFFSET)

#define EZDP_DECODE_MPLS_RETVAL_RESERVED28_31_SIZE 4

#define EZDP_DECODE_MPLS_RETVAL_RESERVED28_31_OFFSET 28

#define EZDP_DECODE_MPLS_RESULT_DECODE_ERROR_SIZE 1

#define EZDP_DECODE_MPLS_RESULT_DECODE_ERROR_OFFSET 0

#define EZDP_DECODE_MPLS_RESULT_DECODE_ERROR_MASK (1 <<
EZDP_DECODE_MPLS_RESULT_DECODE_ERROR_OFFSET)

#define EZDP_DECODE_MPLS_RESULT_RESERVED1_7_SIZE 7

#define EZDP_DECODE_MPLS_RESULT_RESERVED1_7_OFFSET 1

#define EZDP_DECODE_MPLS_RESULT_LAST_ENTRY_IN_STACK_SIZE 2

#define EZDP_DECODE_MPLS_RESULT_LAST_ENTRY_IN_STACK_OFFSET 8

#define EZDP_DECODE_MPLS_RESULT_RESERVED10_15_SIZE 6

#define EZDP_DECODE_MPLS_RESULT_RESERVED10_15_OFFSET 10

#define EZDP_DECODE_MPLS_RESULT_LABEL1_TTL_IS_ZERO_SIZE 1
```

#define EZDP_DECODE_MPLS_RESULT_LABEL1_TTL_IS_ZERO_OFFSET 16

#define EZDP_DECODE_MPLS_RESULT_LABEL1_TTL_IS_ZERO_MASK (1 << EZDP_DECODE_MPLS_RESULT_LABEL1_TTL_IS_ZERO_OFFSET)

#define EZDP_DECODE_MPLS_RESULT_LABEL2_TTL_IS_ZERO_SIZE 1

#define EZDP_DECODE_MPLS_RESULT_LABEL2_TTL_IS_ZERO_OFFSET 17

#define EZDP_DECODE_MPLS_RESULT_LABEL2_TTL_IS_ZERO_MASK (1 << EZDP_DECODE_MPLS_RESULT_LABEL2_TTL_IS_ZERO_OFFSET)

#define EZDP_DECODE_MPLS_RESULT_LABEL3_TTL_IS_ZERO_SIZE 1

#define EZDP_DECODE_MPLS_RESULT_LABEL3_TTL_IS_ZERO_OFFSET 18

#define EZDP_DECODE_MPLS_RESULT_LABEL3_TTL_IS_ZERO_MASK (1 << EZDP_DECODE_MPLS_RESULT_LABEL3_TTL_IS_ZERO_OFFSET)

#define EZDP_DECODE_MPLS_RESULT_LABEL4_TTL_IS_ZERO_SIZE 1

#define EZDP_DECODE_MPLS_RESULT_LABEL4_TTL_IS_ZERO_OFFSET 19

#define EZDP_DECODE_MPLS_RESULT_LABEL4_TTL_IS_ZERO_MASK (1 << EZDP_DECODE_MPLS_RESULT_LABEL4_TTL_IS_ZERO_OFFSET)

#define EZDP_DECODE_MPLS_RESULT_RESERVED20_23_SIZE 4

#define EZDP_DECODE_MPLS_RESULT_RESERVED20_23_OFFSET 20

#define EZDP_DECODE_MPLS_RESULT_LABEL1_TTL_IS_ONE_SIZE 1

#define EZDP_DECODE_MPLS_RESULT_LABEL1_TTL_IS_ONE_OFFSET 24

#define EZDP_DECODE_MPLS_RESULT_LABEL1_TTL_IS_ONE_MASK (1 << EZDP_DECODE_MPLS_RESULT_LABEL1_TTL_IS_ONE_OFFSET)

#define EZDP_DECODE_MPLS_RESULT_LABEL2_TTL_IS_ONE_SIZE 1

#define EZDP_DECODE_MPLS_RESULT_LABEL2_TTL_IS_ONE_OFFSET 25

#define EZDP_DECODE_MPLS_RESULT_LABEL2_TTL_IS_ONE_MASK (1 << EZDP_DECODE_MPLS_RESULT_LABEL2_TTL_IS_ONE_OFFSET)

#define EZDP_DECODE_MPLS_RESULT_LABEL3_TTL_IS_ONE_SIZE 1

#define EZDP_DECODE_MPLS_RESULT_LABEL3_TTL_IS_ONE_OFFSET 26

#define EZDP_DECODE_MPLS_RESULT_LABEL3_TTL_IS_ONE_MASK (1 << EZDP_DECODE_MPLS_RESULT_LABEL3_TTL_IS_ONE_OFFSET)

#define EZDP_DECODE_MPLS_RESULT_LABEL4_TTL_IS_ONE_SIZE 1

```
#define EZDP_DECODE_MPLS_RESULT_LABEL4_TTL_IS_ONE_OFFSET 27

#define EZDP_DECODE_MPLS_RESULT_LABEL4_TTL_IS_ONE_MASK (1 <<
EZDP_DECODE_MPLS_RESULT_LABEL4_TTL_IS_ONE_OFFSET)

#define EZDP_DECODE_MPLS_RESULT_RESERVED28_31_SIZE 4

#define EZDP_DECODE_MPLS_RESULT_RESERVED28_31_OFFSET 28

#define EZDP_DECODE_MPLS_LABEL_RETVAL_END_OF_STACK_SIZE 1

#define EZDP_DECODE_MPLS_LABEL_RETVAL_END_OF_STACK_OFFSET 0

#define EZDP_DECODE_MPLS_LABEL_RETVAL_END_OF_STACK_MASK (1 <<
EZDP_DECODE_MPLS_LABEL_RETVAL_END_OF_STACK_OFFSET)

#define EZDP_DECODE_MPLS_LABEL_RETVAL_RESERVED_LABEL_SIZE 1

#define EZDP_DECODE_MPLS_LABEL_RETVAL_RESERVED_LABEL_OFFSET 1

#define EZDP_DECODE_MPLS_LABEL_RETVAL_RESERVED_LABEL_MASK (1 <<
EZDP_DECODE_MPLS_LABEL_RETVAL_RESERVED_LABEL_OFFSET)

#define EZDP_DECODE_MPLS_LABEL_RETVAL_TTL_IS_ZERO_SIZE 1

#define EZDP_DECODE_MPLS_LABEL_RETVAL_TTL_IS_ZERO_OFFSET 2

#define EZDP_DECODE_MPLS_LABEL_RETVAL_TTL_IS_ZERO_MASK (1 <<
EZDP_DECODE_MPLS_LABEL_RETVAL_TTL_IS_ZERO_OFFSET)

#define EZDP_DECODE_MPLS_LABEL_RETVAL_TTL_IS_ONE_SIZE 1

#define EZDP_DECODE_MPLS_LABEL_RETVAL_TTL_IS_ONE_OFFSET 3

#define EZDP_DECODE_MPLS_LABEL_RETVAL_TTL_IS_ONE_MASK (1 <<
EZDP_DECODE_MPLS_LABEL_RETVAL_TTL_IS_ONE_OFFSET)

#define EZDP_DECODE_MPLS_LABEL_RETVAL_USER_CONFIG0_SIZE 1

#define EZDP_DECODE_MPLS_LABEL_RETVAL_USER_CONFIG0_OFFSET 4

#define EZDP_DECODE_MPLS_LABEL_RETVAL_USER_CONFIG0_MASK (1 <<
EZDP_DECODE_MPLS_LABEL_RETVAL_USER_CONFIG0_OFFSET)

#define EZDP_DECODE_MPLS_LABEL_RETVAL_USER_CONFIG1_SIZE 1

#define EZDP_DECODE_MPLS_LABEL_RETVAL_USER_CONFIG1_OFFSET 5

#define EZDP_DECODE_MPLS_LABEL_RETVAL_USER_CONFIG1_MASK (1 <<
EZDP_DECODE_MPLS_LABEL_RETVAL_USER_CONFIG1_OFFSET)

#define EZDP_DECODE_MPLS_LABEL_RETVAL_USER_CONFIG2_SIZE 1
```

#define EZDP_DECODE_MPLS_LABEL_RETVAL_USER_CONFIG2_OFFSET 6

#define EZDP_DECODE_MPLS_LABEL_RETVAL_USER_CONFIG2_MASK (1 << EZDP_DECODE_MPLS_LABEL_RETVAL_USER_CONFIG2_OFFSET)

#define EZDP_DECODE_MPLS_LABEL_RETVAL_USER_CONFIG3_SIZE 1

#define EZDP_DECODE_MPLS_LABEL_RETVAL_USER_CONFIG3_OFFSET 7

#define EZDP_DECODE_MPLS_LABEL_RETVAL_USER_CONFIG3_MASK (1 << EZDP_DECODE_MPLS_LABEL_RETVAL_USER_CONFIG3_OFFSET)

#define EZDP_DECODE_MPLS_LABEL_RETVAL_EXCEPTION_BIT_SIZE 1

#define EZDP_DECODE_MPLS_LABEL_RETVAL_EXCEPTION_BIT_OFFSET 8

#define EZDP_DECODE_MPLS_LABEL_RETVAL_EXCEPTION_BIT_MASK (1 << EZDP_DECODE_MPLS_LABEL_RETVAL_EXCEPTION_BIT_OFFSET)

#define EZDP_DECODE_MPLS_LABEL_RETVAL_STOP_BIT_SIZE 1

#define EZDP_DECODE_MPLS_LABEL_RETVAL_STOP_BIT_OFFSET 9

#define EZDP_DECODE_MPLS_LABEL_RETVAL_STOP_BIT_MASK (1 << EZDP_DECODE_MPLS_LABEL_RETVAL_STOP_BIT_OFFSET)

#define EZDP_DECODE_MPLS_LABEL_RETVAL_RESERVED10_31_SIZE 22

#define EZDP_DECODE_MPLS_LABEL_RETVAL_RESERVED10_31_OFFSET 10

#define EZDP_DECODE_MPLS_LABEL_RESULT_END_OF_STACK_SIZE 1

#define EZDP_DECODE_MPLS_LABEL_RESULT_END_OF_STACK_OFFSET 0

#define EZDP_DECODE_MPLS_LABEL_RESULT_END_OF_STACK_MASK (1 << EZDP_DECODE_MPLS_LABEL_RESULT_END_OF_STACK_OFFSET)

#define EZDP_DECODE_MPLS_LABEL_RESULT_RESERVED_LABEL_SIZE 1

#define EZDP_DECODE_MPLS_LABEL_RESULT_RESERVED_LABEL_OFFSET 1

#define EZDP_DECODE_MPLS_LABEL_RESULT_RESERVED_LABEL_MASK (1 << EZDP_DECODE_MPLS_LABEL_RESULT_RESERVED_LABEL_OFFSET)

#define EZDP_DECODE_MPLS_LABEL_RESULT_TTL_IS_ZERO_SIZE 1

#define EZDP_DECODE_MPLS_LABEL_RESULT_TTL_IS_ZERO_OFFSET 2

#define EZDP_DECODE_MPLS_LABEL_RESULT_TTL_IS_ZERO_MASK (1 << EZDP_DECODE_MPLS_LABEL_RESULT_TTL_IS_ZERO_OFFSET)

#define EZDP_DECODE_MPLS_LABEL_RESULT_TTL_IS_ONE_SIZE 1

#define EZDP_DECODE_MPLS_LABEL_RESULT_TTL_IS_ONE_OFFSET 3

#define EZDP_DECODE_MPLS_LABEL_RESULT_TTL_IS_ONE_MASK (1 << EZDP_DECODE_MPLS_LABEL_RESULT_TTL_IS_ONE_OFFSET)

#define EZDP_DECODE_MPLS_LABEL_RESULT_USER_CONFIG0_SIZE 1

#define EZDP_DECODE_MPLS_LABEL_RESULT_USER_CONFIG0_OFFSET 4

#define EZDP_DECODE_MPLS_LABEL_RESULT_USER_CONFIG0_MASK (1 << EZDP_DECODE_MPLS_LABEL_RESULT_USER_CONFIG0_OFFSET)

#define EZDP_DECODE_MPLS_LABEL_RESULT_USER_CONFIG1_SIZE 1

#define EZDP_DECODE_MPLS_LABEL_RESULT_USER_CONFIG1_OFFSET 5

#define EZDP_DECODE_MPLS_LABEL_RESULT_USER_CONFIG1_MASK (1 << EZDP_DECODE_MPLS_LABEL_RESULT_USER_CONFIG1_OFFSET)

#define EZDP_DECODE_MPLS_LABEL_RESULT_USER_CONFIG2_SIZE 1

#define EZDP_DECODE_MPLS_LABEL_RESULT_USER_CONFIG2_OFFSET 6

#define EZDP_DECODE_MPLS_LABEL_RESULT_USER_CONFIG2_MASK (1 << EZDP_DECODE_MPLS_LABEL_RESULT_USER_CONFIG2_OFFSET)

#define EZDP_DECODE_MPLS_LABEL_RESULT_USER_CONFIG3_SIZE 1

#define EZDP_DECODE_MPLS_LABEL_RESULT_USER_CONFIG3_OFFSET 7

#define EZDP_DECODE_MPLS_LABEL_RESULT_USER_CONFIG3_MASK (1 << EZDP_DECODE_MPLS_LABEL_RESULT_USER_CONFIG3_OFFSET)

#define EZDP_DECODE_MPLS_LABEL_RESULT_EXCEPTION_BIT_SIZE 1

#define EZDP_DECODE_MPLS_LABEL_RESULT_EXCEPTION_BIT_OFFSET 8

#define EZDP_DECODE_MPLS_LABEL_RESULT_EXCEPTION_BIT_MASK (1 << EZDP_DECODE_MPLS_LABEL_RESULT_EXCEPTION_BIT_OFFSET)

#define EZDP_DECODE_MPLS_LABEL_RESULT_STOP_BIT_SIZE 1

#define EZDP_DECODE_MPLS_LABEL_RESULT_STOP_BIT_OFFSET 9

#define EZDP_DECODE_MPLS_LABEL_RESULT_STOP_BIT_MASK (1 << EZDP_DECODE_MPLS_LABEL_RESULT_STOP_BIT_OFFSET)

#define EZDP_DECODE_MPLS_LABEL_RESULT_RESERVED10_31_SIZE 22

#define EZDP_DECODE_MPLS_LABEL_RESULT_RESERVED10_31_OFFSET 10

Typedef Documentation

typedef uint8_t ezdp_decode_ipv4_control_t

typedef uint16_t ezdp_decode_ipv4_errors_t

typedef uint8_t ezdp_decode_ip_next_protocol_t

typedef uint8_t ezdp_decode_ipv6_control_t

typedef uint16_t ezdp_decode_ipv6_errors_t

typedef uint16_t ezdp_decode_mac_control_t

typedef uint8_t ezdp_decode_mac_errors_t

typedef uint16_t ezdp_decode_mac_protocol_type_t

typedef uint8_t ezdp_decode_tcp_errors_t

typedef uint32_t ezdp_decode_tcp_retval_t

typedef uint32_t ezdp_decode_ip_protocol_retval_t

typedef uint32_t ezdp_decode_eth_type_retval_t

typedef uint32_t ezdp_decode_ipv4_retval_t

typedef uint32_t ezdp_decode_ipv6_retval_t

typedef uint32_t ezdp_decode_mac_retval_t

typedef uint32_t ezdp_decode_mpls_retval_t

typedef uint32_t ezdp_decode_mpls_result_t

typedef uint32_t ezdp_decode_mpls_label_retval_t

typedef uint32_t ezdp_decode_mpls_label_result_t

dpe/dp/include/ezdp_defs.h File Reference

Defines

- #define likely(x) __builtin_expect(!!(x),1)
- #define unlikely(x) __builtin_expect(!!(x),0)
- #define __no_inline __attribute__((noinline))
- #define __unused __attribute__((unused))
- #define __packed __attribute__((packed))
- #define __packed_struct
- #define __aligned_cmem_ext_addr __attribute__ ((aligned (8)))
- #define __cmem
- #define __cmem_var __attribute__((section(".cmem")))
- #define __cmem_shared_var __attribute__((section(".cmem_shared")))
- #define __alter_cmem_var __attribute__((section(".cmem_alter")))
- #define __alter_cmem_shared_var __attribute__((section(".cmem_shared_alter")))
- #define __imem_private_var __attribute__((section(".fmt_slot0")))
- #define __imem_half_cluster_var __attribute__((section(".fmt_slot2")))
- #define __imem_1_cluster_var __attribute__((section(".fmt_slot4")))
- #define __imem_2_cluster_var __attribute__((section(".fmt_slot6")))
- #define __imem_4_cluster_var __attribute__((section(".fmt_slot8")))
- #define __imem_16_cluster_var __attribute__((section(".fmt_slot10")))
- #define __imem_all_cluster_var __attribute__((section(".fmt_slot12")))
- #define __emem_var __attribute__((section(".fmt_slot14")))
- #define __imem_half_cluster_func __attribute__((section(".fmt_slot3")))
- #define __imem_1_cluster_func __attribute__((section(".fmt_slot5")))
- #define __imem_2_cluster_func __attribute__((section(".fmt_slot7")))
- #define __imem_4_cluster_func __attribute__((section(".fmt_slot9")))
- #define __imem_16_cluster_func __attribute__((section(".fmt_slot11")))
- #define __imem_all_cluster_func __attribute__((section(".fmt_slot13")))

Define Documentation

#define likely(x) __builtin_expect(!!(x),1)

#define unlikely(x) __builtin_expect(!!(x),0)

#define __no_inline __attribute__((noinline))

#define __unused __attribute__((unused))

#define __packed __attribute__((packed))

#define __packed_struct

#define __aligned_cmem_ext_addr __attribute__ ((aligned (8)))

#define __cmem

#define __cmem_var __attribute__((section(".cmem")))

#define __cmem_shared_var __attribute__((section(".cmem_shared")))

#define __alter_cmem_var __attribute__((section(".cmem_alter")))

#define __alter_cmem_shared_var __attribute__((section(".cmem_shared_alter")))

#define __imem_private_var __attribute__((section(".fmt_slot0")))

#define __imem_half_cluster_var __attribute__((section(".fmt_slot2")))

#define __imem_1_cluster_var __attribute__((section(".fmt_slot4")))

#define __imem_2_cluster_var __attribute__((section(".fmt_slot6")))

#define __imem_4_cluster_var __attribute__((section(".fmt_slot8")))

#define __imem_16_cluster_var __attribute__((section(".fmt_slot10")))

#define __imem_all_cluster_var __attribute__((section(".fmt_slot12")))

#define __emem_var __attribute__((section(".fmt_slot14")))

#define __imem_half_cluster_func __attribute__((section(".fmt_slot3")))

#define __imem_1_cluster_func __attribute__((section(".fmt_slot5")))

#define __imem_2_cluster_func __attribute__((section(".fmt_slot7")))

#define __imem_4_cluster_func __attribute__((section(".fmt_slot9")))

#define __imem_16_cluster_func __attribute__((section(".fmt_slot11")))

#define __imem_all_cluster_func __attribute__((section(".fmt_slot13")))

dpe/dp/include/ezdp_dma.h File Reference

Functions

- static __always_inline uint32_t ezdp_copy_data_by_ext_addr (struct ezdp_ext_addr __cmem *dst_ptr, struct ezdp_ext_addr __cmem *src_ptr, uint32_t size, uint32_t flags)
- *Copy data between two extended addresses.* static __always_inline void ezdp_copy_data_by_ext_addr_async (struct ezdp_ext_addr __cmem *dst_ptr, struct ezdp_ext_addr __cmem *src_ptr, uint32_t size, uint32_t flags)
- *Non blocking version of ezdp_copy_data_by_ext_addr().* static __always_inline void ezdp_load_data_from_ext_addr (void __cmem *dst_ptr, struct ezdp_ext_addr __cmem *src_ptr, uint32_t size, uint32_t flags)
- *Copy data from an extended address to CMEM.* static __always_inline void ezdp_load_data_from_ext_addr_async (void __cmem *dst_ptr, struct ezdp_ext_addr __cmem *src_ptr, uint32_t size, uint32_t flags)
- *Non blocking version of ezdp_load_data_from_ext_addr().* static __always_inline uint32_t ezdp_load_16_byte_data_from_ext_addr (void __cmem *dst_ptr, struct ezdp_ext_addr *src_ptr, uint32_t flags)
- *Copy 16 bytes from an extended address to CMEM.* static __always_inline void ezdp_load_16_byte_data_from_ext_addr_async (void __cmem *dst_ptr, struct ezdp_ext_addr *src_ptr, uint32_t flags)
- *Non blocking version of ezdp_load_16_byte_data_from_ext_addr().* static __always_inline uint32_t ezdp_load_32_byte_data_from_ext_addr (void __cmem *dst_ptr, struct ezdp_ext_addr *src_ptr, uint32_t flags)
- *Copy 32 bytes from an extended address to CMEM.* static __always_inline void ezdp_load_32_byte_data_from_ext_addr_async (void __cmem *dst_ptr, struct ezdp_ext_addr *src_ptr, uint32_t flags)
- *Non blocking version of ezdp_load_32_byte_data_from_ext_addr().* static __always_inline uint32_t ezdp_store_data_to_ext_addr (struct ezdp_ext_addr __cmem *dst_ptr, void __cmem *src_ptr, uint32_t size, uint32_t flags)
- *Copy data from CMEM to an extended address.* static __always_inline void ezdp_store_data_to_ext_addr_async (struct ezdp_ext_addr __cmem *dst_ptr, void __cmem *src_ptr, uint32_t size, uint32_t flags)
- *Non blocking version of ezdp_store_data_to_ext_addr().* static __always_inline void ezdp_store_16_byte_data_to_ext_addr (struct ezdp_ext_addr *dst_ptr, void __cmem *src_ptr, uint32_t flags)
- *Copy 16 bytes from CMEM to an extended address.* static __always_inline void ezdp_store_16_byte_data_to_ext_addr_async (struct ezdp_ext_addr *dst_ptr, void __cmem *src_ptr, uint32_t flags)
- *Non blocking version of ezdp_store_16_byte_data_to_ext_addr().* static __always_inline void ezdp_store_32_byte_data_to_ext_addr (struct ezdp_ext_addr *dst_ptr, void __cmem *src_ptr, uint32_t flags)
- *Copy 32 bytes from CMEM to an extended address.* static __always_inline void ezdp_store_32_byte_data_to_ext_addr_async (struct ezdp_ext_addr *dst_ptr, void __cmem *src_ptr, uint32_t flags)
- *Non blocking version of ezdp_store_32_byte_data_to_ext_addr().* static __always_inline uint32_t ezdp_load_data_from_sum_addr (struct ezdp_sum_addr __cmem *src_sum_addr_ptr, uint32_t sum_addr_entry_size, uint32_t sum_addr_offset, uint8_t __cmem *dst_ptr, uint32_t size, uint32_t flags)
- *Load data from a summarized address to CMEM.* static __always_inline void ezdp_load_data_from_sum_addr_async (struct ezdp_sum_addr __cmem *src_sum_addr_ptr, uint32_t sum_addr_entry_size, uint32_t sum_addr_offset, uint8_t __cmem *dst_ptr, uint32_t size, uint32_t flags)
- *Non blocking version of ezdp_load_data_from_sum_addr().* static __always_inline uint32_t ezdp_load_16_byte_data_from_sum_addr (ezdp_sum_addr_t sum_addr, uint32_t entry_size, uint32_t offset, uint8_t __cmem *ptr, uint32_t flags)
- *Load 16 bytes from a summarized address to CMEM.* static __always_inline void ezdp_load_16_byte_data_from_sum_addr_async (ezdp_sum_addr_t sum_addr, uint32_t entry_size, uint32_t offset, uint8_t __cmem *ptr, uint32_t flags)
- *Load 16 bytes from a summarized address to CMEM.* static __always_inline uint32_t ezdp_load_32_byte_data_from_sum_addr (ezdp_sum_addr_t sum_addr, uint32_t entry_size, uint32_t offset, uint8_t __cmem *ptr, uint32_t flags)
- *Load 32 bytes from a summarized address to CMEM.* static __always_inline void ezdp_load_32_byte_data_from_sum_addr_async (ezdp_sum_addr_t sum_addr, uint32_t entry_size, uint32_t offset, uint8_t __cmem *ptr, uint32_t flags)

- *Load 32 bytes from a summarized address to CMEM.* static __always_inline uint32_t ezdp_store_data_to_sum_addr (uint8_t __cmem *src_ptr, struct ezdp_sum_addr __cmem *dst_sum_addr_ptr, uint32_t sum_addr_entry_size, uint32_t sum_addr_offset, uint32_t size, uint32_t flags)
- *Store data from CMEM to summarized address.* static __always_inline void ezdp_store_data_to_sum_addr_async (uint8_t __cmem *src_ptr, struct ezdp_sum_addr __cmem *dst_sum_addr_ptr, uint32_t sum_addr_entry_size, uint32_t sum_addr_offset, uint32_t size, uint32_t flags)
- *Non blocking version of ezdp_store_data_to_sum_addr().* static __always_inline void ezdp_store_16_byte_data_to_sum_addr (uint8_t __cmem *ptr, ezdp_sum_addr_t sum_addr, uint32_t entry_size, uint32_t offset, uint32_t flags)
- *Store 16 bytes from CMEM to a summarized address.* static __always_inline void ezdp_store_16_byte_data_to_sum_addr_async (uint8_t __cmem *ptr, ezdp_sum_addr_t sum_addr, uint32_t entry_size, uint32_t offset, uint32_t flags)
- *Non blocking version of ezdp_store_16_byte_data_to_sum_addr().* static __always_inline void ezdp_store_32_byte_data_to_sum_addr (uint8_t __cmem *ptr, ezdp_sum_addr_t sum_addr, uint32_t entry_size, uint32_t offset, uint32_t flags)
- *Store 32 bytes from CMEM to a summarized address.* static __always_inline void ezdp_store_32_byte_data_to_sum_addr_async (uint8_t __cmem *ptr, ezdp_sum_addr_t sum_addr, uint32_t entry_size, uint32_t offset, uint32_t flags)

Non blocking version of ezdp_store_32_byte_data_to_sum_addr().

Function Documentation

static __always_inline uint32_t ezdp_copy_data_by_ext_addr (struct ezdp_ext_addr __cmem * *dst_ptr*, struct ezdp_ext_addr __cmem * *src_ptr*, uint32_t *size*, uint32_t *flags*) [static]

Copy data between two extended addresses.

Parameters:

[in] *dst_ptr* - pointer to destination extended address (in CMEM in 8 byte alignment)
[in] *src_ptr* - pointer to source extended address (in CMEM in 8 byte alignment)
[in] *size* - number of bytes to copy
[in] *flags* - execution flags. Bitwise OR of zero or more flags out of the following:
EZDP_MEMORY_FLAG_OVERWRITE - override ECC (do not merge)
EZDP_MEMORY_FLAG_UNCACHED - do not allocate new cache entry

Note:

In EMEM atomic operation is limited to 128 bytes. In IMEM atomic operation is limited to 32 bytes.

Returns:

uint32_t - 16 bit checksum value

static __always_inline void ezdp_copy_data_by_ext_addr_async (struct ezdp_ext_addr __cmem * *dst_ptr*, struct ezdp_ext_addr __cmem * *src_ptr*, uint32_t *size*, uint32_t *flags*) [static]

Non blocking version of ezdp_copy_data_by_ext_addr().

Parameters:

[in] *dst_ptr* - pointer to destination extended address (in CMEM in 8 byte alignment)
[in] *src_ptr* - pointer to source extended address (in CMEM in 8 byte alignment)
[in] *size* - number of bytes to copy
[in] *flags* - execution flags. Bitwise OR of zero or more flags out of the following:
EZDP_MEMORY_FLAG_OVERWRITE - override ECC (do not merge)
EZDP_MEMORY_FLAG_UNCACHED - do not allocate new cache entry

Note:

Call ezdp_sync() to wait for the operation to complete, confirming that the data was copied to the destination.

Returns:

none

static __always_inline void ezdp_load_data_from_ext_addr (void __cmem * *dst_ptr*, struct ezdp_ext_addr __cmem * *src_ptr*, uint32_t *size*, uint32_t *flags*) [static]

Copy data from an extended address to CMEM.

Parameters:

[out] *dst_ptr* - pointer to the destination array in CMEM where the content is to be copied
[in] *src_ptr* - pointer to the source extended address (in CMEM in 8 byte alignment) to be copied
[in] *size* - number of bytes to copy
[in] *flags* - execution flags. Bitwise OR of zero or more flags out of the following:
EZDP_MEMORY_FLAG_UNCACHED - do not allocate new cache entry

Note:

In EMEM atomic operation is limited to 128 bytes. In IMEM atomic operation is limited to 32 bytes.

Returns:

none

static __always_inline void ezdp_load_data_from_ext_addr_async (void __cmem * *dst_ptr*, struct ezdp_ext_addr __cmem * *src_ptr*, uint32_t *size*, uint32_t *flags*) [static]

Non blocking version of ezdp_load_data_from_ext_addr().

Parameters:

[out] *dst_ptr* - pointer in CMEM to copy data to
[in] *src_ptr* - pointer to source extended address (in CMEM in 8 byte alignment)
[in] *size* - number of bytes to copy
[in] *flags* - execution flags. Bitwise OR of zero or more flags out of the following:
EZDP_MEMORY_FLAG_UNCACHED - do not allocate new cache entry

Note:

Call ezdp_sync() to wait for the operation to complete, confirming that the data was copied to the destination.

Returns:

none

static __always_inline uint32_t ezdp_load_16_byte_data_from_ext_addr (void __cmem * *dst_ptr*, struct ezdp_ext_addr * *src_ptr*, uint32_t *flags*) [static]

Copy 16 bytes from an extended address to CMEM.

Parameters:

[out] *dst_ptr* - pointer to the destination array in CMEM where the content is to be copied
[in] *src_ptr* - pointer to the source extended address
[in] *flags* - execution flags. Bitwise OR of zero or more flags. No flags defined for this function yet.

Note:

The extended address must be 16-byte aligned.

Returns:
uint32_t - first 4 bytes of the copied data

static __always_inline void ezdp_load_16_byte_data_from_ext_addr_async (void __cmem * *dst_ptr*, struct ezdp_ext_addr * *src_ptr*, uint32_t *flags*) [static]

Non blocking version of ezdp_load_16_byte_data_from_ext_addr().

Parameters:
[out] *dst_ptr* - pointer in CMEM to copy data to
[in] *src_ptr* - pointer to source extended address
[in] *flags* - execution flags. Bitwise OR of zero or more flags. No flags defined for this function yet.

Note:
Call ezdp_sync() to wait for the operation to complete, confirming that the data was copied to the destination.

Returns:
none

static __always_inline uint32_t ezdp_load_32_byte_data_from_ext_addr (void __cmem * *dst_ptr*, struct ezdp_ext_addr * *src_ptr*, uint32_t *flags*) [static]

Copy 32 bytes from an extended address to CMEM.

Parameters:
[out] *dst_ptr* - pointer to the destination array in CMEM where the content is to be copied
[in] *src_ptr* - pointer to the source extended address
[in] *flags* - execution flags. Bitwise OR of zero or more flags. No flags defined for this function yet.

Note:
The extended address must be 16-byte aligned.

Returns:
uint32_t - first 4 bytes of the copied data

static __always_inline void ezdp_load_32_byte_data_from_ext_addr_async (void __cmem * *dst_ptr*, struct ezdp_ext_addr * *src_ptr*, uint32_t *flags*) [static]

Non blocking version of ezdp_load_32_byte_data_from_ext_addr().

Parameters:
[out] *dst_ptr* - pointer in CMEM to copy data to
[in] *src_ptr* - pointer to source extended address
[in] *flags* - execution flags. Bitwise OR of zero or more flags. No flags defined for this function yet.

Note:
Call ezdp_sync() to wait for the operation to complete, confirming that the data was copied to the destination.

Returns:
none

static __always_inline uint32_t ezdp_store_data_to_ext_addr (struct ezdp_ext_addr __cmem * *dst_ptr*, void __cmem * *src_ptr*, uint32_t *size*, uint32_t *flags*) [static]

Copy data from CMEM to an extended address.

Parameters:

[in] *dst_ptr* - pointer to destination extended address (in CMEM in 8 byte alignment)
[in] *src_ptr* - pointer in CMEM to copy data from
[in] *size* - number of bytes to copy
[in] *flags* - execution flags. Bitwise OR of zero or more flags out of the following:
EZDP_MEMORY_FLAG_OVERWRITE - override ECC (do not merge)
EZDP_MEMORY_FLAG_UNCACHED - do not allocate new cache entry

Note:

returned 16 bit checksum is applicable only when memory type is external or memory type is internal and msid > 5 In EMEM atomic operation is limited to 128 bytes. In IMEM atomic operation is limited to 32 bytes.

Returns:

uint32_t - 16 bit checksum value

static __always_inline void ezdp_store_data_to_ext_addr_async (struct ezdp_ext_addr __cmem * *dst_ptr*, void __cmem * *src_ptr*, uint32_t *size*, uint32_t *flags*) [static]

Non blocking version of ezdp_store_data_to_ext_addr().

Parameters:

[in] *dst_ptr* - pointer to destination extended address (in CMEM in 8 byte alignment)
[in] *src_ptr* - pointer in CMEM to copy data from
[in] *size* - number of bytes to copy
[in] *flags* - execution flags. Bitwise OR of zero or more flags out of the following:
EZDP_MEMORY_FLAG_OVERWRITE - override ECC (do not merge)
EZDP_MEMORY_FLAG_UNCACHED - do not allocate new cache entry

Note:

Call ezdp_sync() to wait for the operation to complete, confirming that the data was copied to the destination.

Returns:

non

static __always_inline void ezdp_store_16_byte_data_to_ext_addr (struct ezdp_ext_addr * *dst_ptr*, void __cmem * *src_ptr*, uint32_t *flags*) [static]

Copy 16 bytes from CMEM to an extended address.

Parameters:

[in] *dst_ptr* - pointer to destination extended address
[in] *src_ptr* - pointer in CMEM to copy data from
[in] *flags* - execution flags. Bitwise OR of zero or more flags. No flags defined for this function yet.

Note:

The extended address must be 16-byte aligned.

Returns:

void

static __always_inline void ezdp_store_16_byte_data_to_ext_addr_async (struct ezdp_ext_addr * *dst_ptr*, void __cmem * *src_ptr*, uint32_t *flags*) [static]

Non blocking version of ezdp_store_16_byte_data_to_ext_addr().

Parameters:

[in] *dst_ptr* - pointer in CMEM to copy data to
[in] *src_ptr* - pointer to source extended address
[in] *flags* - execution flags. Bitwise OR of zero or more flags. No flags defined for this function yet.

Note:

Call ezdp_sync() to wait for the operation to complete, confirming that the data was copied to the destination.

Returns:

none

static __always_inline void ezdp_store_32_byte_data_to_ext_addr (struct ezdp_ext_addr * *dst_ptr*, void __cmem * *src_ptr*, uint32_t *flags*) `[static]`

Copy 32 bytes from CMEM to an extended address.

Parameters:

[in] *dst_ptr* - pointer to destination extended address
[in] *src_ptr* - pointer in CMEM to copy data from
[in] *flags* - execution flags. Bitwise OR of zero or more flags. No flags defined for this function yet.

Note:

The extended address must be 16-byte aligned.

Returns:

void

static __always_inline void ezdp_store_32_byte_data_to_ext_addr_async (struct ezdp_ext_addr * *dst_ptr*, void __cmem * *src_ptr*, uint32_t *flags*) `[static]`

Non blocking version of ezdp_store_32_byte_data_to_ext_addr().

Parameters:

[in] *dst_ptr* - pointer in CMEM to copy data to
[in] *src_ptr* - pointer to source extended address
[in] *flags* - execution flags. Bitwise OR of zero or more flags. No flags defined for this function yet.

Note:

Call ezdp_sync() to wait for the operation to complete, confirming that the data was copied to the destination.

Returns:

none

static __always_inline uint32_t ezdp_load_data_from_sum_addr (struct ezdp_sum_addr __cmem * *src_sum_addr_ptr*, uint32_t *sum_addr_entry_size*, uint32_t *sum_addr_offset*, uint8_t __cmem * *dst_ptr*, uint32_t *size*, uint32_t *flags*) `[static]`

Load data from a summarized address to CMEM.

Parameters:

[in] *src_sum_addr_ptr* - pointer to summarized address in CMEM

[in] *sum_addr_entry_size* - the entry size in bytes. (Applicable values are 16,32,64,128,256)
[in] *sum_addr_offset* - the offset in bytes to add to base address (Applicable values are 0,16,32,48,64,...,240)
[out] *dst_ptr* - pointer to the destination in CMEM
[in] *size* - the amount of data to load in byte (Applicable values are 0,16,32,48,64,...,240,256)
[in] *flags* - execution flags. Bitwise OR of zero or more flags out of the following:
EZDP_MEMORY_FLAG_UNCACHED - do not allocate new cache entry

Note:

In EMEM atomic operation is limited to 128 bytes. In IMEM atomic operation is limited to 32 bytes.

Returns:

uint32_t - first 4 bytes of the copied data

static __always_inline void ezdp_load_data_from_sum_addr_async (struct ezdp_sum_addr __cmem * *src_sum_addr_ptr*, uint32_t *sum_addr_entry_size*, uint32_t *sum_addr_offset*, uint8_t __cmem * *dst_ptr*, uint32_t *size*, uint32_t *flags*) `[static]`

Non blocking version of ezdp_load_data_from_sum_addr().

Parameters:

[in] *src_sum_addr_ptr* - pointer to summarized address in CMEM
[in] *sum_addr_entry_size* - the entry size in bytes. (Applicable values are 16,32,64,128,256)
[in] *sum_addr_offset* - the offset in bytes to add to base address (Applicable values are 0,16,32,48,64,...,240)
[out] *dst_ptr* - pointer to the destination in CMEM
[in] *size* - the amount of data to load in byte (Applicable values are 0,16,32,48,64,...,240,256)
[in] *flags* - execution flags. Bitwise OR of zero or more flags out of the following:
EZDP_MEMORY_FLAG_UNCACHED - do not allocate new cache entry

Note:

Call ezdp_sync() to wait for the operation to complete, confirming that the data was copied to the destination.

Returns:

void

static __always_inline uint32_t ezdp_load_16_byte_data_from_sum_addr (ezdp_sum_addr_t *sum_addr*, uint32_t *entry_size*, uint32_t *offset*, uint8_t __cmem * *ptr*, uint32_t *flags*) `[static]`

Load 16 bytes from a summarized address to CMEM.

Parameters:

[in] *sum_addr* - summarized address
[in] *entry_size* - size of entry
[in] *offset* - offset into memory
[in] *ptr* - pointer to address in CMEM
[in] *flags* - execution flags. Bitwise OR of zero or more flags out of the following:
EZDP_MEMORY_FLAG_UNCACHED - do not allocate new cache entry

Returns:

uint32_t - first 4 bytes of the copied data

static __always_inline void ezdp_load_16_byte_data_from_sum_addr_async (ezdp_sum_addr_t *sum_addr*, uint32_t *entry_size*, uint32_t *offset*, uint8_t __cmem * *ptr*, uint32_t *flags*) `[static]`

Load 16 bytes from a summarized address to CMEM.

Parameters:

[in] *sum_addr* - summarized address
[in] *entry_size* - size of entry
[in] *offset* - offset into memory
[in] *ptr* - pointer to address in CMEM
[in] *flags* - execution flags. Bitwise OR of zero or more flags out of the following:
EZDP_MEMORY_FLAG_UNCACHED - do not allocate new cache entry

Note:

Call ezdp_sync() to wait for the operation to complete, confirming that the data was copied to the destination.

Returns:

void

static __always_inline uint32_t ezdp_load_32_byte_data_from_sum_addr (ezdp_sum_addr_t *sum_addr*, uint32_t *entry_size*, uint32_t *offset*, uint8_t __cmem * *ptr*, uint32_t *flags*) [static]

Load 32 bytes from a summarized address to CMEM.

Parameters:

[in] *sum_addr* - summarized address
[in] *entry_size* - size of entry
[in] *offset* - offset into memory
[in] *ptr* - pointer to address in CMEM
[in] *flags* - execution flags. Bitwise OR of zero or more flags out of the following:
EZDP_MEMORY_FLAG_UNCACHED - do not allocate new cache entry

Returns:

uint32_t - first 4 bytes of the copied data

static __always_inline void ezdp_load_32_byte_data_from_sum_addr_async (ezdp_sum_addr_t *sum_addr*, uint32_t *entry_size*, uint32_t *offset*, uint8_t __cmem * *ptr*, uint32_t *flags*) [static]

Load 32 bytes from a summarized address to CMEM.

Parameters:

[in] *sum_addr* - summarized address
[in] *entry_size* - size of entry
[in] *offset* - offset into memory
[in] *ptr* - pointer to address in CMEM
[in] *flags* - execution flags. Bitwise OR of zero or more flags out of the following:
EZDP_MEMORY_FLAG_UNCACHED - do not allocate new cache entry

Note:

Call ezdp_sync() to wait for the operation to complete, confirming that the data was copied to the destination.

Returns:

void

static __always_inline uint32_t ezdp_store_data_to_sum_addr (uint8_t __cmem * *src_ptr*, struct ezdp_sum_addr __cmem * *dst_sum_addr_ptr*, uint32_t *sum_addr_entry_size*, uint32_t *sum_addr_offset*, uint32_t *size*, uint32_t *flags*) [static]

Store data from CMEM to summarized address.

Parameters:

[in] *src_ptr* - pointer to the source in CMEM
[out] *dst_sum_addr_ptr* - pointer to summarized address of the destination in CMEM
[in] *sum_addr_entry_size* - the entry size in bytes. (Applicable values are 16,32,64,128,256)
[in] *sum_addr_offset* - the offset in bytes to add to base address (Applicable values are 0,16,32,48,64,...,240)
[in] *size* - the amount of data to load in byte (Applicable values are 0,16,32,48,64,...,240,256)
[in] *flags* - execution flags. Bitwise OR of zero or more flags out of the following:
EZDP_MEMORY_FLAG_OVERWRITE - override ECC (do not merge)
EZDP_MEMORY_FLAG_UNCACHED - do not allocate new cache entry

Note:

returned 16 bit checksum is applicable only when memory type is external or memory type is internal and msid > 5. In EMEM atomic operation is limited to 128 bytes. In IMEM atomic operation is limited to 32 bytes.

Returns:

uint32_t - 16 bit checksum value

static __always_inline void ezdp_store_data_to_sum_addr_async (uint8_t __cmem * *src_ptr*, struct ezdp_sum_addr __cmem * *dst_sum_addr_ptr*, uint32_t *sum_addr_entry_size*, uint32_t *sum_addr_offset*, uint32_t *size*, uint32_t *flags*) [static]

Non blocking version of ezdp_store_data_to_sum_addr().

Parameters:

[in] *src_ptr* - pointer to the source in CMEM
[out] *dst_sum_addr_ptr* - pointer to summarized address of the destination in CMEM
[in] *sum_addr_entry_size* - the entry size in bytes. (Applicable values are 16,32,64,128,256)
[in] *sum_addr_offset* - the offset in bytes to add to base address (Applicable values are 0,16,32,48,64,...,240)
[in] *size* - the amount of data to load in byte (Applicable values are 0,16,32,48,64,...,240,256)
[in] *flags* - execution flags. Bitwise OR of zero or more flags out of the following:
EZDP_MEMORY_FLAG_OVERWRITE - override ECC (do not merge)
EZDP_MEMORY_FLAG_UNCACHED - do not allocate new cache entry

Note:

Call ezdp_sync() to wait for the operation to complete, confirming that the data was copied to the destination.

Returns:

void

static __always_inline void ezdp_store_16_byte_data_to_sum_addr (uint8_t __cmem * *ptr*, ezdp_sum_addr_t *sum_addr*, uint32_t *entry_size*, uint32_t *offset*, uint32_t *flags*) [static]

Store 16 bytes from CMEM to a summarized address.

Parameters:

[in] *ptr* - pointer to address in CMEM

[in] *sum_addr* - summarized address
[in] *entry_size* - size of entry
[in] *offset* - offset into memory
[in] *flags* - execution flags. Bitwise OR of zero or more flags out of the following:
EZDP_MEMORY_FLAG_UNCACHED - do not allocate new cache entry

Returns:

void

static __always_inline void ezdp_store_16_byte_data_to_sum_addr_async (uint8_t __cmem * *ptr*, ezdp_sum_addr_t *sum_addr*, uint32_t *entry_size*, uint32_t *offset*, uint32_t *flags*) [static]

Non blocking version of ezdp_store_16_byte_data_to_sum_addr().

Parameters:

[in] *ptr* - pointer to address in CMEM
[in] *sum_addr* - summarized address
[in] *entry_size* - size of entry
[in] *offset* - offset into memory
[in] *flags* - execution flags. Bitwise OR of zero or more flags out of the following:
EZDP_MEMORY_FLAG_UNCACHED - do not allocate new cache entry

Note:

Call ezdp_sync() to wait for the operation to complete, confirming that the data was copied to the destination.

Returns:

void

static __always_inline void ezdp_store_32_byte_data_to_sum_addr (uint8_t __cmem * *ptr*, ezdp_sum_addr_t *sum_addr*, uint32_t *entry_size*, uint32_t *offset*, uint32_t *flags*) [static]

Store 32 bytes from CMEM to a summarized address.

Parameters:

[in] *ptr* - pointer to address in CMEM
[in] *sum_addr* - summarized address
[in] *entry_size* - size of entry
[in] *offset* - offset into memory
[in] *flags* - execution flags. Bitwise OR of zero or more flags out of the following:
EZDP_MEMORY_FLAG_UNCACHED - do not allocate new cache entry

Returns:

void

static __always_inline void ezdp_store_32_byte_data_to_sum_addr_async (uint8_t __cmem * *ptr*, ezdp_sum_addr_t *sum_addr*, uint32_t *entry_size*, uint32_t *offset*, uint32_t *flags*) [static]

Non blocking version of ezdp_store_32_byte_data_to_sum_addr().

Parameters:

[in] *ptr* - pointer to address in CMEM
[in] *sum_addr* - summarized address
[in] *entry_size* - size of entry
[in] *offset* - offset into memory

[in] *flags* - execution flags. Bitwise OR of zero or more flags out of the following:
EZDP_MEMORY_FLAG_UNCACHED - do not allocate new cache entry

Note:

Call ezdp_sync() to wait for the operation to complete, confirming that the data was copied to the destination.

Returns:

void

dpe/dp/include/ezdp_frame.h File Reference

Functions

- static __always_inline ezdp_buffer_desc_t ezdp_alloc_buf (enum ezdp_buffer_mem_type pool_type, uint32_t buf_budget_id)
- *Allocate a single frame buffer.* static __always_inline void ezdp_free_buf (uint32_t buf_budget_id, ezdp_buffer_desc_t bd_raw_data)
- *Free a single frame buffer.* static __always_inline void ezdp_free_buf_async (uint32_t buf_budget_id, ezdp_buffer_desc_t bd_raw_data)
- *Non blocking version of ezdp_free_buf().* static __always_inline ezdp_buffer_desc_t ezdp_alloc_multi_buf (enum ezdp_buffer_mem_type pool_type, uint32_t buf_budget_id, uint32_t num_of_bufs, struct ezdp_buffer_desc __cmem *bds_ptr)
- *Allocate multiple frame buffers.* static __always_inline void ezdp_alloc_multi_buf_async (enum ezdp_buffer_mem_type pool_type, uint32_t buf_budget_id, uint32_t num_of_bufs, struct ezdp_buffer_desc __cmem *bds_ptr)
- *Non blocking version of ezdp_alloc_multi_buf().* static __always_inline bool ezdp_buf_alloc_failed (ezdp_buffer_desc_t ret)
- *Check if allocation of the buffer failed.* static __always_inline void ezdp_free_multi_buf (uint32_t buf_budget_id, uint32_t num_of_bufs, struct ezdp_buffer_desc __cmem *bds_ptr)
- *Free multiple frame buffers.* static __always_inline void ezdp_free_multi_buf_async (uint32_t buf_budget_id, uint32_t num_of_bufs, struct ezdp_buffer_desc __cmem *bds_ptr)
- *Non blocking version of ezdp_free_multi_buf().* static __always_inline uint32_t ezdp_read_free_buf (enum ezdp_buffer_mem_type pool_type)
- *The number of buffers available to be obtained.* static __always_inline void ezdp_rebudget_buf (enum ezdp_buffer_mem_type pool_type, uint32_t free_buf_budget_id, uint32_t free_num_of_bufs, uint32_t alloc_buf_budget_id, uint32_t alloc_num_of_bufs)
- *Update the budget to which buffers are credited.* static __always_inline void ezdp_rebudget_buf_async (enum ezdp_buffer_mem_type pool_type, uint32_t free_buf_budget_id, uint32_t free_num_of_bufs, uint32_t alloc_buf_budget_id, uint32_t alloc_num_of_bufs)
- *Non blocking version of ezdp_rebudget_buf().* static __always_inline uint32_t ezdp_copy_frame_data (struct ezdp_buffer_desc __cmem *dst_bd_ptr, uint32_t dst_bd_offset, struct ezdp_buffer_desc __cmem *src_bd_ptr, uint32_t src_bd_offset, uint32_t size, uint32_t flags)
- *Copy data between two frame buffers.* static __always_inline void ezdp_copy_frame_data_async (struct ezdp_buffer_desc __cmem *dst_bd_ptr, uint32_t dst_bd_offset, struct ezdp_buffer_desc __cmem *src_bd_ptr, uint32_t src_bd_offset, uint32_t size, uint32_t flags)
- *Non blocking version of ezdp_copy_frame_data().* static __always_inline uint32_t ezdp_clone_frame_data (struct ezdp_buffer_desc __cmem *dst_bd_ptr, struct ezdp_buffer_desc __cmem *src_bd_ptr, uint32_t bd_offset, uint32_t size, uint32_t flags)
- *Copy data between two frame buffers, with the same source and destination offset (optimized).* static __always_inline void ezdp_clone_frame_data_async (struct ezdp_buffer_desc __cmem *dst_bd_ptr, struct ezdp_buffer_desc __cmem *src_bd_ptr, uint32_t bd_offset, uint32_t size, uint32_t flags)
- *Non blocking version of ezdp_clone_frame_data().* static __always_inline uint32_t ezdp_copy_frame_data_to_ext_addr (struct ezdp_ext_addr __cmem *dst_ptr, struct ezdp_buffer_desc __cmem *src_bd_ptr, uint32_t src_bd_offset, uint32_t size, uint32_t flags)
- *Copy data from a frame buffer to an extended address.* static __always_inline void ezdp_copy_frame_data_to_ext_addr_async (struct ezdp_ext_addr __cmem *dst_ptr, struct ezdp_buffer_desc __cmem *src_bd_ptr, uint32_t src_bd_offset, uint32_t size, uint32_t flags)
- *Non blocking version of ezdp_copy_frame_data_to_ext_addr().* static __always_inline uint32_t ezdp_copy_frame_data_from_ext_addr (struct ezdp_buffer_desc __cmem *dst_bd_ptr, uint32_t dst_bd_offset, struct ezdp_ext_addr __cmem *src_ptr, uint32_t size, uint32_t flags)
- *Copy data from an extended address to a frame buffer.* static __always_inline void ezdp_copy_frame_data_from_ext_addr_async (struct ezdp_buffer_desc __cmem *dst_bd_ptr, uint32_t dst_bd_offset, struct ezdp_ext_addr __cmem *src_ptr, uint32_t size, uint32_t flags)
- *Non blocking version of ezdp_copy_frame_data_from_ext_addr_async().* static __always_inline uint32_t ezdp_load_frame_data (void __cmem *dst_ptr, struct ezdp_buffer_desc __cmem *src_bd_ptr, uint32_t src_bd_offset, uint32_t size, uint32_t flags)

- *Copy data from a frame buffer to CMEM.* static __always_inline void ezdp_load_frame_data_async (void __cmem *dst_ptr, struct ezdp_buffer_desc __cmem *src_bd_ptr, uint32_t src_bd_offset, uint32_t size, uint32_t flags)
- *Non blocking version of ezdp_load_frame_data().* static __always_inline uint32_t ezdp_store_frame_data (struct ezdp_buffer_desc __cmem *dst_bd_ptr, uint32_t dst_bd_offset, void __cmem *src_ptr, uint32_t size, uint32_t flags)
- *Copy data from CMEM to a frame buffer.* static __always_inline void ezdp_store_frame_data_async (struct ezdp_buffer_desc __cmem *dst_bd_ptr, uint32_t dst_bd_offset, void __cmem *src_ptr, uint32_t size, uint32_t flags)
- *Non blocking version of ezdp_store_frame_data().* static __always_inline void ezdp_copy_frame_lbd (struct ezdp_buffer_desc __cmem *dst_bd_ptr, uint32_t dst_bd_offset, struct ezdp_buffer_desc __cmem *src_bd_ptr, uint32_t src_bd_offset, uint32_t size, uint32_t flags)
- *Copy LBD data between two frame buffers.* static __always_inline void ezdp_copy_frame_lbd_async (struct ezdp_buffer_desc __cmem *dst_bd_ptr, uint32_t dst_bd_offset, struct ezdp_buffer_desc __cmem *src_bd_ptr, uint32_t src_bd_offset, uint32_t size, uint32_t flags)
- *Non blocking version of ezdp_copy_frame_lbd().* static __always_inline void ezdp_clone_frame_lbd (struct ezdp_buffer_desc __cmem *dst_bd_ptr, struct ezdp_buffer_desc __cmem *src_bd_ptr, uint32_t bd_offset, uint32_t size, uint32_t flags)
- *Copy LBD data between two frame buffers, with the same source and destination offset (optimized).* static __always_inline void ezdp_clone_frame_lbd_async (struct ezdp_buffer_desc __cmem *dst_bd_ptr, struct ezdp_buffer_desc __cmem *src_bd_ptr, uint32_t bd_offset, uint32_t size, uint32_t flags)
- *Non blocking version of ezdp_clone_frame_lbd().* static __always_inline void ezdp_copy_frame_lbd_to_ext_addr (struct ezdp_ext_addr __cmem *dst_ptr, struct ezdp_buffer_desc __cmem *src_bd_ptr, uint32_t src_bd_offset, uint32_t size, uint32_t flags)
- *Copy LBD data from a frame buffer to an extended address.* static __always_inline void ezdp_copy_frame_lbd_to_ext_addr_async (struct ezdp_ext_addr __cmem *dst_ptr, struct ezdp_buffer_desc __cmem *src_bd_ptr, uint32_t src_bd_offset, uint32_t size, uint32_t flags)
- *Non blocking version of ezdp_copy_frame_lbd_to_ext_addr().* static __always_inline void ezdp_copy_frame_lbd_from_ext_addr (struct ezdp_buffer_desc __cmem *dst_bd_ptr, uint32_t dst_bd_offset, struct ezdp_ext_addr __cmem *src_ptr, uint32_t size, uint32_t flags)
- *Copy LBD data from an extended address to a frame buffer.* static __always_inline void ezdp_copy_frame_lbd_from_ext_addr_async (struct ezdp_buffer_desc __cmem *dst_bd_ptr, uint32_t dst_bd_offset, struct ezdp_ext_addr __cmem *src_ptr, uint32_t size, uint32_t flags)
- *Non blocking version of _ezdp_copy_frame_lbd_from_ext_addr().* static __always_inline void ezdp_load_frame_lbd (void __cmem *dst_ptr, struct ezdp_buffer_desc __cmem *src_bd_ptr, uint32_t src_bd_offset, uint32_t size, uint32_t flags)
- *Copy LBD data from a frame buffer to CMEM.* static __always_inline void ezdp_load_frame_lbd_async (void __cmem *dst_ptr, struct ezdp_buffer_desc __cmem *src_bd_ptr, uint32_t src_bd_offset, uint32_t size, uint32_t flags)
- *Non blocking version of ezdp_load_frame_lbd().* static __always_inline void ezdp_store_frame_lbd (struct ezdp_buffer_desc __cmem *dst_bd_ptr, uint32_t dst_bd_offset, void __cmem *src_ptr, uint32_t size, uint32_t flags)
- *Copy LBD data from CMEM to a frame buffer.* static __always_inline void ezdp_store_frame_lbd_async (struct ezdp_buffer_desc __cmem *dst_bd_ptr, uint32_t dst_bd_offset, void __cmem *src_ptr, uint32_t size, uint32_t flags)
- *Non blocking version of ezdp_store_frame_lbd.* static __always_inline ezdp_buffer_desc_t ezdp_alloc_mc_buf (enum ezdp_buffer_mem_type pool_type, uint32_t buf_budget_id, uint16_t ref_counter)
- *Allocate a single frame buffer and set its multicast reference counter.* static __always_inline void ezdp_free_mc_buf (uint32_t buf_budget_id, ezdp_buffer_desc_t bd)
- *Free a multicast frame buffer.* static __always_inline void ezdp_write_mc_buf_counter (ezdp_buffer_desc_t bd, uint16_t value)
- *Set a frame buffer's multicast reference counter.* static __always_inline void ezdp_write_mc_buf_counter_async (ezdp_buffer_desc_t bd, uint16_t value)
- *Non blocking version of ezdp_write_mc_buf_counter().* static __always_inline uint32_t ezdp_read_mc_buf_counter (ezdp_buffer_desc_t bd)
- *Get a frame buffer's multicast reference counter.* static __always_inline uint32_t ezdp_atomic_read_and_inc_mc_buf_counter (ezdp_buffer_desc_t bd)
- *Atomically read and increment a frame buffer's multicast reference counter.* static __always_inline uint32_t ezdp_atomic_read_and_dec_mc_buf_counter (ezdp_buffer_desc_t bd)

- *Atomically read and conditionally decrement a frame buffer's multicast reference counter.* static __always_inline uint32_t ezdp_calc_frame_data_checksum (struct ezdp_buffer_desc __cmem *bd_ptr, uint32_t bd_offset, uint32_t size, uint32_t flags)
- *Calculate checksum of frame data buffer.* static __always_inline uint32_t ezdp_buf_data_len (uint32_t header_offset, uint32_t free_bytes, uint32_t max_length)
- *Calculate the length of the buffer based on header offset and free bytes.* static __always_inline uint32_t ezdp_lbd_len (uint32_t data_buf_count)
- *Calculate LBD buffer length according to BD count.* static __always_inline uint32_t ezdp_calc_header_offset (uint32_t min_header_offset, bool prefer_128B_offset, uint32_t buffer_length)
- *Calculate optimized frame header offset.* static __always_inline void ezdp_inc_tm_imem_buf_ctr (uint32_t value)
- *Increment TM IMEM buffer counter.* static __always_inline void ezdp_inc_tm_imem_buf_ctr_async (uint32_t value)
- *Non blocking version of _ezdp_inc_tm_imem_buf_ctr.* static __always_inline void ezdp_dec_tm_imem_buf_ctr (uint32_t value)
- *Decrement TM IMEM buffer counter.* static __always_inline void ezdp_dec_tm_imem_buf_ctr_async (uint32_t value)
- *Non blocking version of _ezdp_dec_tm_imem_buf_ctr.* static __always_inline uint32_t ezdp_read_tm_imem_buf_ctr (void)
- *Number of IMEM buffers used by frames currently being processed by TM.* static __always_inline int32_t ezdp_get_first_buf (struct ezdp_frame_desc *fd, struct ezdp_buffer_desc *buf_desc, struct ezdp_linked_buffers_desc_line __cmem *lbd_line, ezdp_frame_buf_iterator_state_t *iter_st)
- *Gets first buffer from frame.* static __always_inline int32_t ezdp_get_next_buf (struct ezdp_frame_desc *fd, struct ezdp_buffer_desc *buf_desc, struct ezdp_linked_buffers_desc_line __cmem *lbd_line, ezdp_frame_buf_iterator_state_t *iter_st)
- *Get next buffer from frame.* static __always_inline bool ezdp_init_frame (struct ezdp_frame_desc __cmem *frame_desc, enum ezdp_frame_type frame_type, ezdp_buffer_desc_t buf_desc, uint32_t buf_len, struct ezdp_linked_buffers_desc_line __cmem *lbd_line, uint32_t header_offset, uint32_t budget_id, ezdp_frame_buf_iterator_state_t *iter_st)
- *Create a new frame by setting its frame descriptor params and init frame iterator.* static __always_inline void ezdp_append_buf (struct ezdp_frame_desc __cmem *frame_desc, ezdp_buffer_desc_t buf_desc, uint32_t buf_len, struct ezdp_linked_buffers_desc_line __cmem *lbd_line, ezdp_frame_buf_iterator_state_t *iter_st)
- *Add newlly allocated (by user) buffer to frame pointed by iterator.* static __always_inline void ezdp_sync_frame (struct ezdp_frame_desc __cmem *frame_desc, struct ezdp_linked_buffers_desc_line __cmem *lbd_line, ezdp_frame_buf_iterator_state_t *iter_st, bool force)

Store last LBD line to memory, if required.

Function Documentation

static __always_inline ezdp_buffer_desc_t ezdp_alloc_buf (enum ezdp_buffer_mem_type *pool_type*, uint32_t *buf_budget_id*) [static]

Allocate a single frame buffer.

The returned buffer descriptor's type is initially set to be a data buffer. This can later be changed by the caller to match the actual intended use.

Parameters:

[in] *pool_type* - pool type to allocate buffer from
[in] *buf_budget_id* - budget id for accounting

Returns:

ezdp_buffer_desc_t - use ezdp_buf_alloc_failed API to check if allocation success or fail

static __always_inline void ezdp_free_buf (uint32_t *buf_budget_id*, ezdp_buffer_desc_t *bd_raw_data*) [static]

Free a single frame buffer.

Parameters:

[in] *buf_budget_id* - budget id to charge
[in] *bd_raw_data* - BD raw data

Returns:

none

static __always_inline void ezdp_free_buf_async (uint32_t *buf_budget_id*, ezdp_buffer_desc_t *bd_raw_data*) [static]

Non blocking version of ezdp_free_buf().

Parameters:

[in] *buf_budget_id* - budget id to charge
[in] *bd_raw_data* - BD raw data

Returns:

none

static __always_inline ezdp_buffer_desc_t ezdp_alloc_multi_buf (enum ezdp_buffer_mem_type *pool_type*, uint32_t *buf_budget_id*, uint32_t *num_of_bufs*, struct ezdp_buffer_desc __cmem * *bds_ptr*) [static]

Allocate multiple frame buffers.

Allocate up to 8 frame buffers from the same pool. The allocated BDs are written to the CMEM. In addition, the first allocated BD is returned. The operation either succeeds to allocate all requested resources or fails without allocating any resources.

Parameters:

[in] *pool_type* - pool type to allocate buffer from (IMEM or EMEM)
[in] *num_of_bufs* - number of buffers to allocate (1-8)
[in] *buf_budget_id* - budget id for accounting
[out] *bds_ptr* - pointer to CMEM to write response to

Returns:

First allocated ezdp_buffer_desc - use ezdp_buf_alloc_failed API to check if allocation success or fail

static __always_inline void ezdp_alloc_multi_buf_async (enum ezdp_buffer_mem_type *pool_type*, uint32_t *buf_budget_id*, uint32_t *num_of_bufs*, struct ezdp_buffer_desc __cmem * *bds_ptr*) [static]

Non blocking version of ezdp_alloc_multi_buf().

Parameters:

[in] *pool_type* - pool type to allocate buffer from (IMEM or EMEM pool)
[in] *num_of_bufs* - number of buffer to allocate (1-8)
[in] *buf_budget_id* - budget id for accounting
[out] *bds_ptr* - pointer to CMEM to write response to

Note:

Call ezdp_sync() to wait for the operation to complete, confirming that the allocated BDs were written to CMEM.

Returns:

none use ezdp_buf_alloc_failed API on first allocated buffer to check if allocation success or fail

static __always_inline bool ezdp_buf_alloc_failed (ezdp_buffer_desc_t *ret*) `[static]`

Check if allocation of the buffer failed.

Parameters:

[in] *ret* - return value from buffer allocation API

Returns:

bool - true if allocation failed

static __always_inline void ezdp_free_multi_buf (uint32_t *buf_budget_id*, uint32_t *num_of_bufs*, struct ezdp_buffer_desc __cmem * *bds_ptr*) `[static]`

Free multiple frame buffers.

Free up to 8 frame buffers, not necessarily from the same pool. The BDs to free are passed in CMEM. NULL BDs are ignored.

Parameters:

[in] *num_of_bufs* - number of buffers to free (1-8)
[in] *buf_budget_id* - budget id to charge
[in] *bds_ptr* - pointer to BDs in CMEM to recycle

Returns:

none

static __always_inline void ezdp_free_multi_buf_async (uint32_t *buf_budget_id*, uint32_t *num_of_bufs*, struct ezdp_buffer_desc __cmem * *bds_ptr*) `[static]`

Non blocking version of ezdp_free_multi_buf().

Parameters:

[in] *num_of_bufs* - number of buffers to recycle (1-8)
[in] *buf_budget_id* - budget id to charge
[in] *bds_ptr* - pointer to BDs in CMEM to recycle

Note:

Call ezdp_sync() to wait for the operation to complete, confirming that the BDs were read from CMEM and the request was sent to the BMU.

Returns:

none

static __always_inline uint32_t ezdp_read_free_buf (enum ezdp_buffer_mem_type *pool_type*) `[static]`

The number of buffers available to be obtained.

Parameters:

[in] *pool_type* - buffer pool type

Returns:

uint32_t - number of available/free frame buffers

static __always_inline void ezdp_rebudget_buf (enum ezdp_buffer_mem_type *pool_type*, uint32_t *free_buf_budget_id*, uint32_t *free_num_of_bufs*, uint32_t *alloc_buf_budget_id*, uint32_t *alloc_num_of_bufs*) `[static]`

Update the budget to which buffers are credited.

Adds free_num_of_bufs credits back into the budget selected by free_buf_budget_id and subtracts alloc_num_of_bufs credits from the budget selected by alloc_buf_budget_id. The budget is maintained separately for IMEM and EMEM buffers.

Parameters:

[in] *pool_type* - pool type (IMEM or EMEM pool)
[in] *free_buf_budget_id* - budget ID to reimburse
[in] *free_num_of_bufs* - number of buffers to reimburse. Decrement operation (can be zero)
[in] *alloc_buf_budget_id* - budget ID to charge
[in] *alloc_num_of_bufs* - number of buffers to charge. Increment operation (can be zero)

Returns:

none

static __always_inline void ezdp_rebudget_buf_async (enum ezdp_buffer_mem_type *pool_type*, uint32_t *free_buf_budget_id*, uint32_t *free_num_of_bufs*, uint32_t *alloc_buf_budget_id*, uint32_t *alloc_num_of_bufs*) `[static]`

Non blocking version of ezdp_rebudget_buf().

Parameters:

[in] *pool_type* - pool type (IMEM or EMEM pool)
[in] *free_buf_budget_id* - budget ID to reimburse
[in] *free_num_of_bufs* - number of buffers to reimburse. Decrement operation (can be zero)
[in] *alloc_buf_budget_id* - budget ID to charge
[in] *alloc_num_of_bufs* - number of buffers to charge. Increment operation (can be zero)

Returns:

none

static __always_inline uint32_t ezdp_copy_frame_data (struct ezdp_buffer_desc __cmem * *dst_bd_ptr*, uint32_t *dst_bd_offset*, struct ezdp_buffer_desc __cmem * *src_bd_ptr*, uint32_t *src_bd_offset*, uint32_t *size*, uint32_t *flags*) `[static]`

Copy data between two frame buffers.

Parameters:

[in] *dst_bd_ptr* - pointer to BD (in CMEM with 4 byte alignment) describing the destination buffer
[in] *dst_bd_offset* - offset in the destination buffer to copy data to
[in] *src_bd_ptr* - pointer to BD (in CMEM with 4 byte alignment) describing the source buffer
[in] *src_bd_offset* - offset in the source buffer to copy data from
[in] *size* - number of bytes to copy
[in] *flags* - execution flags. Bitwise OR of zero or more flags out of the following:
EZDP_MEMORY_FLAG_OVERWRITE - override ECC (do not merge)

Returns:

uin32_t - 16 bit checksum value

static __always_inline void ezdp_copy_frame_data_async (struct ezdp_buffer_desc __cmem * *dst_bd_ptr*, uint32_t *dst_bd_offset*, struct ezdp_buffer_desc __cmem * *src_bd_ptr*, uint32_t *src_bd_offset*, uint32_t *size*, uint32_t *flags*) `[static]`

Non blocking version of ezdp_copy_frame_data().

Parameters:

[in] *dst_bd_ptr* - pointer to BD (in CMEM with 4 byte alignment) describing the destination buffer
[in] *dst_bd_offset* - offset in the destination buffer to copy data to
[in] *src_bd_ptr* - pointer to BD (in CMEM with 4 byte alignment) describing the source buffer
[in] *src_bd_offset* - offset in the source buffer to copy data from
[in] *size* - number of bytes to copy
[in] *flags* - execution flags. Bitwise OR of zero or more flags out of the following:
EZDP_MEMORY_FLAG_OVERWRITE - override ECC (do not merge)

Note:

Call ezdp_sync() to wait for the operation to complete, confirming that the data was copied to the destination.

Returns:

none

static __always_inline uint32_t ezdp_clone_frame_data (struct ezdp_buffer_desc __cmem * *dst_bd_ptr*, struct ezdp_buffer_desc __cmem * *src_bd_ptr*, uint32_t *bd_offset*, uint32_t *size*, uint32_t *flags*) `[static]`

Copy data between two frame buffers, with the same source and destination offset (optimized).

Parameters:

[in] *dst_bd_ptr* - pointer to BD (in CMEM with 4 byte alignment) describing the destination buffer
[in] *src_bd_ptr* - pointer to BD (in CMEM with 4 byte alignment) describing the source buffer
[in] *bd_offset* - offset in both the source and destination buffers to copy from/to
[in] *size* - number of bytes to copy
[in] *flags* - execution flags. Bitwise OR of zero or more flags out of the following:
EZDP_MEMORY_FLAG_OVERWRITE - override ECC (do not merge)

Returns:

uin32_t - 16 bit checksum value

static __always_inline void ezdp_clone_frame_data_async (struct ezdp_buffer_desc __cmem * *dst_bd_ptr*, struct ezdp_buffer_desc __cmem * *src_bd_ptr*, uint32_t *bd_offset*, uint32_t *size*, uint32_t *flags*) `[static]`

Non blocking version of ezdp_clone_frame_data().

Parameters:

[in] *dst_bd_ptr* - pointer to BD (in CMEM with 4 byte alignment) describing the destination buffer
[in] *src_bd_ptr* - pointer to BD (in CMEM with 4 byte alignment) describing the source buffer
[in] *bd_offset* - offset in both the source and destination buffers to copy from/to
[in] *size* - number of bytes to copy
[in] *flags* - execution flags. Bitwise OR of zero or more flags out of the following:
EZDP_MEMORY_FLAG_OVERWRITE - override ECC (do not merge)

Note:

Call ezdp_sync() to wait for the operation to complete, confirming that the data was copied to the destination.

Returns:

none

static __always_inline uint32_t ezdp_copy_frame_data_to_ext_addr (struct ezdp_ext_addr __cmem * *dst_ptr*, struct ezdp_buffer_desc __cmem * *src_bd_ptr*, uint32_t *src_bd_offset*, uint32_t *size*, uint32_t *flags*) [static]

Copy data from a frame buffer to an extended address.

Parameters:

[in] *dst_ptr* - pointer to destination extended address (in CMEM in 8 byte alignment)
[in] *src_bd_ptr* - pointer to BD (in CMEM in 4 byte alignment) describing the source buffer
[in] *src_bd_offset* - offset in the source buffer to copy data from
[in] *size* - number of bytes to copy
[in] *flags* - execution flags. Bitwise OR of zero or more flags out of the following:
EZDP_MEMORY_FLAG_OVERWRITE - override ECC (do not merge)

Note:

returned 16 bit checksum is applicable only when memory type is external or memory type is internal and msid > 5

Returns:

uin32_t - 16 bit checksum value

static __always_inline void ezdp_copy_frame_data_to_ext_addr_async (struct ezdp_ext_addr __cmem * *dst_ptr*, struct ezdp_buffer_desc __cmem * *src_bd_ptr*, uint32_t *src_bd_offset*, uint32_t *size*, uint32_t *flags*) [static]

Non blocking version of ezdp_copy_frame_data_to_ext_addr().

Parameters:

[in] *dst_ptr* - pointer to destination extended address (in CMEM in 8 byte alignment)
[in] *src_bd_ptr* - pointer to BD (in CMEM in 4 byte alignment) describing the source buffer
[in] *src_bd_offset* - offset in the source buffer to copy data from
[in] *size* - number of bytes to copy
[in] *flags* - execution flags. Bitwise OR of zero or more flags out of the following:
EZDP_MEMORY_FLAG_OVERWRITE - override ECC (do not merge)

Note:

Call ezdp_sync() to wait for the operation to complete, confirming that the data was copied to the destination.

Returns:

none

static __always_inline uint32_t ezdp_copy_frame_data_from_ext_addr (struct ezdp_buffer_desc __cmem * *dst_bd_ptr*, uint32_t *dst_bd_offset*, struct ezdp_ext_addr __cmem * *src_ptr*, uint32_t *size*, uint32_t *flags*) [static]

Copy data from an extended address to a frame buffer.

Parameters:

[in] *dst_bd_ptr* - pointer to BD (in CMEM in 4 byte alignment) describing the destination buffer
[in] *dst_bd_offset* - offset in the destination buffer to copy data to
[in] *src_ptr* - pointer to source extended address (in CMEM in 8 byte alignment)
[in] *size* - number of bytes to copy

[in] *flags* - execution flags. Bitwise OR of zero or more flags out of the following:
EZDP_MEMORY_FLAG_OVERWRITE - override ECC (do not merge)

Returns:

uint32_t - 16 bit checksum value

static __always_inline void ezdp_copy_frame_data_from_ext_addr_async (struct ezdp_buffer_desc __cmem * *dst_bd_ptr*, uint32_t *dst_bd_offset*, struct ezdp_ext_addr __cmem * *src_ptr*, uint32_t *size*, uint32_t *flags*) `[static]`

Non blocking version of ezdp_copy_frame_data_from_ext_addr_async().

Parameters:

[in] *dst_bd_ptr* - pointer to BD (in CMEM in 4 byte alignment) describing the destination buffer
[in] *dst_bd_offset* - offset in the destination buffer to copy data to
[in] *src_ptr* - pointer to source extended address (in CMEM in 8 byte alignment)
[in] *size* - number of bytes to copy
[in] *flags* - execution flags. Bitwise OR of zero or more flags out of the following:
EZDP_MEMORY_FLAG_OVERWRITE - override ECC (do not merge)

Note:

Call ezdp_sync() to wait for the operation to complete, confirming that the data was copied to the destination.

Returns:

none

static __always_inline uint32_t ezdp_load_frame_data (void __cmem * *dst_ptr*, struct ezdp_buffer_desc __cmem * *src_bd_ptr*, uint32_t *src_bd_offset*, uint32_t *size*, uint32_t *flags*) `[static]`

Copy data from a frame buffer to CMEM.

Parameters:

[out] *dst_ptr* - pointer in CMEM to copy data to
[in] *src_bd_ptr* - pointer to BD (in CMEM in 4 byte alignment) describing the source buffer
[in] *src_bd_offset* - offset in the source buffer to copy data from
[in] *size* - number of bytes to copy
[in] *flags* - execution flags. Bitwise OR of zero or more flags out of the following: TODO

Returns:

uint32_t - first 4 bytes of the copied data, which are zero padded if the requested data length is smaller than 4

static __always_inline void ezdp_load_frame_data_async (void __cmem * *dst_ptr*, struct ezdp_buffer_desc __cmem * *src_bd_ptr*, uint32_t *src_bd_offset*, uint32_t *size*, uint32_t *flags*) `[static]`

Non blocking version of ezdp_load_frame_data().

Parameters:

[out] *dst_ptr* - pointer in CMEM to copy data to
[in] *src_bd_ptr* - pointer to BD (in CMEM in 4 byte alignment) describing the source buffer
[in] *src_bd_offset* - offset in the source buffer to copy data from
[in] *size* - number of bytes to copy
[in] *flags* - execution flags. Bitwise OR of zero or more flags out of the following: TODO

Note:

Call ezdp_sync() to wait for the operation to complete, confirming that the data was copied to the destination.

Returns:

none

static __always_inline uint32_t ezdp_store_frame_data (struct ezdp_buffer_desc __cmem * *dst_bd_ptr*, uint32_t *dst_bd_offset*, void __cmem * *src_ptr*, uint32_t *size*, uint32_t *flags*) `[static]`

Copy data from CMEM to a frame buffer.

Parameters:

[in] *dst_bd_ptr* - pointer to BD (in CMEM in 4 byte alignment) describing the destination buffer
[in] *dst_bd_offset* - offset in the destination buffer to copy data to
[in] *src_ptr* - pointer in CMEM to copy data from
[in] *size* - number of bytes to copy
[in] *flags* - execution flags. Bitwise OR of zero or more flags out of the following:
EZDP_MEMORY_FLAG_OVERWRITE - override ECC (do not merge)

Returns:

uin32_t - 16 bit checksum value

static __always_inline void ezdp_store_frame_data_async (struct ezdp_buffer_desc __cmem * *dst_bd_ptr*, uint32_t *dst_bd_offset*, void __cmem * *src_ptr*, uint32_t *size*, uint32_t *flags*) `[static]`

Non blocking version of ezdp_store_frame_data().

Parameters:

[in] *dst_bd_ptr* - pointer to BD (in CMEM in 4 byte alignment) describing the destination buffer
[in] *dst_bd_offset* - offset in the destination buffer to copy data to
[in] *src_ptr* - pointer in CMEM to copy data from
[in] *size* - number of bytes to copy
[in] *flags* - execution flags. Bitwise OR of zero or more flags out of the following:
EZDP_MEMORY_FLAG_OVERWRITE - override ECC (do not merge)

Note:

Call ezdp_sync() to wait for the operation to complete, confirming that the data was copied to the destination.

Returns:

none

static __always_inline void ezdp_copy_frame_lbd (struct ezdp_buffer_desc __cmem * *dst_bd_ptr*, uint32_t *dst_bd_offset*, struct ezdp_buffer_desc __cmem * *src_bd_ptr*, uint32_t *src_bd_offset*, uint32_t *size*, uint32_t *flags*) `[static]`

Copy LBD data between two frame buffers.

Parameters:

[in] *dst_bd_ptr* - pointer to BD (in CMEM in 4 byte alignment) describing the destination buffer
[in] *dst_bd_offset* - offset in the destination buffer to copy data to
[in] *src_bd_ptr* - pointer to BD (in CMEM in 4 byte alignment) describing the source buffer
[in] *src_bd_offset* - offset in the source buffer to copy data from

[in] *size* - number of bytes to copy
[in] *flags* - execution flags. Bitwise OR of zero or more flags out of the following: TODO

Returns:
none

static __always_inline void ezdp_copy_frame_lbd_async (struct ezdp_buffer_desc __cmem * *dst_bd_ptr*, uint32_t *dst_bd_offset*, struct ezdp_buffer_desc __cmem * *src_bd_ptr*, uint32_t *src_bd_offset*, uint32_t *size*, uint32_t *flags*) `[static]`

Non blocking version of ezdp_copy_frame_lbd().

Parameters:
[in] *dst_bd_ptr* - pointer to BD (in CMEM in 4 byte alignment) describing the destination buffer
[in] *dst_bd_offset* - offset in the destination buffer to copy data to
[in] *src_bd_ptr* - pointer to BD (in CMEM in 4 byte alignment) describing the source buffer
[in] *src_bd_offset* - offset in the source buffer to copy data from
[in] *size* - number of bytes to copy
[in] *flags* - execution flags. Bitwise OR of zero or more flags out of the following: TODO

Note:
Call ezdp_sync() to wait for the operation to complete, confirming that the data was copied to the destination.

Returns:
none

static __always_inline void ezdp_clone_frame_lbd (struct ezdp_buffer_desc __cmem * *dst_bd_ptr*, struct ezdp_buffer_desc __cmem * *src_bd_ptr*, uint32_t *bd_offset*, uint32_t *size*, uint32_t *flags*) `[static]`

Copy LBD data between two frame buffers, with the same source and destination offset (optimized).

Parameters:
[in] *dst_bd_ptr* - pointer to BD (in CMEM in 4 byte alignment) describing the destination buffer
[in] *src_bd_ptr* - pointer to BD (in CMEM in 4 byte alignment) describing the source buffer
[in] *bd_offset* - offset in both the source and destination buffers to copy from/to
[in] *size* - number of bytes to copy
[in] *flags* - execution flags. Bitwise OR of zero or more flags out of the following: TODO

Returns:
none

static __always_inline void ezdp_clone_frame_lbd_async (struct ezdp_buffer_desc __cmem * *dst_bd_ptr*, struct ezdp_buffer_desc __cmem * *src_bd_ptr*, uint32_t *bd_offset*, uint32_t *size*, uint32_t *flags*) `[static]`

Non blocking version of ezdp_clone_frame_lbd().

Parameters:
[in] *dst_bd_ptr* - pointer to BD (in CMEM in 4 byte alignment) describing the destination buffer
[in] *src_bd_ptr* - pointer to BD (in CMEM in 4 byte alignment) describing the source buffer
[in] *bd_offset* - offset in both the source and destination buffers to copy from/to
[in] *size* - number of bytes to copy
[in] *flags* - execution flags. Bitwise OR of zero or more flags out of the following: TODO

Note:

Call ezdp_sync() to wait for the operation to complete, confirming that the data was copied to the destination.

Returns:

none

static __always_inline void ezdp_copy_frame_lbd_to_ext_addr (struct ezdp_ext_addr __cmem * *dst_ptr*, struct ezdp_buffer_desc __cmem * *src_bd_ptr*, uint32_t *src_bd_offset*, uint32_t *size*, uint32_t *flags*) `[static]`

Copy LBD data from a frame buffer to an extended address.

Parameters:

[in] *dst_ptr* - pointer to destination extended address (in CMEM in 8 byte alignment)
[in] *src_bd_ptr* - pointer to BD (in CMEM in 4 byte alignment) describing the source buffer
[in] *src_bd_offset* - offset in the source buffer to copy data from
[in] *size* - number of bytes to copy
[in] *flags* - execution flags. Bitwise OR of zero or more flags out of the following: TODO

Returns:

none

static __always_inline void ezdp_copy_frame_lbd_to_ext_addr_async (struct ezdp_ext_addr __cmem * *dst_ptr*, struct ezdp_buffer_desc __cmem * *src_bd_ptr*, uint32_t *src_bd_offset*, uint32_t *size*, uint32_t *flags*) `[static]`

Non blocking version of ezdp_copy_frame_lbd_to_ext_addr().

Parameters:

[in] *dst_ptr* - pointer to destination extended address (in CMEM in 8 byte alignment)
[in] *src_bd_ptr* - pointer to BD (in CMEM in 4 byte alignment) describing the source buffer
[in] *src_bd_offset* - offset in the source buffer to copy data from
[in] *size* - number of bytes to copy
[in] *flags* - execution flags. Bitwise OR of zero or more flags out of the following: TODO

Note:

Call ezdp_sync() to wait for the operation to complete, confirming that the data was copied to the destination.

Returns:

none

static __always_inline void ezdp_copy_frame_lbd_from_ext_addr (struct ezdp_buffer_desc __cmem * *dst_bd_ptr*, uint32_t *dst_bd_offset*, struct ezdp_ext_addr __cmem * *src_ptr*, uint32_t *size*, uint32_t *flags*) `[static]`

Copy LBD data from an extended address to a frame buffer.

Parameters:

[in] *dst_bd_ptr* - pointer to BD (in CMEM in 4 byte alignment) describing the destination buffer
[in] *dst_bd_offset* - offset in the destination buffer to copy data to
[in] *src_ptr* - pointer to source extended address (in CMEM in 8 byte alignment)
[in] *size* - number of bytes to copy
[in] *flags* - execution flags. Bitwise OR of zero or more flags out of the following: TODO

Returns:

none

static __always_inline void ezdp_copy_frame_lbd_from_ext_addr_async (struct ezdp_buffer_desc __cmem * *dst_bd_ptr*, uint32_t *dst_bd_offset*, struct ezdp_ext_addr __cmem * *src_ptr*, uint32_t *size*, uint32_t *flags*) `[static]`

Non blocking version of _ezdp_copy_frame_lbd_from_ext_addr().

Parameters:

[in] *dst_bd_ptr* - pointer to BD (in CMEM in 4 byte alignment) describing the destination buffer
[in] *dst_bd_offset* - offset in the destination buffer to copy data to
[in] *src_ptr* - pointer to source extended address (in CMEM in 8 byte alignment)
[in] *size* - number of bytes to copy
[in] *flags* - execution flags. Bitwise OR of zero or more flags out of the following: TODO

Note:

Call ezdp_sync() to wait for the operation to complete, confirming that the data was copied to the destination.

Returns:

none

static __always_inline void ezdp_load_frame_lbd (void __cmem * *dst_ptr*, struct ezdp_buffer_desc __cmem * *src_bd_ptr*, uint32_t *src_bd_offset*, uint32_t *size*, uint32_t *flags*) `[static]`

Copy LBD data from a frame buffer to CMEM.

Parameters:

[out] *dst_ptr* - pointer in CMEM to copy data to
[in] *src_bd_ptr* - pointer to BD (in CMEM in 4 byte alignment) describing the source buffer
[in] *src_bd_offset* - offset in the source buffer to copy data from
[in] *size* - number of bytes to copy
[in] *flags* - execution flags. Bitwise OR of zero or more flags out of the following: TODO

Returns:

none

static __always_inline void ezdp_load_frame_lbd_async (void __cmem * *dst_ptr*, struct ezdp_buffer_desc __cmem * *src_bd_ptr*, uint32_t *src_bd_offset*, uint32_t *size*, uint32_t *flags*) `[static]`

Non blocking version of ezdp_load_frame_lbd().

Parameters:

[out] *dst_ptr* - pointer in CMEM to copy data to
[in] *src_bd_ptr* - pointer to BD (in CMEM in 4 byte alignment) describing the source buffer
[in] *src_bd_offset* - offset in the source buffer to copy data from
[in] *size* - number of bytes to copy
[in] *flags* - execution flags. Bitwise OR of zero or more flags out of the following: TODO

Note:

Call ezdp_sync() to wait for the operation to complete, confirming that the data was copied to the destination.

Returns:
none

static __always_inline void ezdp_store_frame_lbd (struct ezdp_buffer_desc __cmem * *dst_bd_ptr*, uint32_t *dst_bd_offset*, void __cmem * *src_ptr*, uint32_t *size*, uint32_t *flags*) `[static]`

Copy LBD data from CMEM to a frame buffer.

Parameters:
[in] *dst_bd_ptr* - pointer to BD (in CMEM in 4 byte alignment) describing the destination buffer
[in] *dst_bd_offset* - offset in the destination buffer to copy data to
[in] *src_ptr* - pointer in CMEM to copy data from
[in] *size* - number of bytes to copy
[in] *flags* - execution flags. Bitwise OR of zero or more flags out of the following: TODO

Returns:
none

static __always_inline void ezdp_store_frame_lbd_async (struct ezdp_buffer_desc __cmem * *dst_bd_ptr*, uint32_t *dst_bd_offset*, void __cmem * *src_ptr*, uint32_t *size*, uint32_t *flags*) `[static]`

Non blocking version of ezdp_store_frame_lbd.

Parameters:
[in] *dst_bd_ptr* - pointer to BD (in CMEM in 4 byte alignment) describing the destination buffer
[in] *dst_bd_offset* - offset in the destination buffer to copy data to
[in] *src_ptr* - pointer in CMEM to copy data from
[in] *size* - number of bytes to copy
[in] *flags* - execution flags. Bitwise OR of zero or more flags out of the following: TODO

Note:
Call ezdp_sync() to wait for the operation to complete, confirming that the data was copied to the destination.

Returns:
none

static __always_inline ezdp_buffer_desc_t ezdp_alloc_mc_buf (enum ezdp_buffer_mem_type *pool_type*, uint32_t *buf_budget_id*, uint16_t *ref_counter*) `[static]`

Allocate a single frame buffer and set its multicast reference counter.

Used for allocation of frame buffers for multicast frames. The buffer is allocated and its multicast reference counter is set to required ref_counter value

Parameters:
[in] *pool_type* - pool type to allocate buffer from
[in] *buf_budget_id* - budget id for accounting
[in] *ref_counter* - reference counter to set for this buffer

Returns:
ezdp_buffer_desc_t - use ezdp_buf_alloc_failed API to check if allocation success or fail

static __always_inline void ezdp_free_mc_buf (uint32_t *buf_budget_id*, ezdp_buffer_desc_t *bd*) `[static]`

Free a multicast frame buffer.

Used for recycling of frame buffers belonging to multicast frames. The buffer's multicast reference counter is decremented and the buffer is recycled if this was the last reference to the buffer.

Parameters:

[in] *buf_budget_id* - budget id to charge
[in] *bd* - BD raw data

Returns:

none

static __always_inline void ezdp_write_mc_buf_counter (ezdp_buffer_desc_t *bd*, uint16_t *value*) [static]

Set a frame buffer's multicast reference counter.

Parameters:

[in] *bd* - BD describing the buffer
[in] *value* - reference count value to set

Returns:

void

static __always_inline void ezdp_write_mc_buf_counter_async (ezdp_buffer_desc_t *bd*, uint16_t *value*) [static]

Non blocking version of ezdp_write_mc_buf_counter().

Parameters:

[in] *bd* - BD describing the buffer
[in] *value* - reference count value to set

Note:

Call ezdp_sync() to wait for the operation to complete, confirming that the value was set.

Returns:

void

static __always_inline uint32_t ezdp_read_mc_buf_counter (ezdp_buffer_desc_t *bd*) [static]

Get a frame buffer's multicast reference counter.

Parameters:

[in] *bd* - BD describing the buffer

Returns:

uint32_t - the 16 LSB are the current counter value

static __always_inline uint32_t ezdp_atomic_read_and_inc_mc_buf_counter (ezdp_buffer_desc_t *bd*) [static]

Atomically read and increment a frame buffer's multicast reference counter.

Parameters:

[in] *bd* - BD describing the buffer

Returns:

uint32_t - the 16 LSB are the value of the counter before the increment

static __always_inline uint32_t ezdp_atomic_read_and_dec_mc_buf_counter (ezdp_buffer_desc_t *bd*) [static]

Atomically read and conditionally decrement a frame buffer's multicast reference counter.

Parameters:

[in] *bd* - BD describing the buffer

Returns:

uint32_t - the 16 LSB are the value of the counter before the increment A value of 1 indicates the last reference has been released A value of 0 indicates an error (reference counter was already at 0)

static __always_inline uint32_t ezdp_calc_frame_data_checksum (struct ezdp_buffer_desc __cmem * *bd_ptr*, uint32_t *bd_offset*, uint32_t *size*, uint32_t *flags*) [static]

Calculate checksum of frame data buffer.

Parameters:

[in] *bd_ptr* - pointer to BD (in CMEM in 4 byte alignment) describing the buffer to calculate checksum on
[in] *bd_offset* - offset in the source buffer to calculate checksum from
[in] *size* - number of bytes to sum
[in] *flags* - execution flags. Bitwise OR of zero or more flags out of the following: TODO

Note:

1. The checksum calculation assumes an even (2 byte aligned) pointer. If the pointer is odd, the checksum result should be swapped. 2. Applicable only for data buffer.

Returns:

uin32_t - 16 bit checksum value

static __always_inline uint32_t ezdp_buf_data_len (uint32_t *header_offset*, uint32_t *free_bytes*, uint32_t *max_length*) [static]

Calculate the length of the buffer based on header offset and free bytes.

Parameters:

[in] *header_offset* - buffer header offset (can be taken from ezdp_frame_desc) - max size is 255 bytes
[in] *free_bytes* - buffer free bytes (can be taken from ezdp_frame_desc) - max size is 255 bytes
[in] *max_length* - max length of buffer - max value 256 bytes

Returns:

buffer length - max 256 bytes

static __always_inline uint32_t ezdp_lbd_len (uint32_t *data_buf_count*) [static]

Calculate LBD buffer length according to BD count.

Parameters:

[in] *data_buf_count* - BD count from FD

Returns:

LBD buffer length in bytes

static __always_inline uint32_t ezdp_calc_header_offset (uint32_t *min_header_offset*, bool *prefer_128B_offset*, uint32_t *buffer_length*) [static]

Calculate optimized frame header offset.

Parameters:

[in] *min_header_offset* - min header offset (Input: 0, 16, 32, 48, ... 240 Bytes)
[in] *prefer_128B_offset* - Prefer header offset bigger than 128 byte
[in] *buffer_length* - buffer length 1-256

Returns:

Calculated optimized header offset

static __always_inline void ezdp_inc_tm_imem_buf_ctr (uint32_t *value*) [static]

Increment TM IMEM buffer counter.

Parameters:

[in] *value* - value to increment

Returns:

none

static __always_inline void ezdp_inc_tm_imem_buf_ctr_async (uint32_t *value*) [static]

Non blocking version of _ezdp_inc_tm_imem_buf_ctr.

Parameters:

[in] *value* - value to increment

Note:

Call ezdp_sync() to wait for the operation to complete.

Returns:

none

static __always_inline void ezdp_dec_tm_imem_buf_ctr (uint32_t *value*) [static]

Decrement TM IMEM buffer counter.

Parameters:

[in] *value* - value to decrement

Returns:

none

static __always_inline void ezdp_dec_tm_imem_buf_ctr_async (uint32_t *value*) [static]

Non blocking version of _ezdp_dec_tm_imem_buf_ctr.

Parameters:

[in] *value* - value to decrement

Note:

Call ezdp_sync() to wait for the operation to complete.

Returns:

none

static __always_inline uint32_t ezdp_read_tm_imem_buf_ctr (void) [static]

Number of IMEM buffers used by frames currently being processed by TM.

Returns:

int32_t - counter value

Note:

The return value can be negative due to delayed updates of the counter from the cpus

static __always_inline int32_t ezdp_get_first_buf (struct ezdp_frame_desc * *fd*, struct ezdp_buffer_desc * *buf_desc*, struct ezdp_linked_buffers_desc_line __cmem * *lbd_line*, ezdp_frame_buf_iterator_state_t * *iter_st*) [static]

Gets first buffer from frame.

Parameters:

[in] *fd* - Pointer to frame descriptor
[out] *buf_desc* - Pointer to buffer descriptor
[out] *lbd_line* - Pointer in CMEM for LBD line
[out] *iter_st* - Pointer to frame buffer iterator state

Note:

: LBD line is loaded only for extended frame type

Returns:

(-1) - no buffer other - length of data in buffer descriptor

static __always_inline int32_t ezdp_get_next_buf (struct ezdp_frame_desc * *fd*, struct ezdp_buffer_desc * *buf_desc*, struct ezdp_linked_buffers_desc_line __cmem * *lbd_line*, ezdp_frame_buf_iterator_state_t * *iter_st*) [static]

Get next buffer from frame.

Parameters:

[in] *fd* - Pointer to frame descriptor
[out] *buf_desc* - Pointer to buffer descriptor
[in,out] *lbd_line* - Pointer in CMEM for last LBD line
[in,out] *iter_st* - Pointer to frame buffer iterator state

Note:

LBD line is re-loaded if needed you must call ezdp_get_first_buf before calling this function

Returns:

(-1) - no more buffers other - length of data in buffer descriptor

static __always_inline bool ezdp_init_frame (struct ezdp_frame_desc __cmem * *frame_desc*, enum ezdp_frame_type *frame_type*, ezdp_buffer_desc_t *buf_desc*, uint32_t *buf_len*, struct ezdp_linked_buffers_desc_line __cmem * *lbd_line*, uint32_t *header_offset*, uint32_t *budget_id*, ezdp_frame_buf_iterator_state_t * *iter_st*) `[static]`

Create a new frame by setting its frame descriptor params and init frame iterator.

Parameters:

[in,out] *frame_desc* - Pointer to frame descriptor
[in] *frame_type* - Frame type: standard or extended
[in] *buf_desc* - New buffer descriptor that will be added to frame
[in] *buf_len* - Buffer descriptor length (max length is 256 bytes)
[in,out] *lbd_line* - Pointer in CMEM for last LBD line
[in] *header_offset* - Header offset in 1st buffer data.
[in] *budget_id* - Buffer data budget id
[in,out] *iter_st* - Pointer to frame buffer iterator state

Note:

: Not all frame descriptor fields are initialized in this function. The following fields are the user's responsibility: class of service logical_id multicast_control job_budget_id transmit_confirmation_flag timestamp_flag transmit_keep_buf_flag gross_checksum_flag efa_control

Returns:

true - success creating new frame false - fail in the process of creating new frame

static __always_inline void ezdp_append_buf (struct ezdp_frame_desc __cmem * *frame_desc*, ezdp_buffer_desc_t *buf_desc*, uint32_t *buf_len*, struct ezdp_linked_buffers_desc_line __cmem * *lbd_line*, ezdp_frame_buf_iterator_state_t * *iter_st*) `[static]`

Add newlly allocated (by user) buffer to frame pointed by iterator.

Parameters:

[in,out] *frame_desc* - Pointer to frame descriptor
[in] *buf_desc* - New buffer descriptor that will be added to frame
[in] *buf_len* - Buffer descriptor length (max length is 256 bytes)
[in,out] *lbd_line* - Pointer in CMEM for last LBD line
[in,out] *iter_st* - Pointer to frame buffer iterator state

Note:

Call ezdp_init_frame before using this function. Assumes that iterator points to the last buffer in the frame.

Returns:

void

static __always_inline void ezdp_sync_frame (struct ezdp_frame_desc __cmem * *frame_desc*, struct ezdp_linked_buffers_desc_line __cmem * *lbd_line*, ezdp_frame_buf_iterator_state_t * *iter_st*, bool *force*) `[static]`

Store last LBD line to memory, if required.

Parameters:

[in] *frame_desc* - Pointer to frame descriptor
[in] *lbd_line* - Pointer in CMEM for last LBD line
[in] *iter_st* - Pointer to frame buffer iterator state
[in] *force* - Force store of the LBD line

Returns:

void

dpe/dp/include/ezdp_frame_defs.h File Reference

Data Structures

- struct ezdp_buffer_desc
- *Buffer descriptor (BD) data structure.* struct ezdp_frame_desc
- *Frame descriptor data structure.* struct ezdp_2step_1588_header
- *2-step 1588 header format definition* struct ezdp_1step_1588_header
- *1-step 1588 header format definition* struct ezdp_buffer_info
- *Buffer descriptor info.* struct ezdp_1588_header
- *1588 header format definition* struct ezdp_linked_buffers_desc_line
- *LBD Line data structure.* struct ezdp_linked_buffers_desc
- *A generic linked buffers descriptor.* struct ezdp_small_linked_buffers_desc
- *Small linked buffers descriptor.May hold up to 3 buffs.* struct ezdp_large_linked_buffers_desc
- *Large linked buffers descriptor.* struct ezdp_ext_linked_buffers_desc

Extended linked buffers descriptor. Defines

- #define EZDP_BUFFER_DATA_SIZE 256
- #define EZDP_MEM_FRAME_DATA_BUFFER_SIZE EZDP_BUFFER_DATA_SIZE
- #define EZDP_BUFFER_DESC_ID_SIZE 28
- #define EZDP_BUFFER_DESC_ID_OFFSET 0
- #define EZDP_BUFFER_DESC_RESERVED28_29_SIZE 2
- #define EZDP_BUFFER_DESC_RESERVED28_29_OFFSET 28
- #define EZDP_BUFFER_DESC_MEM_TYPE_SIZE 1
- #define EZDP_BUFFER_DESC_MEM_TYPE_OFFSET 30
- #define EZDP_BUFFER_DESC_MEM_TYPE_MASK (1 << EZDP_BUFFER_DESC_MEM_TYPE_OFFSET)
- #define EZDP_BUFFER_DESC_VALID_DATA_BUF_SIZE 1
- #define EZDP_BUFFER_DESC_VALID_DATA_BUF_OFFSET 31
- #define EZDP_BUFFER_DESC_VALID_DATA_BUF_MASK (1 << EZDP_BUFFER_DESC_VALID_DATA_BUF_OFFSET)
- #define EZDP_FRAME_DESC_BUF_BUDGET_ID_SIZE 10
- #define EZDP_FRAME_DESC_BUF_BUDGET_ID_OFFSET 0
- #define EZDP_FRAME_DESC_BUF_BUDGET_ID_WORD_SELECT 0
- #define EZDP_FRAME_DESC_BUF_BUDGET_ID_WORD_OFFSET 0
- #define EZDP_FRAME_DESC_RESERVED10_11_SIZE 2
- #define EZDP_FRAME_DESC_RESERVED10_11_OFFSET 10
- #define EZDP_FRAME_DESC_CLASS_OF_SERVICE_SIZE 2
- #define EZDP_FRAME_DESC_CLASS_OF_SERVICE_OFFSET 12
- #define EZDP_FRAME_DESC_CLASS_OF_SERVICE_WORD_SELECT 0
- #define EZDP_FRAME_DESC_CLASS_OF_SERVICE_WORD_OFFSET 12
- #define EZDP_FRAME_DESC_RESERVED14_15_SIZE 2
- #define EZDP_FRAME_DESC_RESERVED14_15_OFFSET 14
- #define EZDP_FRAME_DESC_RESERVED0_1_SIZE 2
- #define EZDP_FRAME_DESC_RESERVED0_1_OFFSET 16
- #define EZDP_FRAME_DESC_GROSS_CHECKSUM_FLAG_SIZE 1
- #define EZDP_FRAME_DESC_GROSS_CHECKSUM_FLAG_OFFSET 18
- #define EZDP_FRAME_DESC_GROSS_CHECKSUM_FLAG_WORD_SELECT 0
- #define EZDP_FRAME_DESC_GROSS_CHECKSUM_FLAG_WORD_OFFSET 18
- #define EZDP_FRAME_DESC_GROSS_CHECKSUM_FLAG_MASK (1 << EZDP_FRAME_DESC_GROSS_CHECKSUM_FLAG_WORD_OFFSET)
- #define EZDP_FRAME_DESC_TRANSMIT_KEEP_BUF_FLAG_SIZE 1
- #define EZDP_FRAME_DESC_TRANSMIT_KEEP_BUF_FLAG_OFFSET 19
- #define EZDP_FRAME_DESC_TRANSMIT_KEEP_BUF_FLAG_WORD_SELECT 0
- #define EZDP_FRAME_DESC_TRANSMIT_KEEP_BUF_FLAG_WORD_OFFSET 19

- #define EZDP_FRAME_DESC_TRANSMIT_KEEP_BUF_FLAG_MASK (1 << EZDP_FRAME_DESC_TRANSMIT_KEEP_BUF_FLAG_WORD_OFFSET)
- #define EZDP_FRAME_DESC_TIMESTAMP_FLAG_SIZE 1
- #define EZDP_FRAME_DESC_TIMESTAMP_FLAG_OFFSET 20
- #define EZDP_FRAME_DESC_TIMESTAMP_FLAG_WORD_SELECT 0
- #define EZDP_FRAME_DESC_TIMESTAMP_FLAG_WORD_OFFSET 20
- #define EZDP_FRAME_DESC_TIMESTAMP_FLAG_MASK (1 << EZDP_FRAME_DESC_TIMESTAMP_FLAG_WORD_OFFSET)
- #define EZDP_FRAME_DESC_TRANSMIT_CONFIRMATION_FLAG_SIZE 1
- #define EZDP_FRAME_DESC_TRANSMIT_CONFIRMATION_FLAG_OFFSET 21
- #define EZDP_FRAME_DESC_TRANSMIT_CONFIRMATION_FLAG_WORD_SELECT 0
- #define EZDP_FRAME_DESC_TRANSMIT_CONFIRMATION_FLAG_WORD_OFFSET 21
- #define EZDP_FRAME_DESC_TRANSMIT_CONFIRMATION_FLAG_MASK (1 << EZDP_FRAME_DESC_TRANSMIT_CONFIRMATION_FLAG_WORD_OFFSET)
- #define EZDP_FRAME_DESC_TYPE_SIZE 2
- #define EZDP_FRAME_DESC_TYPE_OFFSET 22
- #define EZDP_FRAME_DESC_TYPE_WORD_SELECT 0
- #define EZDP_FRAME_DESC_TYPE_WORD_OFFSET 22
- #define EZDP_FRAME_DESC_ECC_SIZE 8
- #define EZDP_FRAME_DESC_ECC_OFFSET 24
- #define EZDP_FRAME_DESC_ECC_WORD_SELECT 0
- #define EZDP_FRAME_DESC_ECC_WORD_OFFSET 24
- #define EZDP_FRAME_DESC_HEADER_OFFSET_SIZE 8
- #define EZDP_FRAME_DESC_HEADER_OFFSET_OFFSET 32
- #define EZDP_FRAME_DESC_HEADER_OFFSET_WORD_SELECT 1
- #define EZDP_FRAME_DESC_HEADER_OFFSET_WORD_OFFSET 0
- #define EZDP_FRAME_DESC_DATA_BUF_COUNT_SIZE 8
- #define EZDP_FRAME_DESC_DATA_BUF_COUNT_OFFSET 40
- #define EZDP_FRAME_DESC_DATA_BUF_COUNT_WORD_SELECT 1
- #define EZDP_FRAME_DESC_DATA_BUF_COUNT_WORD_OFFSET 8
- #define EZDP_FRAME_DESC_FRAME_LENGTH_SIZE 16
- #define EZDP_FRAME_DESC_FRAME_LENGTH_OFFSET 48
- #define EZDP_FRAME_DESC_FRAME_LENGTH_WORD_SELECT 1
- #define EZDP_FRAME_DESC_FRAME_LENGTH_WORD_OFFSET 16
- #define EZDP_FRAME_DESC_BUF_DESC_SIZE 32
- #define EZDP_FRAME_DESC_BUF_DESC_OFFSET 64
- #define EZDP_FRAME_DESC_BUF_DESC_WORD_SELECT 2
- #define EZDP_FRAME_DESC_BUF_DESC_WORD_OFFSET 0
- #define EZDP_FRAME_DESC_JOB_BUDGET_ID_SIZE 10
- #define EZDP_FRAME_DESC_JOB_BUDGET_ID_OFFSET 96
- #define EZDP_FRAME_DESC_JOB_BUDGET_ID_WORD_SELECT 3
- #define EZDP_FRAME_DESC_JOB_BUDGET_ID_WORD_OFFSET 0
- #define EZDP_FRAME_DESC_RESERVED106_110_SIZE 4
- #define EZDP_FRAME_DESC_RESERVED106_110_OFFSET 106
- #define EZDP_FRAME_DESC_MULTICAST_CONTROL_SIZE 2
- #define EZDP_FRAME_DESC_MULTICAST_CONTROL_OFFSET 110
- #define EZDP_FRAME_DESC_MULTICAST_CONTROL_WORD_SELECT 3
- #define EZDP_FRAME_DESC_MULTICAST_CONTROL_WORD_OFFSET 14
- #define EZDP_FRAME_DESC_LOGICAL_ID_SIZE 8
- #define EZDP_FRAME_DESC_LOGICAL_ID_OFFSET 112
- #define EZDP_FRAME_DESC_LOGICAL_ID_WORD_SELECT 3
- #define EZDP_FRAME_DESC_LOGICAL_ID_WORD_OFFSET 16
- #define EZDP_FRAME_DESC_FREE_BYTES_SIZE 8
- #define EZDP_FRAME_DESC_FREE_BYTES_OFFSET 120
- #define EZDP_FRAME_DESC_FREE_BYTES_WORD_SELECT 3
- #define EZDP_FRAME_DESC_FREE_BYTES_WORD_OFFSET 24
- #define EZDP_FRAME_DESC_WORD_COUNT 4

- #define EZDP_2STEP_1588_HEADER_RESERVED0_23_SIZE 24
- #define EZDP_2STEP_1588_HEADER_RESERVED0_23_OFFSET 0
- #define EZDP_2STEP_1588_HEADER_RESERVED24_SIZE 1
- #define EZDP_2STEP_1588_HEADER_RESERVED24_OFFSET 24
- #define EZDP_2STEP_1588_HEADER_RESERVED24_31_SIZE 7
- #define EZDP_2STEP_1588_HEADER_RESERVED24_31_OFFSET 25
- #define EZDP_2STEP_1588_HEADER_RESERVED32_63_SIZE 32
- #define EZDP_2STEP_1588_HEADER_RESERVED32_63_OFFSET 32
- #define EZDP_2STEP_1588_HEADER_BUF_BUDGET_ID_SIZE 10
- #define EZDP_2STEP_1588_HEADER_BUF_BUDGET_ID_OFFSET 64
- #define EZDP_2STEP_1588_HEADER_BUF_BUDGET_ID_WORD_SELECT 2
- #define EZDP_2STEP_1588_HEADER_BUF_BUDGET_ID_WORD_OFFSET 0
- #define EZDP_2STEP_1588_HEADER_RESERVED74_75_SIZE 2
- #define EZDP_2STEP_1588_HEADER_RESERVED74_75_OFFSET 74
- #define EZDP_2STEP_1588_HEADER_CLASS_OF_SERVICE_SIZE 2
- #define EZDP_2STEP_1588_HEADER_CLASS_OF_SERVICE_OFFSET 76
- #define EZDP_2STEP_1588_HEADER_CLASS_OF_SERVICE_WORD_SELECT 2
- #define EZDP_2STEP_1588_HEADER_CLASS_OF_SERVICE_WORD_OFFSET 12
- #define EZDP_2STEP_1588_HEADER_RESERVED76_77_SIZE 2
- #define EZDP_2STEP_1588_HEADER_RESERVED76_77_OFFSET 78
- #define EZDP_2STEP_1588_HEADER_HEADER_OFFSET_SIZE 8
- #define EZDP_2STEP_1588_HEADER_HEADER_OFFSET_OFFSET 80
- #define EZDP_2STEP_1588_HEADER_HEADER_OFFSET_WORD_SELECT 2
- #define EZDP_2STEP_1588_HEADER_HEADER_OFFSET_WORD_OFFSET 16
- #define EZDP_2STEP_1588_HEADER_FREE_BYTES_SIZE 8
- #define EZDP_2STEP_1588_HEADER_FREE_BYTES_OFFSET 88
- #define EZDP_2STEP_1588_HEADER_FREE_BYTES_WORD_SELECT 2
- #define EZDP_2STEP_1588_HEADER_FREE_BYTES_WORD_OFFSET 24
- #define EZDP_2STEP_1588_HEADER_BUF_DESC_SIZE 32
- #define EZDP_2STEP_1588_HEADER_BUF_DESC_OFFSET 96
- #define EZDP_2STEP_1588_HEADER_BUF_DESC_WORD_SELECT 3
- #define EZDP_2STEP_1588_HEADER_BUF_DESC_WORD_OFFSET 0
- #define EZDP_2STEP_1588_HEADER_WORD_COUNT 4
- #define EZDP_1STEP_1588_HEADER_CHECKSUM_SIZE 16
- #define EZDP_1STEP_1588_HEADER_CHECKSUM_OFFSET 0
- #define EZDP_1STEP_1588_HEADER_CHECKSUM_WORD_SELECT 0
- #define EZDP_1STEP_1588_HEADER_CHECKSUM_WORD_OFFSET 0
- #define EZDP_1STEP_1588_HEADER_RESERVED16_23_SIZE 8
- #define EZDP_1STEP_1588_HEADER_RESERVED16_23_OFFSET 16
- #define EZDP_1STEP_1588_HEADER_RESERVED24_SIZE 1
- #define EZDP_1STEP_1588_HEADER_RESERVED24_OFFSET 24
- #define EZDP_1STEP_1588_HEADER_INJECT_CHECKSUM_FLAG_SIZE 1
- #define EZDP_1STEP_1588_HEADER_INJECT_CHECKSUM_FLAG_OFFSET 25
- #define EZDP_1STEP_1588_HEADER_INJECT_CHECKSUM_FLAG_WORD_SELECT 0
- #define EZDP_1STEP_1588_HEADER_INJECT_CHECKSUM_FLAG_WORD_OFFSET 25
- #define EZDP_1STEP_1588_HEADER_INJECT_CHECKSUM_FLAG_MASK (1 << EZDP_1STEP_1588_HEADER_INJECT_CHECKSUM_FLAG_WORD_OFFSET)
- #define EZDP_1STEP_1588_HEADER_WRAP_AROUND_CONDITION_SIZE 1
- #define EZDP_1STEP_1588_HEADER_WRAP_AROUND_CONDITION_OFFSET 26
- #define EZDP_1STEP_1588_HEADER_WRAP_AROUND_CONDITION_WORD_SELECT 0
- #define EZDP_1STEP_1588_HEADER_WRAP_AROUND_CONDITION_WORD_OFFSET 26
- #define EZDP_1STEP_1588_HEADER_WRAP_AROUND_CONDITION_MASK (1 << EZDP_1STEP_1588_HEADER_WRAP_AROUND_CONDITION_WORD_OFFSET)
- #define EZDP_1STEP_1588_HEADER_CORRECTION_ODD_START_SIZE 1
- #define EZDP_1STEP_1588_HEADER_CORRECTION_ODD_START_OFFSET 27
- #define EZDP_1STEP_1588_HEADER_CORRECTION_ODD_START_WORD_SELECT 0
- #define EZDP_1STEP_1588_HEADER_CORRECTION_ODD_START_WORD_OFFSET 27

- #define EZDP_1STEP_1588_HEADER_CORRECTION_ODD_START_MASK (1 << EZDP_1STEP_1588_HEADER_CORRECTION_ODD_START_WORD_OFFSET)
- #define EZDP_1STEP_1588_HEADER_RESERVED28_31_SIZE 4
- #define EZDP_1STEP_1588_HEADER_RESERVED28_31_OFFSET 28
- #define EZDP_1STEP_1588_HEADER_CORRECTION_OFFSET_SIZE 16
- #define EZDP_1STEP_1588_HEADER_CORRECTION_OFFSET_OFFSET 32
- #define EZDP_1STEP_1588_HEADER_CORRECTION_OFFSET_WORD_SELECT 1
- #define EZDP_1STEP_1588_HEADER_CORRECTION_OFFSET_WORD_OFFSET 0
- #define EZDP_1STEP_1588_HEADER_CHECKSUM_OFFSET_SIZE 16
- #define EZDP_1STEP_1588_HEADER_CHECKSUM_OFFSET_OFFSET 48
- #define EZDP_1STEP_1588_HEADER_CHECKSUM_OFFSET_WORD_SELECT 1
- #define EZDP_1STEP_1588_HEADER_CHECKSUM_OFFSET_WORD_OFFSET 16
- #define EZDP_1STEP_1588_HEADER_CORRECTION_SIZE 64
- #define EZDP_1STEP_1588_HEADER_CORRECTION_OFFSET 64
- #define EZDP_1STEP_1588_HEADER_CORRECTION_WORD_SELECT 2
- #define EZDP_1STEP_1588_HEADER_CORRECTION_WORD_OFFSET 0
- #define EZDP_1STEP_1588_HEADER_WORD_COUNT 4
- #define EZDP_LINKED_BUFFER_DESC_LINE_NUMBER_OF_BUFFERS_DESC 3

Typedefs

- typedef uint32_t ezdp_buffer_desc_t
- typedef struct ezdp_frame_buf_iterator_state ezdp_frame_buf_iterator_state_t
- typedef struct ezdp_extract_frame_tail_working_area ezdp_extract_frame_tail_working_area_t
- typedef struct ezdp_convert_std2ext_working_area ezdp_convert_std2ext_working_area_t
- typedef struct ezdp_concat_frames_working_area ezdp_concat_frames_working_area_t
- typedef struct ezdp_trim_frame_head_working_area ezdp_trim_frame_head_working_area_t

Enumerations

- enum ezdp_buffer_mem_type { EZDP_EXT_MEM = 0x0, EZDP_INT_MEM = 0x1 }
  buffer mem type possible values.
- enum ezdp_frame_type { EZDP_NULL_FRAME = 0x0, EZDP_EXT_FRAME = 0x1, EZDP_STD_FRAME = 0x2 }
  frame type possible values.
- enum ezdp_multicast_control { EZDP_UNICAST = 0x0, EZDP_MULTICAST = 0x1, EZDP_BROADCAST = 0x2, EZDP_REPLICA = 0x3 }
  multicast control possible values.
- enum ezdp_1588_type { EZDP_2STEP = 0, EZDP_1STEP = 1 }
  job container command type possible values.
- enum ezdp_linked_buffers_desc_size { EZDP_SMALL_LBD = 1, EZDP_LARGE_LBD = 2, EZDP_EXTENDED_LBD = 16 }
  LBD definition.

Define Documentation

#define EZDP_BUFFER_DATA_SIZE 256

#define EZDP_MEM_FRAME_DATA_BUFFER_SIZE EZDP_BUFFER_DATA_SIZE

#define EZDP_BUFFER_DESC_ID_SIZE 28

#define EZDP_BUFFER_DESC_ID_OFFSET 0

#define EZDP_BUFFER_DESC_RESERVED28_29_SIZE 2

#define EZDP_BUFFER_DESC_RESERVED28_29_OFFSET 28

#define EZDP_BUFFER_DESC_MEM_TYPE_SIZE 1

#define EZDP_BUFFER_DESC_MEM_TYPE_OFFSET 30

#define EZDP_BUFFER_DESC_MEM_TYPE_MASK (1 << EZDP_BUFFER_DESC_MEM_TYPE_OFFSET)

#define EZDP_BUFFER_DESC_VALID_DATA_BUF_SIZE 1

#define EZDP_BUFFER_DESC_VALID_DATA_BUF_OFFSET 31

#define EZDP_BUFFER_DESC_VALID_DATA_BUF_MASK (1 << EZDP_BUFFER_DESC_VALID_DATA_BUF_OFFSET)

#define EZDP_FRAME_DESC_BUF_BUDGET_ID_SIZE 10

#define EZDP_FRAME_DESC_BUF_BUDGET_ID_OFFSET 0

#define EZDP_FRAME_DESC_BUF_BUDGET_ID_WORD_SELECT 0

#define EZDP_FRAME_DESC_BUF_BUDGET_ID_WORD_OFFSET 0

#define EZDP_FRAME_DESC_RESERVED10_11_SIZE 2

#define EZDP_FRAME_DESC_RESERVED10_11_OFFSET 10

#define EZDP_FRAME_DESC_CLASS_OF_SERVICE_SIZE 2

#define EZDP_FRAME_DESC_CLASS_OF_SERVICE_OFFSET 12

#define EZDP_FRAME_DESC_CLASS_OF_SERVICE_WORD_SELECT 0

#define EZDP_FRAME_DESC_CLASS_OF_SERVICE_WORD_OFFSET 12

#define EZDP_FRAME_DESC_RESERVED14_15_SIZE 2

#define EZDP_FRAME_DESC_RESERVED14_15_OFFSET 14

#define EZDP_FRAME_DESC_RESERVED0_1_SIZE 2

```
#define EZDP_FRAME_DESC_RESERVED0_1_OFFSET 16

#define EZDP_FRAME_DESC_GROSS_CHECKSUM_FLAG_SIZE 1

#define EZDP_FRAME_DESC_GROSS_CHECKSUM_FLAG_OFFSET 18

#define EZDP_FRAME_DESC_GROSS_CHECKSUM_FLAG_WORD_SELECT 0

#define EZDP_FRAME_DESC_GROSS_CHECKSUM_FLAG_WORD_OFFSET 18

#define EZDP_FRAME_DESC_GROSS_CHECKSUM_FLAG_MASK (1 <<
EZDP_FRAME_DESC_GROSS_CHECKSUM_FLAG_WORD_OFFSET)

#define EZDP_FRAME_DESC_TRANSMIT_KEEP_BUF_FLAG_SIZE 1

#define EZDP_FRAME_DESC_TRANSMIT_KEEP_BUF_FLAG_OFFSET 19

#define EZDP_FRAME_DESC_TRANSMIT_KEEP_BUF_FLAG_WORD_SELECT 0

#define EZDP_FRAME_DESC_TRANSMIT_KEEP_BUF_FLAG_WORD_OFFSET 19

#define EZDP_FRAME_DESC_TRANSMIT_KEEP_BUF_FLAG_MASK (1 <<
EZDP_FRAME_DESC_TRANSMIT_KEEP_BUF_FLAG_WORD_OFFSET)

#define EZDP_FRAME_DESC_TIMESTAMP_FLAG_SIZE 1

#define EZDP_FRAME_DESC_TIMESTAMP_FLAG_OFFSET 20

#define EZDP_FRAME_DESC_TIMESTAMP_FLAG_WORD_SELECT 0

#define EZDP_FRAME_DESC_TIMESTAMP_FLAG_WORD_OFFSET 20

#define EZDP_FRAME_DESC_TIMESTAMP_FLAG_MASK (1 <<
EZDP_FRAME_DESC_TIMESTAMP_FLAG_WORD_OFFSET)

#define EZDP_FRAME_DESC_TRANSMIT_CONFIRMATION_FLAG_SIZE 1

#define EZDP_FRAME_DESC_TRANSMIT_CONFIRMATION_FLAG_OFFSET 21

#define EZDP_FRAME_DESC_TRANSMIT_CONFIRMATION_FLAG_WORD_SELECT 0

#define EZDP_FRAME_DESC_TRANSMIT_CONFIRMATION_FLAG_WORD_OFFSET 21

#define EZDP_FRAME_DESC_TRANSMIT_CONFIRMATION_FLAG_MASK (1 <<
EZDP_FRAME_DESC_TRANSMIT_CONFIRMATION_FLAG_WORD_OFFSET)

#define EZDP_FRAME_DESC_TYPE_SIZE 2

#define EZDP_FRAME_DESC_TYPE_OFFSET 22

#define EZDP_FRAME_DESC_TYPE_WORD_SELECT 0

#define EZDP_FRAME_DESC_TYPE_WORD_OFFSET 22
```

```
#define EZDP_FRAME_DESC_ECC_SIZE 8

#define EZDP_FRAME_DESC_ECC_OFFSET 24

#define EZDP_FRAME_DESC_ECC_WORD_SELECT 0

#define EZDP_FRAME_DESC_ECC_WORD_OFFSET 24

#define EZDP_FRAME_DESC_HEADER_OFFSET_SIZE 8

#define EZDP_FRAME_DESC_HEADER_OFFSET_OFFSET 32

#define EZDP_FRAME_DESC_HEADER_OFFSET_WORD_SELECT 1

#define EZDP_FRAME_DESC_HEADER_OFFSET_WORD_OFFSET 0

#define EZDP_FRAME_DESC_DATA_BUF_COUNT_SIZE 8

#define EZDP_FRAME_DESC_DATA_BUF_COUNT_OFFSET 40

#define EZDP_FRAME_DESC_DATA_BUF_COUNT_WORD_SELECT 1

#define EZDP_FRAME_DESC_DATA_BUF_COUNT_WORD_OFFSET 8

#define EZDP_FRAME_DESC_FRAME_LENGTH_SIZE 16

#define EZDP_FRAME_DESC_FRAME_LENGTH_OFFSET 48

#define EZDP_FRAME_DESC_FRAME_LENGTH_WORD_SELECT 1

#define EZDP_FRAME_DESC_FRAME_LENGTH_WORD_OFFSET 16

#define EZDP_FRAME_DESC_BUF_DESC_SIZE 32

#define EZDP_FRAME_DESC_BUF_DESC_OFFSET 64

#define EZDP_FRAME_DESC_BUF_DESC_WORD_SELECT 2

#define EZDP_FRAME_DESC_BUF_DESC_WORD_OFFSET 0

#define EZDP_FRAME_DESC_JOB_BUDGET_ID_SIZE 10

#define EZDP_FRAME_DESC_JOB_BUDGET_ID_OFFSET 96

#define EZDP_FRAME_DESC_JOB_BUDGET_ID_WORD_SELECT 3

#define EZDP_FRAME_DESC_JOB_BUDGET_ID_WORD_OFFSET 0

#define EZDP_FRAME_DESC_RESERVED106_110_SIZE 4

#define EZDP_FRAME_DESC_RESERVED106_110_OFFSET 106
```

#define EZDP_FRAME_DESC_MULTICAST_CONTROL_SIZE 2

#define EZDP_FRAME_DESC_MULTICAST_CONTROL_OFFSET 110

#define EZDP_FRAME_DESC_MULTICAST_CONTROL_WORD_SELECT 3

#define EZDP_FRAME_DESC_MULTICAST_CONTROL_WORD_OFFSET 14

#define EZDP_FRAME_DESC_LOGICAL_ID_SIZE 8

#define EZDP_FRAME_DESC_LOGICAL_ID_OFFSET 112

#define EZDP_FRAME_DESC_LOGICAL_ID_WORD_SELECT 3

#define EZDP_FRAME_DESC_LOGICAL_ID_WORD_OFFSET 16

#define EZDP_FRAME_DESC_FREE_BYTES_SIZE 8

#define EZDP_FRAME_DESC_FREE_BYTES_OFFSET 120

#define EZDP_FRAME_DESC_FREE_BYTES_WORD_SELECT 3

#define EZDP_FRAME_DESC_FREE_BYTES_WORD_OFFSET 24

#define EZDP_FRAME_DESC_WORD_COUNT 4

#define EZDP_2STEP_1588_HEADER_RESERVED0_23_SIZE 24

#define EZDP_2STEP_1588_HEADER_RESERVED0_23_OFFSET 0

#define EZDP_2STEP_1588_HEADER_RESERVED24_SIZE 1

#define EZDP_2STEP_1588_HEADER_RESERVED24_OFFSET 24

#define EZDP_2STEP_1588_HEADER_RESERVED24_31_SIZE 7

#define EZDP_2STEP_1588_HEADER_RESERVED24_31_OFFSET 25

#define EZDP_2STEP_1588_HEADER_RESERVED32_63_SIZE 32

#define EZDP_2STEP_1588_HEADER_RESERVED32_63_OFFSET 32

#define EZDP_2STEP_1588_HEADER_BUF_BUDGET_ID_SIZE 10

#define EZDP_2STEP_1588_HEADER_BUF_BUDGET_ID_OFFSET 64

#define EZDP_2STEP_1588_HEADER_BUF_BUDGET_ID_WORD_SELECT 2

#define EZDP_2STEP_1588_HEADER_BUF_BUDGET_ID_WORD_OFFSET 0

#define EZDP_2STEP_1588_HEADER_RESERVED74_75_SIZE 2

#define EZDP_2STEP_1588_HEADER_RESERVED74_75_OFFSET 74

#define EZDP_2STEP_1588_HEADER_CLASS_OF_SERVICE_SIZE 2

#define EZDP_2STEP_1588_HEADER_CLASS_OF_SERVICE_OFFSET 76

#define EZDP_2STEP_1588_HEADER_CLASS_OF_SERVICE_WORD_SELECT 2

#define EZDP_2STEP_1588_HEADER_CLASS_OF_SERVICE_WORD_OFFSET 12

#define EZDP_2STEP_1588_HEADER_RESERVED76_77_SIZE 2

#define EZDP_2STEP_1588_HEADER_RESERVED76_77_OFFSET 78

#define EZDP_2STEP_1588_HEADER_HEADER_OFFSET_SIZE 8

#define EZDP_2STEP_1588_HEADER_HEADER_OFFSET_OFFSET 80

#define EZDP_2STEP_1588_HEADER_HEADER_OFFSET_WORD_SELECT 2

#define EZDP_2STEP_1588_HEADER_HEADER_OFFSET_WORD_OFFSET 16

#define EZDP_2STEP_1588_HEADER_FREE_BYTES_SIZE 8

#define EZDP_2STEP_1588_HEADER_FREE_BYTES_OFFSET 88

#define EZDP_2STEP_1588_HEADER_FREE_BYTES_WORD_SELECT 2

#define EZDP_2STEP_1588_HEADER_FREE_BYTES_WORD_OFFSET 24

#define EZDP_2STEP_1588_HEADER_BUF_DESC_SIZE 32

#define EZDP_2STEP_1588_HEADER_BUF_DESC_OFFSET 96

#define EZDP_2STEP_1588_HEADER_BUF_DESC_WORD_SELECT 3

#define EZDP_2STEP_1588_HEADER_BUF_DESC_WORD_OFFSET 0

#define EZDP_2STEP_1588_HEADER_WORD_COUNT 4

#define EZDP_1STEP_1588_HEADER_CHECKSUM_SIZE 16

#define EZDP_1STEP_1588_HEADER_CHECKSUM_OFFSET 0

#define EZDP_1STEP_1588_HEADER_CHECKSUM_WORD_SELECT 0

#define EZDP_1STEP_1588_HEADER_CHECKSUM_WORD_OFFSET 0

#define EZDP_1STEP_1588_HEADER_RESERVED16_23_SIZE 8

#define EZDP_1STEP_1588_HEADER_RESERVED16_23_OFFSET 16

#define EZDP_1STEP_1588_HEADER_RESERVED24_SIZE 1

#define EZDP_1STEP_1588_HEADER_RESERVED24_OFFSET 24

#define EZDP_1STEP_1588_HEADER_INJECT_CHECKSUM_FLAG_SIZE 1

#define EZDP_1STEP_1588_HEADER_INJECT_CHECKSUM_FLAG_OFFSET 25

#define EZDP_1STEP_1588_HEADER_INJECT_CHECKSUM_FLAG_WORD_SELECT 0

#define EZDP_1STEP_1588_HEADER_INJECT_CHECKSUM_FLAG_WORD_OFFSET 25

#define EZDP_1STEP_1588_HEADER_INJECT_CHECKSUM_FLAG_MASK (1 << EZDP_1STEP_1588_HEADER_INJECT_CHECKSUM_FLAG_WORD_OFFSET)

#define EZDP_1STEP_1588_HEADER_WRAP_AROUND_CONDITION_SIZE 1

#define EZDP_1STEP_1588_HEADER_WRAP_AROUND_CONDITION_OFFSET 26

#define EZDP_1STEP_1588_HEADER_WRAP_AROUND_CONDITION_WORD_SELECT 0

#define EZDP_1STEP_1588_HEADER_WRAP_AROUND_CONDITION_WORD_OFFSET 26

#define EZDP_1STEP_1588_HEADER_WRAP_AROUND_CONDITION_MASK (1 << EZDP_1STEP_1588_HEADER_WRAP_AROUND_CONDITION_WORD_OFFSET)

#define EZDP_1STEP_1588_HEADER_CORRECTION_ODD_START_SIZE 1

#define EZDP_1STEP_1588_HEADER_CORRECTION_ODD_START_OFFSET 27

#define EZDP_1STEP_1588_HEADER_CORRECTION_ODD_START_WORD_SELECT 0

#define EZDP_1STEP_1588_HEADER_CORRECTION_ODD_START_WORD_OFFSET 27

#define EZDP_1STEP_1588_HEADER_CORRECTION_ODD_START_MASK (1 << EZDP_1STEP_1588_HEADER_CORRECTION_ODD_START_WORD_OFFSET)

#define EZDP_1STEP_1588_HEADER_RESERVED28_31_SIZE 4

#define EZDP_1STEP_1588_HEADER_RESERVED28_31_OFFSET 28

#define EZDP_1STEP_1588_HEADER_CORRECTION_OFFSET_SIZE 16

#define EZDP_1STEP_1588_HEADER_CORRECTION_OFFSET_OFFSET 32

#define EZDP_1STEP_1588_HEADER_CORRECTION_OFFSET_WORD_SELECT 1

#define EZDP_1STEP_1588_HEADER_CORRECTION_OFFSET_WORD_OFFSET 0

#define EZDP_1STEP_1588_HEADER_CHECKSUM_OFFSET_SIZE 16

#define EZDP_1STEP_1588_HEADER_CHECKSUM_OFFSET_OFFSET 48

#define EZDP_1STEP_1588_HEADER_CHECKSUM_OFFSET_WORD_SELECT 1

#define EZDP_1STEP_1588_HEADER_CHECKSUM_OFFSET_WORD_OFFSET 16

#define EZDP_1STEP_1588_HEADER_CORRECTION_SIZE 64

#define EZDP_1STEP_1588_HEADER_CORRECTION_OFFSET 64

#define EZDP_1STEP_1588_HEADER_CORRECTION_WORD_SELECT 2

#define EZDP_1STEP_1588_HEADER_CORRECTION_WORD_OFFSET 0

#define EZDP_1STEP_1588_HEADER_WORD_COUNT 4

#define EZDP_LINKED_BUFFER_DESC_LINE_NUMBER_OF_BUFFERS_DESC 3

Typedef Documentation

typedef uint32_t ezdp_buffer_desc_t

typedef struct ezdp_frame_buf_iterator_state ezdp_frame_buf_iterator_state_t

typedef struct ezdp_extract_frame_tail_working_area ezdp_extract_frame_tail_working_area_t

typedef struct ezdp_convert_std2ext_working_area ezdp_convert_std2ext_working_area_t

typedef struct ezdp_concat_frames_working_area ezdp_concat_frames_working_area_t

typedef struct ezdp_trim_frame_head_working_area ezdp_trim_frame_head_working_area_t

---

Enumeration Type Documentation

enum ezdp_buffer_mem_type

buffer mem type possible values.

Enumerator:

EZDP_EXT_MEM Data is located in EMEM.

EZDP_INT_MEM Data is located in IMEM.

enum ezdp_frame_type

frame type possible values.

Enumerator:

EZDP_NULL_FRAME Null frame.

Frame has no allocated buffers. All other ezdp_frame fields are don't-care.

EZDP_EXT_FRAME Multi-buffer frame with separate link buffer descriptor.

All frame data buffers (BDs) are listed in the link buffer descriptor when header buffer descriptor points to the link buffer descriptor.

EZDP_STD_FRAME Multi-buffer frame data with embedded link buffer descriptor.

The buffer pointer has two meanings, differentiated through API usage. In data buffer API it points to the first frame data buffer (which starts after header_offset bytes), while in the LBD buffer API it points to the LBD entries at the beginning of the buffer. The size of the embedded link buffer descriptor (length) may be empty (data_buf_count = 1), 16B (data_buf_count greater than 1 but less than 4) or 32B (data_buf_count greater than 4 but less than 7). The maximum length of the embedded LBD is limited to 32B.

enum ezdp_multicast_control

multicast control possible values.

Enumerator:

EZDP_UNICAST The frame is unicast.

Buffer multicast reference counter value is not applicable.

EZDP_MULTICAST The first buffer and LBD are not multicasted, the rest of the buffers are multicasted.

EZDP_BROADCAST All buffer are multicasted (Buffer multicast reference counter values are applicable to all buffers).

EZDP_REPLICA On receive path, the value indicates that the ezdp_frame.

replication_num field is valid and holds the current replication number. On transmit path to loopback port, this value is used as replication command. On transmit path to external port or in case of drop, this value is similar to UNICAST functionality.

enum ezdp_1588_type

job container command type possible values.

Enumerator:

EZDP_2STEP 2-step 1588 header.

EZDP_1STEP 1-step 1588 header.

enum ezdp_linked_buffers_desc_size

LBD definition.

Enumerator:

EZDP_SMALL_LBD Small LBD may contain up to 3 buffs which are organized in 1 LBD line.

Applicable for EZDP_STD_FRAME frame type.

EZDP_LARGE_LBD Small LBD may contain up to 6 buffs which are organized in 2 LBD line Applicable for EZDP_STD_FRAME frame type.

EZDP_EXTENDED_LBD Extended LBD may contain up to 48 buffs which are organized in 16 LBD lines Applicable for EZDP_EXT_FRAME frame type.

dpe/dp/include/ezdp_job.h File Reference

Defines

- #define ezdp_notifier(NAME) void ezdp_notice_handler(void) __attribute__((alias (#NAME)))

Register notifier function. Typedefs

- typedef void(* ezdp_notifier_t)(void)

IPC notifier handle function. Functions

- static __always_inline ezdp_job_id_t ezdp_alloc_job_id (uint32_t budget_id)
- *Allocate a new job from the PMU.* static __always_inline void ezdp_alloc_job_id_async (ezdp_job_id_t *jobh_ptr, uint32_t budget_id)
- *Non blocking version of ezdp_alloc_job_id().* static __always_inline ezdp_job_id_t ezdp_alloc_multi_job_id (ezdp_job_id_t __cmem *jobhs_ptr, uint32_t budget_id, uint32_t num_of_jobs)
- *Allocate multiple new jobs from the PMU.* static __always_inline void ezdp_alloc_multi_job_id_async (ezdp_job_id_t __cmem *jobhs_ptr, uint32_t budget_id, uint32_t num_of_jobs)
- *Non blocking version of ezdp_alloc_multi_job_id().* static __always_inline bool ezdp_job_alloc_failed (ezdp_job_id_t ret)
- *Check if allocation of the job failed.* static __always_inline void ezdp_free_job_id (ezdp_job_id_t *__cmem jobh_ptr, uint32_t budget_id)
- *Recycle a job to the PMU.* static __always_inline void ezdp_free_job_id_async (ezdp_job_id_t *__cmem jobh_ptr, uint32_t budget_id)
- *Non blocking version of ezdp_free_job_id().* static __always_inline uint32_t ezdp_read_free_job (void)
- *The number of jobs available to be obtained.* static __always_inline void ezdp_rebudget_job (uint32_t free_budget_id, uint32_t free_num_of_jobs, uint32_t alloc_budget_id, uint32_t alloc_num_of_jobs)
- *Update the budget to which jobs are credited.* static __always_inline void ezdp_rebudget_job_async (uint32_t free_budget_id, uint32_t free_num_of_jobs, uint32_t alloc_budget_id, uint32_t alloc_num_of_jobs)
- *Non blocking version of ezdp_rebudget_job().* static __always_inline void ezdp_load_job (ezdp_job_id_t *jobh_ptr, struct ezdp_job_desc __cmem *jd_ptr)
- *Copy the job descriptor from IMEM to CMEM.* static __always_inline void ezdp_load_job_async (ezdp_job_id_t *jobh_ptr, struct ezdp_job_desc __cmem *jd_ptr)
- *Non blocking version of ezdp_load_job().* static __always_inline void ezdp_store_job (ezdp_job_id_t *jobh_ptr, struct ezdp_job_desc __cmem *jd_ptr)
- *Copy the job descriptor from CMEM to IMEM.* static __always_inline void ezdp_store_job_async (ezdp_job_id_t *jobh_ptr, struct ezdp_job_desc __cmem *jd_ptr)
- *Non blocking version of ezdp_store_job().* static __always_inline void ezdp_store_job_container (ezdp_job_id_t *jobh_ptr, struct ezdp_job_container_desc __cmem *jd_ptr)
- *Copy the job container descriptor from CMEM to IMEM.* static __always_inline void ezdp_store_job_container_async (ezdp_job_id_t *jobh_ptr, struct ezdp_job_container_desc __cmem *jd_ptr)
- *Non blocking version of ezdp_store_job_container().* static __always_inline void ezdp_request_job_id (ezdp_job_id_t *__cmem jobh_ptr)
- *Request a new job from the PMU.* static __always_inline void ezdp_wait_for_job_id (void)
- *Suspend execution until a job request completes.* static __always_inline void ezdp_cancel_job_request (void)
- *Cancel a job request from the PMU.* static __always_inline void ezdp_receive_job (ezdp_job_id_t __cmem *jobh_ptr, struct ezdp_job_desc __cmem *jd_ptr, uint32_t flags)
- *Request a new job from the PMU and load it to CMEM.* static __always_inline void ezdp_send_job_id_to_queue (ezdp_job_id_t __cmem *jobh_ptr, uint32_t side, uint32_t target_queue, uint32_t flags)
- *Dispatch the job to another PMU queue.* static __always_inline void ezdp_send_job_id_to_queue_async (ezdp_job_id_t __cmem *jobh_ptr, uint32_t side, uint32_t target_queue, uint32_t flags)
- *Non blocking version of ezdp_send_job_id_to_queue().* static __always_inline void ezdp_send_job_id_to_tm (ezdp_job_id_t __cmem *jobh_ptr, uint32_t side, uint32_t flags)
- *Transmit the job via the TM.* static __always_inline void ezdp_send_job_id_to_tm_async (ezdp_job_id_t __cmem *jobh_ptr, uint32_t side, uint32_t flags)

- *Non blocking version of ezdp_send_job_id_to_tm()*. static __always_inline void ezdp_send_job_id_to_interface (ezdp_job_id_t __cmem *jobh_ptr, uint32_t side, uint32_t output_channel, uint32_t flags)
- *Transmit the job directly to an output queue channel, bypassing the TM*. static __always_inline void ezdp_send_job_id_to_interface_async (ezdp_job_id_t __cmem *jobh_ptr, uint32_t side, uint32_t output_channel, uint32_t flags)
- *Non blocking version of ezdp_send_job_id_to_interface()*. static __always_inline void ezdp_update_job_id_queue (ezdp_job_id_t __cmem *jobh_ptr, uint32_t side, uint32_t target_queue, uint32_t flags)
- *Move the job to another PMU queue without dispatching it*. static __always_inline void ezdp_send_job_to_queue (ezdp_job_id_t __cmem *jobh_ptr, struct ezdp_job_desc __cmem *jd_ptr, uint32_t side, uint32_t target_queue, uint32_t flags)
- *Store the job descriptor and dispatch the job to another queue in the PMU*. static __always_inline void ezdp_send_job_to_tm (ezdp_job_id_t __cmem *jobh_ptr, struct ezdp_job_desc __cmem *jd_ptr, uint32_t side, uint32_t flags)
- *Store the job descriptor and transmit the job via the TM*. static __always_inline void ezdp_send_job_to_interface (ezdp_job_id_t __cmem *jobh_ptr, struct ezdp_job_desc __cmem *jd_ptr, uint32_t side, uint32_t output_channel, uint32_t flags)
- *Store the job descriptor and transmit the job directly to interface by bypassing the TM*. static __always_inline void ezdp_update_job_queue (ezdp_job_id_t *__cmem jobh_ptr, struct ezdp_job_desc __cmem *jd_ptr, uint32_t side, uint32_t target_queue, uint32_t flags)
- *Store the job descriptor and move the job to another PMU queue without dispatching it*. static __always_inline void ezdp_discard_job_id (ezdp_job_id_t *__cmem jobh_ptr)
- *Discard a job and all its associated frame resources*. static __always_inline void ezdp_discard_job_id_async (ezdp_job_id_t *__cmem jobh_ptr)
- *Non blocking version of ezdp_discard_job_id()*. static __always_inline void ezdp_discard_job (ezdp_job_id_t __cmem *jobh_ptr, struct ezdp_job_desc __cmem *jd_ptr)
- *Store the job descriptor and discard a job and all its associated frame resources*. static __always_inline void ezdp_send_job_id_container (ezdp_job_id_t __cmem *jobh_ptr)
- *Send request to PMU to distribute the job container*. static __always_inline void ezdp_send_job_id_container_async (ezdp_job_id_t __cmem *jobh_ptr)
- *Non blocking version of ezdp_send_job_id_container()*. static __always_inline void ezdp_send_job_container (ezdp_job_id_t __cmem *jobh_ptr, struct ezdp_job_container_desc __cmem *jd_container_ptr)
- *Store the job container and send request to PMU to distribute it*. static __always_inline uint32_t ezdp_container_job_count (ezdp_job_container_info_t info)
- *Return the number of the jobs that are in the job container*. static __always_inline ezdp_job_container_info_t ezdp_container_info (uint32_t job_count)
- *Set the info according to the number of jobs in the job container*. static __always_inline void ezdp_notify_cpu (uint32_t target_cpu_id)
- *Notify target CPU*. static __always_inline bool ezdp_notice_pending (void)
- *Check if there is a new notice*. static __always_inline void ezdp_clear_notice (void)
- *Clear notice indication*. static __always_inline void ezdp_wait_for_notice (void)
- *Suspend execution until receiving new notification for CPU*. static __always_inline bool ezdp_check_notice (void)
- *Check if there is a new notice and clear new notice indication*. static __always_inline void ezdp_wait_for_event (void)
- *Suspend execution until job request completed or new notice received*. static __always_inline void ezdp_handle_notice (void)
- *Handle notice*. static __always_inline ezdp_congestion_status_t ezdp_read_congestion_status (enum ezdp_flow_control_node node_type, uint32_t node_id)
- *Read priority drop congestion status*. static __always_inline ezdp_flow_control_status_t ezdp_read_flow_control_status (enum ezdp_budget_type budget_type, enum ezdp_flow_control_node node_type, uint32_t node_id)
- *Read flow control status*. static __always_inline enum ezdp_congestion_level ezdp_read_pmu_input_queue_congestion (uint32_t side, uint32_t queue_id)
- *Read PMU input queue congestion level*. static __always_inline uint32_t ezdp_read_global_budget (enum ezdp_budget_type budget_type)
- *Read global budget counter value*. static __always_inline void ezdp_read_pmu_input_queue_status (uint32_t side, uint32_t queue_id, struct ezdp_input_queue_status *queue_status)

- *Read PMU input queue status from system info.* static __always_inline void ezdp_read_pmu_tm_output_queue_status (uint32_t side, struct ezdp_output_queue_status *queue_status)
- *Read PMU TM output queue status from system info.* static __always_inline void ezdp_read_pmu_discard_output_queue_status (uint32_t side, struct ezdp_output_queue_status *queue_status)
- *Read PMU discard output queue status from system info.* static __always_inline void ezdp_read_pmu_tm_bypass_output_queue_status (uint32_t side, uint32_t queue_id, struct ezdp_output_queue_status *queue_status)
- *Read PMU TM bypass output queue status from system info.* static __always_inline void ezdp_read_pmu_app_schlr_status (uint32_t side, uint32_t app_schlr_id, struct ezdp_app_schlr_status *app_schlr_status)
- *Read PMU application scheduler status from system info.* static __always_inline void ezdp_read_pmu_group_schlr_status (uint32_t side, uint32_t group_schlr_id, struct ezdp_group_schlr_status *group_schlr_status)
- *Read PMU group scheduler status from system info.* static __always_inline void ezdp_init_tm_reporting_desc (uint32_t side, uint32_t level, ezdp_tm_queue_depth_desc_t *tm_desc, char __cmem *work_area_ptr, uint32_t work_area_size)
- *Initialize TM queue depth descriptor.* static __always_inline ezdp_tm_queue_depth_handle_t ezdp_calc_tm_queue_depth_handle (uint32_t queue_id, ezdp_tm_queue_depth_desc_t *tm_desc)
- *Calculate tm queue depth handle.* static __always_inline uint32_t ezdp_get_tm_queue_depth (ezdp_tm_queue_depth_handle_t tm_handle, enum ezdp_report_size report_size)
- *Get entity queue depth.* static __always_inline bool ezdp_valid_tm_queue_depth_handle (ezdp_tm_queue_depth_handle_t tm_handle)

validate tm_handle received from CP/DP API

Define Documentation

#define ezdp_notifier(NAME) void ezdp_notice_handler(void) __attribute__((alias (#NAME)))

Register notifier function.

Returns:
none

Typedef Documentation

typedef void(* ezdp_notifier_t)(void)

IPC notifier handle function.

Function Documentation

static __always_inline ezdp_job_id_t ezdp_alloc_job_id (uint32_t *budget_id*) [static]

Allocate a new job from the PMU.

Parameters:
[in] *budget_id* - budget id to charge

Returns:
ezdp_job_id_t, use ezdp_job_alloc_failed API to check if allocation success or fail

static __always_inline void ezdp_alloc_job_id_async (ezdp_job_id_t * *jobh_ptr*, uint32_t *budget_id*) [static]

Non blocking version of ezdp_alloc_job_id().

Parameters:

[out] *jobh_ptr* - pointer to the job id to write response to
[in] *budget_id* - budget id to charge

Returns:

void, use ezdp_job_alloc_failed API to check if allocation success or fail

static __always_inline ezdp_job_id_t ezdp_alloc_multi_job_id (ezdp_job_id_t __cmem * *jobhs_ptr*, uint32_t *budget_id*, uint32_t *num_of_jobs*) [static]

Allocate multiple new jobs from the PMU.

Parameters:

[out] *jobhs_ptr* - relative offset in CMEM to write response to
[in] *budget_id* - budget id to charge
[in] *num_of_jobs* - the number of job IDs to allocate (possible values: 1-4)

Returns:

ezdp_job_id_t, use ezdp_job_alloc_failed API on returned job_id to check if allocation success or fail

static __always_inline void ezdp_alloc_multi_job_id_async (ezdp_job_id_t __cmem * *jobhs_ptr*, uint32_t *budget_id*, uint32_t *num_of_jobs*) [static]

Non blocking version of ezdp_alloc_multi_job_id().

Parameters:

[out] *jobhs_ptr* - relative offset in CMEM to write response to
[in] *budget_id* - budget id to charge
[in] *num_of_jobs* - the number of job IDs to allocate (possible values: 1-4)

Returns:

void, use ezdp_job_alloc_failed API on first allocated job_id to check if allocation success or fail

static __always_inline bool ezdp_job_alloc_failed (ezdp_job_id_t *ret*) [static]

Check if allocation of the job failed.

Parameters:

[in] *ret* - return value from job allocation API

Returns:

bool - true if allocation failed

static __always_inline void ezdp_free_job_id (ezdp_job_id_t *__cmem *jobh_ptr*, uint32_t *budget_id*) [static]

Recycle a job to the PMU.

Parameters:

[in] *jobh_ptr* - pointer to the job id in CMEM
[in] *budget_id* - budget id to charge

Note:

Any associated frame resources are not recycled.

Returns:

none

static __always_inline void ezdp_free_job_id_async (ezdp_job_id_t *__cmem *jobh_ptr*, uint32_t *budget_id*) [static]

Non blocking version of ezdp_free_job_id().

Parameters:

[in] *jobh_ptr* - pointer to the job id in CMEM
[in] *budget_id* - budget id to charge

Note:

Call ezdp_sync() to wait for the operation to complete, confirming that the job id was read from CMEM and the request was sent to the PMU.

Returns:

void

static __always_inline uint32_t ezdp_read_free_job (void) [static]

The number of jobs available to be obtained.

Returns:

uint32_t - number of available/free jobs

static __always_inline void ezdp_rebudget_job (uint32_t *free_budget_id*, uint32_t *free_num_of_jobs*, uint32_t *alloc_budget_id*, uint32_t *alloc_num_of_jobs*) [static]

Update the budget to which jobs are credited.

Adds free_num_of_indexes credits back into the budget selected by free_budget_id and subtracts alloc_num_of_indexes credits from the budget selected by alloc_budget_id.

Parameters:

[in] *free_budget_id* - budget ID to reimburse
[in] *free_num_of_jobs* - number of jobs to reimburse. Decrement operation (can be zero)
[in] *alloc_budget_id* - budget ID to charge
[in] *alloc_num_of_jobs* - number of jobs to charge. Increment operation (can be zero)

Returns:

none

static __always_inline void ezdp_rebudget_job_async (uint32_t *free_budget_id*, uint32_t *free_num_of_jobs*, uint32_t *alloc_budget_id*, uint32_t *alloc_num_of_jobs*) [static]

Non blocking version of ezdp_rebudget_job().

Parameters:

[in] *free_budget_id* - budget ID to reimburse
[in] *free_num_of_jobs* - number of jobs to reimburse (can be zero)
[in] *alloc_budget_id* - budget ID to charge
[in] *alloc_num_of_jobs* - number of jobs to charge (can be zero)

Returns:

none

static __always_inline void ezdp_load_job (ezdp_job_id_t * *jobh_ptr*, struct ezdp_job_desc __cmem * *jd_ptr*) [static]

Copy the job descriptor from IMEM to CMEM.

Parameters:

[in] *jobh_ptr* - pointer to the job id
[out] *jd_ptr* - pointer to the job descriptor in CMEM

Returns:

none

static __always_inline void ezdp_load_job_async (ezdp_job_id_t * *jobh_ptr*, struct ezdp_job_desc __cmem * *jd_ptr*) [static]

Non blocking version of ezdp_load_job().

Parameters:

[in] *jobh_ptr* - pointer to the job id
[out] *jd_ptr* - pointer to the job descriptor in CMEM

Note:

Call ezdp_sync() to wait for the operation to complete, confirming that the that job descriptor is ready in CMEM.

Returns:

none

static __always_inline void ezdp_store_job (ezdp_job_id_t * *jobh_ptr*, struct ezdp_job_desc __cmem * *jd_ptr*) [static]

Copy the job descriptor from CMEM to IMEM.

Parameters:

[in] *jobh_ptr* - pointer to the job id
[in] *jd_ptr* - pointer to the job descriptor in CMEM

Returns:

none

static __always_inline void ezdp_store_job_async (ezdp_job_id_t * *jobh_ptr*, struct ezdp_job_desc __cmem * *jd_ptr*) [static]

Non blocking version of ezdp_store_job().

Parameters:

[in] *jobh_ptr* - pointer to the job id
[in] *jd_ptr* - pointer to the job descriptor in CMEM

Note:

Call ezdp_sync() to wait for the operation to complete, confirming that the that job descriptor was written to IMEM.

Returns:

none

static __always_inline void ezdp_store_job_container (ezdp_job_id_t * *jobh_ptr*, struct ezdp_job_container_desc __cmem * *jd_ptr*) [static]

Copy the job container descriptor from CMEM to IMEM.

Parameters:

[in] *jobh_ptr* - pointer to the job id
[in] *jd_ptr* - pointer to the job container descriptor in CMEM

Returns:

none

static __always_inline void ezdp_store_job_container_async (ezdp_job_id_t * *jobh_ptr*, struct ezdp_job_container_desc __cmem * *jd_ptr*) [static]

Non blocking version of ezdp_store_job_container().

Parameters:

[in] *jobh_ptr* - pointer to the job id
[in] *jd_ptr* - pointer to the job descriptor in CMEM

Note:

Call ezdp_sync() to wait for the operation to complete, confirming that the that job descriptor was written to IMEM.

Returns:

none

static __always_inline void ezdp_request_job_id (ezdp_job_id_t *__cmem *jobh_ptr*) [static]

Request a new job from the PMU.

Parameters:

[out] *jobh_ptr* - pointer to the job id in CMEM

Note:

- The job request is non blocking. You must call ezdp_wait_for_job_id() to wait for the request to complete before reading the response data in CMEM.

Returns:

none

static __always_inline void ezdp_wait_for_job_id (void) [static]

Suspend execution until a job request completes.

Note:

Should be called after ezdp_request_job_id() to wait for the request to complete. When returning from this routine, the job id is ready in CMEM.

Returns:

none

static __always_inline void ezdp_cancel_job_request (void) [static]

Cancel a job request from the PMU.

Returns:

none

static __always_inline void ezdp_receive_job (ezdp_job_id_t __cmem * *jobh_ptr*, struct ezdp_job_desc __cmem * *jd_ptr*, uint32_t *flags*) [static]

Request a new job from the PMU and load it to CMEM.

Convenience function which requests a new job, waits for the job request to complete, and then loads the job descriptor.

Parameters:

[out] *jobh_ptr* - pointer to the job id in CMEM
[out] *jd_ptr* - pointer to the job descriptor in CMEM
[in] *flags* - execution flags. Bitwise OR of zero or more flags as defined in ezdp_job_flags enum.

Returns:

none

static __always_inline void ezdp_send_job_id_to_queue (ezdp_job_id_t __cmem * *jobh_ptr*, uint32_t *side*, uint32_t *target_queue*, uint32_t *flags*) [static]

Dispatch the job to another PMU queue.

Parameters:

[in] *jobh_ptr* - pointer to the job id in CMEM
[in] *side* - target PMU side
[in] *target_queue* - target PMU queue
[in] *flags* - execution flags. Bitwise OR of zero or more flags as defined in ezdp_job_flags enum.

Returns:

none

static __always_inline void ezdp_send_job_id_to_queue_async (ezdp_job_id_t __cmem * *jobh_ptr*, uint32_t *side*, uint32_t *target_queue*, uint32_t *flags*) [static]

Non blocking version of ezdp_send_job_id_to_queue().

Parameters:

[in] *jobh_ptr* - pointer to the job id in CMEM
[in] *side* - target PMU side
[in] *target_queue* - target PMU queue

[in] *flags* - execution flags. Bitwise OR of zero or more flags as defined in ezdp_job_flags enum.

Note:

Call ezdp_sync() to wait for the operation to complete, confirming that the job id was read from CMEM and the request was sent to the PMU.

Returns:

none

static __always_inline void ezdp_send_job_id_to_tm (ezdp_job_id_t __cmem * *jobh_ptr*, uint32_t *side*, uint32_t *flags*) `[static]`

Transmit the job via the TM.

Return the job back to the PMU and then transmit it via the TM.

Parameters:

[in] *jobh_ptr* - pointer to the job id in CMEM
[in] *side* - target TM side
[in] *flags* - execution flags. Bitwise OR of zero or more flags as defined in ezdp_job_flags enum.

Returns:

none

static __always_inline void ezdp_send_job_id_to_tm_async (ezdp_job_id_t __cmem * *jobh_ptr*, uint32_t *side*, uint32_t *flags*) `[static]`

Non blocking version of ezdp_send_job_id_to_tm().

Parameters:

[in] *jobh_ptr* - pointer to the job id in CMEM
[in] *side* - target TM side
[in] *flags* - execution flags. Bitwise OR of zero or more flags as defined in ezdp_job_flags enum.

Note:

Call ezdp_sync() to wait for the operation to complete, confirming that the job id was read from CMEM and the request was sent to the PMU.

Returns:

none

static __always_inline void ezdp_send_job_id_to_interface (ezdp_job_id_t __cmem * *jobh_ptr*, uint32_t *side*, uint32_t *output_channel*, uint32_t *flags*) `[static]`

Transmit the job directly to an output queue channel, bypassing the TM.

Return the job back to the PMU and then transmit it directly to interface. The interface is selected using output channel and side.

Parameters:

[in] *jobh_ptr* - pointer to the job id in CMEM
[in] *side* - transmission side for job
[in] *output_channel* - output channel in selected side to transmit packet to (define destination port)
[in] *flags* - execution flags. Bitwise OR of zero or more flags as defined in ezdp_job_flags enum.

Returns:

none

static __always_inline void ezdp_send_job_id_to_interface_async (ezdp_job_id_t __cmem * *jobh_ptr*, uint32_t *side*, uint32_t *output_channel*, uint32_t *flags*) [static]

Non blocking version of ezdp_send_job_id_to_interface().

Parameters:

[in] *jobh_ptr* - pointer to the job id in CMEM
[in] *side* - transmission side for job
[in] *output_channel* - output channel in selected side to transmit packet to (define destination port)
[in] *flags* - execution flags. Bitwise OR of zero or more flags as defined in ezdp_job_flags enum.

Note:

Call ezdp_sync() to wait for the operation to complete, confirming that the job id was read from CMEM and the request was sent to the PMU.

Returns:

none

static __always_inline void ezdp_update_job_id_queue (ezdp_job_id_t __cmem * *jobh_ptr*, uint32_t *side*, uint32_t *target_queue*, uint32_t *flags*) [static]

Move the job to another PMU queue without dispatching it.

When called, a request is sent to the PMU to move the job to another queue, thus removing it from the previous ordering and inserting it to a new queue with new ordering. However, the job is not dispatched and is still owned by the calling thread which can continue to work on it.

Parameters:

[in] *jobh_ptr* - pointer to the job id in CMEM
[in] *side* - target PMU side
[in] *target_queue* - target PMU queue
[in] *flags* - execution flags. Bitwise OR of zero or more flags as defined in ezdp_job_flags enum.

Returns:

none

static __always_inline void ezdp_send_job_to_queue (ezdp_job_id_t __cmem * *jobh_ptr*, struct ezdp_job_desc __cmem * *jd_ptr*, uint32_t *side*, uint32_t *target_queue*, uint32_t *flags*) [static]

Store the job descriptor and dispatch the job to another queue in the PMU.

Convenience function which stores the job descriptor and then dispatches the job.

Parameters:

[in] *jobh_ptr* - pointer to the job id in CMEM
[in] *jd_ptr* - pointer to the job descriptor in CMEM
[in] *side* - target PMU side
[in] *target_queue* - target PMU queue
[in] *flags* - execution flags. Bitwise OR of zero or more flags as defined in ezdp_job_flags enum.

Returns:

none

static __always_inline void ezdp_send_job_to_tm (ezdp_job_id_t __cmem * *jobh_ptr*, struct ezdp_job_desc __cmem * *jd_ptr*, uint32_t *side*, uint32_t *flags*) [static]

Store the job descriptor and transmit the job via the TM.

Return the job back to the PMU and then transmit it via the TM. Convenience function which stores the job descriptor and then forwards the job to the TM.

Parameters:

[in] *jobh_ptr* - pointer to the job id in CMEM
[in] *jd_ptr* - pointer to the job descriptor in CMEM
[in] *side* - target TM side
[in] *flags* - execution flags. Bitwise OR of zero or more flags as defined in ezdp_job_flags enum.

Returns:

none

static __always_inline void ezdp_send_job_to_interface (ezdp_job_id_t __cmem * *jobh_ptr*, struct ezdp_job_desc __cmem * *jd_ptr*, uint32_t *side*, uint32_t *output_channel*, uint32_t *flags*) `[static]`

Store the job descriptor and transmit the job directly to interface by bypassing the TM.

Return the job back to the PMU and then transmit directly to output channel. Convenience function which stores the job descriptor and then transmit the job directly to interface. The interface is selected using output channel and side

Parameters:

[in] *jobh_ptr* - pointer to the job id in CMEM
[in] *jd_ptr* - pointer to the job descriptor in CMEM
[in] *side* - the side job should be transmitted
[in] *output_channel* - output channel for transmit; used to map to destination port
[in] *flags* - execution flags. Bitwise OR of zero or more flags as defined in ezdp_job_flags enum.

Returns:

none

static __always_inline void ezdp_update_job_queue (ezdp_job_id_t *__cmem *jobh_ptr*, struct ezdp_job_desc __cmem * *jd_ptr*, uint32_t *side*, uint32_t *target_queue*, uint32_t *flags*) `[static]`

Store the job descriptor and move the job to another PMU queue without dispatching it.

When called, a request is sent to the PMU to move the job to another queue, thus removing it from the previous ordering and inserting it to a new queue with new ordering. However, the job is not dispatched and is still owned by the calling thread which can continue to work on it.

Parameters:

[in] *jobh_ptr* - pointer to the job id in CMEM
[in] *jd_ptr* - pointer to the job descriptor in CMEM
[in] *side* - target PMU side
[in] *target_queue* - target PMU queue
[in] *flags* - execution flags. Bitwise OR of zero or more flags as defined in ezdp_job_flags enum.

Returns:

none

static __always_inline void ezdp_discard_job_id (ezdp_job_id_t *__cmem *jobh_ptr*) `[static]`

Discard a job and all its associated frame resources.

Parameters:

[in] *jobh_ptr* - pointer to the job id in CMEM

Note:

All associated frame resources are recycled. TKB flag is implied to be zero (resources are recycled regardless of FD[TKB] state). Discard job with NULL frame or non valid_data_buf is not supported

Returns:

none

static __always_inline void ezdp_discard_job_id_async (ezdp_job_id_t *__cmem *jobh_ptr*) [static]

Non blocking version of ezdp_discard_job_id().

Parameters:

[in] *jobh_ptr* - pointer to the job id in CMEM

Note:

Call ezdp_sync() to wait for the operation to complete, confirming that the job id was read from CMEM and the request was sent to the PMU.

Returns:

void

static __always_inline void ezdp_discard_job (ezdp_job_id_t __cmem * *jobh_ptr*, struct ezdp_job_desc __cmem * *jd_ptr*) [static]

Store the job descriptor and discard a job and all its associated frame resources.

Convenience function which stores the job descriptor and then forwards the job back to PMU for discarding.

Parameters:

[in] *jobh_ptr* - pointer to the job id in CMEM
[in] *jd_ptr* - pointer to the job descriptor in CMEM

Returns:

none

static __always_inline void ezdp_send_job_id_container (ezdp_job_id_t __cmem * *jobh_ptr*) [static]

Send request to PMU to distribute the job container.

When called, the job container is send to the PMU. It distribute the container job requests while preserving order with other jobs in the queue and recycles the job id after reading all job requests. The job container may contain a link to an extension container, which gets further expanded and recycled in a similar manner.

*

Parameters:

[in] *jobh_ptr* - pointer to the job id in CMEM, which points to the job container

Note:

job_id must point to the job container, which contains multiple jobs

Returns:

none

static __always_inline void ezdp_send_job_id_container_async (ezdp_job_id_t __cmem * *jobh_ptr*) [static]

Non blocking version of ezdp_send_job_id_container().

Parameters:

[in] *jobh_ptr* - pointer to the job id in CMEM, which point to the job container

Note:

Call ezdp_sync() to wait for the operation to complete, confirming that the job id was read from CMEM and the request was sent to the PMU.

Returns:

none

static __always_inline void ezdp_send_job_container (ezdp_job_id_t __cmem * *jobh_ptr*, struct ezdp_job_container_desc __cmem * *jd_container_ptr*) [static]

Store the job container and send request to PMU to distribute it.

Convenience function which stores the job descriptor and then call ezdp_send_job_id_container.

Parameters:

[in] *jobh_ptr* - pointer to the job id in CMEM
[in] *jd_container_ptr* - pointer to the job container in CMEM

Returns:

none

static __always_inline uint32_t ezdp_container_job_count (ezdp_job_container_info_t *info*) [static]

Return the number of the jobs that are in the job container.

Parameters:

[in] *info* - job container info

Returns:

1 - 7

static __always_inline ezdp_job_container_info_t ezdp_container_info (uint32_t *job_count*) [static]

Set the info according to the number of jobs in the job container.

Parameters:

[in] *job_count* - job count in the job container, include extension

Returns:

ezdp_job_container_info_t

static __always_inline void ezdp_notify_cpu (uint32_t *target_cpu_id*) [static]

Notify target CPU.

Turn on notice flag in target CPU and select this CPU available for execution

Parameters:

[in] *target_cpu_id* - The ID of the CPU

Note:

Used to implement IPC

Returns:

none

static __always_inline bool ezdp_notice_pending (void) [static]

Check if there is a new notice.

Returns:

bool - true when having new notice indication

static __always_inline void ezdp_clear_notice (void) [static]

Clear notice indication.

Returns:

none

static __always_inline void ezdp_wait_for_notice (void) [static]

Suspend execution until receiving new notification for CPU.

Note:

New notice indication is cleared automatically

Returns:

none

static __always_inline bool ezdp_check_notice (void) [static]

Check if there is a new notice and clear new notice indication.

Returns:

bool - true when having new notice indication

static __always_inline void ezdp_wait_for_event (void) [static]

Suspend execution until job request completed or new notice received.

Note:

Use ezdp_check_notice or ezdp_notice_pending to check if job or notice received.

Returns:

none

static __always_inline void ezdp_handle_notice (void) [static]

Handle notice.

If there is a new notice, call registered notifier function.

Returns:
none

static __always_inline ezdp_congestion_status_t ezdp_read_congestion_status (enum ezdp_flow_control_node *node_type*, uint32_t *node_id*) [static]

Read priority drop congestion status.

Parameters:
[in] *node_type* - flow control node type
[in] *node_id* - index of the node For the channel nodes: 0 - 255 For the port nodes: 0 - 127 For group nodes: 0 - 15 For global node: 0/value is ignored

Returns:
ezdp_congestion_status_t - according to struct ezdp_congestion_status

static __always_inline ezdp_flow_control_status_t ezdp_read_flow_control_status (enum ezdp_budget_type *budget_type*, enum ezdp_flow_control_node *node_type*, uint32_t *node_id*) [static]

Read flow control status.

Parameters:
[in] *budget_type* - flow control budget type
[in] *node_type* - flow control node type
[in] *node_id* - index of the node For the channel nodes: 0 - 255 For the port nodes: 0 - 127 For group nodes: 0 - 15 For global node: 0/value is ignored

Returns:
ezdp_flow_control_status_t - according to struct ezdp_flow_control_status

static __always_inline enum ezdp_congestion_level ezdp_read_pmu_input_queue_congestion (uint32_t *side*, uint32_t *queue_id*) [static]

Read PMU input queue congestion level.

Parameters:
[in] *side* - PMU side {0,1}
[in] *queue_id* - pmu queue id {0 - 127}

Returns:
enum ezdp_congestion_level - congestion level

static __always_inline uint32_t ezdp_read_global_budget (enum ezdp_budget_type *budget_type*) [static]

Read global budget counter value.

Parameters:

[in] *budget_type* - flow control budget type

Returns:

uint32_t - value of the budget

static __always_inline void ezdp_read_pmu_input_queue_status (uint32_t *side*, uint32_t *queue_id*, struct ezdp_input_queue_status * *queue_status*) [static]

Read PMU input queue status from system info.

Parameters:

[in] *side* - PMU side {0,1}
[in] *queue_id* - pmu queue id {0 - 127}
[in] *queue_status* - point to write queue status into

Returns:

none

static __always_inline void ezdp_read_pmu_tm_output_queue_status (uint32_t *side*, struct ezdp_output_queue_status * *queue_status*) [static]

Read PMU TM output queue status from system info.

Parameters:

[in] *side* - PMU side {0,1}
[in] *queue_status* - point to write queue status into

Returns:

none

static __always_inline void ezdp_read_pmu_discard_output_queue_status (uint32_t *side*, struct ezdp_output_queue_status * *queue_status*) [static]

Read PMU discard output queue status from system info.

Parameters:

[in] *side* - PMU side {0,1}
[in] *queue_status* - point to write queue status into

Returns:

none

static __always_inline void ezdp_read_pmu_tm_bypass_output_queue_status (uint32_t *side*, uint32_t *queue_id*, struct ezdp_output_queue_status * *queue_status*) [static]

Read PMU TM bypass output queue status from system info.

Parameters:

[in] *side* - PMU side {0,1}
[in] *queue_id* - queue id {0-3}
[in] *queue_status* - point to write queue status into

Returns:
none

static __always_inline void ezdp_read_pmu_app_schlr_status (uint32_t *side*, uint32_t *app_schlr_id*, struct ezdp_app_schlr_status * *app_schlr_status*) [static]

Read PMU application scheduler status from system info.

Parameters:
[in] *side* - PMU side {0,1}
[in] *app_schlr_id* - application scheduler id {0-7}
[in] *app_schlr_status* - point to write status into

Returns:
none

static __always_inline void ezdp_read_pmu_group_schlr_status (uint32_t *side*, uint32_t *group_schlr_id*, struct ezdp_group_schlr_status * *group_schlr_status*) [static]

Read PMU group scheduler status from system info.

Parameters:
[in] *side* - PMU side {0,1}
[in] *group_schlr_id* - group scheduler id {0-15}
[in] *group_schlr_status*- point to write status into

Returns:
none

static __always_inline void ezdp_init_tm_reporting_desc (uint32_t *side*, uint32_t *level*, ezdp_tm_queue_depth_desc_t * *tm_desc*, char __cmem * *work_area_ptr*, uint32_t *work_area_size*) [static]

Initialize TM queue depth descriptor.

Parameters:
[in] *side* - TM side (0 / 1)
[in] *level* - TM level (0..4)
[out] *tm_desc* - TM Queue depth descriptor
[in] *work_area_ptr* - work area pointer (temporary memory on CMEM to be used by the function). The size of the temporary memory is determined by EZDP_TM_REPORT_WORK_AREA_SIZE
[in] *work_area_size* - size of work area pointer

Returns:
void

static __always_inline ezdp_tm_queue_depth_handle_t ezdp_calc_tm_queue_depth_handle (uint32_t *queue_id*, ezdp_tm_queue_depth_desc_t * *tm_desc*) [static]

Calculate tm queue depth handle.

Parameters:
[in] *queue_id* - TM Queue-ID

[in] *tm_desc* - TM reporting description

Note:

For better performance, use CP calculation instead of dp calculation

Returns:

ezdp_tm_queue_depth_handle_t

static __always_inline uint32_t ezdp_get_tm_queue_depth (ezdp_tm_queue_depth_handle_t *tm_handle*, enum ezdp_report_size *report_size*) [static]

Get entity queue depth.

Parameters:

[in] *tm_handle* - 32bit TM handle value for queue and level received from CP/DP API
[in] *report_size* - TM reporting size as configured by CP (1B/2B/4B)

Note:

API relevant only if report is enabled for the requested queue, etc valid tm_handle

Returns:

entity queue depth

static __always_inline bool ezdp_valid_tm_queue_depth_handle (ezdp_tm_queue_depth_handle_t *tm_handle*) [static]

validate tm_handle received from CP/DP API

Parameters:

[in] *tm_handle* - 32bit TM handle value for queue and level received from CP/DP API

Returns:

bool - true if tm_handle is valid

dpe/dp/include/ezdp_job_defs.h File Reference

Data Structures

- struct ezdp_job_rx_interface_info
- *Info field for incoming job from external RX interfaces.* struct ezdp_job_rx_loopback_info
- *Info field for incoming job from loopback ports.* struct ezdp_job_rx_confirmation_info
- *Info field for incoming job from TX confirmation ports.* struct ezdp_job_rx_timer_info
- *Info field for incoming timer job (PMU Timer).* struct ezdp_job_rx_user_info
- *Info field for incoming frame job from generic user forwarding.* struct ezdp_job_rx_info
- *Job receive info.* struct ezdp_job_tx_info
- *Info field for transmitting frame job (TM mode is full or tm qos bypass).* struct ezdp_job_queue_cmd_info
- *Job container send to queue request info.* struct ezdp_job_transmit_cmd_info
- *Job container send out request info.* struct ezdp_job_discard_cmd_info
- *Job container discard request info.* struct ezdp_congestion_status
- *System priority drop congestion status.* struct ezdp_flow_control_status
- *Flow control status.* struct ezdp_input_queue_status
- *PMU physical input queue status definition (based on PMU system info).* struct ezdp_output_queue_status
- *PMU output queue status definition (based on PMU system info).* struct ezdp_app_schlr_status
- *PMU application scheduler status (based on PMU system info).* struct ezdp_group_schlr_status
- *PMU group scheduler status (based on PMU system info).* struct ezdp_job_container_cmd_desc
- *Job container request.* struct ezdp_job_container_desc
- *Job container descriptor.* struct ezdp_job_desc

job descriptor data structure Defines

- #define EZDP_TM_REPORT_WORK_AREA_SIZE sizeof(struct ezdp_init_tm_report_wa)
- *Work area minimal required size definitions.* #define EZDP_JOB_RX_INTERFACE_INFO_TIMESTAMP_SEC_SIZE 8
- #define EZDP_JOB_RX_INTERFACE_INFO_TIMESTAMP_SEC_OFFSET 0
- #define EZDP_JOB_RX_INTERFACE_INFO_TIMESTAMP_SEC_WORD_SELECT 0
- #define EZDP_JOB_RX_INTERFACE_INFO_TIMESTAMP_SEC_WORD_OFFSET 0
- #define EZDP_JOB_RX_INTERFACE_INFO_CRC_CHECKED_FLAG_SIZE 1
- #define EZDP_JOB_RX_INTERFACE_INFO_CRC_CHECKED_FLAG_OFFSET 8
- #define EZDP_JOB_RX_INTERFACE_INFO_CRC_CHECKED_FLAG_WORD_SELECT 0
- #define EZDP_JOB_RX_INTERFACE_INFO_CRC_CHECKED_FLAG_WORD_OFFSET 8
- #define EZDP_JOB_RX_INTERFACE_INFO_CRC_CHECKED_FLAG_MASK (1 << EZDP_JOB_RX_INTERFACE_INFO_CRC_CHECKED_FLAG_WORD_OFFSET)
- #define EZDP_JOB_RX_INTERFACE_INFO_CRC_OK_FLAG_SIZE 1
- #define EZDP_JOB_RX_INTERFACE_INFO_CRC_OK_FLAG_OFFSET 9
- #define EZDP_JOB_RX_INTERFACE_INFO_CRC_OK_FLAG_WORD_SELECT 0
- #define EZDP_JOB_RX_INTERFACE_INFO_CRC_OK_FLAG_WORD_OFFSET 9
- #define EZDP_JOB_RX_INTERFACE_INFO_CRC_OK_FLAG_MASK (1 << EZDP_JOB_RX_INTERFACE_INFO_CRC_OK_FLAG_WORD_OFFSET)
- #define EZDP_JOB_RX_INTERFACE_INFO_RESERVED10_SIZE 1
- #define EZDP_JOB_RX_INTERFACE_INFO_RESERVED10_OFFSET 10
- #define EZDP_JOB_RX_INTERFACE_INFO_RESERVED11_SIZE 1
- #define EZDP_JOB_RX_INTERFACE_INFO_RESERVED11_OFFSET 11
- #define EZDP_JOB_RX_INTERFACE_INFO_TRUNCATION_FLAG_SIZE 1
- #define EZDP_JOB_RX_INTERFACE_INFO_TRUNCATION_FLAG_OFFSET 12
- #define EZDP_JOB_RX_INTERFACE_INFO_TRUNCATION_FLAG_WORD_SELECT 0
- #define EZDP_JOB_RX_INTERFACE_INFO_TRUNCATION_FLAG_WORD_OFFSET 12
- #define EZDP_JOB_RX_INTERFACE_INFO_TRUNCATION_FLAG_MASK (1 << EZDP_JOB_RX_INTERFACE_INFO_TRUNCATION_FLAG_WORD_OFFSET)
- #define EZDP_JOB_RX_INTERFACE_INFO_ICU_SUCC_PARSING_FLAG_SIZE 1
- #define EZDP_JOB_RX_INTERFACE_INFO_ICU_SUCC_PARSING_FLAG_OFFSET 13
- #define EZDP_JOB_RX_INTERFACE_INFO_ICU_SUCC_PARSING_FLAG_WORD_SELECT 0

- #define EZDP_JOB_RX_INTERFACE_INFO_ICU_SUCC_PARSING_FLAG_WORD_OFFSET 13
- #define EZDP_JOB_RX_INTERFACE_INFO_ICU_SUCC_PARSING_FLAG_MASK (1 << EZDP_JOB_RX_INTERFACE_INFO_ICU_SUCC_PARSING_FLAG_WORD_OFFSET)
- #define EZDP_JOB_RX_INTERFACE_INFO_RESERVED14_SIZE 1
- #define EZDP_JOB_RX_INTERFACE_INFO_RESERVED14_OFFSET 14
- #define EZDP_JOB_RX_INTERFACE_INFO_RESERVED15_SIZE 1
- #define EZDP_JOB_RX_INTERFACE_INFO_RESERVED15_OFFSET 15
- #define EZDP_JOB_RX_INTERFACE_INFO_IMEM_BUF_COUNT_SIZE 6
- #define EZDP_JOB_RX_INTERFACE_INFO_IMEM_BUF_COUNT_OFFSET 16
- #define EZDP_JOB_RX_INTERFACE_INFO_IMEM_BUF_COUNT_WORD_SELECT 0
- #define EZDP_JOB_RX_INTERFACE_INFO_IMEM_BUF_COUNT_WORD_OFFSET 16
- #define EZDP_JOB_RX_INTERFACE_INFO_GLOBAL_CONGESTION_LEVEL_SIZE 2
- #define EZDP_JOB_RX_INTERFACE_INFO_GLOBAL_CONGESTION_LEVEL_OFFSET 22
- #define EZDP_JOB_RX_INTERFACE_INFO_GLOBAL_CONGESTION_LEVEL_WORD_SELECT 0
- #define EZDP_JOB_RX_INTERFACE_INFO_GLOBAL_CONGESTION_LEVEL_WORD_OFFSET 22
- #define EZDP_JOB_RX_INTERFACE_INFO_IMEM_BUF_CONGESTION_LEVEL_SIZE 2
- #define EZDP_JOB_RX_INTERFACE_INFO_IMEM_BUF_CONGESTION_LEVEL_OFFSET 24
- #define EZDP_JOB_RX_INTERFACE_INFO_IMEM_BUF_CONGESTION_LEVEL_WORD_SELECT 0
- #define EZDP_JOB_RX_INTERFACE_INFO_IMEM_BUF_CONGESTION_LEVEL_WORD_OFFSET 24
- #define EZDP_JOB_RX_INTERFACE_INFO_EMEM_BUF_CONGESTION_LEVEL_SIZE 2
- #define EZDP_JOB_RX_INTERFACE_INFO_EMEM_BUF_CONGESTION_LEVEL_OFFSET 26
- #define EZDP_JOB_RX_INTERFACE_INFO_EMEM_BUF_CONGESTION_LEVEL_WORD_SELECT 0
- #define EZDP_JOB_RX_INTERFACE_INFO_EMEM_BUF_CONGESTION_LEVEL_WORD_OFFSET 26
- #define EZDP_JOB_RX_INTERFACE_INFO_JOB_CONGESTION_LEVEL_SIZE 2
- #define EZDP_JOB_RX_INTERFACE_INFO_JOB_CONGESTION_LEVEL_OFFSET 28
- #define EZDP_JOB_RX_INTERFACE_INFO_JOB_CONGESTION_LEVEL_WORD_SELECT 0
- #define EZDP_JOB_RX_INTERFACE_INFO_JOB_CONGESTION_LEVEL_WORD_OFFSET 28
- #define EZDP_JOB_RX_INTERFACE_INFO_PMU_QUEUE_CONGESTION_LEVEL_SIZE 2
- #define EZDP_JOB_RX_INTERFACE_INFO_PMU_QUEUE_CONGESTION_LEVEL_OFFSET 30
- #define EZDP_JOB_RX_INTERFACE_INFO_PMU_QUEUE_CONGESTION_LEVEL_WORD_SELECT 0
- #define EZDP_JOB_RX_INTERFACE_INFO_PMU_QUEUE_CONGESTION_LEVEL_WORD_OFFSET 30
- #define EZDP_JOB_RX_INTERFACE_INFO_TIMESTAMP_NSEC_SIZE 32
- #define EZDP_JOB_RX_INTERFACE_INFO_TIMESTAMP_NSEC_OFFSET 32
- #define EZDP_JOB_RX_INTERFACE_INFO_TIMESTAMP_NSEC_WORD_SELECT 1
- #define EZDP_JOB_RX_INTERFACE_INFO_TIMESTAMP_NSEC_WORD_OFFSET 0
- #define EZDP_JOB_RX_INTERFACE_INFO_WORD_COUNT 2
- #define EZDP_JOB_RX_LOOPBACK_INFO_RESERVED0_15_SIZE 16
- #define EZDP_JOB_RX_LOOPBACK_INFO_RESERVED0_15_OFFSET 0
- #define EZDP_JOB_RX_LOOPBACK_INFO_REPLICATION_ID_SIZE 16
- #define EZDP_JOB_RX_LOOPBACK_INFO_REPLICATION_ID_OFFSET 16
- #define EZDP_JOB_RX_LOOPBACK_INFO_REPLICATION_ID_WORD_SELECT 0
- #define EZDP_JOB_RX_LOOPBACK_INFO_REPLICATION_ID_WORD_OFFSET 16
- #define EZDP_JOB_RX_LOOPBACK_INFO_RESERVED32_63_SIZE 32
- #define EZDP_JOB_RX_LOOPBACK_INFO_RESERVED32_63_OFFSET 32
- #define EZDP_JOB_RX_LOOPBACK_INFO_WORD_COUNT 2
- #define EZDP_JOB_RX_CONFIRMATION_INFO_TIMESTAMP_SEC_SIZE 8
- #define EZDP_JOB_RX_CONFIRMATION_INFO_TIMESTAMP_SEC_OFFSET 0
- #define EZDP_JOB_RX_CONFIRMATION_INFO_TIMESTAMP_SEC_WORD_SELECT 0
- #define EZDP_JOB_RX_CONFIRMATION_INFO_TIMESTAMP_SEC_WORD_OFFSET 0
- #define EZDP_JOB_RX_CONFIRMATION_INFO_RESERVED8_31_SIZE 24
- #define EZDP_JOB_RX_CONFIRMATION_INFO_RESERVED8_31_OFFSET 8
- #define EZDP_JOB_RX_CONFIRMATION_INFO_TIMESTAMP_NSEC_SIZE 32
- #define EZDP_JOB_RX_CONFIRMATION_INFO_TIMESTAMP_NSEC_OFFSET 32
- #define EZDP_JOB_RX_CONFIRMATION_INFO_TIMESTAMP_NSEC_WORD_SELECT 1
- #define EZDP_JOB_RX_CONFIRMATION_INFO_TIMESTAMP_NSEC_WORD_OFFSET 0
- #define EZDP_JOB_RX_CONFIRMATION_INFO_WORD_COUNT 2

- #define EZDP_JOB_RX_TIMER_INFO_RESERVED0_8_SIZE 8
- #define EZDP_JOB_RX_TIMER_INFO_RESERVED0_8_OFFSET 0
- #define EZDP_JOB_RX_TIMER_INFO_TIMER_ID_SIZE 8
- #define EZDP_JOB_RX_TIMER_INFO_TIMER_ID_OFFSET 8
- #define EZDP_JOB_RX_TIMER_INFO_TIMER_ID_WORD_SELECT 0
- #define EZDP_JOB_RX_TIMER_INFO_TIMER_ID_WORD_OFFSET 8
- #define EZDP_JOB_RX_TIMER_INFO_RESERVED16_31_SIZE 16
- #define EZDP_JOB_RX_TIMER_INFO_RESERVED16_31_OFFSET 16
- #define EZDP_JOB_RX_TIMER_INFO_EVENT_ID_SIZE 32
- #define EZDP_JOB_RX_TIMER_INFO_EVENT_ID_OFFSET 32
- #define EZDP_JOB_RX_TIMER_INFO_EVENT_ID_WORD_SELECT 1
- #define EZDP_JOB_RX_TIMER_INFO_EVENT_ID_WORD_OFFSET 0
- #define EZDP_JOB_RX_TIMER_INFO_WORD_COUNT 2
- #define EZDP_JOB_RX_USER_INFO_USER_DATA_INFO0_SIZE 32
- #define EZDP_JOB_RX_USER_INFO_USER_DATA_INFO0_OFFSET 0
- #define EZDP_JOB_RX_USER_INFO_USER_DATA_INFO0_WORD_SELECT 0
- #define EZDP_JOB_RX_USER_INFO_USER_DATA_INFO0_WORD_OFFSET 0
- #define EZDP_JOB_RX_USER_INFO_USER_DATA_INFO1_SIZE 32
- #define EZDP_JOB_RX_USER_INFO_USER_DATA_INFO1_OFFSET 32
- #define EZDP_JOB_RX_USER_INFO_USER_DATA_INFO1_WORD_SELECT 1
- #define EZDP_JOB_RX_USER_INFO_USER_DATA_INFO1_WORD_OFFSET 0
- #define EZDP_JOB_RX_USER_INFO_WORD_COUNT 2
- #define EZDP_JOB_RX_INFO_GROSS_CHECKSUM_SIZE 16
- #define EZDP_JOB_RX_INFO_GROSS_CHECKSUM_OFFSET 64
- #define EZDP_JOB_RX_INFO_GROSS_CHECKSUM_WORD_SELECT 2
- #define EZDP_JOB_RX_INFO_GROSS_CHECKSUM_WORD_OFFSET 0
- #define EZDP_JOB_RX_INFO_RESERVED112_127_SIZE 16
- #define EZDP_JOB_RX_INFO_RESERVED112_127_OFFSET 80
- #define EZDP_JOB_RX_INFO_SOURCE_QUEUE_SIZE 7
- #define EZDP_JOB_RX_INFO_SOURCE_QUEUE_OFFSET 96
- #define EZDP_JOB_RX_INFO_SOURCE_QUEUE_WORD_SELECT 3
- #define EZDP_JOB_RX_INFO_SOURCE_QUEUE_WORD_OFFSET 0
- #define EZDP_JOB_RX_INFO_SIDE_SIZE 1
- #define EZDP_JOB_RX_INFO_SIDE_OFFSET 103
- #define EZDP_JOB_RX_INFO_SIDE_WORD_SELECT 3
- #define EZDP_JOB_RX_INFO_SIDE_WORD_OFFSET 7
- #define EZDP_JOB_RX_INFO_SIDE_MASK (1 << EZDP_JOB_RX_INFO_SIDE_WORD_OFFSET)
- #define EZDP_JOB_RX_INFO_RESERVED104_107_SIZE 4
- #define EZDP_JOB_RX_INFO_RESERVED104_107_OFFSET 104
- #define EZDP_JOB_RX_INFO_RESERVED108_109_SIZE 2
- #define EZDP_JOB_RX_INFO_RESERVED108_109_OFFSET 108
- #define EZDP_JOB_RX_INFO_SEQ_NUMBER_VALID_SIZE 1
- #define EZDP_JOB_RX_INFO_SEQ_NUMBER_VALID_OFFSET 110
- #define EZDP_JOB_RX_INFO_SEQ_NUMBER_VALID_WORD_SELECT 3
- #define EZDP_JOB_RX_INFO_SEQ_NUMBER_VALID_WORD_OFFSET 14
- #define EZDP_JOB_RX_INFO_SEQ_NUMBER_VALID_MASK (1 << EZDP_JOB_RX_INFO_SEQ_NUMBER_VALID_WORD_OFFSET)
- #define EZDP_JOB_RX_INFO_IS_SERVICE_READY_SIZE 1
- #define EZDP_JOB_RX_INFO_IS_SERVICE_READY_OFFSET 111
- #define EZDP_JOB_RX_INFO_IS_SERVICE_READY_WORD_SELECT 3
- #define EZDP_JOB_RX_INFO_IS_SERVICE_READY_WORD_OFFSET 15
- #define EZDP_JOB_RX_INFO_IS_SERVICE_READY_MASK (1 << EZDP_JOB_RX_INFO_IS_SERVICE_READY_WORD_OFFSET)
- #define EZDP_JOB_RX_INFO_SEQ_NUMBER_SIZE 16
- #define EZDP_JOB_RX_INFO_SEQ_NUMBER_OFFSET 112
- #define EZDP_JOB_RX_INFO_SEQ_NUMBER_WORD_SELECT 3
- #define EZDP_JOB_RX_INFO_SEQ_NUMBER_WORD_OFFSET 16

- #define EZDP_JOB_RX_INFO_WORD_COUNT 4
- #define EZDP_JOB_TX_INFO_PACKET_SWITCH_ID_SELECT_SIZE 9
- #define EZDP_JOB_TX_INFO_PACKET_SWITCH_ID_SELECT_OFFSET 16
- #define EZDP_JOB_TX_INFO_PACKET_SWITCH_ID_SELECT_WORD_SELECT 0
- #define EZDP_JOB_TX_INFO_PACKET_SWITCH_ID_SELECT_WORD_OFFSET 16
- #define EZDP_JOB_TX_INFO_RESERVED25_29_SIZE 4
- #define EZDP_JOB_TX_INFO_RESERVED25_29_OFFSET 25
- #define EZDP_JOB_TX_INFO_WRED_COLOR_SIZE 3
- #define EZDP_JOB_TX_INFO_WRED_COLOR_OFFSET 29
- #define EZDP_JOB_TX_INFO_WRED_COLOR_WORD_SELECT 0
- #define EZDP_JOB_TX_INFO_WRED_COLOR_WORD_OFFSET 29
- #define EZDP_JOB_TX_INFO_FLOW_ID_SIZE 19
- #define EZDP_JOB_TX_INFO_FLOW_ID_OFFSET 32
- #define EZDP_JOB_TX_INFO_FLOW_ID_WORD_SELECT 1
- #define EZDP_JOB_TX_INFO_FLOW_ID_WORD_OFFSET 0
- #define EZDP_JOB_TX_INFO_SIDE_SIZE 1
- #define EZDP_JOB_TX_INFO_SIDE_OFFSET 51
- #define EZDP_JOB_TX_INFO_SIDE_WORD_SELECT 1
- #define EZDP_JOB_TX_INFO_SIDE_WORD_OFFSET 19
- #define EZDP_JOB_TX_INFO_SIDE_MASK (1 << EZDP_JOB_TX_INFO_SIDE_WORD_OFFSET)
- #define EZDP_JOB_TX_INFO_RESERVED52_55_SIZE 4
- #define EZDP_JOB_TX_INFO_RESERVED52_55_OFFSET 52
- #define EZDP_JOB_TX_INFO_PACKET_SWITCH_MODE_SIZE 3
- #define EZDP_JOB_TX_INFO_PACKET_SWITCH_MODE_OFFSET 56
- #define EZDP_JOB_TX_INFO_PACKET_SWITCH_MODE_WORD_SELECT 1
- #define EZDP_JOB_TX_INFO_PACKET_SWITCH_MODE_WORD_OFFSET 24
- #define EZDP_JOB_TX_INFO_RESERVED59_SIZE 1
- #define EZDP_JOB_TX_INFO_RESERVED59_OFFSET 59
- #define EZDP_JOB_TX_INFO_QOS_BYPASS_SIZE 1
- #define EZDP_JOB_TX_INFO_QOS_BYPASS_OFFSET 60
- #define EZDP_JOB_TX_INFO_QOS_BYPASS_WORD_SELECT 1
- #define EZDP_JOB_TX_INFO_QOS_BYPASS_WORD_OFFSET 28
- #define EZDP_JOB_TX_INFO_QOS_BYPASS_MASK (1 << EZDP_JOB_TX_INFO_QOS_BYPASS_WORD_OFFSET)
- #define EZDP_JOB_TX_INFO_RESERVED61_SIZE 1
- #define EZDP_JOB_TX_INFO_RESERVED61_OFFSET 61
- #define EZDP_JOB_TX_INFO_DROP_MODE_SIZE 2
- #define EZDP_JOB_TX_INFO_DROP_MODE_OFFSET 62
- #define EZDP_JOB_TX_INFO_DROP_MODE_WORD_SELECT 1
- #define EZDP_JOB_TX_INFO_DROP_MODE_WORD_OFFSET 30
- #define EZDP_JOB_TX_INFO_STAT_STREAM_ID_SIZE 24
- #define EZDP_JOB_TX_INFO_STAT_STREAM_ID_OFFSET 64
- #define EZDP_JOB_TX_INFO_STAT_STREAM_ID_WORD_SELECT 2
- #define EZDP_JOB_TX_INFO_STAT_STREAM_ID_WORD_OFFSET 0
- #define EZDP_JOB_TX_INFO_RESERVED88_90_SIZE 2
- #define EZDP_JOB_TX_INFO_RESERVED88_90_OFFSET 88
- #define EZDP_JOB_TX_INFO_STAT_CODE_PROFILE2_SIZE 3
- #define EZDP_JOB_TX_INFO_STAT_CODE_PROFILE2_OFFSET 90
- #define EZDP_JOB_TX_INFO_STAT_CODE_PROFILE2_WORD_SELECT 2
- #define EZDP_JOB_TX_INFO_STAT_CODE_PROFILE2_WORD_OFFSET 26
- #define EZDP_JOB_TX_INFO_STAT_CODE_PROFILE1_SIZE 3
- #define EZDP_JOB_TX_INFO_STAT_CODE_PROFILE1_OFFSET 93
- #define EZDP_JOB_TX_INFO_STAT_CODE_PROFILE1_WORD_SELECT 2
- #define EZDP_JOB_TX_INFO_STAT_CODE_PROFILE1_WORD_OFFSET 29
- #define EZDP_JOB_TX_INFO_INTER_PACKET_GAP_SIZE 5
- #define EZDP_JOB_TX_INFO_INTER_PACKET_GAP_OFFSET 96
- #define EZDP_JOB_TX_INFO_INTER_PACKET_GAP_WORD_SELECT 3

- #define EZDP_JOB_TX_INFO_INTER_PACKET_GAP_WORD_OFFSET 0
- #define EZDP_JOB_TX_INFO_INTER_PACKET_GAP_CONTROL_SIZE 1
- #define EZDP_JOB_TX_INFO_INTER_PACKET_GAP_CONTROL_OFFSET 101
- #define EZDP_JOB_TX_INFO_INTER_PACKET_GAP_CONTROL_WORD_SELECT 3
- #define EZDP_JOB_TX_INFO_INTER_PACKET_GAP_CONTROL_WORD_OFFSET 5
- #define EZDP_JOB_TX_INFO_INTER_PACKET_GAP_CONTROL_MASK (1 << EZDP_JOB_TX_INFO_INTER_PACKET_GAP_CONTROL_WORD_OFFSET)
- #define EZDP_JOB_TX_INFO_RESERVED102_103_SIZE 2
- #define EZDP_JOB_TX_INFO_RESERVED102_103_OFFSET 102
- #define EZDP_JOB_TX_INFO_WRED_CLASS_SCALE_PROFILE_SIZE 8
- #define EZDP_JOB_TX_INFO_WRED_CLASS_SCALE_PROFILE_OFFSET 104
- #define EZDP_JOB_TX_INFO_WRED_CLASS_SCALE_PROFILE_WORD_SELECT 3
- #define EZDP_JOB_TX_INFO_WRED_CLASS_SCALE_PROFILE_WORD_OFFSET 8
- #define EZDP_JOB_TX_INFO_WRED_FLOW_SCALE_PROFILE_SIZE 8
- #define EZDP_JOB_TX_INFO_WRED_FLOW_SCALE_PROFILE_OFFSET 112
- #define EZDP_JOB_TX_INFO_WRED_FLOW_SCALE_PROFILE_WORD_SELECT 3
- #define EZDP_JOB_TX_INFO_WRED_FLOW_SCALE_PROFILE_WORD_OFFSET 16
- #define EZDP_JOB_TX_INFO_WRED_CLASS_TEMPLATE_PROFILE_SIZE 4
- #define EZDP_JOB_TX_INFO_WRED_CLASS_TEMPLATE_PROFILE_OFFSET 120
- #define EZDP_JOB_TX_INFO_WRED_CLASS_TEMPLATE_PROFILE_WORD_SELECT 3
- #define EZDP_JOB_TX_INFO_WRED_CLASS_TEMPLATE_PROFILE_WORD_OFFSET 24
- #define EZDP_JOB_TX_INFO_WRED_FLOW_TEMPLATE_PROFILE_SIZE 4
- #define EZDP_JOB_TX_INFO_WRED_FLOW_TEMPLATE_PROFILE_OFFSET 124
- #define EZDP_JOB_TX_INFO_WRED_FLOW_TEMPLATE_PROFILE_WORD_SELECT 3
- #define EZDP_JOB_TX_INFO_WRED_FLOW_TEMPLATE_PROFILE_WORD_OFFSET 28
- #define EZDP_JOB_TX_INFO_WORD_COUNT 4
- #define EZDP_JOB_QUEUE_CMD_INFO_TARGET_QUEUE_SIZE 7
- #define EZDP_JOB_QUEUE_CMD_INFO_TARGET_QUEUE_OFFSET 0
- #define EZDP_JOB_QUEUE_CMD_INFO_SIDE_SIZE 1
- #define EZDP_JOB_QUEUE_CMD_INFO_SIDE_OFFSET 7
- #define EZDP_JOB_QUEUE_CMD_INFO_SIDE_MASK (1 << EZDP_JOB_QUEUE_CMD_INFO_SIDE_OFFSET)
- #define EZDP_JOB_QUEUE_CMD_INFO_RESERVED8_15_SIZE 8
- #define EZDP_JOB_QUEUE_CMD_INFO_RESERVED8_15_OFFSET 8
- #define EZDP_JOB_TRANSMIT_CMD_INFO_OUTPUT_CHANNEL_SIZE 10
- #define EZDP_JOB_TRANSMIT_CMD_INFO_OUTPUT_CHANNEL_OFFSET 0
- #define EZDP_JOB_TRANSMIT_CMD_INFO_SIDE_SIZE 1
- #define EZDP_JOB_TRANSMIT_CMD_INFO_SIDE_OFFSET 10
- #define EZDP_JOB_TRANSMIT_CMD_INFO_SIDE_MASK (1 << EZDP_JOB_TRANSMIT_CMD_INFO_SIDE_OFFSET)
- #define EZDP_JOB_TRANSMIT_CMD_INFO_DESTINATION_SIZE 1
- #define EZDP_JOB_TRANSMIT_CMD_INFO_DESTINATION_OFFSET 11
- #define EZDP_JOB_TRANSMIT_CMD_INFO_DESTINATION_MASK (1 << EZDP_JOB_TRANSMIT_CMD_INFO_DESTINATION_OFFSET)
- #define EZDP_JOB_TRANSMIT_CMD_INFO_RESERVED12_15_SIZE 4
- #define EZDP_JOB_TRANSMIT_CMD_INFO_RESERVED12_15_OFFSET 12
- #define EZDP_JOB_DISCARD_CMD_INFO_RESERVED0_9_SIZE 10
- #define EZDP_JOB_DISCARD_CMD_INFO_RESERVED0_9_OFFSET 0
- #define EZDP_JOB_DISCARD_CMD_INFO_SIDE_SIZE 1
- #define EZDP_JOB_DISCARD_CMD_INFO_SIDE_OFFSET 10
- #define EZDP_JOB_DISCARD_CMD_INFO_SIDE_MASK (1 << EZDP_JOB_DISCARD_CMD_INFO_SIDE_OFFSET)
- #define EZDP_JOB_DISCARD_CMD_INFO_RESERVED11_15_SIZE 5
- #define EZDP_JOB_DISCARD_CMD_INFO_RESERVED11_15_OFFSET 11
- #define EZDP_CONGESTION_STATUS_IMEM_BUF_CONGESTION_LEVEL_SIZE 2
- #define EZDP_CONGESTION_STATUS_IMEM_BUF_CONGESTION_LEVEL_OFFSET 0
- #define EZDP_CONGESTION_STATUS_IMEM_BUF_GUARANTEE_SIZE 1

- #define EZDP_CONGESTION_STATUS_IMEM_BUF_GUARANTEE_OFFSET 2
- #define EZDP_CONGESTION_STATUS_IMEM_BUF_GUARANTEE_MASK (1 << EZDP_CONGESTION_STATUS_IMEM_BUF_GUARANTEE_OFFSET)
- #define EZDP_CONGESTION_STATUS_RESERVED3_SIZE 1
- #define EZDP_CONGESTION_STATUS_RESERVED3_OFFSET 3
- #define EZDP_CONGESTION_STATUS_EMEM_BUF_CONGESTION_LEVEL_SIZE 2
- #define EZDP_CONGESTION_STATUS_EMEM_BUF_CONGESTION_LEVEL_OFFSET 4
- #define EZDP_CONGESTION_STATUS_EMEM_BUF_GUARANTEE_SIZE 1
- #define EZDP_CONGESTION_STATUS_EMEM_BUF_GUARANTEE_OFFSET 6
- #define EZDP_CONGESTION_STATUS_EMEM_BUF_GUARANTEE_MASK (1 << EZDP_CONGESTION_STATUS_EMEM_BUF_GUARANTEE_OFFSET)
- #define EZDP_CONGESTION_STATUS_RESERVED7_SIZE 1
- #define EZDP_CONGESTION_STATUS_RESERVED7_OFFSET 7
- #define EZDP_CONGESTION_STATUS_JOB_CONGESTION_LEVEL_SIZE 2
- #define EZDP_CONGESTION_STATUS_JOB_CONGESTION_LEVEL_OFFSET 8
- #define EZDP_CONGESTION_STATUS_JOB_GUARANTEE_SIZE 1
- #define EZDP_CONGESTION_STATUS_JOB_GUARANTEE_OFFSET 10
- #define EZDP_CONGESTION_STATUS_JOB_GUARANTEE_MASK (1 << EZDP_CONGESTION_STATUS_JOB_GUARANTEE_OFFSET)
- #define EZDP_CONGESTION_STATUS_RESERVED11_SIZE 1
- #define EZDP_CONGESTION_STATUS_RESERVED11_OFFSET 11
- #define EZDP_CONGESTION_STATUS_PORT_CONGESTION_LEVEL_SIZE 2
- #define EZDP_CONGESTION_STATUS_PORT_CONGESTION_LEVEL_OFFSET 12
- #define EZDP_CONGESTION_STATUS_RESERVED14_15_SIZE 2
- #define EZDP_CONGESTION_STATUS_RESERVED14_15_OFFSET 14
- #define EZDP_FLOW_CONTROL_STATUS_CONGESTION_LEVEL_SIZE 3
- #define EZDP_FLOW_CONTROL_STATUS_CONGESTION_LEVEL_OFFSET 0
- #define EZDP_FLOW_CONTROL_STATUS_ENABLE_SIZE 1
- #define EZDP_FLOW_CONTROL_STATUS_ENABLE_OFFSET 3
- #define EZDP_FLOW_CONTROL_STATUS_ENABLE_MASK (1 << EZDP_FLOW_CONTROL_STATUS_ENABLE_OFFSET)
- #define EZDP_FLOW_CONTROL_STATUS_RESERVED4_7_SIZE 4
- #define EZDP_FLOW_CONTROL_STATUS_RESERVED4_7_OFFSET 4
- #define EZDP_INPUT_QUEUE_STATUS_DISPATCHED_JOB_SIZE 16
- #define EZDP_INPUT_QUEUE_STATUS_DISPATCHED_JOB_OFFSET 0
- #define EZDP_INPUT_QUEUE_STATUS_DISPATCHED_JOB_WORD_SELECT 0
- #define EZDP_INPUT_QUEUE_STATUS_DISPATCHED_JOB_WORD_OFFSET 0
- #define EZDP_INPUT_QUEUE_STATUS_CONGESTION_LEVEL_SIZE 2
- #define EZDP_INPUT_QUEUE_STATUS_CONGESTION_LEVEL_OFFSET 16
- #define EZDP_INPUT_QUEUE_STATUS_CONGESTION_LEVEL_WORD_SELECT 0
- #define EZDP_INPUT_QUEUE_STATUS_CONGESTION_LEVEL_WORD_OFFSET 16
- #define EZDP_INPUT_QUEUE_STATUS_READY_SIZE 1
- #define EZDP_INPUT_QUEUE_STATUS_READY_OFFSET 18
- #define EZDP_INPUT_QUEUE_STATUS_READY_WORD_SELECT 0
- #define EZDP_INPUT_QUEUE_STATUS_READY_WORD_OFFSET 18
- #define EZDP_INPUT_QUEUE_STATUS_READY_MASK (1 << EZDP_INPUT_QUEUE_STATUS_READY_WORD_OFFSET)
- #define EZDP_INPUT_QUEUE_STATUS_RESERVED19_31_SIZE 13
- #define EZDP_INPUT_QUEUE_STATUS_RESERVED19_31_OFFSET 19
- #define EZDP_INPUT_QUEUE_STATUS_OUTSTANDING_JOB_SIZE 16
- #define EZDP_INPUT_QUEUE_STATUS_OUTSTANDING_JOB_OFFSET 32
- #define EZDP_INPUT_QUEUE_STATUS_OUTSTANDING_JOB_WORD_SELECT 1
- #define EZDP_INPUT_QUEUE_STATUS_OUTSTANDING_JOB_WORD_OFFSET 0
- #define EZDP_INPUT_QUEUE_STATUS_SIZE_SIZE 16
- #define EZDP_INPUT_QUEUE_STATUS_SIZE_OFFSET 48
- #define EZDP_INPUT_QUEUE_STATUS_SIZE_WORD_SELECT 1
- #define EZDP_INPUT_QUEUE_STATUS_SIZE_WORD_OFFSET 16

- #define EZDP_INPUT_QUEUE_STATUS_WORD_COUNT 2
- #define EZDP_OUTPUT_QUEUE_STATUS_SIZE_SIZE 16
- #define EZDP_OUTPUT_QUEUE_STATUS_SIZE_OFFSET 0
- #define EZDP_OUTPUT_QUEUE_STATUS_READY_SIZE 1
- #define EZDP_OUTPUT_QUEUE_STATUS_READY_OFFSET 16
- #define EZDP_OUTPUT_QUEUE_STATUS_READY_MASK (1 << EZDP_OUTPUT_QUEUE_STATUS_READY_OFFSET)
- #define EZDP_OUTPUT_QUEUE_STATUS_CONGESTION_SIZE 1
- #define EZDP_OUTPUT_QUEUE_STATUS_CONGESTION_OFFSET 17
- #define EZDP_OUTPUT_QUEUE_STATUS_CONGESTION_MASK (1 << EZDP_OUTPUT_QUEUE_STATUS_CONGESTION_OFFSET)
- #define EZDP_OUTPUT_QUEUE_STATUS_RESERVED18_31_SIZE 14
- #define EZDP_OUTPUT_QUEUE_STATUS_RESERVED18_31_OFFSET 18
- #define EZDP_APP_SCHLR_STATUS_DISPATCHED_JOB_SIZE 13
- #define EZDP_APP_SCHLR_STATUS_DISPATCHED_JOB_OFFSET 0
- #define EZDP_APP_SCHLR_STATUS_RESERVED13_SIZE 1
- #define EZDP_APP_SCHLR_STATUS_RESERVED13_OFFSET 13
- #define EZDP_APP_SCHLR_STATUS_BUSY_SIZE 1
- #define EZDP_APP_SCHLR_STATUS_BUSY_OFFSET 14
- #define EZDP_APP_SCHLR_STATUS_BUSY_MASK (1 << EZDP_APP_SCHLR_STATUS_BUSY_OFFSET)
- #define EZDP_APP_SCHLR_STATUS_ENABLE_SIZE 1
- #define EZDP_APP_SCHLR_STATUS_ENABLE_OFFSET 15
- #define EZDP_APP_SCHLR_STATUS_ENABLE_MASK (1 << EZDP_APP_SCHLR_STATUS_ENABLE_OFFSET)
- #define EZDP_GROUP_SCHLR_STATUS_DISPATCHED_JOB_SIZE 13
- #define EZDP_GROUP_SCHLR_STATUS_DISPATCHED_JOB_OFFSET 0
- #define EZDP_GROUP_SCHLR_STATUS_RESERVED13_15_SIZE 3
- #define EZDP_GROUP_SCHLR_STATUS_RESERVED13_15_OFFSET 13
- #define EZDP_JOB_CONTAINER_CMD_DESC_JOB_ID_SIZE 16
- #define EZDP_JOB_CONTAINER_CMD_DESC_JOB_ID_OFFSET 16
- #define EZDP_JOB_CONTAINER_CMD_DESC_RESERVED0_11_SIZE 12
- #define EZDP_JOB_CONTAINER_CMD_DESC_RESERVED0_11_OFFSET 0
- #define EZDP_JOB_CONTAINER_CMD_DESC_COMMAND_SIZE 4
- #define EZDP_JOB_CONTAINER_CMD_DESC_COMMAND_OFFSET 12
- #define EZDP_JOB_CONTAINER_DESC_RESERVED0_15_SIZE 16
- #define EZDP_JOB_CONTAINER_DESC_RESERVED0_15_OFFSET 0
- #define EZDP_JOB_CONTAINER_DESC_JOB_BUDGET_ID_SIZE 10
- #define EZDP_JOB_CONTAINER_DESC_JOB_BUDGET_ID_OFFSET 16
- #define EZDP_JOB_CONTAINER_DESC_JOB_BUDGET_ID_WORD_SELECT 0
- #define EZDP_JOB_CONTAINER_DESC_JOB_BUDGET_ID_WORD_OFFSET 16
- #define EZDP_JOB_CONTAINER_DESC_INFO_SIZE 3
- #define EZDP_JOB_CONTAINER_DESC_INFO_OFFSET 26
- #define EZDP_JOB_CONTAINER_DESC_INFO_WORD_SELECT 0
- #define EZDP_JOB_CONTAINER_DESC_INFO_WORD_OFFSET 26
- #define EZDP_JOB_CONTAINER_DESC_RESERVED29_31_SIZE 3
- #define EZDP_JOB_CONTAINER_DESC_RESERVED29_31_OFFSET 29
- #define EZDP_JOB_CONTAINER_DESC_WORD_COUNT 8
- #define EZDP_JOB_CONTAINER_DESC_MAX_NUM_OF_JOBS 7

Typedefs

- typedef uint32_t ezdp_job_id_t
- *Job id struct definition.* typedef uint16_t ezdp_job_queue_cmd_info_t
- typedef uint16_t ezdp_job_transmit_cmd_info_t
- typedef uint16_t ezdp_job_discard_cmd_info_t
- typedef uint16_t ezdp_congestion_status_t
- typedef uint8_t ezdp_flow_control_status_t

- typedef uint32_t ezdp_output_queue_status_t
- typedef uint16_t ezdp_app_schlr_status_t
- typedef uint16_t ezdp_group_schlr_status_t
- typedef uint32_t ezdp_job_container_cmd_desc_t
- typedef uint32_t ezdp_job_container_info_t

Enumerations

- enum ezdp_flow_control_node { EZDP_CHANNEL_NODE = 0x0, EZDP_PORT_NODE = 0x1, EZDP_GROUP_NODE = 0x2, EZDP_GLOBAL_NODE = 0x3 }
  Flow control congestion node types.
- enum ezdp_budget_type { EZDP_INT_MEM_BUF_BUDGET = 0x0, EZDP_EXT_MEM_BUF_BUDGET = 0x1, EZDP_JOB_BUDGET = 0x2 }
  Flow control budget types.
- enum ezdp_congestion_level { EZDP_LOW_LEVEL = 0x0, EZDP_MEDIUM_LEVEL = 0x1, EZDP_HIGH_LEVEL = 0x2, EZDP_CRITICAL_LEVEL = 0x3 }
  congestion level possible values.
- enum ezdp_tx_packet_switch_mode { EZDP_TOPOLOGY = 0x0, EZDP_FIXED_BASE = 0x1, EZDP_EXPLICIT_PSID = 0x2 }
  tx packet switch mode possible values.
- enum ezdp_tx_drop_mode { EZDP_CAN_DROP = 0x0, EZDP_DONT_DROP = 0x2, EZDP_NEVER_DROP = 0x3 }
  tx drop mode possible values.
- enum ezdp_job_transmit_dest { EZDP_TM_DEST = EZASM_JOB_TRANSMIT_MODE_TM, EZDP_INTERFACE_DEST = EZASM_JOB_TRANSMIT_MODE_TM_BYPASS }
  job transmit dest possible values.
- enum ezdp_flow_control_congestion_level { EZDP_CONGESTION_LEVEL_0 = 0x0, EZDP_CONGESTION_LEVEL_1 = 0x4, EZDP_CONGESTION_LEVEL_2 = 0x5, EZDP_CONGESTION_LEVEL_3 = 0x6, EZDP_CONGESTION_LEVEL_4 = 0x7 }
  flow control congestion level possible values.
- enum ezdp_job_container_cmd { EZDP_TRANSMIT = EZASM_JOB_NEXT_DST_TRANSMIT, EZDP_FREE = EZASM_JOB_NEXT_DST_TERMINATE, EZDP_DISCARD = EZASM_JOB_NEXT_DST_DISCARD, EZDP_QUEUE = EZASM_JOB_NEXT_DST_PMU, EZDP_QUEUE_WITH_SEQ_NUM = EZASM_JOB_NEXT_DST_PMU_WITH_PQ_SERVICE }
  job container command type possible values.
- enum ezdp_job_flags { EZDP_ALLOW_REORDER = 0x1, EZDP_SET_SEQ_NUM = 0x2, EZDP_HANDLE_NOTICE = 0x4 }
  job flags.
- enum ezdp_report_size { EZDP_8BITS_REPORT = EZDP_DRIVER_REPORT_SIZE_TYPE_8BITS_REPORT, EZDP_16BITS_REPORT = EZDP_DRIVER_REPORT_SIZE_TYPE_16BITS_REPORT, EZDP_32BITS_REPORT = EZDP_DRIVER_REPORT_SIZE_TYPE_32BITS_REPORT }
  report size possible values.

Define Documentation

#define EZDP_TM_REPORT_WORK_AREA_SIZE sizeof(struct ezdp_init_tm_report_wa)

Work area minimal required size definitions.

```
#define EZDP_JOB_RX_INTERFACE_INFO_TIMESTAMP_SEC_SIZE 8

#define EZDP_JOB_RX_INTERFACE_INFO_TIMESTAMP_SEC_OFFSET 0

#define EZDP_JOB_RX_INTERFACE_INFO_TIMESTAMP_SEC_WORD_SELECT 0

#define EZDP_JOB_RX_INTERFACE_INFO_TIMESTAMP_SEC_WORD_OFFSET 0

#define EZDP_JOB_RX_INTERFACE_INFO_CRC_CHECKED_FLAG_SIZE 1

#define EZDP_JOB_RX_INTERFACE_INFO_CRC_CHECKED_FLAG_OFFSET 8

#define EZDP_JOB_RX_INTERFACE_INFO_CRC_CHECKED_FLAG_WORD_SELECT 0

#define EZDP_JOB_RX_INTERFACE_INFO_CRC_CHECKED_FLAG_WORD_OFFSET 8

#define EZDP_JOB_RX_INTERFACE_INFO_CRC_CHECKED_FLAG_MASK (1 <<
EZDP_JOB_RX_INTERFACE_INFO_CRC_CHECKED_FLAG_WORD_OFFSET)

#define EZDP_JOB_RX_INTERFACE_INFO_CRC_OK_FLAG_SIZE 1

#define EZDP_JOB_RX_INTERFACE_INFO_CRC_OK_FLAG_OFFSET 9

#define EZDP_JOB_RX_INTERFACE_INFO_CRC_OK_FLAG_WORD_SELECT 0

#define EZDP_JOB_RX_INTERFACE_INFO_CRC_OK_FLAG_WORD_OFFSET 9

#define EZDP_JOB_RX_INTERFACE_INFO_CRC_OK_FLAG_MASK (1 <<
EZDP_JOB_RX_INTERFACE_INFO_CRC_OK_FLAG_WORD_OFFSET)

#define EZDP_JOB_RX_INTERFACE_INFO_RESERVED10_SIZE 1

#define EZDP_JOB_RX_INTERFACE_INFO_RESERVED10_OFFSET 10

#define EZDP_JOB_RX_INTERFACE_INFO_RESERVED11_SIZE 1

#define EZDP_JOB_RX_INTERFACE_INFO_RESERVED11_OFFSET 11

#define EZDP_JOB_RX_INTERFACE_INFO_TRUNCATION_FLAG_SIZE 1

#define EZDP_JOB_RX_INTERFACE_INFO_TRUNCATION_FLAG_OFFSET 12

#define EZDP_JOB_RX_INTERFACE_INFO_TRUNCATION_FLAG_WORD_SELECT 0

#define EZDP_JOB_RX_INTERFACE_INFO_TRUNCATION_FLAG_WORD_OFFSET 12

#define EZDP_JOB_RX_INTERFACE_INFO_TRUNCATION_FLAG_MASK (1 <<
EZDP_JOB_RX_INTERFACE_INFO_TRUNCATION_FLAG_WORD_OFFSET)

#define EZDP_JOB_RX_INTERFACE_INFO_ICU_SUCC_PARSING_FLAG_SIZE 1

#define EZDP_JOB_RX_INTERFACE_INFO_ICU_SUCC_PARSING_FLAG_OFFSET 13
```

```
#define EZDP_JOB_RX_INTERFACE_INFO_ICU_SUCC_PARSING_FLAG_WORD_SELECT 0

#define EZDP_JOB_RX_INTERFACE_INFO_ICU_SUCC_PARSING_FLAG_WORD_OFFSET 13

#define EZDP_JOB_RX_INTERFACE_INFO_ICU_SUCC_PARSING_FLAG_MASK (1 <<
EZDP_JOB_RX_INTERFACE_INFO_ICU_SUCC_PARSING_FLAG_WORD_OFFSET)

#define EZDP_JOB_RX_INTERFACE_INFO_RESERVED14_SIZE 1

#define EZDP_JOB_RX_INTERFACE_INFO_RESERVED14_OFFSET 14

#define EZDP_JOB_RX_INTERFACE_INFO_RESERVED15_SIZE 1

#define EZDP_JOB_RX_INTERFACE_INFO_RESERVED15_OFFSET 15

#define EZDP_JOB_RX_INTERFACE_INFO_IMEM_BUF_COUNT_SIZE 6

#define EZDP_JOB_RX_INTERFACE_INFO_IMEM_BUF_COUNT_OFFSET 16

#define EZDP_JOB_RX_INTERFACE_INFO_IMEM_BUF_COUNT_WORD_SELECT 0

#define EZDP_JOB_RX_INTERFACE_INFO_IMEM_BUF_COUNT_WORD_OFFSET 16

#define EZDP_JOB_RX_INTERFACE_INFO_GLOBAL_CONGESTION_LEVEL_SIZE 2

#define EZDP_JOB_RX_INTERFACE_INFO_GLOBAL_CONGESTION_LEVEL_OFFSET 22

#define EZDP_JOB_RX_INTERFACE_INFO_GLOBAL_CONGESTION_LEVEL_WORD_SELECT 0

#define EZDP_JOB_RX_INTERFACE_INFO_GLOBAL_CONGESTION_LEVEL_WORD_OFFSET 22

#define EZDP_JOB_RX_INTERFACE_INFO_IMEM_BUF_CONGESTION_LEVEL_SIZE 2

#define EZDP_JOB_RX_INTERFACE_INFO_IMEM_BUF_CONGESTION_LEVEL_OFFSET 24

#define EZDP_JOB_RX_INTERFACE_INFO_IMEM_BUF_CONGESTION_LEVEL_WORD_SELECT 0

#define
EZDP_JOB_RX_INTERFACE_INFO_IMEM_BUF_CONGESTION_LEVEL_WORD_OFFSET 24

#define EZDP_JOB_RX_INTERFACE_INFO_EMEM_BUF_CONGESTION_LEVEL_SIZE 2

#define EZDP_JOB_RX_INTERFACE_INFO_EMEM_BUF_CONGESTION_LEVEL_OFFSET 26

#define
EZDP_JOB_RX_INTERFACE_INFO_EMEM_BUF_CONGESTION_LEVEL_WORD_SELECT 0

#define
EZDP_JOB_RX_INTERFACE_INFO_EMEM_BUF_CONGESTION_LEVEL_WORD_OFFSET 26

#define EZDP_JOB_RX_INTERFACE_INFO_JOB_CONGESTION_LEVEL_SIZE 2

#define EZDP_JOB_RX_INTERFACE_INFO_JOB_CONGESTION_LEVEL_OFFSET 28
```

#define EZDP_JOB_RX_INTERFACE_INFO_JOB_CONGESTION_LEVEL_WORD_SELECT 0

#define EZDP_JOB_RX_INTERFACE_INFO_JOB_CONGESTION_LEVEL_WORD_OFFSET 28

#define EZDP_JOB_RX_INTERFACE_INFO_PMU_QUEUE_CONGESTION_LEVEL_SIZE 2

#define EZDP_JOB_RX_INTERFACE_INFO_PMU_QUEUE_CONGESTION_LEVEL_OFFSET 30

#define EZDP_JOB_RX_INTERFACE_INFO_PMU_QUEUE_CONGESTION_LEVEL_WORD_SELECT 0

#define EZDP_JOB_RX_INTERFACE_INFO_PMU_QUEUE_CONGESTION_LEVEL_WORD_OFFSET 30

#define EZDP_JOB_RX_INTERFACE_INFO_TIMESTAMP_NSEC_SIZE 32

#define EZDP_JOB_RX_INTERFACE_INFO_TIMESTAMP_NSEC_OFFSET 32

#define EZDP_JOB_RX_INTERFACE_INFO_TIMESTAMP_NSEC_WORD_SELECT 1

#define EZDP_JOB_RX_INTERFACE_INFO_TIMESTAMP_NSEC_WORD_OFFSET 0

#define EZDP_JOB_RX_INTERFACE_INFO_WORD_COUNT 2

#define EZDP_JOB_RX_LOOPBACK_INFO_RESERVED0_15_SIZE 16

#define EZDP_JOB_RX_LOOPBACK_INFO_RESERVED0_15_OFFSET 0

#define EZDP_JOB_RX_LOOPBACK_INFO_REPLICATION_ID_SIZE 16

#define EZDP_JOB_RX_LOOPBACK_INFO_REPLICATION_ID_OFFSET 16

#define EZDP_JOB_RX_LOOPBACK_INFO_REPLICATION_ID_WORD_SELECT 0

#define EZDP_JOB_RX_LOOPBACK_INFO_REPLICATION_ID_WORD_OFFSET 16

#define EZDP_JOB_RX_LOOPBACK_INFO_RESERVED32_63_SIZE 32

#define EZDP_JOB_RX_LOOPBACK_INFO_RESERVED32_63_OFFSET 32

#define EZDP_JOB_RX_LOOPBACK_INFO_WORD_COUNT 2

#define EZDP_JOB_RX_CONFIRMATION_INFO_TIMESTAMP_SEC_SIZE 8

#define EZDP_JOB_RX_CONFIRMATION_INFO_TIMESTAMP_SEC_OFFSET 0

#define EZDP_JOB_RX_CONFIRMATION_INFO_TIMESTAMP_SEC_WORD_SELECT 0

#define EZDP_JOB_RX_CONFIRMATION_INFO_TIMESTAMP_SEC_WORD_OFFSET 0

#define EZDP_JOB_RX_CONFIRMATION_INFO_RESERVED8_31_SIZE 24

#define EZDP_JOB_RX_CONFIRMATION_INFO_RESERVED8_31_OFFSET 8

```
#define EZDP_JOB_RX_CONFIRMATION_INFO_TIMESTAMP_NSEC_SIZE 32

#define EZDP_JOB_RX_CONFIRMATION_INFO_TIMESTAMP_NSEC_OFFSET 32

#define EZDP_JOB_RX_CONFIRMATION_INFO_TIMESTAMP_NSEC_WORD_SELECT 1

#define EZDP_JOB_RX_CONFIRMATION_INFO_TIMESTAMP_NSEC_WORD_OFFSET 0

#define EZDP_JOB_RX_CONFIRMATION_INFO_WORD_COUNT 2

#define EZDP_JOB_RX_TIMER_INFO_RESERVED0_8_SIZE 8

#define EZDP_JOB_RX_TIMER_INFO_RESERVED0_8_OFFSET 0

#define EZDP_JOB_RX_TIMER_INFO_TIMER_ID_SIZE 8

#define EZDP_JOB_RX_TIMER_INFO_TIMER_ID_OFFSET 8

#define EZDP_JOB_RX_TIMER_INFO_TIMER_ID_WORD_SELECT 0

#define EZDP_JOB_RX_TIMER_INFO_TIMER_ID_WORD_OFFSET 8

#define EZDP_JOB_RX_TIMER_INFO_RESERVED16_31_SIZE 16

#define EZDP_JOB_RX_TIMER_INFO_RESERVED16_31_OFFSET 16

#define EZDP_JOB_RX_TIMER_INFO_EVENT_ID_SIZE 32

#define EZDP_JOB_RX_TIMER_INFO_EVENT_ID_OFFSET 32

#define EZDP_JOB_RX_TIMER_INFO_EVENT_ID_WORD_SELECT 1

#define EZDP_JOB_RX_TIMER_INFO_EVENT_ID_WORD_OFFSET 0

#define EZDP_JOB_RX_TIMER_INFO_WORD_COUNT 2

#define EZDP_JOB_RX_USER_INFO_USER_DATA_INFO0_SIZE 32

#define EZDP_JOB_RX_USER_INFO_USER_DATA_INFO0_OFFSET 0

#define EZDP_JOB_RX_USER_INFO_USER_DATA_INFO0_WORD_SELECT 0

#define EZDP_JOB_RX_USER_INFO_USER_DATA_INFO0_WORD_OFFSET 0

#define EZDP_JOB_RX_USER_INFO_USER_DATA_INFO1_SIZE 32

#define EZDP_JOB_RX_USER_INFO_USER_DATA_INFO1_OFFSET 32

#define EZDP_JOB_RX_USER_INFO_USER_DATA_INFO1_WORD_SELECT 1

#define EZDP_JOB_RX_USER_INFO_USER_DATA_INFO1_WORD_OFFSET 0
```

```
#define EZDP_JOB_RX_USER_INFO_WORD_COUNT 2

#define EZDP_JOB_RX_INFO_GROSS_CHECKSUM_SIZE 16

#define EZDP_JOB_RX_INFO_GROSS_CHECKSUM_OFFSET 64

#define EZDP_JOB_RX_INFO_GROSS_CHECKSUM_WORD_SELECT 2

#define EZDP_JOB_RX_INFO_GROSS_CHECKSUM_WORD_OFFSET 0

#define EZDP_JOB_RX_INFO_RESERVED112_127_SIZE 16

#define EZDP_JOB_RX_INFO_RESERVED112_127_OFFSET 80

#define EZDP_JOB_RX_INFO_SOURCE_QUEUE_SIZE 7

#define EZDP_JOB_RX_INFO_SOURCE_QUEUE_OFFSET 96

#define EZDP_JOB_RX_INFO_SOURCE_QUEUE_WORD_SELECT 3

#define EZDP_JOB_RX_INFO_SOURCE_QUEUE_WORD_OFFSET 0

#define EZDP_JOB_RX_INFO_SIDE_SIZE 1

#define EZDP_JOB_RX_INFO_SIDE_OFFSET 103

#define EZDP_JOB_RX_INFO_SIDE_WORD_SELECT 3

#define EZDP_JOB_RX_INFO_SIDE_WORD_OFFSET 7

#define EZDP_JOB_RX_INFO_SIDE_MASK (1 << EZDP_JOB_RX_INFO_SIDE_WORD_OFFSET)

#define EZDP_JOB_RX_INFO_RESERVED104_107_SIZE 4

#define EZDP_JOB_RX_INFO_RESERVED104_107_OFFSET 104

#define EZDP_JOB_RX_INFO_RESERVED108_109_SIZE 2

#define EZDP_JOB_RX_INFO_RESERVED108_109_OFFSET 108

#define EZDP_JOB_RX_INFO_SEQ_NUMBER_VALID_SIZE 1

#define EZDP_JOB_RX_INFO_SEQ_NUMBER_VALID_OFFSET 110

#define EZDP_JOB_RX_INFO_SEQ_NUMBER_VALID_WORD_SELECT 3

#define EZDP_JOB_RX_INFO_SEQ_NUMBER_VALID_WORD_OFFSET 14

#define EZDP_JOB_RX_INFO_SEQ_NUMBER_VALID_MASK (1 <<
EZDP_JOB_RX_INFO_SEQ_NUMBER_VALID_WORD_OFFSET)

#define EZDP_JOB_RX_INFO_IS_SERVICE_READY_SIZE 1
```

```
#define EZDP_JOB_RX_INFO_IS_SERVICE_READY_OFFSET 111

#define EZDP_JOB_RX_INFO_IS_SERVICE_READY_WORD_SELECT 3

#define EZDP_JOB_RX_INFO_IS_SERVICE_READY_WORD_OFFSET 15

#define EZDP_JOB_RX_INFO_IS_SERVICE_READY_MASK (1 <<
EZDP_JOB_RX_INFO_IS_SERVICE_READY_WORD_OFFSET)

#define EZDP_JOB_RX_INFO_SEQ_NUMBER_SIZE 16

#define EZDP_JOB_RX_INFO_SEQ_NUMBER_OFFSET 112

#define EZDP_JOB_RX_INFO_SEQ_NUMBER_WORD_SELECT 3

#define EZDP_JOB_RX_INFO_SEQ_NUMBER_WORD_OFFSET 16

#define EZDP_JOB_RX_INFO_WORD_COUNT 4

#define EZDP_JOB_TX_INFO_PACKET_SWITCH_ID_SELECT_SIZE 9

#define EZDP_JOB_TX_INFO_PACKET_SWITCH_ID_SELECT_OFFSET 16

#define EZDP_JOB_TX_INFO_PACKET_SWITCH_ID_SELECT_WORD_SELECT 0

#define EZDP_JOB_TX_INFO_PACKET_SWITCH_ID_SELECT_WORD_OFFSET 16

#define EZDP_JOB_TX_INFO_RESERVED25_29_SIZE 4

#define EZDP_JOB_TX_INFO_RESERVED25_29_OFFSET 25

#define EZDP_JOB_TX_INFO_WRED_COLOR_SIZE 3

#define EZDP_JOB_TX_INFO_WRED_COLOR_OFFSET 29

#define EZDP_JOB_TX_INFO_WRED_COLOR_WORD_SELECT 0

#define EZDP_JOB_TX_INFO_WRED_COLOR_WORD_OFFSET 29

#define EZDP_JOB_TX_INFO_FLOW_ID_SIZE 19

#define EZDP_JOB_TX_INFO_FLOW_ID_OFFSET 32

#define EZDP_JOB_TX_INFO_FLOW_ID_WORD_SELECT 1

#define EZDP_JOB_TX_INFO_FLOW_ID_WORD_OFFSET 0

#define EZDP_JOB_TX_INFO_SIDE_SIZE 1

#define EZDP_JOB_TX_INFO_SIDE_OFFSET 51

#define EZDP_JOB_TX_INFO_SIDE_WORD_SELECT 1
```

```
#define EZDP_JOB_TX_INFO_SIDE_WORD_OFFSET 19

#define EZDP_JOB_TX_INFO_SIDE_MASK (1 << EZDP_JOB_TX_INFO_SIDE_WORD_OFFSET)

#define EZDP_JOB_TX_INFO_RESERVED52_55_SIZE 4

#define EZDP_JOB_TX_INFO_RESERVED52_55_OFFSET 52

#define EZDP_JOB_TX_INFO_PACKET_SWITCH_MODE_SIZE 3

#define EZDP_JOB_TX_INFO_PACKET_SWITCH_MODE_OFFSET 56

#define EZDP_JOB_TX_INFO_PACKET_SWITCH_MODE_WORD_SELECT 1

#define EZDP_JOB_TX_INFO_PACKET_SWITCH_MODE_WORD_OFFSET 24

#define EZDP_JOB_TX_INFO_RESERVED59_SIZE 1

#define EZDP_JOB_TX_INFO_RESERVED59_OFFSET 59

#define EZDP_JOB_TX_INFO_QOS_BYPASS_SIZE 1

#define EZDP_JOB_TX_INFO_QOS_BYPASS_OFFSET 60

#define EZDP_JOB_TX_INFO_QOS_BYPASS_WORD_SELECT 1

#define EZDP_JOB_TX_INFO_QOS_BYPASS_WORD_OFFSET 28

#define EZDP_JOB_TX_INFO_QOS_BYPASS_MASK (1 <<
EZDP_JOB_TX_INFO_QOS_BYPASS_WORD_OFFSET)

#define EZDP_JOB_TX_INFO_RESERVED61_SIZE 1

#define EZDP_JOB_TX_INFO_RESERVED61_OFFSET 61

#define EZDP_JOB_TX_INFO_DROP_MODE_SIZE 2

#define EZDP_JOB_TX_INFO_DROP_MODE_OFFSET 62

#define EZDP_JOB_TX_INFO_DROP_MODE_WORD_SELECT 1

#define EZDP_JOB_TX_INFO_DROP_MODE_WORD_OFFSET 30

#define EZDP_JOB_TX_INFO_STAT_STREAM_ID_SIZE 24

#define EZDP_JOB_TX_INFO_STAT_STREAM_ID_OFFSET 64

#define EZDP_JOB_TX_INFO_STAT_STREAM_ID_WORD_SELECT 2

#define EZDP_JOB_TX_INFO_STAT_STREAM_ID_WORD_OFFSET 0

#define EZDP_JOB_TX_INFO_RESERVED88_90_SIZE 2
```

#define EZDP_JOB_TX_INFO_RESERVED88_90_OFFSET 88

#define EZDP_JOB_TX_INFO_STAT_CODE_PROFILE2_SIZE 3

#define EZDP_JOB_TX_INFO_STAT_CODE_PROFILE2_OFFSET 90

#define EZDP_JOB_TX_INFO_STAT_CODE_PROFILE2_WORD_SELECT 2

#define EZDP_JOB_TX_INFO_STAT_CODE_PROFILE2_WORD_OFFSET 26

#define EZDP_JOB_TX_INFO_STAT_CODE_PROFILE1_SIZE 3

#define EZDP_JOB_TX_INFO_STAT_CODE_PROFILE1_OFFSET 93

#define EZDP_JOB_TX_INFO_STAT_CODE_PROFILE1_WORD_SELECT 2

#define EZDP_JOB_TX_INFO_STAT_CODE_PROFILE1_WORD_OFFSET 29

#define EZDP_JOB_TX_INFO_INTER_PACKET_GAP_SIZE 5

#define EZDP_JOB_TX_INFO_INTER_PACKET_GAP_OFFSET 96

#define EZDP_JOB_TX_INFO_INTER_PACKET_GAP_WORD_SELECT 3

#define EZDP_JOB_TX_INFO_INTER_PACKET_GAP_WORD_OFFSET 0

#define EZDP_JOB_TX_INFO_INTER_PACKET_GAP_CONTROL_SIZE 1

#define EZDP_JOB_TX_INFO_INTER_PACKET_GAP_CONTROL_OFFSET 101

#define EZDP_JOB_TX_INFO_INTER_PACKET_GAP_CONTROL_WORD_SELECT 3

#define EZDP_JOB_TX_INFO_INTER_PACKET_GAP_CONTROL_WORD_OFFSET 5

#define EZDP_JOB_TX_INFO_INTER_PACKET_GAP_CONTROL_MASK (1 << EZDP_JOB_TX_INFO_INTER_PACKET_GAP_CONTROL_WORD_OFFSET)

#define EZDP_JOB_TX_INFO_RESERVED102_103_SIZE 2

#define EZDP_JOB_TX_INFO_RESERVED102_103_OFFSET 102

#define EZDP_JOB_TX_INFO_WRED_CLASS_SCALE_PROFILE_SIZE 8

#define EZDP_JOB_TX_INFO_WRED_CLASS_SCALE_PROFILE_OFFSET 104

#define EZDP_JOB_TX_INFO_WRED_CLASS_SCALE_PROFILE_WORD_SELECT 3

#define EZDP_JOB_TX_INFO_WRED_CLASS_SCALE_PROFILE_WORD_OFFSET 8

#define EZDP_JOB_TX_INFO_WRED_FLOW_SCALE_PROFILE_SIZE 8

#define EZDP_JOB_TX_INFO_WRED_FLOW_SCALE_PROFILE_OFFSET 112

```
#define EZDP_JOB_TX_INFO_WRED_FLOW_SCALE_PROFILE_WORD_SELECT 3

#define EZDP_JOB_TX_INFO_WRED_FLOW_SCALE_PROFILE_WORD_OFFSET 16

#define EZDP_JOB_TX_INFO_WRED_CLASS_TEMPLATE_PROFILE_SIZE 4

#define EZDP_JOB_TX_INFO_WRED_CLASS_TEMPLATE_PROFILE_OFFSET 120

#define EZDP_JOB_TX_INFO_WRED_CLASS_TEMPLATE_PROFILE_WORD_SELECT 3

#define EZDP_JOB_TX_INFO_WRED_CLASS_TEMPLATE_PROFILE_WORD_OFFSET 24

#define EZDP_JOB_TX_INFO_WRED_FLOW_TEMPLATE_PROFILE_SIZE 4

#define EZDP_JOB_TX_INFO_WRED_FLOW_TEMPLATE_PROFILE_OFFSET 124

#define EZDP_JOB_TX_INFO_WRED_FLOW_TEMPLATE_PROFILE_WORD_SELECT 3

#define EZDP_JOB_TX_INFO_WRED_FLOW_TEMPLATE_PROFILE_WORD_OFFSET 28

#define EZDP_JOB_TX_INFO_WORD_COUNT 4

#define EZDP_JOB_QUEUE_CMD_INFO_TARGET_QUEUE_SIZE 7

#define EZDP_JOB_QUEUE_CMD_INFO_TARGET_QUEUE_OFFSET 0

#define EZDP_JOB_QUEUE_CMD_INFO_SIDE_SIZE 1

#define EZDP_JOB_QUEUE_CMD_INFO_SIDE_OFFSET 7

#define EZDP_JOB_QUEUE_CMD_INFO_SIDE_MASK (1 <<
EZDP_JOB_QUEUE_CMD_INFO_SIDE_OFFSET)

#define EZDP_JOB_QUEUE_CMD_INFO_RESERVED8_15_SIZE 8

#define EZDP_JOB_QUEUE_CMD_INFO_RESERVED8_15_OFFSET 8

#define EZDP_JOB_TRANSMIT_CMD_INFO_OUTPUT_CHANNEL_SIZE 10

#define EZDP_JOB_TRANSMIT_CMD_INFO_OUTPUT_CHANNEL_OFFSET 0

#define EZDP_JOB_TRANSMIT_CMD_INFO_SIDE_SIZE 1

#define EZDP_JOB_TRANSMIT_CMD_INFO_SIDE_OFFSET 10

#define EZDP_JOB_TRANSMIT_CMD_INFO_SIDE_MASK (1 <<
EZDP_JOB_TRANSMIT_CMD_INFO_SIDE_OFFSET)

#define EZDP_JOB_TRANSMIT_CMD_INFO_DESTINATION_SIZE 1

#define EZDP_JOB_TRANSMIT_CMD_INFO_DESTINATION_OFFSET 11
```

#define EZDP_JOB_TRANSMIT_CMD_INFO_DESTINATION_MASK (1 << EZDP_JOB_TRANSMIT_CMD_INFO_DESTINATION_OFFSET)

#define EZDP_JOB_TRANSMIT_CMD_INFO_RESERVED12_15_SIZE 4

#define EZDP_JOB_TRANSMIT_CMD_INFO_RESERVED12_15_OFFSET 12

#define EZDP_JOB_DISCARD_CMD_INFO_RESERVED0_9_SIZE 10

#define EZDP_JOB_DISCARD_CMD_INFO_RESERVED0_9_OFFSET 0

#define EZDP_JOB_DISCARD_CMD_INFO_SIDE_SIZE 1

#define EZDP_JOB_DISCARD_CMD_INFO_SIDE_OFFSET 10

#define EZDP_JOB_DISCARD_CMD_INFO_SIDE_MASK (1 << EZDP_JOB_DISCARD_CMD_INFO_SIDE_OFFSET)

#define EZDP_JOB_DISCARD_CMD_INFO_RESERVED11_15_SIZE 5

#define EZDP_JOB_DISCARD_CMD_INFO_RESERVED11_15_OFFSET 11

#define EZDP_CONGESTION_STATUS_IMEM_BUF_CONGESTION_LEVEL_SIZE 2

#define EZDP_CONGESTION_STATUS_IMEM_BUF_CONGESTION_LEVEL_OFFSET 0

#define EZDP_CONGESTION_STATUS_IMEM_BUF_GUARANTEE_SIZE 1

#define EZDP_CONGESTION_STATUS_IMEM_BUF_GUARANTEE_OFFSET 2

#define EZDP_CONGESTION_STATUS_IMEM_BUF_GUARANTEE_MASK (1 << EZDP_CONGESTION_STATUS_IMEM_BUF_GUARANTEE_OFFSET)

#define EZDP_CONGESTION_STATUS_RESERVED3_SIZE 1

#define EZDP_CONGESTION_STATUS_RESERVED3_OFFSET 3

#define EZDP_CONGESTION_STATUS_EMEM_BUF_CONGESTION_LEVEL_SIZE 2

#define EZDP_CONGESTION_STATUS_EMEM_BUF_CONGESTION_LEVEL_OFFSET 4

#define EZDP_CONGESTION_STATUS_EMEM_BUF_GUARANTEE_SIZE 1

#define EZDP_CONGESTION_STATUS_EMEM_BUF_GUARANTEE_OFFSET 6

#define EZDP_CONGESTION_STATUS_EMEM_BUF_GUARANTEE_MASK (1 << EZDP_CONGESTION_STATUS_EMEM_BUF_GUARANTEE_OFFSET)

#define EZDP_CONGESTION_STATUS_RESERVED7_SIZE 1

#define EZDP_CONGESTION_STATUS_RESERVED7_OFFSET 7

#define EZDP_CONGESTION_STATUS_JOB_CONGESTION_LEVEL_SIZE 2

#define EZDP_CONGESTION_STATUS_JOB_CONGESTION_LEVEL_OFFSET 8

#define EZDP_CONGESTION_STATUS_JOB_GUARANTEE_SIZE 1

#define EZDP_CONGESTION_STATUS_JOB_GUARANTEE_OFFSET 10

#define EZDP_CONGESTION_STATUS_JOB_GUARANTEE_MASK (1 << EZDP_CONGESTION_STATUS_JOB_GUARANTEE_OFFSET)

#define EZDP_CONGESTION_STATUS_RESERVED11_SIZE 1

#define EZDP_CONGESTION_STATUS_RESERVED11_OFFSET 11

#define EZDP_CONGESTION_STATUS_PORT_CONGESTION_LEVEL_SIZE 2

#define EZDP_CONGESTION_STATUS_PORT_CONGESTION_LEVEL_OFFSET 12

#define EZDP_CONGESTION_STATUS_RESERVED14_15_SIZE 2

#define EZDP_CONGESTION_STATUS_RESERVED14_15_OFFSET 14

#define EZDP_FLOW_CONTROL_STATUS_CONGESTION_LEVEL_SIZE 3

#define EZDP_FLOW_CONTROL_STATUS_CONGESTION_LEVEL_OFFSET 0

#define EZDP_FLOW_CONTROL_STATUS_ENABLE_SIZE 1

#define EZDP_FLOW_CONTROL_STATUS_ENABLE_OFFSET 3

#define EZDP_FLOW_CONTROL_STATUS_ENABLE_MASK (1 << EZDP_FLOW_CONTROL_STATUS_ENABLE_OFFSET)

#define EZDP_FLOW_CONTROL_STATUS_RESERVED4_7_SIZE 4

#define EZDP_FLOW_CONTROL_STATUS_RESERVED4_7_OFFSET 4

#define EZDP_INPUT_QUEUE_STATUS_DISPATCHED_JOB_SIZE 16

#define EZDP_INPUT_QUEUE_STATUS_DISPATCHED_JOB_OFFSET 0

#define EZDP_INPUT_QUEUE_STATUS_DISPATCHED_JOB_WORD_SELECT 0

#define EZDP_INPUT_QUEUE_STATUS_DISPATCHED_JOB_WORD_OFFSET 0

#define EZDP_INPUT_QUEUE_STATUS_CONGESTION_LEVEL_SIZE 2

#define EZDP_INPUT_QUEUE_STATUS_CONGESTION_LEVEL_OFFSET 16

#define EZDP_INPUT_QUEUE_STATUS_CONGESTION_LEVEL_WORD_SELECT 0

#define EZDP_INPUT_QUEUE_STATUS_CONGESTION_LEVEL_WORD_OFFSET 16

#define EZDP_INPUT_QUEUE_STATUS_READY_SIZE 1

#define EZDP_INPUT_QUEUE_STATUS_READY_OFFSET 18

#define EZDP_INPUT_QUEUE_STATUS_READY_WORD_SELECT 0

#define EZDP_INPUT_QUEUE_STATUS_READY_WORD_OFFSET 18

#define EZDP_INPUT_QUEUE_STATUS_READY_MASK (1 << EZDP_INPUT_QUEUE_STATUS_READY_WORD_OFFSET)

#define EZDP_INPUT_QUEUE_STATUS_RESERVED19_31_SIZE 13

#define EZDP_INPUT_QUEUE_STATUS_RESERVED19_31_OFFSET 19

#define EZDP_INPUT_QUEUE_STATUS_OUTSTANDING_JOB_SIZE 16

#define EZDP_INPUT_QUEUE_STATUS_OUTSTANDING_JOB_OFFSET 32

#define EZDP_INPUT_QUEUE_STATUS_OUTSTANDING_JOB_WORD_SELECT 1

#define EZDP_INPUT_QUEUE_STATUS_OUTSTANDING_JOB_WORD_OFFSET 0

#define EZDP_INPUT_QUEUE_STATUS_SIZE_SIZE 16

#define EZDP_INPUT_QUEUE_STATUS_SIZE_OFFSET 48

#define EZDP_INPUT_QUEUE_STATUS_SIZE_WORD_SELECT 1

#define EZDP_INPUT_QUEUE_STATUS_SIZE_WORD_OFFSET 16

#define EZDP_INPUT_QUEUE_STATUS_WORD_COUNT 2

#define EZDP_OUTPUT_QUEUE_STATUS_SIZE_SIZE 16

#define EZDP_OUTPUT_QUEUE_STATUS_SIZE_OFFSET 0

#define EZDP_OUTPUT_QUEUE_STATUS_READY_SIZE 1

#define EZDP_OUTPUT_QUEUE_STATUS_READY_OFFSET 16

#define EZDP_OUTPUT_QUEUE_STATUS_READY_MASK (1 << EZDP_OUTPUT_QUEUE_STATUS_READY_OFFSET)

#define EZDP_OUTPUT_QUEUE_STATUS_CONGESTION_SIZE 1

#define EZDP_OUTPUT_QUEUE_STATUS_CONGESTION_OFFSET 17

#define EZDP_OUTPUT_QUEUE_STATUS_CONGESTION_MASK (1 << EZDP_OUTPUT_QUEUE_STATUS_CONGESTION_OFFSET)

#define EZDP_OUTPUT_QUEUE_STATUS_RESERVED18_31_SIZE 14

#define EZDP_OUTPUT_QUEUE_STATUS_RESERVED18_31_OFFSET 18

```
#define EZDP_APP_SCHLR_STATUS_DISPATCHED_JOB_SIZE 13

#define EZDP_APP_SCHLR_STATUS_DISPATCHED_JOB_OFFSET 0

#define EZDP_APP_SCHLR_STATUS_RESERVED13_SIZE 1

#define EZDP_APP_SCHLR_STATUS_RESERVED13_OFFSET 13

#define EZDP_APP_SCHLR_STATUS_BUSY_SIZE 1

#define EZDP_APP_SCHLR_STATUS_BUSY_OFFSET 14

#define EZDP_APP_SCHLR_STATUS_BUSY_MASK (1 <<
EZDP_APP_SCHLR_STATUS_BUSY_OFFSET)

#define EZDP_APP_SCHLR_STATUS_ENABLE_SIZE 1

#define EZDP_APP_SCHLR_STATUS_ENABLE_OFFSET 15

#define EZDP_APP_SCHLR_STATUS_ENABLE_MASK (1 <<
EZDP_APP_SCHLR_STATUS_ENABLE_OFFSET)

#define EZDP_GROUP_SCHLR_STATUS_DISPATCHED_JOB_SIZE 13

#define EZDP_GROUP_SCHLR_STATUS_DISPATCHED_JOB_OFFSET 0

#define EZDP_GROUP_SCHLR_STATUS_RESERVED13_15_SIZE 3

#define EZDP_GROUP_SCHLR_STATUS_RESERVED13_15_OFFSET 13

#define EZDP_JOB_CONTAINER_CMD_DESC_JOB_ID_SIZE 16

#define EZDP_JOB_CONTAINER_CMD_DESC_JOB_ID_OFFSET 16

#define EZDP_JOB_CONTAINER_CMD_DESC_RESERVED0_11_SIZE 12

#define EZDP_JOB_CONTAINER_CMD_DESC_RESERVED0_11_OFFSET 0

#define EZDP_JOB_CONTAINER_CMD_DESC_COMMAND_SIZE 4

#define EZDP_JOB_CONTAINER_CMD_DESC_COMMAND_OFFSET 12

#define EZDP_JOB_CONTAINER_DESC_RESERVED0_15_SIZE 16

#define EZDP_JOB_CONTAINER_DESC_RESERVED0_15_OFFSET 0

#define EZDP_JOB_CONTAINER_DESC_JOB_BUDGET_ID_SIZE 10

#define EZDP_JOB_CONTAINER_DESC_JOB_BUDGET_ID_OFFSET 16

#define EZDP_JOB_CONTAINER_DESC_JOB_BUDGET_ID_WORD_SELECT 0

#define EZDP_JOB_CONTAINER_DESC_JOB_BUDGET_ID_WORD_OFFSET 16
```

#define EZDP_JOB_CONTAINER_DESC_INFO_SIZE 3

#define EZDP_JOB_CONTAINER_DESC_INFO_OFFSET 26

#define EZDP_JOB_CONTAINER_DESC_INFO_WORD_SELECT 0

#define EZDP_JOB_CONTAINER_DESC_INFO_WORD_OFFSET 26

#define EZDP_JOB_CONTAINER_DESC_RESERVED29_31_SIZE 3

#define EZDP_JOB_CONTAINER_DESC_RESERVED29_31_OFFSET 29

#define EZDP_JOB_CONTAINER_DESC_WORD_COUNT 8

#define EZDP_JOB_CONTAINER_DESC_MAX_NUM_OF_JOBS 7

Typedef Documentation

typedef uint32_t ezdp_job_id_t

Job id struct definition.

typedef uint16_t ezdp_job_queue_cmd_info_t

typedef uint16_t ezdp_job_transmit_cmd_info_t

typedef uint16_t ezdp_job_discard_cmd_info_t

typedef uint16_t ezdp_congestion_status_t

typedef uint8_t ezdp_flow_control_status_t

typedef uint32_t ezdp_output_queue_status_t

typedef uint16_t ezdp_app_schlr_status_t

typedef uint16_t ezdp_group_schlr_status_t

typedef uint32_t ezdp_job_container_cmd_desc_t

typedef uint32_t ezdp_job_container_info_t

Enumeration Type Documentation

enum ezdp_flow_control_node

Flow control congestion node types.

Enumerator:

EZDP_CHANNEL_NODE Lowest level - channel.

Each incoming/outgoing packet is accompanied a channel (AKA source tag). The max number of channel nodes is 256

EZDP_PORT_NODE Second level - port.

Usually represent port in the system (physical ports, AUX ports, Loop-back, user defined) The max number of port nodes is 128

EZDP_GROUP_NODE Third level - group.

Group can aggregate few port nodes The max number of port nodes is 16

EZDP_GLOBAL_NODE The last level - global.

Only 1 global node exist in the system.

enum ezdp_budget_type

Flow control budget types.

Enumerator:

EZDP_INT_MEM_BUF_BUDGET Internal memory frame buffer budget.

EZDP_EXT_MEM_BUF_BUDGET External memory frame buffer budget.

EZDP_JOB_BUDGET Job budget.

enum ezdp_congestion_level

congestion level possible values.

Enumerator:

EZDP_LOW_LEVEL First threshold not crossed.

EZDP_MEDIUM_LEVEL Crossing first threshold.

EZDP_HIGH_LEVEL Crossing second threshold.

EZDP_CRITICAL_LEVEL Crossing max threshold.

enum ezdp_tx_packet_switch_mode

tx packet switch mode possible values.

Enumerator:

EZDP_TOPOLOGY Topology based entry selection.

L0/L1 entity selects base table entry according to TM queue topology, 3 lsb of packet_switch_id_select selects offset, and the table entry provides an output channel. Applicable only for TM Mode = FULL.

EZDP_FIXED_BASE Preconfigured fixed base entry selection.

TM packet switch id table base is taken from TM configuration and 3 lsb of packet_switch_id_select selects an offset in that entry. This method bypasses TM topological

L0/L1 queue selection when addressing the port switching table. Typically this flow is used by implicit TM loopback flows to one of eight PMU loopback channels (channel IDs 128-143). It can also be used for specific transmission path with eight selectable output channels. Applicable for TM Modes FULL and BYPASS.

EZDP_EXPLICIT_PSID Explicit PSID selection.

In this mode, SW puts an explicit PSID in the 16-bit INFO fields carried in the TM packet header. Explicitly selects an entry from the packet switch id table. Applicable for TM Modes FULL and BYPASS.

enum ezdp_tx_drop_mode

tx drop mode possible values.

Enumerator:

EZDP_CAN_DROP Standard TM drop policy.

The packet can be dropped from TM queues by policer WRED, or by other TM scheduling levels.

EZDP_DONT_DROP No Drop Packet.

The packet is protected from getting dropped along TM data paths (either by policer WRED or other TM scheduling levels). The packet could be dropped if the TM queue it belongs to is being flushed through configuration access.

EZDP_NEVER_DROP Never Drop Packet (SYNC).

In addition to not being dropped by TM data paths, the NEVER_DROP packet is also not dropped on a flushed queue (the NEVER_DROP packet is used to mark the end of the queue flush).

enum ezdp_job_transmit_dest

job transmit dest possible values.

Enumerator:

EZDP_TM_DEST Forward to TM.

EZDP_INTERFACE_DEST Forward directly to output interface.

enum ezdp_flow_control_congestion_level

flow control congestion level possible values.

Enumerator:

EZDP_CONGESTION_LEVEL_0 First threshold not crossed.

EZDP_CONGESTION_LEVEL_1 Crossing first threshold.

EZDP_CONGESTION_LEVEL_2 Crossing second threshold.

EZDP_CONGESTION_LEVEL_3 Crossing third threshold.

EZDP_CONGESTION_LEVEL_4 Crossing fourth threshold.

enum ezdp_job_container_cmd

job container command type possible values.

Enumerator:

EZDP_TRANSMIT Transmit the job either via TM or directly to output channel bypassing the TM.

EZDP_FREE Recycle a job to the PMU.

NOTE: Any associated frame resources are not recycle.

EZDP_DISCARD Discard a job.

NOTE: All associated frame resources are recycled.

EZDP_QUEUE Dispatch the job to another PMU queue.

EZDP_QUEUE_WITH_SEQ_NUM Dispatch the job to another PMU queue and request target queue sequence numbering service.

enum ezdp_job_flags

job flags.

Enumerator:

EZDP_ALLOW_REORDER Allow jobs to be unordered.

EZDP_SET_SEQ_NUM Request sequence numbering - applicable only when changing the PMU queue.

EZDP_HANDLE_NOTICE Allow notice handling, by calling to ezdp_notice_handler function, before receiving the job.

enum ezdp_report_size

report size possible values.

Enumerator:

EZDP_8BITS_REPORT 8 bits report size

EZDP_16BITS_REPORT 16 bits report size

EZDP_32BITS_REPORT 32 bits report size

dpe/dp/include/ezdp_lock.h File Reference

Functions

- static __always_inline uint32_t ezdp_init_spinlock_ext_addr (ezdp_spinlock_t *spinlock_ref, struct ezdp_ext_addr *addr)
- *Initialize resources required for a spin lock.* static __always_inline uint32_t ezdp_init_spinlock_sum_addr (ezdp_spinlock_t *spinlock_ref, ezdp_sum_addr_t addr)
- *Initialize resources required for a spin lock.* static __always_inline uint32_t ezdp_lock_spinlock (ezdp_spinlock_t *spinlock_ref)
- *Lock a spin lock.* static __always_inline uint32_t ezdp_try_lock_spinlock (ezdp_spinlock_t *spinlock_ref)
- *Lock a spin lock with limited number of attempts.* static __always_inline uint32_t ezdp_unlock_spinlock (ezdp_spinlock_t *spinlock_ref)
- *Release a spin lock which was locked.* static __always_inline void ezdp_init_qlock (ezdp_qlock_t *qlock_ref, ezdp_sum_addr_t addr, ezdp_mem_pool_t *link_pool)
- *Initialize queue lock structure.* static __always_inline ezdp_sum_addr_t ezdp_destroy_qlock (ezdp_qlock_t *qlock_ref)
- *Get queue lock address.* static __always_inline ezdp_qlock_slot_t ezdp_alloc_qlock_slot (ezdp_qlock_t *qlock_ref)
- *allocate queue lock slot.* static __always_inline void ezdp_free_qlock_slot (ezdp_qlock_t *qlock_ref, ezdp_qlock_slot_t qlock_slot)
- *free queue lock slot.* static __always_inline bool ezdp_lock_qlock (ezdp_qlock_t *qlock_ref, ezdp_qlock_slot_t qlock_slot, char *work_area_ptr, uint32_t work_area_size)
- *Try to lock queue lock.* static __always_inline bool ezdp_order_lock_qlock (ezdp_qlock_t *qlock_ref, ezdp_qlock_slot_t qlock_slot, uint16_t *order_lock, uint16_t sequence, char *work_area_ptr, uint32_t work_area_size)
- *Try to lock queue lock (with order).* static __always_inline void ezdp_enqueue_qlock (ezdp_qlock_t *qlock_ref, ezdp_qlock_slot_t qlock_slot, uint8_t *data, uint8_t data_size, char *work_area_ptr, uint32_t work_area_size)
- *enqueue data to queue lock.* static __always_inline bool ezdp_dequeue_qlock (ezdp_qlock_t *qlock_ref, uint8_t *data, uint8_t data_size, char *work_area_ptr, uint32_t work_area_size)
- *dequeue data from queue lock.* static __always_inline bool ezdp_try_unlock_qlock (ezdp_qlock_t *qlock_ref)

Try to unlock queue lock.

Function Documentation

static __always_inline uint32_t ezdp_init_spinlock_ext_addr (ezdp_spinlock_t * *spinlock_ref*, struct ezdp_ext_addr * *addr*) [static]

Initialize resources required for a spin lock.

Parameters:

[out] *spinlock_ref* - Reference to spinlock object
[in] *addr* - address of spinlock (1 byte)

Returns:

0 - success

static __always_inline uint32_t ezdp_init_spinlock_sum_addr (ezdp_spinlock_t * *spinlock_ref*, ezdp_sum_addr_t *addr*) [static]

Initialize resources required for a spin lock.

Parameters:

[out] *spinlock_ref* - Reference to spinlock object

[in] *addr* - address of spinlock (1 byte)

Returns:

0 - success

static __always_inline uint32_t ezdp_lock_spinlock (ezdp_spinlock_t * *spinlock_ref*) [static]

Lock a spin lock.

Calling thread shall acquire the lock if it is not held by another thread. Otherwise, the thread shall spin until the lock becomes available.

Parameters:

[in] *spinlock_ref* - Reference to spinlock object

Returns:

0 - success

static __always_inline uint32_t ezdp_try_lock_spinlock (ezdp_spinlock_t * *spinlock_ref*) [static]

Lock a spin lock with limited number of attempts.

Calling thread shall acquire the lock if it is not held by another thread. Otherwise, the thread shall spin until the lock becomes available or number of tries reach maximum.

Parameters:

[in] *spinlock_ref* - Reference to spinlock object

Returns:

0 - success 1 - try lock num_of_tries with no success

static __always_inline uint32_t ezdp_unlock_spinlock (ezdp_spinlock_t * *spinlock_ref*) [static]

Release a spin lock which was locked.

Parameters:

[in] *spinlock_ref* - Reference to spinlock object

Returns:

0 - success 2 - try to unlock twice

static __always_inline void ezdp_init_qlock (ezdp_qlock_t * *qlock_ref*, ezdp_sum_addr_t *addr*, ezdp_mem_pool_t * *link_pool*) [static]

Initialize queue lock structure.

Parameters:

[out] *qlock_ref* - Reference to queue lock object
[in] *addr* - address of queue lock (4B)
[in] *link_pool* - memory pool for queue links (16B each)

Returns:

void

static __always_inline ezdp_sum_addr_t ezdp_destroy_qlock (ezdp_qlock_t * *qlock_ref*) [static]

Get queue lock address.

Parameters:

[in] *qlock_ref* - Reference to queue lock object

Returns:

Address of queue lock

static __always_inline ezdp_qlock_slot_t ezdp_alloc_qlock_slot (ezdp_qlock_t * *qlock_ref*) [static]

allocate queue lock slot.

allocate queue lock slot. slot should be validated with ezdp_is_null_qlock_slot() before usage.

Parameters:

[in] *qlock_ref* - Reference to queue lock object

Returns:

queue lock slot to be used in ezdp_try_to_lock_qlock() and ezdp_enqueue_qlock()

static __always_inline void ezdp_free_qlock_slot (ezdp_qlock_t * *qlock_ref*, ezdp_qlock_slot_t *qlock_slot*) [static]

free queue lock slot.

Parameters:

[in] *qlock_ref* - Reference to queue lock object
[in] *qlock_slot* - queue lock slot received from ezdp_alloc_qlock_slot()

Returns:

queue lock slot to be used in ezdp_try_to_lock_qlock() and ezdp_enqueue_qlock()

static __always_inline bool ezdp_lock_qlock (ezdp_qlock_t * *qlock_ref*, ezdp_qlock_slot_t *qlock_slot*, char * *work_area_ptr*, uint32_t *work_area_size*) [static]

Try to lock queue lock.

Parameters:

[in] *qlock_ref* - Reference to queue lock object
[in] *qlock_slot* - queue lock slot received from ezdp_alloc_qlock_slot()
[in] *work_area_ptr* - work area pointer (temporary memory to be used by the function). The size of the temporary memory is determined by EZDP_QLOCK_WORK_AREA_SIZE
[in] *work_area_size* - size of work area pointer

Returns:

true iff lock queue was locked.

static __always_inline bool ezdp_order_lock_qlock (ezdp_qlock_t * *qlock_ref*, ezdp_qlock_slot_t *qlock_slot*, uint16_t * *order_lock*, uint16_t *sequence*, char * *work_area_ptr*, uint32_t *work_area_size*) [static]

Try to lock queue lock (with order).

Parameters:

[in] *qlock_ref* - Reference to queue lock object
[in] *qlock_slot* - queue lock slot received from ezdp_alloc_qlock_slot()
[in] *order_lock* - order lock
[in] *sequence* - sequence
[in] *work_area_ptr* - work area pointer (temporary memory to be used by the function). The size of the temporary memory is determined by EZDP_QLOCK_WORK_AREA_SIZE
[in] *work_area_size* - size of work area pointer

Note:

The API is experimental

Returns:

true if lock queue was locked.

static __always_inline void ezdp_enqueue_qlock (ezdp_qlock_t * *qlock_ref*, ezdp_qlock_slot_t *qlock_slot*, uint8_t * *data*, uint8_t *data_size*, char * *work_area_ptr*, uint32_t *work_area_size*) `[static]`

enqueue data to queue lock.

Parameters:

[in] *qlock_ref* - Reference to queue lock object
[in] *qlock_slot* - queue lock slot received from ezdp_alloc_qlock_slot()
[in] *data* - data (up to 12 bytes)
[in] *data_size* - data size
[in] *work_area_ptr* - work area pointer (temporary memory to be used by the function). The size of the temporary memory is determined by EZDP_QLOCK_WORK_AREA_SIZE
[in] *work_area_size* - size of work area pointer

Returns:

void

static __always_inline bool ezdp_dequeue_qlock (ezdp_qlock_t * *qlock_ref*, uint8_t * *data*, uint8_t *data_size*, char * *work_area_ptr*, uint32_t *work_area_size*) `[static]`

dequeue data from queue lock.

Parameters:

[in] *qlock_ref* - Reference to queue lock object
[out] *data* - data (up to 12 bytes)
[in] *data_size* - data size
[in] *work_area_ptr* - work area pointer (temporary memory to be used by the function). The size of the temporary memory is determined by EZDP_QLOCK_WORK_AREA_SIZE
[in] *work_area_size* - size of work area pointer

Returns:

true iff next data is ready in data out argument.

static __always_inline bool ezdp_try_unlock_qlock (ezdp_qlock_t * *qlock_ref*) `[static]`

Try to unlock queue lock.

Parameters:

[in] *qlock_ref* - Reference to queue lock object

Returns:

true if queue lock was unlocked. false if queue is not empty.

dpe/dp/include/ezdp_lock_defs.h File Reference

Defines

- #define EZDP_QLOCK_WORK_AREA_SIZE sizeof(struct ezdp_qlock_working_area)
- *Work area minimal required size definitions.* #define EZDP_NULL_QLOCK_SLOT EZDP_NULL_SUM_ADDR

Typedefs

- typedef struct ezdp_spinlock ezdp_spinlock_t
- typedef struct ezdp_qlock ezdp_qlock_t
- typedef ezdp_sum_addr_t ezdp_qlock_slot_t

Define Documentation

#define EZDP_QLOCK_WORK_AREA_SIZE sizeof(struct ezdp_qlock_working_area)

Work area minimal required size definitions.

#define EZDP_NULL_QLOCK_SLOT EZDP_NULL_SUM_ADDR

Typedef Documentation

typedef struct ezdp_spinlock ezdp_spinlock_t

typedef struct ezdp_qlock ezdp_qlock_t

typedef ezdp_sum_addr_t ezdp_qlock_slot_t

dpe/dp/include/ezdp_math.h File Reference

Enumerations

- enum ezdp_bit_mode { EZDP_BIT_MODE_VALUE = EZASM_BM_BIT_MODE_VALUE, EZDP_BIT_MODE_INVERSE = EZASM_BM_BIT_MODE_INVERSE, EZDP_BIT_MODE_FALSE = EZASM_BM_BIT_MODE_FALSE, EZDP_BIT_MODE_TRUE = EZASM_BM_BIT_MODE_TRUE }
  Bit manipulation mode possible values.
- enum ezdp_reflect_resolution { EZDP_REFLECT_RESOLUTION_1_BYTE = EZASM_RFLT_RESOLUTION_1_BYTE, EZDP_REFLECT_RESOLUTION_2_BYTE = EZASM_RFLT_RESOLUTION_2_BYTE, EZDP_REFLECT_RESOLUTION_4_BYTE = EZASM_RFLT_RESOLUTION_4_BYTE }
  Bit manipulation mode possible values.
- enum ezdp_hash_base_matrix { EZDP_HASH_BASE_MATRIX_HASH_BASE_MATRIX_0 = 0x0, EZDP_HASH_BASE_MATRIX_HASH_BASE_MATRIX_1 = 0x1 }
  Hash base matrix possible values.
- enum ezdp_hash_permutation { EZDP_HASH_PERMUTATION_0 = 0x0, EZDP_HASH_PERMUTATION_1 = 0x1, EZDP_HASH_PERMUTATION_2 = 0x2, EZDP_HASH_PERMUTATION_3 = 0x3 }
  Hash permutation possible values.

Functions

- static __always_inline uint32_t ezdp_add (uint32_t src1, uint32_t src2, uint32_t src1_pos, uint32_t src2_pos, uint32_t size)
- *Add selected bits of src1 to selected bits of src2.* static __always_inline uint32_t ezdp_sub (uint32_t src1, uint32_t src2, uint32_t src1_pos, uint32_t src2_pos, uint32_t size)
- *Subtract selected bits of src2 from selected bits of src1.* static __always_inline uint32_t ezdp_and (uint32_t src1, uint32_t src2, uint32_t src1_pos, uint32_t src2_pos, uint32_t size)
- *Perform logical 'AND' between selected bits of src1 and src2.* static __always_inline uint32_t ezdp_or (uint32_t src1, uint32_t src2, uint32_t src1_pos, uint32_t src2_pos, uint32_t size)
- *Perform logical 'OR' between selected bits of src1 and src2.* static __always_inline uint32_t ezdp_not (uint32_t src, uint32_t src_pos, uint32_t size)
- *Perform logical 'NOT' between selected bits of src1 and src2.* static __always_inline uint32_t ezdp_xor (uint32_t src1, uint32_t src2, uint32_t src1_pos, uint32_t src2_pos, uint32_t size)
- *Perform logical 'XOR' between selected bits of src1 and src2.* static __always_inline uint32_t ezdp_fxor8 (uint32_t src1, uint32_t src2, uint32_t src1_pos, uint32_t src2_pos, uint32_t size)
- *Apply an 8 bit 'folded xor' operation on selected bits of src1 and src2.* static __always_inline uint32_t ezdp_fxor16 (uint32_t src1, uint32_t src2, uint32_t src1_pos, uint32_t src2_pos, uint32_t size)
- *Apply a 16 bit 'folded xor' operation on selected bits of src1 and selected bits of src2.* static __always_inline uint32_t ezdp_shift_left (uint32_t src1, uint32_t src2, uint32_t src1_pos, uint32_t src2_pos, uint32_t size)
- *Perform a 'shift left' operation on a set of bits in src1, with shift size selected by 5 adjacent bits of src2.* static __always_inline uint32_t ezdp_shift_right (uint32_t src1, uint32_t src2, uint32_t src1_pos, uint32_t src2_pos, uint32_t size)
- *Perform a 'shift right' operation on a set of bits in src1, with shift size selected by 5 adjacent bits of src2.* static __always_inline uint32_t ezdp_count_bits (uint32_t src, uint32_t src_pos, uint32_t size)
- *Count the number of bits with value of '1' in a set of bits in src.* static __always_inline uint32_t ezdp_div (uint32_t src1, uint32_t src2, uint32_t src1_pos, uint32_t src2_pos)
- *Divide 8 selected bits of src1 by 4 selected bits of src2.* static __always_inline uint32_t ezdp_mod (uint32_t src1, uint32_t src2, uint32_t src1_pos, uint32_t src2_pos)
- *Modulus 8 selected bits of src1 by 4 selected bits of src2.* static __always_inline uint32_t ezdp_pow_of_2 (uint32_t exp, uint32_t dst_pos)
- *Calculate the value of 2^exp and place into any position in dst.* static __always_inline uint32_t ezdp_merge_pow_of_2 (uint32_t src, uint32_t exp, uint32_t dst_pos, uint32_t size)
- *Calculate the value of 2^exp and merge into any position in src.* static __always_inline uint32_t ezdp_set_bit (uint32_t src, uint32_t idx, uint32_t dst_pos, uint32_t size)

- *Set a single bit in src to 'one'.* static __always_inline uint32_t ezdp_clear_bit (uint32_t src, uint32_t idx, uint32_t dst_pos, uint32_t size)
- *Clear a single bit in src (sets to 'zero').* static __always_inline uint32_t ezdp_find_first_one (uint32_t src, uint32_t src_pos, uint32_t src_size)
- *Find the position of the first 'one' in the range src[src_pos+size-1 : src_pos] (from lsb to msb).* static __always_inline uint32_t ezdp_find_first_zero (uint32_t src, uint32_t src_pos, uint32_t src_size)
- *Find the position of the first 'zero' in the range src[src_pos+size-1 : src_pos] (from lsb to msb).* static __always_inline uint32_t ezdp_get_bitfield (uint32_t src, uint32_t dest_pos, uint32_t src_pos, uint32_t size)
- *Get a set of adjacent bits from src and place into any position in dst.* static __always_inline uint32_t ezdp_merge_bitfield (uint32_t src1, uint32_t src2, uint32_t dest_pos, uint32_t src2_pos, uint32_t size)
- *Get a set of adjacent bits from src2 and merge into any position in src1.* static __always_inline uint32_t ezdp_get_2_bitfields (uint32_t src1, uint32_t src2, uint32_t dest_pos1, uint32_t src1_pos, uint32_t size1, uint32_t dest_pos2, uint32_t src2_pos, uint32_t size2)
- *Get 2 sets of adjacent bits from src1 and src2 and place into two locations in dst.* static __always_inline uint32_t ezdp_merge_2_bitfields (uint32_t src1, uint32_t src2, uint32_t dest_pos1, uint32_t src1_pos, uint32_t size1, uint32_t dest_pos2, uint32_t src2_pos, uint32_t size2)
- *Get 2 sets of adjacent bits from src1 and src2 and merge into two locations in src1.* static __always_inline uint32_t ezdp_get_bit (uint32_t src, uint32_t dest_pos, uint32_t src_pos)
- *Get any bit from src and place in any position in dst.* static __always_inline uint32_t ezdp_merge_bit (uint32_t src1, uint32_t src2, uint32_t dest_pos, uint32_t src2_pos)
- *Get any bit from src2 and merge it any position in src1.* static __always_inline uint32_t ezdp_get_2_bits (uint32_t src, uint32_t dest_pos1, ezdp_bit_mode mode1, uint32_t src_pos1, uint32_t dest_pos2, ezdp_bit_mode mode2, uint32_t src_pos2)
- *Get two separate bits from src and place in two separate locations in dst.* static __always_inline uint32_t ezdp_merge_2_bits (uint32_t src1, uint32_t src2, uint32_t dest_pos1, ezdp_bit_mode mode1, uint32_t src2_pos1, uint32_t dest_pos2, ezdp_bit_mode mode2, uint32_t src2_pos2)
- *Get two separate bits from src2 and merge into two separate locations in src1.* static __always_inline uint32_t ezdp_get_3_bits (uint32_t src, uint32_t dest_pos1, ezdp_bit_mode mode1, uint32_t src_pos1, uint32_t dest_pos2, ezdp_bit_mode mode2, uint32_t src_pos2, uint32_t dest_pos3, ezdp_bit_mode mode3, uint32_t src_pos3)
- *Get three separate bits from src and place in three separate locations in dst.* static __always_inline uint32_t ezdp_merge_3_bits (uint32_t src1, uint32_t src2, uint32_t dest_pos1, ezdp_bit_mode mode1, uint32_t src2_pos1, uint32_t dest_pos2, ezdp_bit_mode mode2, uint32_t src2_pos2, uint32_t dest_pos3, ezdp_bit_mode mode3, uint32_t src2_pos3)
- *Get three separate bits from src2 and merge into three separate locations in src1.* static __always_inline uint32_t ezdp_get_4_bits (uint32_t src, uint32_t dest_pos1, ezdp_bit_mode mode1, uint32_t src_pos1, uint32_t dest_pos2, ezdp_bit_mode mode2, uint32_t src_pos2, uint32_t dest_pos3, ezdp_bit_mode mode3, uint32_t src_pos3, uint32_t dest_pos4, ezdp_bit_mode mode4, uint32_t src_pos4)
- *Get four separate bits from src and place in four separate locations in dst.* static __always_inline uint32_t ezdp_merge_4_bits (uint32_t src1, uint32_t src2, uint32_t dest_pos1, ezdp_bit_mode mode1, uint32_t src2_pos1, uint32_t dest_pos2, ezdp_bit_mode mode2, uint32_t src2_pos2, uint32_t dest_pos3, ezdp_bit_mode mode3, uint32_t src2_pos3, uint32_t dest_pos4, ezdp_bit_mode mode4, uint32_t src2_pos4)
- *Get four separate bits from src2 and merge into four separate locations in src1.* static __always_inline uint32_t ezdp_combine_4_bits (uint32_t src, uint32_t dest_pos, uint32_t src_pos1, uint32_t src_pos2, uint32_t src_pos3, uint32_t src_pos4)
- *Get four separate bits from src and place into 4 adjacent bits in dst.* static __always_inline uint32_t ezdp_combine_merge_4_bits (uint32_t src1, uint32_t src2, uint32_t dest_pos, uint32_t src2_pos1, uint32_t src2_pos2, uint32_t src2_pos3, uint32_t src2_pos4)
- *Get four separate bits from src2 to merge into 4 adjacent bits in src1.* static __always_inline uint32_t ezdp_split_4_bits (uint32_t src, uint32_t dest_pos1, uint32_t dest_pos2, uint32_t dest_pos3, uint32_t dest_pos4, uint32_t src_pos)
- *Get four adjacent bits from src and place in four separate positions in destination.* static __always_inline uint32_t ezdp_split_merge_4_bits (uint32_t src1, uint32_t src2, uint32_t dest_pos1, uint32_t dest_pos2, uint32_t dest_pos3, uint32_t dest_pos4, uint32_t src2_pos)
- *Get four adjacent bits from src2 and merge into four separate positions in src1.* static __always_inline uint32_t ezdp_get_4_bytes (uint32_t src1, uint32_t src2, ezasm_src_pos_index index0, ezasm_src_pos_index index1, ezasm_src_pos_index index2, ezasm_src_pos_index index3)
- *Extract any four bytes from src1 and src2.* static __always_inline uint32_t ezdp_reflect_bits (uint32_t data, ezdp_reflect_resolution resolution)

- *Perform bit swap in resolution of 1, 2 or 4 bytes.* static __always_inline uint32_t ezdp_hash (uint32_t src1, uint32_t src2, uint32_t hash_size, uint32_t input_size, uint32_t input_offset, ezdp_hash_base_matrix base_matrix, const ezdp_hash_permutation perm)
- *General purpose hash function.* static __always_inline uint32_t ezdp_hash32 (uint32_t src, uint32_t hash_size, uint32_t permut_id, uint32_t base_matrix)
- *General purpose hash function for 32-bit input.* static __always_inline uint32_t ezdp_hash64 (uint32_t src1, uint32_t src2, uint32_t hash_size, uint32_t permut_id, uint32_t base_matrix)
- *General purpose hash function for 64-bit input.* static __always_inline uint32_t ezdp_bulk_hash (uint8_t __cmem *data, uint32_t size)
- *General purpose hash function for up to 64-byte input.* static __always_inline uint32_t ezdp_calc_crc16 (uint32_t crc_value, uint8_t input_value, bool input_value_bit_rflt)
- *Perform CRC16 calculation.* static __always_inline uint32_t ezdp_calc_crc32 (uint32_t crc_value, uint8_t input_value, bool input_value_bit_rflt)
- *Perform CRC32 calculation.* static __always_inline uint32_t ezdp_add_checksum (uint32_t checksum_value, uint32_t add_value)
- *Add value to checksum.* static __always_inline uint32_t ezdp_sub_checksum (uint32_t checksum_value, uint32_t sub_value)

Subtract value from checksum.

Enumeration Type Documentation

enum ezdp_bit_mode

Bit manipulation mode possible values.

Enumerator:

EZDP_BIT_MODE_VALUE Copy value of bit.

EZDP_BIT_MODE_INVERSE Copy inverse of bit.

EZDP_BIT_MODE_FALSE Copy false to bit - Clear bit.

EZDP_BIT_MODE_TRUE Copy true to bit - Set bit.

enum ezdp_reflect_resolution

Bit manipulation mode possible values.

Enumerator:

EZDP_REFLECT_RESOLUTION_1_BYTE Reflect 1 byte resolution.

EZDP_REFLECT_RESOLUTION_2_BYTE Reflect 2 byte resolution.

EZDP_REFLECT_RESOLUTION_4_BYTE Reflect 4 byte resolution.

enum ezdp_hash_base_matrix

Hash base matrix possible values.

Enumerator:

EZDP_HASH_BASE_MATRIX_HASH_BASE_MATRIX_0 The base matrix for the hashing will be 0.

EZDP_HASH_BASE_MATRIX_HASH_BASE_MATRIX_1 The base matrix for the hashing will be 1.

enum ezdp_hash_permutation

Hash permutation possible values.

Enumerator:

EZDP_HASH_PERMUTATION_0	The hashing function will use permutation 0.
EZDP_HASH_PERMUTATION_1	The hashing function will use permutation 1.
EZDP_HASH_PERMUTATION_2	The hashing function will use permutation 2.
EZDP_HASH_PERMUTATION_3	The hashing function will use permutation 3.

Function Documentation

static __always_inline uint32_t ezdp_add (uint32_t *src1*, uint32_t *src2*, uint32_t *src1_pos*, uint32_t *src2_pos*, uint32_t *size*) `[static]`

Add selected bits of src1 to selected bits of src2.

Parameters:

[in] *src1* - source 1
[in] *src2* - source 2
[in] *src1_pos* - source 1 starting bit position possible values are: 0, 8, 16, 32
[in] *src2_pos* - source 2 starting bit position possible values are: 0, 8, 16, 32
[in] *size* - number of bits to add

Returns:

The function returns: src1[src1_pos+size-1 : src1_pos] + src2[src2_pos+size-1 : src2_pos]

static __always_inline uint32_t ezdp_sub (uint32_t *src1*, uint32_t *src2*, uint32_t *src1_pos*, uint32_t *src2_pos*, uint32_t *size*) `[static]`

Subtract selected bits of src2 from selected bits of src1.

Parameters:

[in] *src1* - source 1
[in] *src2* - source 2
[in] *src1_pos* - source 1 starting bit position possible values are: 0, 8, 16, 32
[in] *src2_pos* - source 2 starting bit position possible values are: 0, 8, 16, 32
[in] *size* - number of bits to subtract

Returns:

The function returns: src1[src1_pos+size-1 : src1_pos] - src2[src2_pos+size-1 : src2_pos]

static __always_inline uint32_t ezdp_and (uint32_t *src1*, uint32_t *src2*, uint32_t *src1_pos*, uint32_t *src2_pos*, uint32_t *size*) `[static]`

Perform logical 'AND' between selected bits of src1 and src2.

Parameters:

[in] *src1* - source 1

[in] *src2* - source 2
[in] *src1_pos* - source 1 starting bit position possible values are: 0, 8, 16, 32
[in] *src2_pos* - source 2 starting bit position possible values are: 0, 8, 16, 32
[in] *size* - number of bits to apply 'and' to

Returns:

The function returns: src1[src1_pos+size-1 : src1_pos] & src2[src2_pos+size-1 : src2_pos]

static __always_inline uint32_t ezdp_or (uint32_t *src1*, uint32_t *src2*, uint32_t *src1_pos*, uint32_t *src2_pos*, uint32_t *size*) `[static]`

Perform logical 'OR' between selected bits of src1 and src2.

Parameters:

[in] *src1* - source 1
[in] *src2* - source 2
[in] *src1_pos* - source 1 starting bit position possible values are: 0, 8, 16, 32
[in] *src2_pos* - source 2 starting bit position possible values are: 0, 8, 16, 32
[in] *size* - number of bits to apply 'or' to

Returns:

The function returns: src1[src1_pos+size-1 : src1_pos] | src2[src2_pos+size-1 : src2_pos]

static __always_inline uint32_t ezdp_not (uint32_t *src*, uint32_t *src_pos*, uint32_t *size*) `[static]`

Perform logical 'NOT' between selected bits of src1 and src2.

Parameters:

[in] *src* - source 2
[in] *src_pos* - source 2 starting bit position possible values are: 0, 8, 16, 32
[in] *size* - number of bits to apply 'not' to

Returns:

The function returns: ~ src2[src2_pos+size-1 : src2_pos]

static __always_inline uint32_t ezdp_xor (uint32_t *src1*, uint32_t *src2*, uint32_t *src1_pos*, uint32_t *src2_pos*, uint32_t *size*) `[static]`

Perform logical 'XOR' between selected bits of src1 and src2.

Parameters:

[in] *src1* - source 1
[in] *src2* - source 2
[in] *src1_pos* - source 1 starting bit position possible values are: 0, 8, 16, 32
[in] *src2_pos* - source 2 starting bit position possible values are: 0, 8, 16, 32
[in] *size* - number of bits to apply 'xor' to

Returns:

The function returns: src1[src1_pos+size-1 : src1_pos] ^ src2[src2_pos+size-1 : src2_pos]

static __always_inline uint32_t ezdp_fxor8 (uint32_t *src1*, uint32_t *src2*, uint32_t *src1_pos*, uint32_t *src2_pos*, uint32_t *size*) `[static]`

Apply an 8 bit 'folded xor' operation on selected bits of src1 and src2.

The selected bits from sources are padded with zeros to 32-bits.

Parameters:

[in] *src1* - source 1
[in] *src2* - source 2
[in] *src1_pos* - source 1 starting bit position possible values are: 0, 8, 16, 32
[in] *src2_pos* - source 2 starting bit position possible values are: 0, 8, 16, 32
[in] *size* - number of bits to apply operation on

Returns:

The function returns: folded_xor8(src1[src1_pos+size-1 : src1_pos], src2[src2_pos+size-1 : src2_pos])

static __always_inline uint32_t ezdp_fxor16 (uint32_t *src1*, uint32_t *src2*, uint32_t *src1_pos*, uint32_t *src2_pos*, uint32_t *size*) `[static]`

Apply a 16 bit 'folded xor' operation on selected bits of src1 and selected bits of src2.

The selected bits from sources are padded with zeros to 32-bits.

Parameters:

[in] *src1* - source 1
[in] *src2* - source 2
[in] *src1_pos* - source 1 starting bit position possible values are: 0, 8, 16, 32
[in] *src2_pos* - source 2 starting bit position possible values are: 0, 8, 16, 32
[in] *size* - number of bits to apply operation on

Returns:

The function returns: folded_xor16(src1[src1_pos+size-1 : src1_pos], src2[src2_pos+size-1 : src2_pos])

static __always_inline uint32_t ezdp_shift_left (uint32_t *src1*, uint32_t *src2*, uint32_t *src1_pos*, uint32_t *src2_pos*, uint32_t *size*) `[static]`

Perform a 'shift left' operation on a set of bits in src1, with shift size selected by 5 adjacent bits of src2.

Parameters:

[in] *src1* - source 1
[in] *src2* - source 2 (only 4 bits are used)
[in] *src1_pos* - source 1 starting bit position possible values are: 0, 8, 16, 32
[in] *src2_pos* - source 2 starting bit position possible values are: 0, 8, 16, 32
[in] *size* - number of bits to apply 'shift left' on

Note:

only 5 bits of the src2 are used.

Returns:

The function returns: src1[src1_pos+size-1 : src1_pos] << src2[src2_pos+4 : src2_pos]

static __always_inline uint32_t ezdp_shift_right (uint32_t *src1*, uint32_t *src2*, uint32_t *src1_pos*, uint32_t *src2_pos*, uint32_t *size*) `[static]`

Perform a 'shift right' operation on a set of bits in src1, with shift size selected by 5 adjacent bits of src2.

Parameters:

[in] *src1* - source 1
[in] *src2* - source 2 (only 4 bits are used)
[in] *src1_pos* - source 1 starting bit position possible values are: 0, 8, 16, 32
[in] *src2_pos* - source 2 starting bit position possible values are: 0, 8, 16, 32
[in] *size* - number of bits to apply 'shift right' on

Note:

only 5 bits of the src2 are used.

Returns:

The function returns: src1[src1_pos+size-1 : src1_pos] >> src2[src2_pos+4 : src2_pos]

static __always_inline uint32_t ezdp_count_bits (uint32_t *src*, uint32_t *src_pos*, uint32_t *size*) `[static]`

Count the number of bits with value of '1' in a set of bits in src.

Parameters:

[in] *src* - source
[in] *src_pos* - source 2 starting bit position possible values are: 0, 8, 16, 32
[in] *size* - number of bits to count '1' bits in

Returns:

The function returns the number of '1' bits in src2[src2_pos+size-1 : src2_pos]

static __always_inline uint32_t ezdp_div (uint32_t *src1*, uint32_t *src2*, uint32_t *src1_pos*, uint32_t *src2_pos*) `[static]`

Divide 8 selected bits of src1 by 4 selected bits of src2.

Parameters:

[in] *src1* - source 1
[in] *src2* - source 2
[in] *src1_pos* - source 1 starting bit position possible values are: 0, 8, 16, 32
[in] *src2_pos* - source 2 starting bit position possible values are: 0, 8, 16, 32

Returns:

The function returns: src1[src1_pos+7 : src1_pos] / src2[src2_pos+3 : src2_pos]

static __always_inline uint32_t ezdp_mod (uint32_t *src1*, uint32_t *src2*, uint32_t *src1_pos*, uint32_t *src2_pos*) `[static]`

Modulus 8 selected bits of src1 by 4 selected bits of src2.

Parameters:

[in] *src1* - source 1
[in] *src2* - source 2
[in] *src1_pos* - source 1 starting bit position possible values are: 0, 8, 16, 32
[in] *src2_pos* - source 2 starting bit position possible values are: 0, 8, 16, 32

Returns:

The function returns: src1[src1_pos+7 : src1_pos] % src2[src2_pos+3 : src2_pos]

static __always_inline uint32_t ezdp_pow_of_2 (uint32_t *exp*, uint32_t *dst_pos*) `[static]`

Calculate the value of 2^exp and place into any position in dst.

Parameters:

[in] *exp* - exponent value exp < 32
[in] *dst_pos* - the starting bit position of the field in the destination

Returns:

The function returns: result[31 : 0] = 0; result[dst_pos + exp] = 1;

static __always_inline uint32_t ezdp_merge_pow_of_2 (uint32_t *src*, uint32_t *exp*, uint32_t *dst_pos*, uint32_t *size*) [static]

Calculate the value of 2^exp and merge into any position in src.

Parameters:

[in] *src* - source
[in] *exp* - exponent value exp < size
[in] *dst_pos* - the starting bit position of the field in the destination 0 <= dst_pos <= 31
[in] *size* - the size of the field 1 <= size <= 32

Returns:

The function returns: result[31 : 0] = src; result[size-1+dst_pos : dst_pos] = 0; // the field is clears
result[dst_pos + exp] = 1; // the bit is set

static __always_inline uint32_t ezdp_set_bit (uint32_t *src*, uint32_t *idx*, uint32_t *dst_pos*, uint32_t *size*) [static]

Set a single bit in src to 'one'.

Parameters:

[in] *src* - source
[in] *idx* - bit to set in the field idx < size
[in] *dst_pos* - the starting bit position of the field in the destination 0 <= dst_pos <= 31
[in] *size* - the size of the field 1 <= size <= 32

Returns:

The function returns: result[31 : 0] = src; result[dst_pos + idx] = 1; // the bit is set

static __always_inline uint32_t ezdp_clear_bit (uint32_t *src*, uint32_t *idx*, uint32_t *dst_pos*, uint32_t *size*) [static]

Clear a single bit in src (sets to 'zero').

Parameters:

[in] *src* - source
[in] *idx* - bit to clear in the field idx < size
[in] *dst_pos* - the starting bit position of the field in the destination 0 <= dst_pos <= 31
[in] *size* - the size of the field 1 <= size <= 32

Returns:

The function returns: result[31 : 0] = src; result[dst_pos + idx] = 0; // the bit is cleared

static __always_inline uint32_t ezdp_find_first_one (uint32_t *src*, uint32_t *src_pos*, uint32_t *src_size*) [static]

Find the position of the first 'one' in the range src[src_pos+size-1 : src_pos] (from lsb to msb).

Note:

The index returned is relative to src_pos, counting from lsb to msb. If no 'one' is present, -1 is returned.

Parameters:

[in] *src* - source
[in] *src_pos* - source starting bit position
[in] *src_size* - number of bits to count from src

Returns:

The function returns the result calculated in the following way: K = find_first_one (src2[src2_pos+size-1 : src2_pos]); result = {0..0,K};

static __always_inline uint32_t ezdp_find_first_zero (uint32_t *src*, uint32_t *src_pos*, uint32_t *src_size*) [static]

Find the position of the first 'zero' in the range src[src_pos+size-1 : src_pos] (from lsb to msb).

Note:

The index returned is relative to src_pos, counting from lsb to msb. If no 'zero' is present, -1 is returned.

Parameters:

[in] *src* - source
[in] *src_pos* - source starting bit position
[in] *src_size* - number of bits to count from src

Returns:

The function returns the result calculated in the following way: K = find_first_zero (src2[src2_pos+size-1 : src2_pos]); result = {0..0,K};

static __always_inline uint32_t ezdp_get_bitfield (uint32_t *src*, uint32_t *dest_pos*, uint32_t *src_pos*, uint32_t *size*) [static]

Get a set of adjacent bits from src and place into any position in dst.

Parameters:

[in] *src* - source
[in] *dest_pos* - destination starting bit position
[in] *src_pos* - source starting bit position
[in] *size* - number of bits to extract

Returns:

The function returns the result calculated in the following way: result = 0; result[dest_pos+size-1 : dest_pos] = src[src_pos+size-1 : src_pos]

static __always_inline uint32_t ezdp_merge_bitfield (uint32_t *src1*, uint32_t *src2*, uint32_t *dest_pos*, uint32_t *src2_pos*, uint32_t *size*) [static]

Get a set of adjacent bits from src2 and merge into any position in src1.

Parameters:

[in] *src1* - source 1
[in] *src2* - source 2
[in] *dest_pos* - destination starting bit position
[in] *src2_pos* - source 2 starting bit position
[in] *size* - number of bits to merge

Returns:

The function returns the result calculated in the following way: result = src1; result[dest_pos+size-1 : dest_pos = src2[src2_pos+size-1 : src2_pos]

static __always_inline uint32_t ezdp_get_2_bitfields (uint32_t *src1*, uint32_t *src2*, uint32_t *dest_pos1*, uint32_t *src1_pos*, uint32_t *size1*, uint32_t *dest_pos2*, uint32_t *src2_pos*, uint32_t *size2*) `[static]`

Get 2 sets of adjacent bits from src1 and src2 and place into two locations in dst.

Parameters:

[in] *src1* - source 1
[in] *src2* - source 2
[in] *dest_pos1* - destination starting bit position to put bits from source 1
[in] *src1_pos* - source 1 starting bit position
[in] *size1* - number of bits to extract from src1
[in] *dest_pos2* - destination starting bit position to put bits from source 2
[in] *src2_pos* - source 2 starting bit position
[in] *size2* - number of bits to extract from src2

Returns:

The function returns the result calculated in the following way: result = 0; result[dest_pos1+size1-1 : dest_pos1] = src1[src1_pos+size1-1 : src1_pos] result[dest_pos2+size2-1 : dest_pos2] = src2[src2_pos+size1-1 : src2_pos]

static __always_inline uint32_t ezdp_merge_2_bitfields (uint32_t *src1*, uint32_t *src2*, uint32_t *dest_pos1*, uint32_t *src1_pos*, uint32_t *size1*, uint32_t *dest_pos2*, uint32_t *src2_pos*, uint32_t *size2*) `[static]`

Get 2 sets of adjacent bits from src1 and src2 and merge into two locations in src1.

Parameters:

[in] *src1* - source 1
[in] *src2* - source 2
[in] *dest_pos1* - destination starting bit position to put bits from source 1
[in] *src1_pos* - source 1 starting bit position
[in] *size1* - number of bits to extract from src1
[in] *dest_pos2* - destination starting bit position to put bits from source 2
[in] *src2_pos* - source 2 starting bit position
[in] *size2* - number of bits to extract from src2

Returns:

The function returns the result calculated in the following way: result = src1; result[dest_pos1+size1-1 : dest_pos1] = src1[src1_pos+size1-1 : src1_pos] result[dest_pos2+size2-1 : dest_pos2] = src2[src2_pos+size1-1 : src2_pos]

static __always_inline uint32_t ezdp_get_bit (uint32_t *src*, uint32_t *dest_pos*, uint32_t *src_pos*) `[static]`

Get any bit from src and place in any position in dst.

Parameters:

[in] *src* - source
[in] *dest_pos* - destination bit position
[in] *src_pos* - source bit position

Returns:

The function returns the result calculated in the following way: result = 0; result[dst_pos] = src[src_pos]

static __always_inline uint32_t ezdp_merge_bit (uint32_t *src1*, uint32_t *src2*, uint32_t *dest_pos*, uint32_t *src2_pos*) `[static]`

Get any bit from src2 and merge it any position in src1.

Parameters:

[in] *src1* - source 1
[in] *src2* - source 2
[in] *dest_pos* - destination bit position
[in] *src2_pos* - source 2 bit position

Returns:

The function returns the result calculated in the following way: result = src1; result[dst_pos] = src2[src2_pos]

static __always_inline uint32_t ezdp_get_2_bits (uint32_t *src*, uint32_t *dest_pos1*, ezdp_bit_mode *mode1*, uint32_t *src_pos1*, uint32_t *dest_pos2*, ezdp_bit_mode *mode2*, uint32_t *src_pos2*) `[static]`

Get two separate bits from src and place in two separate locations in dst.

The value of each bit can be copied as is, inverted, or set to either true (1) or false (0).

Parameters:

[in] *src* - source
[in] *dest_pos1* - destination bit 1 position
[in] *mode1* - how to copy bit 1 (value, inverse, true, false)
[in] *src_pos1* - source bit 1 position
[in] *dest_pos2* - destination bit 2 position
[in] *mode2* - how to copy bit 2 (value, inverse, true, false)
[in] *src_pos2* - source bit 2 position

Returns:

The function returns the result calculated in the following way: result = 0; result[dest_pos1] = { src[src_pos1] or ~ src[src_pos1] or 1 or 0 } - based on mode1 result[dest_pos2] = { src[src_pos2] or ~ src[src_pos2] or 1 or 0 } - based on mode2

static __always_inline uint32_t ezdp_merge_2_bits (uint32_t *src1*, uint32_t *src2*, uint32_t *dest_pos1*, ezdp_bit_mode *mode1*, uint32_t *src2_pos1*, uint32_t *dest_pos2*, ezdp_bit_mode *mode2*, uint32_t *src2_pos2*) `[static]`

Get two separate bits from src2 and merge into two separate locations in src1.

The value of each bit can be copied as is, inverted, or set to either true (1) or false (0).

Parameters:

[in] *src1* - source 1
[in] *src2* - source 2
[in] *dest_pos1* - destination bit 1 position
[in] *mode1* - how to copy bit 1 (value, inverse, true, false)
[in] *src2_pos1* - source 2 bit 1 position
[in] *dest_pos2* - destination bit 2 position
[in] *mode2* - how to copy bit 2 (value, inverse, true, false)
[in] *src2_pos2* - source 2 bit 2 position

Returns:

The function returns the result calculated in the following way: result = src1; result[dest_pos1] = { src[src2_pos1] or ~ src[src2_pos1] or 1 or 0 } - based on mode1 result[dest_pos2] = { src[src2_pos2] or ~ src[src2_pos2] or 1 or 0 } - based on mode2

static __always_inline uint32_t ezdp_get_3_bits (uint32_t *src*, uint32_t *dest_pos1*, ezdp_bit_mode *mode1*, uint32_t *src_pos1*, uint32_t *dest_pos2*, ezdp_bit_mode *mode2*, uint32_t *src_pos2*, uint32_t *dest_pos3*, ezdp_bit_mode *mode3*, uint32_t *src_pos3*) `[static]`

Get three separate bits from src and place in three separate locations in dst.

The value of each bit can be copied as is, inverted, or set to either true (1) or false (0).

Parameters:

[in] *src* - source
[in] *dest_pos1* - destination bit1 position
[in] *mode1* - how to copy bit 1 (value, inverse, true, false)
[in] *src_pos1* - source bit 1 position
[in] *dest_pos2* - destination bit 2 position
[in] *mode2* - how to copy bit 2 (value, inverse, true, false)
[in] *src_pos2* - source bit 2 position
[in] *dest_pos3* - destination bit 3 position
[in] *mode3* - how to copy bit 3 (value, inverse, true, false)
[in] *src_pos3* - source bit 3 position

Returns:

The function returns the result calculated in the following way: result = 0; result[dest_pos1] = { src[src_pos1] or ~ src[src_pos1] or 1 or 0 } - based on mode1 result[dest_pos2] = { src[src_pos2] or ~ src[src_pos2] or 1 or 0 } - based on mode2 result[dest_pos3] = { src[src_pos3] or ~ src[src_pos3] or 1 or 0 } - based on mode3

static __always_inline uint32_t ezdp_merge_3_bits (uint32_t *src1*, uint32_t *src2*, uint32_t *dest_pos1*, ezdp_bit_mode *mode1*, uint32_t *src2_pos1*, uint32_t *dest_pos2*, ezdp_bit_mode *mode2*, uint32_t *src2_pos2*, uint32_t *dest_pos3*, ezdp_bit_mode *mode3*, uint32_t *src2_pos3*) `[static]`

Get three separate bits from src2 and merge into three separate locations in src1.

The value of each bit can be copied as is, inverted, or set to either true (1) or false (0).

Parameters:

[in] *src1* - source 1
[in] *src2* - source 2
[in] *dest_pos1* - destination bit 1 position
[in] *mode1* - how to copy bit 1 (value, inverse, true, false)
[in] *src2_pos1* - source 2 bit 1 position
[in] *dest_pos2* - destination bit 2 position
[in] *mode2* - how to copy bit 2 (value, inverse, true, false)
[in] *src2_pos2* - source 2 bit 2 position
[in] *dest_pos3* - destination bit 3 position
[in] *mode3* - how to copy bit 3 (value, inverse, true, false)
[in] *src2_pos3* - source 2 bit 3 position

Returns:

The function returns the result calculated in the following way: result = src1; result[dest_pos1] = { src[src2_pos1] or ~ src[src2_pos1] or 1 or 0 } - based on mode1 result[dest_pos2] = { src[src2_pos2] or ~ src[src2_pos2] or 1 or 0 } - based on mode2 result[dest_pos3] = { src[src2_pos3] or ~ src[src2_pos3] or 1 or 0 } - based on mode3

static __always_inline uint32_t ezdp_get_4_bits (uint32_t *src*, uint32_t *dest_pos1*, ezdp_bit_mode *mode1*, uint32_t *src_pos1*, uint32_t *dest_pos2*, ezdp_bit_mode *mode2*, uint32_t *src_pos2*, uint32_t *dest_pos3*, ezdp_bit_mode *mode3*, uint32_t *src_pos3*, uint32_t *dest_pos4*, ezdp_bit_mode *mode4*, uint32_t *src_pos4*) `[static]`

Get four separate bits from src and place in four separate locations in dst.

The value of each bit can be copied as is, inverted, or set to either true (1) or false (0).

Parameters:

[in] *src* - source
[in] *dest_pos1* - destination bit 1 position
[in] *mode1* - how to copy bit 1 (value, inverse, true, false)
[in] *src_pos1* - source bit 1 position
[in] *dest_pos2* - destination bit2 position
[in] *mode2* - how to copy bit 2 (value, inverse, true, false)
[in] *src_pos2* - source bit 2 position
[in] *dest_pos3* - destination bit3 position
[in] *mode3* - how to copy bit 3 (value, inverse, true, false)
[in] *src_pos3* - source bit 3 position
[in] *dest_pos4* - destination bit4 position
[in] *mode4* - how to copy bit 4 (value, inverse, true, false)
[in] *src_pos4* - source bit 4 position

Returns:

The function return result calculated in the following way: result = 0; result[dest_pos1] = { src[src_pos1] or ~ src[src_pos1] or 1 or 0 } - based on mode1 result[dest_pos2] = { src[src_pos2] or ~ src[src_pos2] or 1 or 0 } - based on mode2 result[dest_pos3] = { src[src_pos3] or ~ src[src_pos3] or 1 or 0 } - based on mode3 result[dest_pos4] = { src[src_pos4] or ~ src[src_pos4] or 1 or 0 } - based on mode4

static __always_inline uint32_t ezdp_merge_4_bits (uint32_t *src1*, uint32_t *src2*, uint32_t *dest_pos1*, ezdp_bit_mode *mode1*, uint32_t *src2_pos1*, uint32_t *dest_pos2*, ezdp_bit_mode *mode2*, uint32_t *src2_pos2*, uint32_t *dest_pos3*, ezdp_bit_mode *mode3*, uint32_t *src2_pos3*, uint32_t *dest_pos4*, ezdp_bit_mode *mode4*, uint32_t *src2_pos4*) `[static]`

Get four separate bits from src2 and merge into four separate locations in src1.

The value of each bit can be copied as is, inverted, or set to either true (1) or false (0).

Parameters:

[in] *src1* - source 1
[in] *src2* - source 2
[in] *dest_pos1* - destination bit 1 position
[in] *mode1* - how to copy bit 1 (value, inverse, true, false)
[in] *src2_pos1* - source 2 bit1 position
[in] *dest_pos2* - destination bit 2 position
[in] *mode2* - how to copy bit 2 (value, inverse, true, false)
[in] *src2_pos2* - source 2 bit 2 position
[in] *dest_pos3* - destination bit 3 position
[in] *mode3* - how to copy bit 3 (value, inverse, true, false)
[in] *src2_pos3* - source 2 bit 3 position
[in] *dest_pos4* - destination bit 4 position
[in] *mode4* - how to copy bit 4 (value, inverse, true, false)
[in] *src2_pos4* - source 2 bit 4 position

Returns:

The function return result calculated in the following way: result = src1; result[dest_pos1] = { src[src2_pos1] or ~ src[src2_pos1] or 1 or 0 } - based on mode1 result[dest_pos2] = { src[src2_pos2] or ~ src[src2_pos2] or 1 or 0 } - based on mode2 result[dest_pos3] = { src[src2_pos3] or ~ src[src2_pos3] or 1 or 0 } - based on mode3 result[dest_pos4] = { src[src2_pos4] or ~ src[src2_pos4] or 1 or 0 } - based on mode4

static __always_inline uint32_t ezdp_combine_4_bits (uint32_t *src*, uint32_t *dest_pos*, uint32_t *src_pos1*, uint32_t *src_pos2*, uint32_t *src_pos3*, uint32_t *src_pos4*) `[static]`

Get four separate bits from src and place into 4 adjacent bits in dst.

Parameters:

[in] *src* - source
[in] *dest_pos* - destination bit position
[in] *src_pos1* - source bit 1 position
[in] *src_pos2* - source bit 2 position
[in] *src_pos3* - source bit 3 position
[in] *src_pos4* - source bit 4 position

Returns:

The function returns the result calculated in the following way: result = 0; result[dest_pos] = src[src_pos1] result[dest_pos+1] = src[src_pos2] result[dest_pos+2] = src[src_pos3] result[dest_pos+3] = src[src_pos4]

static __always_inline uint32_t ezdp_combine_merge_4_bits (uint32_t *src1*, uint32_t *src2*, uint32_t *dest_pos*, uint32_t *src2_pos1*, uint32_t *src2_pos2*, uint32_t *src2_pos3*, uint32_t *src2_pos4*) `[static]`

Get four separate bits from src2 to merge into 4 adjacent bits in src1.

Parameters:

[in] *src1* - source 1
[in] *src2* - source 2
[in] *dest_pos* - destination bit position
[in] *src2_pos1* - source 2 bit 1 position
[in] *src2_pos2* - source 2 bit 2 position
[in] *src2_pos3* - source 2 bit 3 position
[in] *src2_pos4* - source 2 bit 4 position

Returns:

The function returns the result calculated in the following way: result = src1; result[dest_pos] = src[src2_pos1] result[dest_pos+1] = src[src2_pos2] result[dest_pos+2] = src[src2_pos3] result[dest_pos+3] = src[src2_pos4]

static __always_inline uint32_t ezdp_split_4_bits (uint32_t *src*, uint32_t *dest_pos1*, uint32_t *dest_pos2*, uint32_t *dest_pos3*, uint32_t *dest_pos4*, uint32_t *src_pos*) `[static]`

Get four adjacent bits from src and place in four separate positions in destination.

Parameters:

[in] *src* - source
[in] *dest_pos1* - destination bit 1 position
[in] *dest_pos2* - destination bit 2 position
[in] *dest_pos3* - destination bit 3 position
[in] *dest_pos4* - destination bit 4 position
[in] *src_pos* - source bit position

Returns:

The function returns the result calculated in the following way: result = 0; result[dest_pos1] = src[src_pos] result[dest_pos2] = src[src_pos+1] result[dest_pos3] = src[src_pos+2] result[dest_pos4] = src[src_pos4+3

static __always_inline uint32_t ezdp_split_merge_4_bits (uint32_t *src1*, uint32_t *src2*, uint32_t *dest_pos1*, uint32_t *dest_pos2*, uint32_t *dest_pos3*, uint32_t *dest_pos4*, uint32_t *src2_pos*) `[static]`

Get four adjacent bits from src2 and merge into four separate positions in src1.

Parameters:

[in] *src1* - source 1
[in] *src2* - source 2
[in] *dest_pos1* - destination bit1 position
[in] *dest_pos2* - destination bit2 position
[in] *dest_pos3* - destination bit3 position
[in] *dest_pos4* - destination bit4 position
[in] *src2_pos* - source 2 bit position

Returns:

The function returns the result calculated in the following way: result = src1; result[dest_pos1] = src2[src2_pos1] result[dest_pos2] = src2[src2_pos2] result[dest_pos3] = src2[src2_pos3] result[dest_pos4] = src2[src2_pos4]

static __always_inline uint32_t ezdp_get_4_bytes (uint32_t *src1*, uint32_t *src2*, ezasm_src_pos_index *index0*, ezasm_src_pos_index *index1*, ezasm_src_pos_index *index2*, ezasm_src_pos_index *index3*) `[static]`

Extract any four bytes from src1 and src2.

Each byte of result register gets the relevant byte from either src1 or src2 as indicated by the index fields.

Parameters:

[in] *src1* - source 1
[in] *src2* - source 2
[in] *index0* - the index of byte 0 in the result register
[in] *index1* - the index of byte 1 in the result register
[in] *index2* - the index of byte 2 in the result register
[in] *index3* - the index of byte 3 in the result register

Note:

index3 must be taken from src2 (index3 >= 4).

Returns:

The function returns the extracted 4 bytes

static __always_inline uint32_t ezdp_reflect_bits (uint32_t *data*, ezdp_reflect_resolution *resolution*) `[static]`

Perform bit swap in resolution of 1, 2 or 4 bytes.

Example: data = 0x00 00 00 01 resolution = 1: Result 0x00 00 00 80 data = 0x00 00 00 01 resolution = 2: Result 0x00 00 80 00 data = 0x00 00 00 01 resolution = 4: Result 0x80 00 00 00

Note:

In case of invalid resolution parameter, no reflection will be made.

Parameters:

[in] *data* - data for reflection
[in] *resolution* - reflection resolution: 1B, 2B or 4B.

Returns:

reflected data

static __always_inline uint32_t ezdp_hash (uint32_t *src1*, uint32_t *src2*, uint32_t *hash_size*, uint32_t *input_size*, uint32_t *input_offset*, ezdp_hash_base_matrix *base_matrix*, const ezdp_hash_permutation *perm*) `[static]`

General purpose hash function.

Different hash values can be obtained by using different base matrixes. The hashing will be applied to a window of bytes from the input src1.src2 (concatenated). The window is defined as input_size bytes, starting from the input_offset byte (starting the count from the MSB). Example: src1.src2 = [b0, b1, b2, b3].[b4, b5, b6, b7] input_size = 5 input_offset = 2 the window will be: [b2, b3, b4, b5, b6]

Parameters:

[in] *src1* - source 1 value (input byte 0 to 3)
[in] *src2* - source 2 value (input byte 4 to 7)
[in] *hash_size* - output hash size in bits 1 <= hash_size <= 32
[in] *input_size* - input size in bytes 1 <= input_size <= 8
[in] *input_offset* - input offset in bytes (starting from the MSB) 0 <= input_offset <= 3
[in] *base_matrix* - base matrix to use
[in] *perm* - permutation to use

Returns:

the hash result

static __always_inline uint32_t ezdp_hash32 (uint32_t *src*, uint32_t *hash_size*, uint32_t *permut_id*, uint32_t *base_matrix*) `[static]`

General purpose hash function for 32-bit input.

Different hash value can be obtained by using different permutations and base matrixes.

Parameters:

[in] *src* - source value
[in] *hash_size* - output hash size
[in] *permut_id* - permutation id
[in] *base_matrix* - base matrix to use

Returns:

32 bit hash value

static __always_inline uint32_t ezdp_hash64 (uint32_t *src1*, uint32_t *src2*, uint32_t *hash_size*, uint32_t *permut_id*, uint32_t *base_matrix*) `[static]`

General purpose hash function for 64-bit input.

Different hash value can be obtained by using different permutations and base matrixes.

Note:

Input of less than 64 bits should be aligned to lsb of src2

Parameters:

[in] *src1* - source 1 value
[in] *src2* - source 2 value
[in] *hash_size* - output hash size
[in] *permut_id* - permutation number
[in] *base_matrix* - base matrix to use

Returns:

32 bit hash result

static __always_inline uint32_t ezdp_bulk_hash (uint8_t __cmem * *data*, uint32_t *size*) [static]

General purpose hash function for up to 64-byte input.

Parameters:

[in] *data* - pointer to the data on CMEM
[in] *size* - size of the data

Returns:

32 bit hash result

static __always_inline uint32_t ezdp_calc_crc16 (uint32_t *crc_value*, uint8_t *input_value*, bool *input_value_bit_rflt*) [static]

Perform CRC16 calculation.

The calculation is using the standard polynomial with the following coefficients: 0x1021.

Parameters:

[in] *crc_value* - 16 LSB are the initial or previous CRC16 value
[in] *input_value* - next CRC input byte
[in] *input_value_bit_rflt* - whether the input value is reflected before the CRC calculation or not

Returns:

uint32_t - the 16 LSB represents the result of the CRC16 calculation

static __always_inline uint32_t ezdp_calc_crc32 (uint32_t *crc_value*, uint8_t *input_value*, bool *input_value_bit_rflt*) [static]

Perform CRC32 calculation.

The calculation is using the standard polynomial with the following coefficients: 0x04C11DB7.

Parameters:

[in] *crc_value* - initial or previous CRC32 value
[in] *input_value* - next CRC input byte
[in] *input_value_bit_rflt* - whether the input value is reflected before the CRC calculation or not

Returns:

uint32_t - the result of the CRC32 calculation

static __always_inline uint32_t ezdp_add_checksum (uint32_t *checksum_value*, uint32_t *add_value*) [static]

Add value to checksum.

Parameters:

[in] *checksum_value* - initial or intermediate checksum value
[in] *add_value* - value to be added to checksum

Note:

The checksum calculation assumes an even (2 byte aligned) offset. If the pointer is odd, the checksum result should be swapped.

Returns:

uin32_t - 16 bit new checksum result

static __always_inline uint32_t ezdp_sub_checksum (uint32_t *checksum_value*, uint32_t *sub_value*) `[static]`

Subtract value from checksum.

Parameters:

[in] *checksum_value* - initial or intermediate checksum value
[in] *sub_value* - value to be subtract from checksum

Note:

The checksum calculation assumes an even (2 byte aligned) offset. If the pointer is odd, the checksum result should be swapped.

Returns:

uin32_t - 16 bit new checksum result

dpe/dp/include/ezdp_memory.h File Reference

Functions

- static __always_inline uint32_t ezdp_calc_checksum_ext_addr (struct ezdp_ext_addr __cmem *src_ptr, uint32_t size, uint32_t flags)
- *Calculate checksum of data on extended address.* static __always_inline uint32_t ezdp_calc_checksum (void __cmem *ptr, uint32_t size)
- *Calculate checksum on a block of memory in CMEM.* static __always_inline bool ezdp_is_null_sum_addr (ezdp_sum_addr_t sum_addr)
- *Check if a summarized address is null.* static __always_inline ezdp_sum_addr_t ezdp_calc_sum_addr (ezdp_sum_addr_table_desc_t addr_desc, uint32_t entry_size, uint32_t key)
- *Calculate summarized address from address descriptor and key.* static __always_inline void ezdp_sum_addr_to_ext_addr (struct ezdp_ext_addr *ext_addr, ezdp_sum_addr_t sum_addr, uint32_t entry_size)
- *Calculate extended address from summarized address.* static __always_inline ezdp_sum_addr_t ezdp_ext_addr_to_sum_addr (struct ezdp_ext_addr *ext_addr, uint32_t entry_size)
- *Calculate summarized address from extended address.* static __always_inline uint32_t ezdp_calc_sum_addr_offset (struct ezdp_ext_addr *ext_addr, uint32_t entry_size)
- *Calculate offset of a summarized address from extended address.* static __always_inline ezdp_sum_addr_t ezdp_scramble_sum_addr (ezdp_sum_addr_t addr, const uint32_t entry_size)
- *Scramble given summarized address.* static __always_inline void ezdp_scramble_ext_addr (struct ezdp_ext_addr *addr, const uint32_t entry_size)

Scramble given extended address.

Function Documentation

static __always_inline uint32_t ezdp_calc_checksum_ext_addr (struct ezdp_ext_addr __cmem * *src_ptr*, uint32_t *size*, uint32_t *flags*) [static]

Calculate checksum of data on extended address.

Parameters:

[in] *src_ptr* - pointer in CMEM to the extended address
[in] *size* - the amount of data to sum in byte
[in] *flags* - execution flags. Bitwise OR of zero or more flags out of the following:
EZDP_MEMORY_FLAG_UNCACHED - do not allocate new cache entry

Note:

The checksum calculation assumes an even (2 byte aligned) offset. If the pointer is odd, the checksum result should be swapped.

Returns:

uin32_t - the calculated checksum value

static __always_inline uint32_t ezdp_calc_checksum (void __cmem * *ptr*, uint32_t *size*) [static]

Calculate checksum on a block of memory in CMEM.

Parameters:

[in] *ptr* - pointer to CMEM to calculate checksum on
[in] *size* - number of bytes to sum

Note:

The checksum calculation assumes an even (2 byte aligned) offset. If the pointer is odd, the checksum result should be swapped.

Returns:

uin32_t - 16 bit checksum value

static __always_inline bool ezdp_is_null_sum_addr (ezdp_sum_addr_t *sum_addr*) [static]

Check if a summarized address is null.

Parameters:

[in] *sum_addr* - summarized address

Returns:

bool - true iff the sum_addr == EZDP_NULL_SUM_ADDR

static __always_inline ezdp_sum_addr_t ezdp_calc_sum_addr (ezdp_sum_addr_table_desc_t *addr_desc*, uint32_t *entry_size*, uint32_t *key*) [static]

Calculate summarized address from address descriptor and key.

Parameters:

[in] *addr_desc* - table summarized address descriptor
[in] *entry_size* - size of a single entry
[in] *key* - index into table

Returns:

uint32_t - the calculated summarized address

static __always_inline void ezdp_sum_addr_to_ext_addr (struct ezdp_ext_addr * *ext_addr*, ezdp_sum_addr_t *sum_addr*, uint32_t *entry_size*) [static]

Calculate extended address from summarized address.

Parameters:

[out] *ext_addr* - pointer to the extended address
[in] *sum_addr* - summarized address
[in] *entry_size* - size of entry

Returns:

void

static __always_inline ezdp_sum_addr_t ezdp_ext_addr_to_sum_addr (struct ezdp_ext_addr * *ext_addr*, uint32_t *entry_size*) [static]

Calculate summarized address from extended address.

Parameters:

[in] *ext_addr* - pointer to extended address
[in] *entry_size* - size of entry

Note:

If the extended address is not a multiple of the entry_size, call ezdp_calc_sum_addr_offset() to calculate the offset of the summarized address.

Returns:

ezdp_sum_addr_t - the calculated summarized address

static __always_inline uint32_t ezdp_calc_sum_addr_offset (struct ezdp_ext_addr * *ext_addr*, uint32_t *entry_size*) `[static]`

Calculate offset of a summarized address from extended address.

Parameters:

[in] *ext_addr* - pointer to extended address
[in] *entry_size* - size of entry

Returns:

uint32_t - the offset of the summarized address

static __always_inline ezdp_sum_addr_t ezdp_scramble_sum_addr (ezdp_sum_addr_t *addr*, const uint32_t *entry_size*) `[static]`

Scramble given summarized address.

Parameters:

[in] *addr* - summarized address
[in] *entry_size* - size of entry (power of 2)

Note:

For entry size equal/bigger than 64 or equal/smaller than 8, function will return the same address (no scrambling will be done) For such sizes scrambling DDR memory scatter is not required (done internally by HW)

Returns:

scrambled ezdp_sum_addr_t

static __always_inline void ezdp_scramble_ext_addr (struct ezdp_ext_addr * *addr*, const uint32_t *entry_size*) `[static]`

Scramble given extended address.

Parameters:

[in,out] *addr* - extended address
[in] *entry_size* - size of entry (power of 2)

Note:

For entry size equal/bigger than 64 or equal/smaller than 8, function will return the same address (no scrambling will be done) For such sizes scrambling DDR memory scatter is not required (done internally by HW)

Returns:

void

dpe/dp/include/ezdp_memory_defs.h File Reference

Data Structures

- struct ezdp_ext_addr
- *Extended address definition.* struct ezdp_sum_addr
- *Summarized Address data structure.* struct ezdp_pci_addr
- *PCI Address data structure.* struct ezdp_sum_addr_table_desc
- *Structure definition table entry data structure.* struct ezdp_dual_add32_result
- *The result of the atomic dual add32 instruction.* struct ezdp_dual_add64_result

The result of the atomic dual add64 instruction. Defines

- #define EZDP_EXT_ADDR_ADDRESS_MSB_SIZE 4
- #define EZDP_EXT_ADDR_ADDRESS_MSB_OFFSET 0
- #define EZDP_EXT_ADDR_ADDRESS_MSB_WORD_SELECT 0
- #define EZDP_EXT_ADDR_ADDRESS_MSB_WORD_OFFSET 0
- #define EZDP_EXT_ADDR_RESERVED4_7_SIZE 4
- #define EZDP_EXT_ADDR_RESERVED4_7_OFFSET 4
- #define EZDP_EXT_ADDR_MSID_SIZE 5
- #define EZDP_EXT_ADDR_MSID_OFFSET 8
- #define EZDP_EXT_ADDR_MSID_WORD_SELECT 0
- #define EZDP_EXT_ADDR_MSID_WORD_OFFSET 8
- #define EZDP_EXT_ADDR_MEM_TYPE_SIZE 1
- #define EZDP_EXT_ADDR_MEM_TYPE_OFFSET 13
- #define EZDP_EXT_ADDR_MEM_TYPE_WORD_SELECT 0
- #define EZDP_EXT_ADDR_MEM_TYPE_WORD_OFFSET 13
- #define EZDP_EXT_ADDR_MEM_TYPE_MASK (1 << EZDP_EXT_ADDR_MEM_TYPE_WORD_OFFSET)
- #define EZDP_EXT_ADDR_RESERVED14_15_SIZE 2
- #define EZDP_EXT_ADDR_RESERVED14_15_OFFSET 14
- #define EZDP_EXT_ADDR_RESERVED16_31_SIZE 16
- #define EZDP_EXT_ADDR_RESERVED16_31_OFFSET 16
- #define EZDP_EXT_ADDR_ADDRESS_SIZE 32
- #define EZDP_EXT_ADDR_ADDRESS_OFFSET 32
- #define EZDP_EXT_ADDR_ADDRESS_WORD_SELECT 1
- #define EZDP_EXT_ADDR_ADDRESS_WORD_OFFSET 0
- #define EZDP_EXT_ADDR_WORD_COUNT 2
- #define EZDP_SUM_ADDR_ELEMENT_INDEX_SIZE 27
- #define EZDP_SUM_ADDR_ELEMENT_INDEX_OFFSET 0
- #define EZDP_SUM_ADDR_MSID_SIZE 4
- #define EZDP_SUM_ADDR_MSID_OFFSET 27
- #define EZDP_SUM_ADDR_MEM_TYPE_SIZE 1
- #define EZDP_SUM_ADDR_MEM_TYPE_OFFSET 31
- #define EZDP_SUM_ADDR_MEM_TYPE_MASK (1 << EZDP_SUM_ADDR_MEM_TYPE_OFFSET)
- #define EZDP_PCI_ADDR_ADDRESS_MSB_SIZE 4
- #define EZDP_PCI_ADDR_ADDRESS_MSB_OFFSET 0
- #define EZDP_PCI_ADDR_ADDRESS_MSB_WORD_SELECT 0
- #define EZDP_PCI_ADDR_ADDRESS_MSB_WORD_OFFSET 0
- #define EZDP_PCI_ADDR_RESERVED4_7_SIZE 4
- #define EZDP_PCI_ADDR_RESERVED4_7_OFFSET 4
- #define EZDP_PCI_ADDR_MSID_SIZE 5
- #define EZDP_PCI_ADDR_MSID_OFFSET 8
- #define EZDP_PCI_ADDR_MSID_WORD_SELECT 0
- #define EZDP_PCI_ADDR_MSID_WORD_OFFSET 8
- #define EZDP_PCI_ADDR_MEM_TYPE_SIZE 1
- #define EZDP_PCI_ADDR_MEM_TYPE_OFFSET 13

- #define EZDP_PCI_ADDR_MEM_TYPE_WORD_SELECT 0
- #define EZDP_PCI_ADDR_MEM_TYPE_WORD_OFFSET 13
- #define EZDP_PCI_ADDR_MEM_TYPE_MASK (1 << EZDP_PCI_ADDR_MEM_TYPE_WORD_OFFSET)
- #define EZDP_PCI_ADDR_RESERVED14_15_SIZE 2
- #define EZDP_PCI_ADDR_RESERVED14_15_OFFSET 14
- #define EZDP_PCI_ADDR_VIRT_FUNC_SIZE 8
- #define EZDP_PCI_ADDR_VIRT_FUNC_OFFSET 16
- #define EZDP_PCI_ADDR_VIRT_FUNC_WORD_SELECT 0
- #define EZDP_PCI_ADDR_VIRT_FUNC_WORD_OFFSET 16
- #define EZDP_PCI_ADDR_PHY_FUNC_SIZE 4
- #define EZDP_PCI_ADDR_PHY_FUNC_OFFSET 24
- #define EZDP_PCI_ADDR_PHY_FUNC_WORD_SELECT 0
- #define EZDP_PCI_ADDR_PHY_FUNC_WORD_OFFSET 24
- #define EZDP_PCI_ADDR_VIRT_FUNC_EN_SIZE 1
- #define EZDP_PCI_ADDR_VIRT_FUNC_EN_OFFSET 28
- #define EZDP_PCI_ADDR_VIRT_FUNC_EN_WORD_SELECT 0
- #define EZDP_PCI_ADDR_VIRT_FUNC_EN_WORD_OFFSET 28
- #define EZDP_PCI_ADDR_VIRT_FUNC_EN_MASK (1 << EZDP_PCI_ADDR_VIRT_FUNC_EN_WORD_OFFSET)
- #define EZDP_PCI_ADDR_RESERVED29_30_SIZE 2
- #define EZDP_PCI_ADDR_RESERVED29_30_OFFSET 29
- #define EZDP_PCI_ADDR_ADDR_TYPE_SIZE 1
- #define EZDP_PCI_ADDR_ADDR_TYPE_OFFSET 31
- #define EZDP_PCI_ADDR_ADDR_TYPE_WORD_SELECT 0
- #define EZDP_PCI_ADDR_ADDR_TYPE_WORD_OFFSET 31
- #define EZDP_PCI_ADDR_ADDR_TYPE_MASK (1 << EZDP_PCI_ADDR_ADDR_TYPE_WORD_OFFSET)
- #define EZDP_PCI_ADDR_ADDRESS_SIZE 32
- #define EZDP_PCI_ADDR_ADDRESS_OFFSET 32
- #define EZDP_PCI_ADDR_ADDRESS_WORD_SELECT 1
- #define EZDP_PCI_ADDR_ADDRESS_WORD_OFFSET 0
- #define EZDP_PCI_ADDR_WORD_COUNT 2
- #define EZDP_SUM_ADDR_TABLE_DESC_BASE_INDEX_SIZE 16
- #define EZDP_SUM_ADDR_TABLE_DESC_BASE_INDEX_OFFSET 0
- #define EZDP_SUM_ADDR_TABLE_DESC_MSID_SIZE 4
- #define EZDP_SUM_ADDR_TABLE_DESC_MSID_OFFSET 16
- #define EZDP_SUM_ADDR_TABLE_DESC_KEY_SIZE_SIZE 5
- #define EZDP_SUM_ADDR_TABLE_DESC_KEY_SIZE_OFFSET 20
- #define EZDP_SUM_ADDR_TABLE_DESC_RESERVED25_26_SIZE 2
- #define EZDP_SUM_ADDR_TABLE_DESC_RESERVED25_26_OFFSET 25
- #define EZDP_SUM_ADDR_TABLE_DESC_MEM_TYPE_SIZE 1
- #define EZDP_SUM_ADDR_TABLE_DESC_MEM_TYPE_OFFSET 27
- #define EZDP_SUM_ADDR_TABLE_DESC_MEM_TYPE_MASK (1 << EZDP_SUM_ADDR_TABLE_DESC_MEM_TYPE_OFFSET)
- #define EZDP_SUM_ADDR_TABLE_DESC_KEY_SHUFF_BITS_SIZE 3
- #define EZDP_SUM_ADDR_TABLE_DESC_KEY_SHUFF_BITS_OFFSET 28
- #define EZDP_SUM_ADDR_TABLE_DESC_KEY_SHUFF_EN_SIZE 1
- #define EZDP_SUM_ADDR_TABLE_DESC_KEY_SHUFF_EN_OFFSET 31
- #define EZDP_SUM_ADDR_TABLE_DESC_KEY_SHUFF_EN_MASK (1 << EZDP_SUM_ADDR_TABLE_DESC_KEY_SHUFF_EN_OFFSET)
- #define EZDP_DUAL_ADD32_RESULT_ORIGINAL_VALUE2_SIZE 32
- #define EZDP_DUAL_ADD32_RESULT_ORIGINAL_VALUE2_OFFSET 0
- #define EZDP_DUAL_ADD32_RESULT_ORIGINAL_VALUE2_WORD_SELECT 0
- #define EZDP_DUAL_ADD32_RESULT_ORIGINAL_VALUE2_WORD_OFFSET 0
- #define EZDP_DUAL_ADD32_RESULT_ORIGINAL_VALUE1_SIZE 32
- #define EZDP_DUAL_ADD32_RESULT_ORIGINAL_VALUE1_OFFSET 32

- #define EZDP_DUAL_ADD32_RESULT_ORIGINAL_VALUE1_WORD_SELECT 1
- #define EZDP_DUAL_ADD32_RESULT_ORIGINAL_VALUE1_WORD_OFFSET 0
- #define EZDP_DUAL_ADD32_RESULT_WORD_COUNT 2
- #define EZDP_DUAL_ADD64_RESULT_ORIGINAL_VALUE2_SIZE 64
- #define EZDP_DUAL_ADD64_RESULT_ORIGINAL_VALUE2_OFFSET 0
- #define EZDP_DUAL_ADD64_RESULT_ORIGINAL_VALUE2_WORD_SELECT 0
- #define EZDP_DUAL_ADD64_RESULT_ORIGINAL_VALUE2_WORD_OFFSET 0
- #define EZDP_DUAL_ADD64_RESULT_ORIGINAL_VALUE1_SIZE 64
- #define EZDP_DUAL_ADD64_RESULT_ORIGINAL_VALUE1_OFFSET 64
- #define EZDP_DUAL_ADD64_RESULT_ORIGINAL_VALUE1_WORD_SELECT 2
- #define EZDP_DUAL_ADD64_RESULT_ORIGINAL_VALUE1_WORD_OFFSET 0
- #define EZDP_DUAL_ADD64_RESULT_WORD_COUNT 4
- #define EZDP_NULL_SUM_ADDR 0x00000000

Typedefs

- typedef uint32_t ezdp_sum_addr_t
- typedef uint32_t ezdp_sum_addr_table_desc_t

Enumerations

- enum ezdp_internal_mem_space { EZDP_PRIVATE_DATA = 0x0, EZDP_STACK = 0x1, EZDP_HALF_CLUSTER_DATA = 0x2, EZDP_HALF_CLUSTER_CODE = 0x3, EZDP_1_CLUSTER_DATA = 0x4, EZDP_1_CLUSTER_CODE = 0x5, EZDP_2_CLUSTER_DATA = 0x6, EZDP_2_CLUSTER_CODE = 0x7, EZDP_4_CLUSTER_DATA = 0x8, EZDP_4_CLUSTER_CODE = 0x9, EZDP_16_CLUSTER_DATA = 0xa, EZDP_16_CLUSTER_CODE = 0xb, EZDP_ALL_CLUSTER_DATA = 0xc, EZDP_ALL_CLUSTER_CODE = 0xd, EZDP_ALL_CLUSTER_IO = 0xe, EZDP_ALL_CLUSTER_DATA_EXT_MEM = 0xf }

  Internal memory space types.
- enum ezdp_mem_space_type { EZDP_INTERNAL_MS = 0x0, EZDP_EXTERNAL_MS = 0x1 }

  mem space type possible values.
- enum ezdp_pci_addr_type { EZDP_PCI_ADDR_TYPE_UNTRANSLATED = 0x0, EZDP_PCI_ADDR_TYPE_TRANSLATED = 0x1 }

  pci addr type possible values.
- enum ezdp_dma_flags { EZDP_MEMORY_FLAG_CLASS_0 = 0, EZDP_MEMORY_FLAG_CLASS_1 = EZDP_MEMORY_IMP_FLAG_CLASS_BIT_0, EZDP_MEMORY_FLAG_CLASS_2 = EZDP_MEMORY_IMP_FLAG_CLASS_BIT_1, EZDP_MEMORY_FLAG_CLASS_3 = EZDP_MEMORY_IMP_FLAG_CLASS_BIT_0 | EZDP_MEMORY_IMP_FLAG_CLASS_BIT_1, EZDP_MEMORY_FLAG_UNCACHED = EZDP_MEMORY_IMP_FLAG_UNCACHED, EZDP_MEMORY_FLAG_OVERWRITE = EZDP_MEMORY_IMP_FLAG_OVERWRITE, EZDP_PCI_FLAG_NO_SNOOP = EZDP_PCI_IMP_FLAG_NO_SNOOP, EZDP_PCI_FLAG_RELEX_ORDERED = EZDP_PCI_IMP_FLAG_RELEX_ORDERED, EZDP_PCI_FLAG_ATU_BYPASS = EZDP_PCI_IMP_FLAG_ATU_BYPASS }

  dma flags.

Define Documentation

#define EZDP_EXT_ADDR_ADDRESS_MSB_SIZE 4

#define EZDP_EXT_ADDR_ADDRESS_MSB_OFFSET 0

#define EZDP_EXT_ADDR_ADDRESS_MSB_WORD_SELECT 0

#define EZDP_EXT_ADDR_ADDRESS_MSB_WORD_OFFSET 0

#define EZDP_EXT_ADDR_RESERVED4_7_SIZE 4

#define EZDP_EXT_ADDR_RESERVED4_7_OFFSET 4

#define EZDP_EXT_ADDR_MSID_SIZE 5

#define EZDP_EXT_ADDR_MSID_OFFSET 8

#define EZDP_EXT_ADDR_MSID_WORD_SELECT 0

#define EZDP_EXT_ADDR_MSID_WORD_OFFSET 8

#define EZDP_EXT_ADDR_MEM_TYPE_SIZE 1

#define EZDP_EXT_ADDR_MEM_TYPE_OFFSET 13

#define EZDP_EXT_ADDR_MEM_TYPE_WORD_SELECT 0

#define EZDP_EXT_ADDR_MEM_TYPE_WORD_OFFSET 13

#define EZDP_EXT_ADDR_MEM_TYPE_MASK (1 << EZDP_EXT_ADDR_MEM_TYPE_WORD_OFFSET)

#define EZDP_EXT_ADDR_RESERVED14_15_SIZE 2

#define EZDP_EXT_ADDR_RESERVED14_15_OFFSET 14

#define EZDP_EXT_ADDR_RESERVED16_31_SIZE 16

#define EZDP_EXT_ADDR_RESERVED16_31_OFFSET 16

#define EZDP_EXT_ADDR_ADDRESS_SIZE 32

#define EZDP_EXT_ADDR_ADDRESS_OFFSET 32

#define EZDP_EXT_ADDR_ADDRESS_WORD_SELECT 1

#define EZDP_EXT_ADDR_ADDRESS_WORD_OFFSET 0

#define EZDP_EXT_ADDR_WORD_COUNT 2

#define EZDP_SUM_ADDR_ELEMENT_INDEX_SIZE 27

```
#define EZDP_SUM_ADDR_ELEMENT_INDEX_OFFSET 0

#define EZDP_SUM_ADDR_MSID_SIZE 4

#define EZDP_SUM_ADDR_MSID_OFFSET 27

#define EZDP_SUM_ADDR_MEM_TYPE_SIZE 1

#define EZDP_SUM_ADDR_MEM_TYPE_OFFSET 31

#define EZDP_SUM_ADDR_MEM_TYPE_MASK (1 << EZDP_SUM_ADDR_MEM_TYPE_OFFSET)

#define EZDP_PCI_ADDR_ADDRESS_MSB_SIZE 4

#define EZDP_PCI_ADDR_ADDRESS_MSB_OFFSET 0

#define EZDP_PCI_ADDR_ADDRESS_MSB_WORD_SELECT 0

#define EZDP_PCI_ADDR_ADDRESS_MSB_WORD_OFFSET 0

#define EZDP_PCI_ADDR_RESERVED4_7_SIZE 4

#define EZDP_PCI_ADDR_RESERVED4_7_OFFSET 4

#define EZDP_PCI_ADDR_MSID_SIZE 5

#define EZDP_PCI_ADDR_MSID_OFFSET 8

#define EZDP_PCI_ADDR_MSID_WORD_SELECT 0

#define EZDP_PCI_ADDR_MSID_WORD_OFFSET 8

#define EZDP_PCI_ADDR_MEM_TYPE_SIZE 1

#define EZDP_PCI_ADDR_MEM_TYPE_OFFSET 13

#define EZDP_PCI_ADDR_MEM_TYPE_WORD_SELECT 0

#define EZDP_PCI_ADDR_MEM_TYPE_WORD_OFFSET 13

#define EZDP_PCI_ADDR_MEM_TYPE_MASK (1 <<
EZDP_PCI_ADDR_MEM_TYPE_WORD_OFFSET)

#define EZDP_PCI_ADDR_RESERVED14_15_SIZE 2

#define EZDP_PCI_ADDR_RESERVED14_15_OFFSET 14

#define EZDP_PCI_ADDR_VIRT_FUNC_SIZE 8

#define EZDP_PCI_ADDR_VIRT_FUNC_OFFSET 16

#define EZDP_PCI_ADDR_VIRT_FUNC_WORD_SELECT 0
```

```
#define EZDP_PCI_ADDR_VIRT_FUNC_WORD_OFFSET 16

#define EZDP_PCI_ADDR_PHY_FUNC_SIZE 4

#define EZDP_PCI_ADDR_PHY_FUNC_OFFSET 24

#define EZDP_PCI_ADDR_PHY_FUNC_WORD_SELECT 0

#define EZDP_PCI_ADDR_PHY_FUNC_WORD_OFFSET 24

#define EZDP_PCI_ADDR_VIRT_FUNC_EN_SIZE 1

#define EZDP_PCI_ADDR_VIRT_FUNC_EN_OFFSET 28

#define EZDP_PCI_ADDR_VIRT_FUNC_EN_WORD_SELECT 0

#define EZDP_PCI_ADDR_VIRT_FUNC_EN_WORD_OFFSET 28

#define EZDP_PCI_ADDR_VIRT_FUNC_EN_MASK (1 <<
EZDP_PCI_ADDR_VIRT_FUNC_EN_WORD_OFFSET)

#define EZDP_PCI_ADDR_RESERVED29_30_SIZE 2

#define EZDP_PCI_ADDR_RESERVED29_30_OFFSET 29

#define EZDP_PCI_ADDR_ADDR_TYPE_SIZE 1

#define EZDP_PCI_ADDR_ADDR_TYPE_OFFSET 31

#define EZDP_PCI_ADDR_ADDR_TYPE_WORD_SELECT 0

#define EZDP_PCI_ADDR_ADDR_TYPE_WORD_OFFSET 31

#define EZDP_PCI_ADDR_ADDR_TYPE_MASK (1 <<
EZDP_PCI_ADDR_ADDR_TYPE_WORD_OFFSET)

#define EZDP_PCI_ADDR_ADDRESS_SIZE 32

#define EZDP_PCI_ADDR_ADDRESS_OFFSET 32

#define EZDP_PCI_ADDR_ADDRESS_WORD_SELECT 1

#define EZDP_PCI_ADDR_ADDRESS_WORD_OFFSET 0

#define EZDP_PCI_ADDR_WORD_COUNT 2

#define EZDP_SUM_ADDR_TABLE_DESC_BASE_INDEX_SIZE 16

#define EZDP_SUM_ADDR_TABLE_DESC_BASE_INDEX_OFFSET 0

#define EZDP_SUM_ADDR_TABLE_DESC_MSID_SIZE 4

#define EZDP_SUM_ADDR_TABLE_DESC_MSID_OFFSET 16
```

#define EZDP_SUM_ADDR_TABLE_DESC_KEY_SIZE_SIZE 5

#define EZDP_SUM_ADDR_TABLE_DESC_KEY_SIZE_OFFSET 20

#define EZDP_SUM_ADDR_TABLE_DESC_RESERVED25_26_SIZE 2

#define EZDP_SUM_ADDR_TABLE_DESC_RESERVED25_26_OFFSET 25

#define EZDP_SUM_ADDR_TABLE_DESC_MEM_TYPE_SIZE 1

#define EZDP_SUM_ADDR_TABLE_DESC_MEM_TYPE_OFFSET 27

#define EZDP_SUM_ADDR_TABLE_DESC_MEM_TYPE_MASK (1 << EZDP_SUM_ADDR_TABLE_DESC_MEM_TYPE_OFFSET)

#define EZDP_SUM_ADDR_TABLE_DESC_KEY_SHUFF_BITS_SIZE 3

#define EZDP_SUM_ADDR_TABLE_DESC_KEY_SHUFF_BITS_OFFSET 28

#define EZDP_SUM_ADDR_TABLE_DESC_KEY_SHUFF_EN_SIZE 1

#define EZDP_SUM_ADDR_TABLE_DESC_KEY_SHUFF_EN_OFFSET 31

#define EZDP_SUM_ADDR_TABLE_DESC_KEY_SHUFF_EN_MASK (1 << EZDP_SUM_ADDR_TABLE_DESC_KEY_SHUFF_EN_OFFSET)

#define EZDP_DUAL_ADD32_RESULT_ORIGINAL_VALUE2_SIZE 32

#define EZDP_DUAL_ADD32_RESULT_ORIGINAL_VALUE2_OFFSET 0

#define EZDP_DUAL_ADD32_RESULT_ORIGINAL_VALUE2_WORD_SELECT 0

#define EZDP_DUAL_ADD32_RESULT_ORIGINAL_VALUE2_WORD_OFFSET 0

#define EZDP_DUAL_ADD32_RESULT_ORIGINAL_VALUE1_SIZE 32

#define EZDP_DUAL_ADD32_RESULT_ORIGINAL_VALUE1_OFFSET 32

#define EZDP_DUAL_ADD32_RESULT_ORIGINAL_VALUE1_WORD_SELECT 1

#define EZDP_DUAL_ADD32_RESULT_ORIGINAL_VALUE1_WORD_OFFSET 0

#define EZDP_DUAL_ADD32_RESULT_WORD_COUNT 2

#define EZDP_DUAL_ADD64_RESULT_ORIGINAL_VALUE2_SIZE 64

#define EZDP_DUAL_ADD64_RESULT_ORIGINAL_VALUE2_OFFSET 0

#define EZDP_DUAL_ADD64_RESULT_ORIGINAL_VALUE2_WORD_SELECT 0

#define EZDP_DUAL_ADD64_RESULT_ORIGINAL_VALUE2_WORD_OFFSET 0

#define EZDP_DUAL_ADD64_RESULT_ORIGINAL_VALUE1_SIZE 64

#define EZDP_DUAL_ADD64_RESULT_ORIGINAL_VALUE1_OFFSET 64

#define EZDP_DUAL_ADD64_RESULT_ORIGINAL_VALUE1_WORD_SELECT 2

#define EZDP_DUAL_ADD64_RESULT_ORIGINAL_VALUE1_WORD_OFFSET 0

#define EZDP_DUAL_ADD64_RESULT_WORD_COUNT 4

#define EZDP_NULL_SUM_ADDR 0x00000000

Typedef Documentation

typedef uint32_t ezdp_sum_addr_t

typedef uint32_t ezdp_sum_addr_table_desc_t

Enumeration Type Documentation

enum ezdp_internal_mem_space

Internal memory space types.

Enumerator:

EZDP_PRIVATE_DATA Thread private data memory space (PDMEM).

EZDP_STACK PDEMEM is an extension to PDMEM with the ability to access/cache private data on EMEM with programmed window size mapped to LPMEM, providing low miss penalty on selected area (PDEMEM).

EZDP_HALF_CLUSTER_DATA Sub cluster data memory space type (LDMEM).

EZDP_HALF_CLUSTER_CODE Sub cluster code memory space type (LCMEM).

EZDP_1_CLUSTER_DATA Single cluster data memory space type (IDMEM1).

EZDP_1_CLUSTER_CODE Single cluster code memory space type (ICMEM1).

EZDP_2_CLUSTER_DATA Dual cluster data memory space type (IDMEM2).

EZDP_2_CLUSTER_CODE Dual cluster code memory space type (ICMEM2).

EZDP_4_CLUSTER_DATA Quad cluster data memory space type (IDMEM4).

EZDP_4_CLUSTER_CODE Quad cluster code memory space type (ICMEM4).

EZDP_16_CLUSTER_DATA 16 cluster data memory space type (IDMEM16).

EZDP_16_CLUSTER_CODE 16 cluster code memory space type (ICMEM16).

EZDP_ALL_CLUSTER_DATA All cluster data memory space type (IDMEMG).

EZDP_ALL_CLUSTER_CODE All cluster code memory space type (ICMEMG).

EZDP_ALL_CLUSTER_IO Entire SOC shared IO data (seen by all SOC threads and IO peripherals) (IDMEMGIO).

EZDP_ALL_CLUSTER_DATA_EXT_MEM External memory shared data (seen by all SOC threads) (EDMEMG).

enum ezdp_mem_space_type

mem space type possible values.

Enumerator:

EZDP_INTERNAL_MS Internal memory space.

msid is according to type of the internal memory space. See ezdp_internal_mem_space.

EZDP_EXTERNAL_MS External memory space.

msid is map, according to the configuration, to one of the configured MSID.

enum ezdp_pci_addr_type

pci addr type possible values.

Enumerator:

EZDP_PCI_ADDR_TYPE_UNTRANSLATED PCIe untranslated address.

EZDP_PCI_ADDR_TYPE_TRANSLATED PCIe translated address.

enum ezdp_dma_flags

dma flags.

Enumerator:

EZDP_MEMORY_FLAG_CLASS_0 Class 0 (Default class).

EZDP_MEMORY_FLAG_CLASS_1 Class 1.

EZDP_MEMORY_FLAG_CLASS_2 Class 2.

EZDP_MEMORY_FLAG_CLASS_3 Class 3.

EZDP_MEMORY_FLAG_UNCACHED Do not allocate new cache entry.

EZDP_MEMORY_FLAG_OVERWRITE Calculate the ECC without first reading the current ECC, then override the current ECC with the result.

EZDP_PCI_FLAG_NO_SNOOP PCI no snoop.

EZDP_PCI_FLAG_RELEX_ORDERED PCI relaxed order enable.

EZDP_PCI_FLAG_ATU_BYPASS PCI Address Translation Unit bypass enable.

dpe/dp/include/ezdp_pci.h File Reference

Functions

- static __always_inline void ezdp_init_pci_queue_desc (uint32_t endpoint, uint32_t queue_id, ezdp_pci_queue_desc_t *pci_queue_desc, char __cmem *work_area_ptr, uint32_t work_area_size)
- *Initialize and get PCI message queue description.* static __always_inline void ezdp_get_pci_msg (ezdp_pci_queue_desc_t *pci_queue_desc, uint32_t msg_index, struct ezdp_pci_msg __cmem *msg)
- *Get message from PCI queue according to given index.* static __always_inline void ezdp_set_pci_msgq_read_index (ezdp_pci_queue_desc_t *pci_queue_desc, uint32_t value, char __cmem *work_area_ptr, uint32_t work_area_size)
- *Set read index of PCI message queue.* static __always_inline void ezdp_set_pci_msgq_read_index_async (ezdp_pci_queue_desc_t *pci_queue_desc, uint32_t value, char __cmem *work_area_ptr, uint32_t work_area_size)
- *Set read index of PCI message queue.* static __always_inline uint32_t ezdp_get_pci_msgq_write_index (ezdp_pci_queue_desc_t *pci_queue_desc, char __cmem *work_area_ptr, uint32_t work_area_size)
- *Get write index of PCI message queue.* static __always_inline uint32_t ezdp_get_pci_msgq_read_index (ezdp_pci_queue_desc_t *pci_queue_desc, char __cmem *work_area_ptr, uint32_t work_area_size)
- *Get read index of PCI message queue.* static __always_inline uint32_t ezdp_copy_frame_data_to_pci (struct ezdp_pci_addr __cmem *dst_pci_ptr, struct ezdp_buffer_desc __cmem *src_bd_ptr, uint32_t src_bd_offset, uint32_t size, uint32_t traffic_class, uint32_t flags)
- *Copy data from a frame buffer to PCI address.* static __always_inline void ezdp_copy_frame_data_to_pci_async (struct ezdp_pci_addr __cmem *dst_pci_ptr, struct ezdp_buffer_desc __cmem *src_bd_ptr, uint32_t src_bd_offset, uint32_t size, uint32_t traffic_class, uint32_t flags)
- *Non blocking version of ezdp_copy_frame_data_to_pci().* static __always_inline void ezdp_copy_frame_data_from_pci (struct ezdp_buffer_desc __cmem *dst_bd_ptr, uint32_t dst_bd_offset, struct ezdp_pci_addr __cmem *src_pci_ptr, uint32_t size, uint32_t traffic_class, uint32_t flags)
- *Copy data from PCI address to a frame buffer.* static __always_inline void ezdp_copy_frame_data_from_pci_async (struct ezdp_buffer_desc __cmem *dst_bd_ptr, uint32_t dst_bd_offset, struct ezdp_pci_addr __cmem *src_pci_ptr, uint32_t size, uint32_t traffic_class, uint32_t flags)
- *Non blocking version of ezdp_copy_frame_data_from_pci().* static __always_inline void ezdp_load_data_from_pci (void __cmem *dst_ptr, struct ezdp_pci_addr __cmem *pci_ptr, uint32_t size, uint32_t traffic_class, uint32_t flags)
- *Copy data from a PCI address to CMEM.* static __always_inline void ezdp_load_data_from_pci_async (void __cmem *dst_ptr, struct ezdp_pci_addr __cmem *pci_ptr, uint32_t size, uint32_t traffic_class, uint32_t flags)
- *Non blocking version of ezdp_load_data_from_pci().* static __always_inline uint32_t ezdp_store_data_to_pci (struct ezdp_pci_addr __cmem *pci_ptr, void __cmem *src_ptr, uint32_t size, uint32_t traffic_class, uint32_t flags)
- *Copy data from CMEM to PCI address.* static __always_inline void ezdp_store_data_to_pci_async (struct ezdp_pci_addr __cmem *pci_ptr, void __cmem *src_ptr, uint32_t size, uint32_t traffic_class, uint32_t flags)
- *Non blocking version of ezdp_store_data_to_pci().* static __always_inline uint32_t ezdp_copy_pci_data_to_ext_addr (struct ezdp_ext_addr __cmem *dst_ptr, struct ezdp_pci_addr __cmem *src_pci_ptr, uint32_t size, uint32_t traffic_class, uint32_t flags)
- *Copy PCI data to extended addresses.* static __always_inline void ezdp_copy_pci_data_to_ext_addr_async (struct ezdp_ext_addr __cmem *dst_ptr, struct ezdp_pci_addr __cmem *src_pci_ptr, uint32_t size, uint32_t traffic_class, uint32_t flags)
- *Non blocking version of ezdp_copy_pci_data_to_ext_addr().* static __always_inline uint32_t ezdp_copy_pci_data_from_ext_addr (struct ezdp_pci_addr __cmem *dst_pci_ptr, struct ezdp_ext_addr __cmem *src_ptr, uint32_t size, uint32_t traffic_class, uint32_t flags)
- *Copy extended address data to PCI.* static __always_inline void ezdp_copy_pci_data_from_ext_addr_async (struct ezdp_pci_addr __cmem *dst_pci_ptr, struct ezdp_ext_addr __cmem *src_ptr, uint32_t size, uint32_t traffic_class, uint32_t flags)
- *Non blocking version of ezdp_copy_pci_data_from_ext_addr().* static __always_inline uint32_t ezdp_translate_pci_addr (void __cmem *dst_ptr, struct ezdp_pci_addr __cmem *pci_ptr, uint32_t traffic_class, uint32_t flags)
- *PCI address translation request; result will be saved in CMEM.* static __always_inline void ezdp_translate_pci_addr_async (void __cmem *dst_ptr, struct ezdp_pci_addr __cmem *pci_ptr, uint32_t traffic_class, uint32_t flags)

- *Non blocking version of ezdp_translate_pci_addr().* static __always_inline uint32_t ezdp_translate_pci_addr_to_ext_addr (struct ezdp_ext_addr __cmem *dst_ptr, struct ezdp_pci_addr __cmem *src_pci_ptr, uint32_t traffic_class, uint32_t flags)
- *Copy PCI data to extended addresses.* static __always_inline void ezdp_translate_pci_addr_to_ext_addr_async (struct ezdp_ext_addr __cmem *dst_ptr, struct ezdp_pci_addr __cmem *src_pci_ptr, uint32_t traffic_class, uint32_t flags)
- *Non blocking version of ezdp_translate_pci_addr_to_ext_addr().* static __always_inline uint32_t ezdp_send_message_to_pci (struct ezdp_pci_addr __cmem *pci_ptr, void __cmem *src_ptr, uint32_t size, uint32_t traffic_class, uint32_t msg_code, uint32_t flags)
- *Copy data from CMEM to PCI address.* static __always_inline void ezdp_send_message_to_pci_async (struct ezdp_pci_addr __cmem *pci_ptr, void __cmem *src_ptr, uint32_t size, uint32_t traffic_class, uint32_t msg_code, uint32_t flags)
- *Non blocking version of ezdp_send_message_to_pci().* static __always_inline void ezdp_send_interrupt_to_pci (uint32_t endpoint, uint32_t msid, uint32_t phy_func, uint32_t virt_func, bool virt_func_en, uint32_t msix_vec_id, char __cmem *work_area_ptr, uint32_t work_area_size)
- *Send interrupt message to PCI.* static __always_inline void ezdp_send_interrupt_to_pci_async (uint32_t endpoint, uint32_t msid, uint32_t phy_func, uint32_t virt_func, bool virt_func_en, uint32_t msix_vec_id, char __cmem *work_area_ptr, uint32_t work_area_size)
- *Non blocking version of ezdp_send_interrupt_to_pci().* uint32_t ezdp_get_pci_ctrl_reg (uint32_t endpoint, uint32_t pf, uint32_t reg_id, uint32_t *value)
- *Get the value of the PCIe vendor specific configuration space register.* uint32_t ezdp_set_pci_ctrl_reg (uint32_t endpoint, uint32_t pf, uint32_t reg_id, uint32_t value)

Set the value of the PCIe vendor specific configuration space register.

Function Documentation

static __always_inline void ezdp_init_pci_queue_desc (uint32_t *endpoint*, uint32_t *queue_id*, ezdp_pci_queue_desc_t * *pci_queue_desc*, char __cmem * *work_area_ptr*, uint32_t *work_area_size*) [static]

Initialize and get PCI message queue description.

Parameters:

[in] *endpoint* - PCI end point number
[in] *queue_id* - PCI queue ID (0/1/2/3)
[out] *pci_queue_desc* - message queue description
[in] *work_area_ptr* - pointer to work area (temporary memory in CMEM to be used by the function). The size of the temporary memory is determined by EZDP_INIT_PCI_QUEUE_DESC_WORK_AREA_SIZE
[in] *work_area_size* - size of work area pointer

Returns:

none

static __always_inline void ezdp_get_pci_msg (ezdp_pci_queue_desc_t * *pci_queue_desc*, uint32_t *msg_index*, struct ezdp_pci_msg __cmem * *msg*) [static]

Get message from PCI queue according to given index.

Parameters:

[in] *pci_queue_desc* - message queue description
[out] *msg_index* - message index
[out] *msg* - message from queue

Returns:

none

static __always_inline void ezdp_set_pci_msgq_read_index (ezdp_pci_queue_desc_t * *pci_queue_desc*, uint32_t *value*, char __cmem * *work_area_ptr*, uint32_t *work_area_size*) [static]

Set read index of PCI message queue.

Parameters:

[in] *pci_queue_desc* - message queue description
[in] *value* - new read index value
[in] *work_area_ptr* - pointer to work area (temporary memory in CMEM to be used by the function). The size of the temporary memory is determined by EZDP_PCI_RW_INDEX_WORK_AREA_SIZE
[in] *work_area_size* - size of work area pointer

Returns:

none

static __always_inline void ezdp_set_pci_msgq_read_index_async (ezdp_pci_queue_desc_t * *pci_queue_desc*, uint32_t *value*, char __cmem * *work_area_ptr*, uint32_t *work_area_size*) [static]

Set read index of PCI message queue.

Parameters:

[in] *pci_queue_desc* - message queue description
[in] *value* - new read index value
[in] *work_area_ptr* - pointer to work area (temporary memory in CMEM to be used by the function). The size of the temporary memory is determined by EZDP_PCI_RW_INDEX_WORK_AREA_SIZE
[in] *work_area_size* - size of work area pointer

Returns:

none

static __always_inline uint32_t ezdp_get_pci_msgq_write_index (ezdp_pci_queue_desc_t * *pci_queue_desc*, char __cmem * *work_area_ptr*, uint32_t *work_area_size*) [static]

Get write index of PCI message queue.

Parameters:

[in] *pci_queue_desc* - message queue description
[in] *work_area_ptr* - pointer to work area (temporary memory in CMEM to be used by the function). The size of the temporary memory is determined by EZDP_PCI_RW_INDEX_WORK_AREA_SIZE
[in] *work_area_size* - size of work area pointer

Returns:

uint32_t write index value

static __always_inline uint32_t ezdp_get_pci_msgq_read_index (ezdp_pci_queue_desc_t * *pci_queue_desc*, char __cmem * *work_area_ptr*, uint32_t *work_area_size*) [static]

Get read index of PCI message queue.

Parameters:

[in] *pci_queue_desc* - message queue description

[in] *work_area_ptr* - pointer to work area (temporary memory in CMEM to be used by the function). The size of the temporary memory is determined by EZDP_PCI_RW_INDEX_WORK_AREA_SIZE
[in] *work_area_size* - size of work area pointer

Returns:
uint32_t read index value

static __always_inline uint32_t ezdp_copy_frame_data_to_pci (struct ezdp_pci_addr __cmem * *dst_pci_ptr*, struct ezdp_buffer_desc __cmem * *src_bd_ptr*, uint32_t *src_bd_offset*, uint32_t *size*, uint32_t *traffic_class*, uint32_t *flags*) [static]

Copy data from a frame buffer to PCI address.

Parameters:
[in] *dst_pci_ptr* - pointer to destination PCI address (in CMEM in 8 byte alignment)
[in] *src_bd_ptr* - pointer to source frame describing the source buffer
[in] *src_bd_offset* - offset in the source buffer to copy data from
[in] *size* - number of bytes to copy
[in] *traffic_class* - PCI traffic class
[in] *flags* - execution flags for memory. Bitwise OR of zero or more flags out of the following:
EZDP_MEMORY_FLAG_OVERWRITE - override ECC (do not merge)
EZDP_PCI_FLAG_NO_SNOOP - PCI no snoop EZDP_PCI_FLAG_RELEX_ORDERED - PCI relaxed order enable EZDP_PCI_FLAG_ATU_BYPASS - PCI Address Translation Unit bypass enable

Returns:
uin32_t - 16 bit checksum value

static __always_inline void ezdp_copy_frame_data_to_pci_async (struct ezdp_pci_addr __cmem * *dst_pci_ptr*, struct ezdp_buffer_desc __cmem * *src_bd_ptr*, uint32_t *src_bd_offset*, uint32_t *size*, uint32_t *traffic_class*, uint32_t *flags*) [static]

Non blocking version of ezdp_copy_frame_data_to_pci().

Parameters:
[in] *dst_pci_ptr* - pointer to destination PCI address (in CMEM in 8 byte alignment)
[in] *src_bd_ptr* - pointer to source frame describing the source buffer
[in] *src_bd_offset* - offset in the source buffer to copy data from
[in] *size* - number of bytes to copy
[in] *traffic_class* - PCI traffic class
[in] *flags* - execution flags for memory. Bitwise OR of zero or more flags out of the following:
EZDP_MEMORY_FLAG_OVERWRITE - override ECC (do not merge)
EZDP_PCI_FLAG_NO_SNOOP - PCI no snoop EZDP_PCI_FLAG_RELEX_ORDERED - PCI relaxed order enable EZDP_PCI_FLAG_ATU_BYPASS - PCI Address Translation Unit bypass enable

Note:
Call ezdp_sync() to wait for the operation to complete, confirming that the data was copied to the destination.

Returns:
none

static __always_inline void ezdp_copy_frame_data_from_pci (struct ezdp_buffer_desc __cmem * *dst_bd_ptr*, uint32_t *dst_bd_offset*, struct ezdp_pci_addr __cmem * *src_pci_ptr*, uint32_t *size*, uint32_t *traffic_class*, uint32_t *flags*) [static]

Copy data from PCI address to a frame buffer.

Parameters:

[in] *dst_bd_ptr* - pointer to frame describing the destination buffer
[in] *dst_bd_offset* - offset in the destination buffer to copy data to
[in] *src_pci_ptr* - pointer to source PCI address (in CMEM in 8 byte alignment)
[in] *size* - number of bytes to copy
[in] *traffic_class* - PCI traffic class
[in] *flags* - execution flags for memory. Bitwise OR of zero or more flags out of the following:
EZDP_MEMORY_FLAG_OVERWRITE - override ECC (do not merge)
EZDP_PCI_FLAG_NO_SNOOP - PCI no snoop EZDP_PCI_FLAG_RELEX_ORDERED - PCI relaxed order enable EZDP_PCI_FLAG_ATU_BYPASS - PCI Address Translation Unit bypass enable

Returns:

none

static __always_inline void ezdp_copy_frame_data_from_pci_async (struct ezdp_buffer_desc __cmem * *dst_bd_ptr*, uint32_t *dst_bd_offset*, struct ezdp_pci_addr __cmem * *src_pci_ptr*, uint32_t *size*, uint32_t *traffic_class*, uint32_t *flags*) `[static]`

Non blocking version of ezdp_copy_frame_data_from_pci().

Parameters:

[in] *dst_bd_ptr* - pointer to frame describing the destination buffer
[in] *dst_bd_offset* - offset in the destination buffer to copy data to
[in] *src_pci_ptr* - pointer to source PCI address (in CMEM in 8 byte alignment)
[in] *size* - number of bytes to copy
[in] *traffic_class* - PCI traffic class
[in] *flags* - execution flags for memory. Bitwise OR of zero or more flags out of the following:
EZDP_MEMORY_FLAG_OVERWRITE - override ECC (do not merge)
EZDP_PCI_FLAG_NO_SNOOP - PCI no snoop EZDP_PCI_FLAG_RELEX_ORDERED - PCI relaxed order enable EZDP_PCI_FLAG_ATU_BYPASS - PCI Address Translation Unit bypass enable

Note:

Call ezdp_sync() to wait for the operation to complete, confirming that the data was copied to the destination.

Returns:

none

static __always_inline void ezdp_load_data_from_pci (void __cmem * *dst_ptr*, struct ezdp_pci_addr __cmem * *pci_ptr*, uint32_t *size*, uint32_t *traffic_class*, uint32_t *flags*) `[static]`

Copy data from a PCI address to CMEM.

Parameters:

[out] *dst_ptr* - pointer to the destination array in CMEM where the content is to be copied
[in] *pci_ptr* - pointer to PCI address (in CMEM in 8 byte alignment) to be copied
[in] *size* - number of bytes to copy
[in] *traffic_class* - PCI traffic class
[in] *flags* - execution memory flags. Bitwise OR of zero or more flags out of the following:
EZDP_MEMORY_FLAG_UNCACHED - do not allocate new cache entry
EZDP_PCI_FLAG_NO_SNOOP - PCI no snoop EZDP_PCI_FLAG_RELEX_ORDERED - PCI relaxed order enable EZDP_PCI_FLAG_ATU_BYPASS - PCI Address Translation Unit bypass enable

Returns:

none

static __always_inline void ezdp_load_data_from_pci_async (void __cmem * *dst_ptr*, struct ezdp_pci_addr __cmem * *pci_ptr*, uint32_t *size*, uint32_t *traffic_class*, uint32_t *flags*) [static]

Non blocking version of ezdp_load_data_from_pci().

Parameters:

[out] *dst_ptr* - pointer in CMEM to copy data to
[in] *pci_ptr* - pointer to PCI address (in CMEM in 8 byte alignment)
[in] *size* - number of bytes to copy
[in] *traffic_class* - PCI traffic class
[in] *flags* - execution memory flags. Bitwise OR of zero or more flags out of the following:
EZDP_MEMORY_FLAG_UNCACHED - do not allocate new cache entry
EZDP_PCI_FLAG_NO_SNOOP - PCI no snoop EZDP_PCI_FLAG_RELEX_ORDERED - PCI relaxed order enable EZDP_PCI_FLAG_ATU_BYPASS - PCI Address Translation Unit bypass enable

Note:

Call ezdp_sync() to wait for the operation to complete, confirming that the data was copied to the destination.

Returns:

none

static __always_inline uint32_t ezdp_store_data_to_pci (struct ezdp_pci_addr __cmem * *pci_ptr*, void __cmem * *src_ptr*, uint32_t *size*, uint32_t *traffic_class*, uint32_t *flags*) [static]

Copy data from CMEM to PCI address.

Parameters:

[in] *pci_ptr* - pointer to PCI address (in CMEM in 8 byte alignment)
[in] *src_ptr* - pointer in CMEM to copy data from
[in] *size* - number of bytes to copy
[in] *traffic_class* - PCI traffic class
[in] *flags* - execution flags for memory. Bitwise OR of zero or more flags out of the following:
EZDP_MEMORY_FLAG_OVERWRITE - override ECC (do not merge)
EZDP_MEMORY_FLAG_UNCACHED - do not allocate new cache entry
EZDP_PCI_FLAG_NO_SNOOP - PCI no snoop EZDP_PCI_FLAG_RELEX_ORDERED - PCI relaxed order enable EZDP_PCI_FLAG_ATU_BYPASS - PCI Address Translation Unit bypass enable

Returns:

uint32_t - 16 bit checksum value

static __always_inline void ezdp_store_data_to_pci_async (struct ezdp_pci_addr __cmem * *pci_ptr*, void __cmem * *src_ptr*, uint32_t *size*, uint32_t *traffic_class*, uint32_t *flags*) [static]

Non blocking version of ezdp_store_data_to_pci().

Parameters:

[in] *pci_ptr* - pointer to PCI address (in CMEM in 8 byte alignment)
[in] *src_ptr* - pointer in CMEM to copy data from
[in] *size* - number of bytes to copy
[in] *traffic_class* - PCI traffic class
[in] *flags* - execution flags for memory. Bitwise OR of zero or more flags out of the following:
EZDP_MEMORY_FLAG_OVERWRITE - override ECC (do not merge)
EZDP_MEMORY_FLAG_UNCACHED - do not allocate new cache entry

EZDP_PCI_FLAG_NO_SNOOP - PCI no snoop EZDP_PCI_FLAG_RELEX_ORDERED - PCI relaxed order enable EZDP_PCI_FLAG_ATU_BYPASS - PCI Address Translation Unit bypass enable

Note:

Call ezdp_sync() to wait for the operation to complete, confirming that the data was copied to the destination.

Returns:

non

static __always_inline uint32_t ezdp_copy_pci_data_to_ext_addr (struct ezdp_ext_addr __cmem * *dst_ptr*, struct ezdp_pci_addr __cmem * *src_pci_ptr*, uint32_t *size*, uint32_t *traffic_class*, uint32_t *flags*) `[static]`

Copy PCI data to extended addresses.

Parameters:

[in] *dst_ptr* - pointer to destination extended address (in CMEM in 8 byte alignment)
[in] *src_pci_ptr* - pointer to source PCI address (in CMEM in 8 byte alignment)
[in] *size* - number of bytes to copy
[in] *traffic_class* - PCI traffic class
[in] *flags* - execution flags for memory. Bitwise OR of zero or more flags out of the following:
EZDP_MEMORY_FLAG_OVERWRITE - override ECC (do not merge)
EZDP_MEMORY_FLAG_UNCACHED - do not allocate new cache entry
EZDP_PCI_FLAG_NO_SNOOP - PCI no snoop EZDP_PCI_FLAG_RELEX_ORDERED - PCI relaxed order enable EZDP_PCI_FLAG_ATU_BYPASS - PCI Address Translation Unit bypass enable

Returns:

uint32_t - first 4 bytes of the copied data

static __always_inline void ezdp_copy_pci_data_to_ext_addr_async (struct ezdp_ext_addr __cmem * *dst_ptr*, struct ezdp_pci_addr __cmem * *src_pci_ptr*, uint32_t *size*, uint32_t *traffic_class*, uint32_t *flags*) `[static]`

Non blocking version of ezdp_copy_pci_data_to_ext_addr().

Parameters:

[in] *dst_ptr* - pointer to destination extended address (in CMEM in 8 byte alignment)
[in] *src_pci_ptr* - pointer to source PCI address (in CMEM in 8 byte alignment)
[in] *size* - number of bytes to copy
[in] *traffic_class* - PCI traffic class
[in] *flags* - execution flags for memory. Bitwise OR of zero or more flags out of the following:
EZDP_MEMORY_FLAG_OVERWRITE - override ECC (do not merge)
EZDP_MEMORY_FLAG_UNCACHED - do not allocate new cache entry
EZDP_PCI_FLAG_NO_SNOOP - PCI no snoop EZDP_PCI_FLAG_RELEX_ORDERED - PCI relaxed order enable EZDP_PCI_FLAG_ATU_BYPASS - PCI Address Translation Unit bypass enable

Note:

Call ezdp_sync() to wait for the operation to complete, confirming that the data was copied to the destination.

Returns:

none

static __always_inline uint32_t ezdp_copy_pci_data_from_ext_addr (struct ezdp_pci_addr __cmem * *dst_pci_ptr*, struct ezdp_ext_addr __cmem * *src_ptr*, uint32_t *size*, uint32_t *traffic_class*, uint32_t *flags*) `[static]`

Copy extended address data to PCI.

Parameters:

[in] *dst_pci_ptr* - pointer to destination PCI address (in CMEM in 8 byte alignment)
[in] *src_ptr* - pointer to source extended address (in CMEM in 8 byte alignment)
[in] *size* - number of bytes to copy
[in] *traffic_class* - PCI traffic class
[in] *flags* - execution flags for memory. Bitwise OR of zero or more flags out of the following:
EZDP_MEMORY_FLAG_OVERWRITE - override ECC (do not merge)
EZDP_MEMORY_FLAG_UNCACHED - do not allocate new cache entry
EZDP_PCI_FLAG_NO_SNOOP - PCI no snoop EZDP_PCI_FLAG_RELEX_ORDERED - PCI relaxed order enable EZDP_PCI_FLAG_ATU_BYPASS - PCI Address Translation Unit bypass enable

Returns:

uint32_t - first 4 bytes of the copied data

static __always_inline void ezdp_copy_pci_data_from_ext_addr_async (struct ezdp_pci_addr __cmem * *dst_pci_ptr*, struct ezdp_ext_addr __cmem * *src_ptr*, uint32_t *size*, uint32_t *traffic_class*, uint32_t *flags*) [static]

Non blocking version of ezdp_copy_pci_data_from_ext_addr().

Parameters:

[in] *dst_pci_ptr* - pointer to destination PCI address (in CMEM in 8 byte alignment)
[in] *src_ptr* - pointer to source extended address (in CMEM in 8 byte alignment)
[in] *size* - number of bytes to copy
[in] *traffic_class* - PCI traffic class
[in] *flags* - execution flags for memory. Bitwise OR of zero or more flags out of the following:
EZDP_MEMORY_FLAG_OVERWRITE - override ECC (do not merge)
EZDP_MEMORY_FLAG_UNCACHED - do not allocate new cache entry
EZDP_PCI_FLAG_NO_SNOOP - PCI no snoop EZDP_PCI_FLAG_RELEX_ORDERED - PCI relaxed order enable EZDP_PCI_FLAG_ATU_BYPASS - PCI Address Translation Unit bypass enable

Note:

Call ezdp_sync() to wait for the operation to complete, confirming that the data was copied to the destination.

Returns:

none

static __always_inline uint32_t ezdp_translate_pci_addr (void __cmem * *dst_ptr*, struct ezdp_pci_addr __cmem * *pci_ptr*, uint32_t *traffic_class*, uint32_t *flags*) [static]

PCI address translation request; result will be saved in CMEM.

Parameters:

[out] *dst_ptr* - pointer to the destination array in CMEM where the content is to be copied
[in] *pci_ptr* - pointer to PCI address (in CMEM in 8 byte alignment) to be copied
[in] *traffic_class* - PCI traffic class
[in] *flags* - execution flags for memory. Bitwise OR of zero or more flags out of the following:
EZDP_MEMORY_FLAG_UNCACHED - do not allocate new cache entry
EZDP_PCI_FLAG_NO_SNOOP - PCI no snoop EZDP_PCI_FLAG_RELEX_ORDERED - PCI relaxed order enable EZDP_PCI_FLAG_ATU_BYPASS - PCI Address Translation Unit bypass enable

Returns:

uint32_t - first 4 bytes of the copied data

static __always_inline void ezdp_translate_pci_addr_async (void __cmem * *dst_ptr*, struct ezdp_pci_addr __cmem * *pci_ptr*, uint32_t *traffic_class*, uint32_t *flags*) [static]

Non blocking version of ezdp_translate_pci_addr().

Parameters:

[out] *dst_ptr* - pointer in CMEM to copy data to
[in] *pci_ptr* - pointer to PCI address (in CMEM in 8 byte alignment)
[in] *traffic_class* - The PCI traffic class
[in] *flags* - execution flags for memory. Bitwise OR of zero or more flags out of the following:
EZDP_MEMORY_FLAG_UNCACHED - do not allocate new cache entry
EZDP_PCI_FLAG_NO_SNOOP - PCI no snoop EZDP_PCI_FLAG_RELEX_ORDERED - PCI relaxed order enable EZDP_PCI_FLAG_ATU_BYPASS - PCI Address Translation Unit bypass enable

Note:

Call ezdp_sync() to wait for the operation to complete, confirming that the data was copied to the destination.

Returns:

none

static __always_inline uint32_t ezdp_translate_pci_addr_to_ext_addr (struct ezdp_ext_addr __cmem * *dst_ptr*, struct ezdp_pci_addr __cmem * *src_pci_ptr*, uint32_t *traffic_class*, uint32_t *flags*) [static]

Copy PCI data to extended addresses.

Parameters:

[in] *dst_ptr* - pointer to destination extended address (in CMEM in 8 byte alignment)
[in] *src_pci_ptr* - pointer to source PCI address (in CMEM in 8 byte alignment)
[in] *traffic_class* - PCI traffic class
[in] *flags* - execution flags for memory. Bitwise OR of zero or more flags out of the following:
EZDP_MEMORY_FLAG_OVERWRITE - override ECC (do not merge)
EZDP_MEMORY_FLAG_UNCACHED - do not allocate new cache entry
EZDP_PCI_FLAG_NO_SNOOP - PCI no snoop EZDP_PCI_FLAG_RELEX_ORDERED - PCI relaxed order enable EZDP_PCI_FLAG_ATU_BYPASS - PCI Address Translation Unit bypass enable

Returns:

uint32_t - first 4 bytes of the copied data

static __always_inline void ezdp_translate_pci_addr_to_ext_addr_async (struct ezdp_ext_addr __cmem * *dst_ptr*, struct ezdp_pci_addr __cmem * *src_pci_ptr*, uint32_t *traffic_class*, uint32_t *flags*) [static]

Non blocking version of ezdp_translate_pci_addr_to_ext_addr().

Parameters:

[in] *dst_ptr* - pointer to destination extended address (in CMEM in 8 byte alignment)
[in] *src_pci_ptr* - pointer to source PCI address (in CMEM in 8 byte alignment)
[in] *traffic_class* - PCI traffic class
[in] *flags* - execution flags for memory. Bitwise OR of zero or more flags out of the following:
EZDP_MEMORY_FLAG_OVERWRITE - override ECC (do not merge)
EZDP_MEMORY_FLAG_UNCACHED - do not allocate new cache entry
EZDP_PCI_FLAG_NO_SNOOP - PCI no snoop EZDP_PCI_FLAG_RELEX_ORDERED - PCI relaxed order enable EZDP_PCI_FLAG_ATU_BYPASS - PCI Address Translation Unit bypass enable

Note:

Call ezdp_sync() to wait for the operation to complete, confirming that the data was copied to the destination.

Returns:

none

static __always_inline uint32_t ezdp_send_message_to_pci (struct ezdp_pci_addr __cmem * *pci_ptr*, void __cmem * *src_ptr*, uint32_t *size*, uint32_t *traffic_class*, uint32_t *msg_code*, uint32_t *flags*) `[static]`

Copy data from CMEM to PCI address.

Parameters:

[in] *pci_ptr* - pointer to PCI address (in CMEM in 8 byte alignment)
[in] *src_ptr* - pointer in CMEM to copy data from
[in] *size* - number of bytes to copy
[in] *traffic_class* - PCI traffic class
[in] *msg_code* - PCI Message Code
[in] *flags* - execution flags for memory. Bitwise OR of zero or more flags out of the following:
EZDP_MEMORY_FLAG_OVERWRITE - override ECC (do not merge)
EZDP_MEMORY_FLAG_UNCACHED - do not allocate new cache entry
EZDP_PCI_FLAG_NO_SNOOP - PCI no snoop EZDP_PCI_FLAG_RELEX_ORDERED - PCI relaxed order enable EZDP_PCI_FLAG_ATU_BYPASS - PCI Address Translation Unit bypass enable

Returns:

uint32_t - first 4 bytes of the copied data

static __always_inline void ezdp_send_message_to_pci_async (struct ezdp_pci_addr __cmem * *pci_ptr*, void __cmem * *src_ptr*, uint32_t *size*, uint32_t *traffic_class*, uint32_t *msg_code*, uint32_t *flags*) `[static]`

Non blocking version of ezdp_send_message_to_pci().

Parameters:

[in] *pci_ptr* - pointer to PCI address (in CMEM in 8 byte alignment)
[in] *src_ptr* - pointer in CMEM to copy data from
[in] *size* - number of bytes to copy
[in] *traffic_class* - PCI traffic class
[in] *msg_code* - PCI Message Code
[in] *flags* - execution flags for pci. Bitwise OR of zero or more flags out of the following:
EZDP_MEMORY_FLAG_OVERWRITE - override ECC (do not merge)
EZDP_MEMORY_FLAG_UNCACHED - do not allocate new cache entry
EZDP_PCI_FLAG_NO_SNOOP - PCI no snoop EZDP_PCI_FLAG_RELEX_ORDERED - PCI relaxed order enable EZDP_PCI_FLAG_ATU_BYPASS - PCI Address Translation Unit bypass enable

Note:

Call ezdp_sync() to wait for the operation to complete, confirming that the data was copied to the destination.

Returns:

non

static __always_inline void ezdp_send_interrupt_to_pci (uint32_t *endpoint*, uint32_t *msid*, uint32_t *phy_func*, uint32_t *virt_func*, bool *virt_func_en*, uint32_t *msix_vec_id*, char __cmem * *work_area_ptr*, uint32_t *work_area_size*) `[static]`

Send interrupt message to PCI.

Parameters:

[in] *endpoint* - PCI endpoint number
[in] *msid* - MSID of the endpoint
[in] *phy_func* - physical function number
[in] *virt_func* - virtual function number
[in] *virt_func_en* - enable virtual function usage
[in] *msix_vec_id* - MSI-X vector ID
[in] *work_area_ptr* - pointer to work area (temporary memory in CMEM to be used by the function). The size of the temporary memory is determined by EZDP_PCI_INTERRUPT_WORK_AREA_SIZE
[in] *work_area_size* - size of work area pointer

Returns:

none

static __always_inline void ezdp_send_interrupt_to_pci_async (uint32_t *endpoint*, uint32_t *msid*, uint32_t *phy_func*, uint32_t *virt_func*, bool *virt_func_en*, uint32_t *msix_vec_id*, char __cmem * *work_area_ptr*, uint32_t *work_area_size*) `[static]`

Non blocking version of ezdp_send_interrupt_to_pci().

Parameters:

[in] *endpoint* - PCI endpoint number
[in] *msid* - MSID of the endpoint
[in] *phy_func* - physical function number
[in] *virt_func* - virtual function number
[in] *virt_func_en* - enable virtual function usage
[in] *msix_vec_id* - MSI-X vector ID
[in] *work_area_ptr* - pointer to work area (temporary memory in CMEM to be used by the function). The size of the temporary memory is determined by EZDP_PCI_INTERRUPT_WORK_AREA_SIZE
[in] *work_area_size* - size of work area pointer

Note:

Call ezdp_sync() to wait for the operation to complete, confirming that the data was copied to the destination.

Returns:

none

uint32_t ezdp_get_pci_ctrl_reg (uint32_t *endpoint*, uint32_t *pf*, uint32_t *reg_id*, uint32_t * *value*)

Get the value of the PCIe vendor specific configuration space register.

Parameters:

[in] *endpoint* - PCI endpoint number
[in] *pf* - physical function number
[in] *reg_id* - register ID (0..3)
[out] *value* - register value

Note:

This function performs a Linux system call. Avoid calling it from packet processing code

Returns:

- 0 (operation success), EIO (OS failed). The function may also fail for any of the errors specified for routines: open,ioctl,close. In such case the errno value of the failure routine is returned. Use ezdp_get_err_msg() API to get the detail error message of the failure

uint32_t ezdp_set_pci_ctrl_reg (uint32_t *endpoint*, uint32_t *pf*, uint32_t *reg_id*, uint32_t *value*)

Set the value of the PCIe vendor specific configuration space register.

Parameters:

[in] *endpoint* - PCI end point number
[in] *pf* - physical function number
[in] *reg_id* - register ID (0..3)
[in] *value* - register value

Note:

This function performs a Linux system call. Avoid calling it from packet processing code

Returns:

- 0 (operation success) EIO (OS failed). The function may also fail for any of the errors specified for routines open,ioctl,close. In such case the errno value of the failure routine is returned Use ezdp_get_err_msg() API to get the detail error message of the failure

dpe/dp/include/ezdp_pci_defs.h File Reference

Data Structures

- struct ezdp_pci_info
- *PCI info for describing to which endpoint, physical function, virtual function and queue the frame is to be sent.* struct ezdp_pci_msg_ctrl
- *PCI message control.* struct ezdp_pci_msg_payload_elbi
- *PCI ELBI message payload.* struct ezdp_pci_msg_payload_ats
- *PCI ATS message payload.* struct ezdp_pci_msg_payload_msix
- *PCI MSIX message payload.* struct ezdp_pci_msg
- *Message from PCI queue.* struct ezdp_driver_desc_flags
- *TX/RX descriptor flags structure.* struct ezdp_driver_desc

TX/RX descriptor. Defines

- #define EZDP_PCI_VERSION_MAJOR 2
- #define EZDP_PCI_VERSION_MINOR 1
- #define EZDP_PCI_INFO_VIRT_FUNC_SIZE 7
- #define EZDP_PCI_INFO_VIRT_FUNC_OFFSET 0
- #define EZDP_PCI_INFO_ENDPOINT_SIZE 1
- #define EZDP_PCI_INFO_ENDPOINT_OFFSET 7
- #define EZDP_PCI_INFO_ENDPOINT_MASK (1 << EZDP_PCI_INFO_ENDPOINT_OFFSET)
- #define EZDP_PCI_INFO_PHYS_FUNC_SIZE 2
- #define EZDP_PCI_INFO_PHYS_FUNC_OFFSET 8
- #define EZDP_PCI_INFO_QUEUE_SIZE 5
- #define EZDP_PCI_INFO_QUEUE_OFFSET 10
- #define EZDP_PCI_INFO_VIRT_FUNC_EN_SIZE 1
- #define EZDP_PCI_INFO_VIRT_FUNC_EN_OFFSET 15
- #define EZDP_PCI_INFO_VIRT_FUNC_EN_MASK (1 << EZDP_PCI_INFO_VIRT_FUNC_EN_OFFSET)
- #define EZDP_PCI_INFO_RESERVED16_32_SIZE 16
- #define EZDP_PCI_INFO_RESERVED16_32_OFFSET 16
- #define EZDP_PCI_MSG_CTRL_VIRT_FUNC_SIZE 7
- #define EZDP_PCI_MSG_CTRL_VIRT_FUNC_OFFSET 0
- #define EZDP_PCI_MSG_CTRL_RESERVED8_SIZE 1
- #define EZDP_PCI_MSG_CTRL_RESERVED8_OFFSET 7
- #define EZDP_PCI_MSG_CTRL_PHY_FUNC_SIZE 2
- #define EZDP_PCI_MSG_CTRL_PHY_FUNC_OFFSET 8
- #define EZDP_PCI_MSG_CTRL_RESERVED10_11_SIZE 2
- #define EZDP_PCI_MSG_CTRL_RESERVED10_11_OFFSET 10
- #define EZDP_PCI_MSG_CTRL_BAR_NUM_SIZE 3
- #define EZDP_PCI_MSG_CTRL_BAR_NUM_OFFSET 12
- #define EZDP_PCI_MSG_CTRL_VIRT_FUNC_EN_SIZE 1
- #define EZDP_PCI_MSG_CTRL_VIRT_FUNC_EN_OFFSET 15
- #define EZDP_PCI_MSG_CTRL_VIRT_FUNC_EN_MASK (1 << EZDP_PCI_MSG_CTRL_VIRT_FUNC_EN_OFFSET)
- #define EZDP_PCI_MSG_PAYLOAD_ELBI_RESERVED_SIZE 32
- #define EZDP_PCI_MSG_PAYLOAD_ELBI_RESERVED_OFFSET 0
- #define EZDP_PCI_MSG_PAYLOAD_ELBI_ADDRESS_SIZE 32
- #define EZDP_PCI_MSG_PAYLOAD_ELBI_ADDRESS_OFFSET 32
- #define EZDP_PCI_MSG_PAYLOAD_ELBI_ADDRESS_WORD_SELECT 1
- #define EZDP_PCI_MSG_PAYLOAD_ELBI_ADDRESS_WORD_OFFSET 0
- #define EZDP_PCI_MSG_PAYLOAD_ELBI_DATA_SIZE 32
- #define EZDP_PCI_MSG_PAYLOAD_ELBI_DATA_OFFSET 64
- #define EZDP_PCI_MSG_PAYLOAD_ELBI_DATA_WORD_SELECT 2
- #define EZDP_PCI_MSG_PAYLOAD_ELBI_DATA_WORD_OFFSET 0
- #define EZDP_PCI_MSG_PAYLOAD_ELBI_WORD_COUNT 3

- #define EZDP_PCI_MSG_PAYLOAD_ATS_RESERVED_SIZE 32
- #define EZDP_PCI_MSG_PAYLOAD_ATS_RESERVED_OFFSET 0
- #define EZDP_PCI_MSG_PAYLOAD_ATS_DATA_MSB_SIZE 32
- #define EZDP_PCI_MSG_PAYLOAD_ATS_DATA_MSB_OFFSET 32
- #define EZDP_PCI_MSG_PAYLOAD_ATS_DATA_MSB_WORD_SELECT 1
- #define EZDP_PCI_MSG_PAYLOAD_ATS_DATA_MSB_WORD_OFFSET 0
- #define EZDP_PCI_MSG_PAYLOAD_ATS_DATA_LSB_SIZE 32
- #define EZDP_PCI_MSG_PAYLOAD_ATS_DATA_LSB_OFFSET 64
- #define EZDP_PCI_MSG_PAYLOAD_ATS_DATA_LSB_WORD_SELECT 2
- #define EZDP_PCI_MSG_PAYLOAD_ATS_DATA_LSB_WORD_OFFSET 0
- #define EZDP_PCI_MSG_PAYLOAD_ATS_WORD_COUNT 3
- #define EZDP_PCI_MSG_PAYLOAD_MSIX_RESERVED0_31_SIZE 32
- #define EZDP_PCI_MSG_PAYLOAD_MSIX_RESERVED0_31_OFFSET 0
- #define EZDP_PCI_MSG_PAYLOAD_MSIX_RESERVED_32_63_SIZE 32
- #define EZDP_PCI_MSG_PAYLOAD_MSIX_RESERVED_32_63_OFFSET 32
- #define EZDP_PCI_MSG_PAYLOAD_MSIX_VECTOR_INDEX_SIZE 2
- #define EZDP_PCI_MSG_PAYLOAD_MSIX_VECTOR_INDEX_OFFSET 64
- #define EZDP_PCI_MSG_PAYLOAD_MSIX_VECTOR_INDEX_WORD_SELECT 2
- #define EZDP_PCI_MSG_PAYLOAD_MSIX_VECTOR_INDEX_WORD_OFFSET 0
- #define EZDP_PCI_MSG_PAYLOAD_MSIX_RESERVED66_95_SIZE 30
- #define EZDP_PCI_MSG_PAYLOAD_MSIX_RESERVED66_95_OFFSET 66
- #define EZDP_PCI_MSG_PAYLOAD_MSIX_WORD_COUNT 3
- #define EZDP_PCI_MSG_CTRL_SIZE 16
- #define EZDP_PCI_MSG_CTRL_OFFSET 0
- #define EZDP_PCI_MSG_CTRL_WORD_SELECT 0
- #define EZDP_PCI_MSG_CTRL_WORD_OFFSET 0
- #define EZDP_PCI_MSG_MSG_SIZE 8
- #define EZDP_PCI_MSG_MSG_OFFSET 16
- #define EZDP_PCI_MSG_MSG_WORD_SELECT 0
- #define EZDP_PCI_MSG_MSG_WORD_OFFSET 16
- #define EZDP_PCI_MSG_ECC_SIZE 8
- #define EZDP_PCI_MSG_ECC_OFFSET 24
- #define EZDP_PCI_MSG_WORD_COUNT 4
- #define EZDP_DRIVER_DESC_FLAGS_DATA_SIZE 1
- #define EZDP_DRIVER_DESC_FLAGS_DATA_OFFSET 0
- #define EZDP_DRIVER_DESC_FLAGS_DATA_MASK (1 << EZDP_DRIVER_DESC_FLAGS_DATA_OFFSET)
- #define EZDP_DRIVER_DESC_FLAGS_OWNER_SIZE 1
- #define EZDP_DRIVER_DESC_FLAGS_OWNER_OFFSET 1
- #define EZDP_DRIVER_DESC_FLAGS_OWNER_MASK (1 << EZDP_DRIVER_DESC_FLAGS_OWNER_OFFSET)
- #define EZDP_DRIVER_DESC_FLAGS_ERROR_SIZE 1
- #define EZDP_DRIVER_DESC_FLAGS_ERROR_OFFSET 2
- #define EZDP_DRIVER_DESC_FLAGS_ERROR_MASK (1 << EZDP_DRIVER_DESC_FLAGS_ERROR_OFFSET)
- #define EZDP_DRIVER_DESC_FLAGS_TYPE_SIZE 5
- #define EZDP_DRIVER_DESC_FLAGS_TYPE_OFFSET 3
- #define EZDP_DRIVER_DESC_BUF_DATA_ADDR_SIZE 64
- #define EZDP_DRIVER_DESC_BUF_DATA_ADDR_OFFSET 0
- #define EZDP_DRIVER_DESC_BUF_DATA_ADDR_WORD_SELECT 0
- #define EZDP_DRIVER_DESC_BUF_DATA_ADDR_WORD_OFFSET 0
- #define EZDP_DRIVER_DESC_LEN_SIZE 32
- #define EZDP_DRIVER_DESC_LEN_OFFSET 64
- #define EZDP_DRIVER_DESC_LEN_WORD_SELECT 2
- #define EZDP_DRIVER_DESC_LEN_WORD_OFFSET 0
- #define EZDP_DRIVER_DESC_TOTAL_SIZE 8
- #define EZDP_DRIVER_DESC_TOTAL_OFFSET 96

- #define EZDP_DRIVER_DESC_TOTAL_WORD_SELECT 3
- #define EZDP_DRIVER_DESC_TOTAL_WORD_OFFSET 0
- #define EZDP_DRIVER_DESC_FLAGS_SIZE 8
- #define EZDP_DRIVER_DESC_FLAGS_OFFSET 104
- #define EZDP_DRIVER_DESC_FLAGS_WORD_SELECT 3
- #define EZDP_DRIVER_DESC_FLAGS_WORD_OFFSET 8
- #define EZDP_DRIVER_DESC_SUB_TYPE_SIZE 16
- #define EZDP_DRIVER_DESC_SUB_TYPE_OFFSET 112
- #define EZDP_DRIVER_DESC_SUB_TYPE_WORD_SELECT 3
- #define EZDP_DRIVER_DESC_SUB_TYPE_WORD_OFFSET 16
- #define EZDP_DRIVER_DESC_WORD_COUNT 4
- #define EZDP_PCI_INTERRUPT_WORK_AREA_SIZE sizeof(struct ezdp_send_pci_interrupt_working_area)
- *Send MSI-X to PCI working area.* #define EZDP_INIT_PCI_QUEUE_DESC_WORK_AREA_SIZE sizeof(struct ezdp_init_msgq_desc_working_area)
- *Initialized PCI queue descriptor working area.* #define EZDP_PCI_RW_INDEX_WORK_AREA_SIZE sizeof(struct ezdp_ctrl_line)

Get PCI queue read/write index working area. Typedefs

- typedef uint32_t ezdp_pci_info_t
- typedef uint16_t ezdp_pci_msg_ctrl_t
- typedef uint8_t ezdp_driver_desc_flags_t
- typedef enum ezdp_pci_queue_type ezdp_pci_queue_type_t
- *PCI queue type.* typedef struct ezdp_pci_msgq_desc ezdp_pci_queue_desc_t
- *PCI message queue descriptor.* typedef struct ezdp_init_msgq_desc_working_area ezdp_init_pci_queue_desc_working_area_t

Initialized PCI queue descriptor working area. Enumerations

- enum ezdp_pci_msg_type { EZDP_PCI_MSG_ERROR = 0x0, EZDP_PCI_MSG_RESET_REQUEST = 0x1, EZDP_PCI_MSG_FUNCTION_LEVEL_RESET = 0x2, EZDP_PCI_MSG_PM = 0x3, EZDP_PCI_MSG_OBFF_IDLE = 0x4, EZDP_PCI_MSG_OBFF_STATE = 0x5, EZDP_PCI_MSG_OBFF_ACTIVE = 0x6, EZDP_PCI_MSG_ATS_INVALID = 0x7, EZDP_PCI_MSG_ELBI = 0x8, EZDP_PCI_MSG_VPD_0 = 0x9, EZDP_PCI_MSG_VPD_1 = 0xa, EZDP_PCI_MSG_VPD_2 = 0xb, EZDP_PCI_MSG_VPD_3 = 0xc, EZDP_PCI_MSG_MSIX = 0xd, EZDP_PCI_MSG_NONE = 0xe }

  pci msg type possible values.

Define Documentation

#define EZDP_PCI_VERSION_MAJOR 2

#define EZDP_PCI_VERSION_MINOR 1

#define EZDP_PCI_INFO_VIRT_FUNC_SIZE 7

#define EZDP_PCI_INFO_VIRT_FUNC_OFFSET 0

#define EZDP_PCI_INFO_ENDPOINT_SIZE 1

#define EZDP_PCI_INFO_ENDPOINT_OFFSET 7

#define EZDP_PCI_INFO_ENDPOINT_MASK (1 << EZDP_PCI_INFO_ENDPOINT_OFFSET)

#define EZDP_PCI_INFO_PHYS_FUNC_SIZE 2

#define EZDP_PCI_INFO_PHYS_FUNC_OFFSET 8

#define EZDP_PCI_INFO_QUEUE_SIZE 5

#define EZDP_PCI_INFO_QUEUE_OFFSET 10

#define EZDP_PCI_INFO_VIRT_FUNC_EN_SIZE 1

#define EZDP_PCI_INFO_VIRT_FUNC_EN_OFFSET 15

#define EZDP_PCI_INFO_VIRT_FUNC_EN_MASK (1 << EZDP_PCI_INFO_VIRT_FUNC_EN_OFFSET)

#define EZDP_PCI_INFO_RESERVED16_32_SIZE 16

#define EZDP_PCI_INFO_RESERVED16_32_OFFSET 16

#define EZDP_PCI_MSG_CTRL_VIRT_FUNC_SIZE 7

#define EZDP_PCI_MSG_CTRL_VIRT_FUNC_OFFSET 0

#define EZDP_PCI_MSG_CTRL_RESERVED8_SIZE 1

#define EZDP_PCI_MSG_CTRL_RESERVED8_OFFSET 7

#define EZDP_PCI_MSG_CTRL_PHY_FUNC_SIZE 2

#define EZDP_PCI_MSG_CTRL_PHY_FUNC_OFFSET 8

#define EZDP_PCI_MSG_CTRL_RESERVED10_11_SIZE 2

#define EZDP_PCI_MSG_CTRL_RESERVED10_11_OFFSET 10

#define EZDP_PCI_MSG_CTRL_BAR_NUM_SIZE 3

```
#define EZDP_PCI_MSG_CTRL_BAR_NUM_OFFSET 12

#define EZDP_PCI_MSG_CTRL_VIRT_FUNC_EN_SIZE 1

#define EZDP_PCI_MSG_CTRL_VIRT_FUNC_EN_OFFSET 15

#define EZDP_PCI_MSG_CTRL_VIRT_FUNC_EN_MASK (1 <<
EZDP_PCI_MSG_CTRL_VIRT_FUNC_EN_OFFSET)

#define EZDP_PCI_MSG_PAYLOAD_ELBI_RESERVED_SIZE 32

#define EZDP_PCI_MSG_PAYLOAD_ELBI_RESERVED_OFFSET 0

#define EZDP_PCI_MSG_PAYLOAD_ELBI_ADDRESS_SIZE 32

#define EZDP_PCI_MSG_PAYLOAD_ELBI_ADDRESS_OFFSET 32

#define EZDP_PCI_MSG_PAYLOAD_ELBI_ADDRESS_WORD_SELECT 1

#define EZDP_PCI_MSG_PAYLOAD_ELBI_ADDRESS_WORD_OFFSET 0

#define EZDP_PCI_MSG_PAYLOAD_ELBI_DATA_SIZE 32

#define EZDP_PCI_MSG_PAYLOAD_ELBI_DATA_OFFSET 64

#define EZDP_PCI_MSG_PAYLOAD_ELBI_DATA_WORD_SELECT 2

#define EZDP_PCI_MSG_PAYLOAD_ELBI_DATA_WORD_OFFSET 0

#define EZDP_PCI_MSG_PAYLOAD_ELBI_WORD_COUNT 3

#define EZDP_PCI_MSG_PAYLOAD_ATS_RESERVED_SIZE 32

#define EZDP_PCI_MSG_PAYLOAD_ATS_RESERVED_OFFSET 0

#define EZDP_PCI_MSG_PAYLOAD_ATS_DATA_MSB_SIZE 32

#define EZDP_PCI_MSG_PAYLOAD_ATS_DATA_MSB_OFFSET 32

#define EZDP_PCI_MSG_PAYLOAD_ATS_DATA_MSB_WORD_SELECT 1

#define EZDP_PCI_MSG_PAYLOAD_ATS_DATA_MSB_WORD_OFFSET 0

#define EZDP_PCI_MSG_PAYLOAD_ATS_DATA_LSB_SIZE 32

#define EZDP_PCI_MSG_PAYLOAD_ATS_DATA_LSB_OFFSET 64

#define EZDP_PCI_MSG_PAYLOAD_ATS_DATA_LSB_WORD_SELECT 2

#define EZDP_PCI_MSG_PAYLOAD_ATS_DATA_LSB_WORD_OFFSET 0

#define EZDP_PCI_MSG_PAYLOAD_ATS_WORD_COUNT 3
```

```
#define EZDP_PCI_MSG_PAYLOAD_MSIX_RESERVED0_31_SIZE 32

#define EZDP_PCI_MSG_PAYLOAD_MSIX_RESERVED0_31_OFFSET 0

#define EZDP_PCI_MSG_PAYLOAD_MSIX_RESERVED_32_63_SIZE 32

#define EZDP_PCI_MSG_PAYLOAD_MSIX_RESERVED_32_63_OFFSET 32

#define EZDP_PCI_MSG_PAYLOAD_MSIX_VECTOR_INDEX_SIZE 2

#define EZDP_PCI_MSG_PAYLOAD_MSIX_VECTOR_INDEX_OFFSET 64

#define EZDP_PCI_MSG_PAYLOAD_MSIX_VECTOR_INDEX_WORD_SELECT 2

#define EZDP_PCI_MSG_PAYLOAD_MSIX_VECTOR_INDEX_WORD_OFFSET 0

#define EZDP_PCI_MSG_PAYLOAD_MSIX_RESERVED66_95_SIZE 30

#define EZDP_PCI_MSG_PAYLOAD_MSIX_RESERVED66_95_OFFSET 66

#define EZDP_PCI_MSG_PAYLOAD_MSIX_WORD_COUNT 3

#define EZDP_PCI_MSG_CTRL_SIZE 16

#define EZDP_PCI_MSG_CTRL_OFFSET 0

#define EZDP_PCI_MSG_CTRL_WORD_SELECT 0

#define EZDP_PCI_MSG_CTRL_WORD_OFFSET 0

#define EZDP_PCI_MSG_MSG_SIZE 8

#define EZDP_PCI_MSG_MSG_OFFSET 16

#define EZDP_PCI_MSG_MSG_WORD_SELECT 0

#define EZDP_PCI_MSG_MSG_WORD_OFFSET 16

#define EZDP_PCI_MSG_ECC_SIZE 8

#define EZDP_PCI_MSG_ECC_OFFSET 24

#define EZDP_PCI_MSG_WORD_COUNT 4

#define EZDP_DRIVER_DESC_FLAGS_DATA_SIZE 1

#define EZDP_DRIVER_DESC_FLAGS_DATA_OFFSET 0

#define EZDP_DRIVER_DESC_FLAGS_DATA_MASK (1 <<
EZDP_DRIVER_DESC_FLAGS_DATA_OFFSET)

#define EZDP_DRIVER_DESC_FLAGS_OWNER_SIZE 1
```

#define EZDP_DRIVER_DESC_FLAGS_OWNER_OFFSET 1

#define EZDP_DRIVER_DESC_FLAGS_OWNER_MASK (1 << EZDP_DRIVER_DESC_FLAGS_OWNER_OFFSET)

#define EZDP_DRIVER_DESC_FLAGS_ERROR_SIZE 1

#define EZDP_DRIVER_DESC_FLAGS_ERROR_OFFSET 2

#define EZDP_DRIVER_DESC_FLAGS_ERROR_MASK (1 << EZDP_DRIVER_DESC_FLAGS_ERROR_OFFSET)

#define EZDP_DRIVER_DESC_FLAGS_TYPE_SIZE 5

#define EZDP_DRIVER_DESC_FLAGS_TYPE_OFFSET 3

#define EZDP_DRIVER_DESC_BUF_DATA_ADDR_SIZE 64

#define EZDP_DRIVER_DESC_BUF_DATA_ADDR_OFFSET 0

#define EZDP_DRIVER_DESC_BUF_DATA_ADDR_WORD_SELECT 0

#define EZDP_DRIVER_DESC_BUF_DATA_ADDR_WORD_OFFSET 0

#define EZDP_DRIVER_DESC_LEN_SIZE 32

#define EZDP_DRIVER_DESC_LEN_OFFSET 64

#define EZDP_DRIVER_DESC_LEN_WORD_SELECT 2

#define EZDP_DRIVER_DESC_LEN_WORD_OFFSET 0

#define EZDP_DRIVER_DESC_TOTAL_SIZE 8

#define EZDP_DRIVER_DESC_TOTAL_OFFSET 96

#define EZDP_DRIVER_DESC_TOTAL_WORD_SELECT 3

#define EZDP_DRIVER_DESC_TOTAL_WORD_OFFSET 0

#define EZDP_DRIVER_DESC_FLAGS_SIZE 8

#define EZDP_DRIVER_DESC_FLAGS_OFFSET 104

#define EZDP_DRIVER_DESC_FLAGS_WORD_SELECT 3

#define EZDP_DRIVER_DESC_FLAGS_WORD_OFFSET 8

#define EZDP_DRIVER_DESC_SUB_TYPE_SIZE 16

#define EZDP_DRIVER_DESC_SUB_TYPE_OFFSET 112

#define EZDP_DRIVER_DESC_SUB_TYPE_WORD_SELECT 3

#define EZDP_DRIVER_DESC_SUB_TYPE_WORD_OFFSET 16

#define EZDP_DRIVER_DESC_WORD_COUNT 4

#define EZDP_PCI_INTERRUPT_WORK_AREA_SIZE sizeof(struct ezdp_send_pci_interrupt_working_area)

Send MSI-X to PCI working area.

#define EZDP_INIT_PCI_QUEUE_DESC_WORK_AREA_SIZE sizeof(struct ezdp_init_msgq_desc_working_area)

Initialized PCI queue descriptor working area.

#define EZDP_PCI_RW_INDEX_WORK_AREA_SIZE sizeof(struct ezdp_ctrl_line)

Get PCI queue read/write index working area.

Typedef Documentation

typedef uint32_t ezdp_pci_info_t

typedef uint16_t ezdp_pci_msg_ctrl_t

typedef uint8_t ezdp_driver_desc_flags_t

typedef enum ezdp_pci_queue_type ezdp_pci_queue_type_t

PCI queue type.

typedef struct ezdp_pci_msgq_desc ezdp_pci_queue_desc_t

PCI message queue descriptor.

typedef struct ezdp_init_msgq_desc_working_area ezdp_init_pci_queue_desc_working_area_t

Initialized PCI queue descriptor working area.

Enumeration Type Documentation

enum ezdp_pci_msg_type

pci msg type possible values.

Enumerator:

EZDP_PCI_MSG_ERROR Error event.

EZDP_PCI_MSG_RESET_REQUEST Reset Request Due link down.

EZDP_PCI_MSG_FUNCTION_LEVEL_RESET Function Level Reset.

EZDP_PCI_MSG_PM Power Management.

EZDP_PCI_MSG_OBFF_IDLE Optimized Buffer Flush and Fill idle.

EZDP_PCI_MSG_OBFF_STATE Optimized Buffer Flush and Fill message.

EZDP_PCI_MSG_OBFF_ACTIVE Optimized Buffer Flush and Fill cpu active.

EZDP_PCI_MSG_ATS_INVALID Address Translate Service invalid.

EZDP_PCI_MSG_ELBI External Local Bus Interface.

EZDP_PCI_MSG_VPD_0 Vital product data capability 0.

EZDP_PCI_MSG_VPD_1 Vital product data capability 1.

EZDP_PCI_MSG_VPD_2 Vital product data capability 2.

EZDP_PCI_MSG_VPD_3 Vital product data capability 3.

EZDP_PCI_MSG_MSIX Message Interrupt X.

EZDP_PCI_MSG_NONE No message in queue.

dpe/dp/include/ezdp_pool.h File Reference

Functions

- static __always_inline uint32_t ezdp_alloc_index (uint32_t pool_id)
- *Allocate a single index from an index pool.* static __always_inline void ezdp_free_index (uint32_t pool_id, uint32_t free_index)
- *Free a single index from an index pool.* static __always_inline void ezdp_free_index_async (uint32_t pool_id, uint32_t free_index)
- *Non blocking version of ezdp_free_index().* static __always_inline uint32_t ezdp_alloc_multi_index (uint32_t pool_id, uint32_t num_of_indexes, uint32_t __cmem *index_arrays_ptr)
- *Allocate multiple indexes from an index pool.* static __always_inline void ezdp_alloc_multi_index_async (uint32_t pool_id, uint32_t num_of_indexes, uint32_t __cmem *index_arrays_ptr)
- *Non blocking version of ezdp_alloc_multi_index().* static __always_inline void ezdp_free_multi_index (uint32_t pool_id, uint32_t num_of_indexes, uint32_t __cmem *index_arrays_ptr)
- *Free multiple indexes from an index pool.* static __always_inline void ezdp_free_multi_index_async (uint32_t pool_id, uint32_t num_of_indexes, uint32_t __cmem *index_arrays_ptr)
- *Non blocking version of ezdp_free_multi_buf().* static __always_inline uint32_t ezdp_read_free_indexes (uint32_t pool_id)
- *The number of indexes available to be obtained.* static __always_inline void ezdp_init_memory_pool (ezdp_mem_pool_t *memory_pool, struct ezdp_mem_pool_config *config)
- *Initialize a memory pool.* static __always_inline ezdp_sum_addr_t ezdp_alloc_obj (ezdp_mem_pool_t *memory_pool)
- *Allocate a single object from a memory pool.* static __always_inline void ezdp_free_obj (ezdp_mem_pool_t *memory_pool, ezdp_sum_addr_t free_obj)
- *Free a single object from a memory pool.* static __always_inline ezdp_sum_addr_t ezdp_get_obj (ezdp_mem_pool_t *memory_pool, uint32_t object_id)
- *Get object based on object id.* static __always_inline uint32_t ezdp_read_free_objs (ezdp_mem_pool_t *memory_pool)

The number of objects available to be obtained.

Function Documentation

static __always_inline uint32_t ezdp_alloc_index (uint32_t *pool_id*) [static]

Allocate a single index from an index pool.

Parameters:

[in] *pool_id* - the pool to allocate from

Returns:

allocated index EZDP_NULL_INDEX in case of error

static __always_inline void ezdp_free_index (uint32_t *pool_id*, uint32_t *free_index*) [static]

Free a single index from an index pool.

Parameters:

[in] *pool_id* - the pool to recycle to
[in] *free_index* - index to release

Returns:

none

static __always_inline void ezdp_free_index_async (uint32_t *pool_id*, uint32_t *free_index*) [static]

Non blocking version of ezdp_free_index().

Parameters:

[in] *pool_id* - the pool to recycle to
[in] *free_index* - index to release

Note:

Call ezdp_sync() to wait for the operation to complete.

Returns:

none

static __always_inline uint32_t ezdp_alloc_multi_index (uint32_t *pool_id*, uint32_t *num_of_indexes*, uint32_t __cmem * *index_arrays_ptr*) [static]

Allocate multiple indexes from an index pool.

Allocate up to 3 indexes from the same pool. The allocated indexes are written to the CMEM. In addition, the first allocated index is returned. The operation either succeeds to allocate all requested resources or fails without allocating any resources.

Parameters:

[in] *pool_id* - the pool to allocate from
[in] *num_of_indexes* - number of indexes to allocate (1-3)
[out] *index_arrays_ptr* - pointer to CMEM to write response to

Returns:

First allocated index EZDP_NULL_INDEX in case of error

static __always_inline void ezdp_alloc_multi_index_async (uint32_t *pool_id*, uint32_t *num_of_indexes*, uint32_t __cmem * *index_arrays_ptr*) [static]

Non blocking version of ezdp_alloc_multi_index().

Parameters:

[in] *pool_id* - the pool to allocate from
[in] *num_of_indexes* - number of indexes to allocate (1-3)
[out] *index_arrays_ptr* - pointer to CMEM to write response to

Note:

Call ezdp_sync() to wait for the operation to complete.

Returns:

First allocated index EZDP_NULL_INDEX in case of error

static __always_inline void ezdp_free_multi_index (uint32_t *pool_id*, uint32_t *num_of_indexes*, uint32_t __cmem * *index_arrays_ptr*) [static]

Free multiple indexes from an index pool.

Up to 8 indexes can be freed in one command.

Parameters:

[in] *pool_id* - the pool to recycle to
[in] *num_of_indexes* - number of indexes to recycle (1-8)

[in] *index_arrays_ptr* - pointer to array of indexes (in CMEM) to free

Returns:

non

static __always_inline void ezdp_free_multi_index_async (uint32_t *pool_id*, uint32_t *num_of_indexes*, uint32_t __cmem * *index_arrays_ptr*) `[static]`

Non blocking version of ezdp_free_multi_buf().

Parameters:

[in] *pool_id* - the pool to recycle to
[in] *num_of_indexes* - number of indexes to recycle (1-8)
[in] *index_arrays_ptr* - pointer to array of indexes (in CMEM) to free

Note:

Call ezdp_sync() to wait for the operation to complete.

Returns:

none

static __always_inline uint32_t ezdp_read_free_indexes (uint32_t *pool_id*) `[static]`

The number of indexes available to be obtained.

Parameters:

[in] *pool_id* - pool id {0-63}

Returns:

uint32_t - number of available/free indexes

static __always_inline void ezdp_init_memory_pool (ezdp_mem_pool_t * *memory_pool*, struct ezdp_mem_pool_config * *config*) `[static]`

Initialize a memory pool.

Parameters:

[out] *memory_pool* - the pool to initialize
[in] *config* - the configuration information for initialization

Returns:

void

static __always_inline ezdp_sum_addr_t ezdp_alloc_obj (ezdp_mem_pool_t * *memory_pool*) `[static]`

Allocate a single object from a memory pool.

Parameters:

[in] *memory_pool* - the pool to allocate from

Returns:

Summarized address (ezdp_sum_addr_t) of the allocated object. In case of error, null summarized address is returned. Use ezdp_is_null_sum_addr API to check it.

static __always_inline void ezdp_free_obj (ezdp_mem_pool_t * *memory_pool*, ezdp_sum_addr_t *free_obj*) `[static]`

Free a single object from a memory pool.

Parameters:

[in] *memory_pool* - the pool to recycle to
[in] *free_obj* - object to release

Returns:

none

static __always_inline ezdp_sum_addr_t ezdp_get_obj (ezdp_mem_pool_t * *memory_pool*, uint32_t *object_id*) `[static]`

Get object based on object id.

Parameters:

[in] *memory_pool* - the pool to calculate block address from
[in] *object_id* - block index

Returns:

Summarized address (ezdp_sum_addr_t) of the block. In case of error, null summarized address is returned. Use ezdp_is_null_sum_addr API to check it.

static __always_inline uint32_t ezdp_read_free_objs (ezdp_mem_pool_t * *memory_pool*) `[static]`

The number of objects available to be obtained.

Parameters:

[in] *memory_pool* - the pool

Returns:

uint32_t - number of available/free indexes

dpe/dp/include/ezdp_pool_defs.h File Reference

Data Structures

- struct ezdp_mem_pool_config

memory pool configuration data structure Defines

- #define EZDP_NULL_INDEX 0xFFFFFFFF

Typedefs

- typedef struct ezdp_memory_pool ezdp_mem_pool_t

Define Documentation

#define EZDP_NULL_INDEX 0xFFFFFFFF

Typedef Documentation

typedef struct ezdp_memory_pool ezdp_mem_pool_t

dpe/dp/include/ezdp_processor.h File Reference

Defines

- #define EZDP_MAX_HW_THREADS 16
- #define EZDP_MAX_HW_CORES 16
- #define EZDP_MAX_HW_CLUSTERS 16
- #define EZDP_MAX_CPUS_ID (EZDP_MAX_THREADS * EZDP_MAX_CORES * EZDP_MAX_CLUSTERS)

Functions

- static uint32_t ezdp_get_cpu_id (void)
- *Get the logical id of the processor that the process is running on (0-4095).* static uint32_t ezdp_get_thread_id (void)
- *Get the id of the thread (within the core) that the process is running on (0-15).* static uint32_t ezdp_get_core_id (void)
- *Get the ID of the core (within the cluster) that the process is running on (0-15).* static uint32_t ezdp_get_cluster_id (void)
- *Get the ID of the cluster that the process is running on (0-15).* static uint32_t ezdp_calc_cpu_id (uint8_t hw_cluster_id, uint8_t hw_core_id, uint8_t hw_thread_id)
- *Calculate the logical ID of a processor.* static void ezdp_sync (void)
- *Relinquish the execution unit until all outstanding transactions complete.* static void ezdp_rsync (void)
- *Relinquish the execution unit until all outstanding read transactions complete.* static void ezdp_mb (void)
- *Wait until all outstanding memory accesses complete.* static void ezdp_rmb (void)
- *Wait until all outstanding memory read accesses complete.* static void ezdp_wmb (void)

Wait until all outstanding memory write accesses complete.

Define Documentation

#define EZDP_MAX_HW_THREADS 16

#define EZDP_MAX_HW_CORES 16

#define EZDP_MAX_HW_CLUSTERS 16

#define EZDP_MAX_CPUS_ID (EZDP_MAX_THREADS * EZDP_MAX_CORES * EZDP_MAX_CLUSTERS)

Function Documentation

static uint32_t ezdp_get_cpu_id (void) `[inline, static]`

Get the logical id of the processor that the process is running on (0-4095).

Returns:
The logical processor id (0-4095)

static uint32_t ezdp_get_thread_id (void) `[inline, static]`

Get the id of the thread (within the core) that the process is running on (0-15).

Returns:
The thread ID (0-15)

static uint32_t ezdp_get_core_id (void) `[inline, static]`

Get the ID of the core (within the cluster) that the process is running on (0-15).

Returns:
The core ID (0-15)

static uint32_t ezdp_get_cluster_id (void) `[inline, static]`

Get the ID of the cluster that the process is running on (0-15).

Returns:
The cluster ID (0-15)

static uint32_t ezdp_calc_cpu_id (uint8_t *hw_cluster_id*, uint8_t *hw_core_id*, uint8_t *hw_thread_id*) `[inline, static]`

Calculate the logical ID of a processor.

Parameters:
[in] *hw_cluster_id* - hardware ID of the cluster
[in] *hw_core_id* - hardware ID of the core (within the cluster)
[in] *hw_thread_id* - hardware ID of the thread (within the core)

Returns:
Logical processor ID (0-4095)

static void ezdp_sync (void) `[inline, static]`

Relinquish the execution unit until all outstanding transactions complete.

When called, the hardware performs a context switch to another eligible thread to utilize the execution unit while the existing thread waits for its transaction to complete.

Returns:
none

static void ezdp_rsync (void) `[inline, static]`

Relinquish the execution unit until all outstanding read transactions complete.

When called, the hardware performs a context switch to another eligible thread to utilize the execution unit while the existing thread waits for its transaction to complete.

Returns:
none

static void ezdp_mb (void) `[inline, static]`

Wait until all outstanding memory accesses complete.

Returns:
none

static void ezdp_rmb (void) `[inline, static]`

Wait until all outstanding memory read accesses complete.

Returns:
none

static void ezdp_wmb (void) `[inline, static]`

Wait until all outstanding memory write accesses complete.

Returns:
none

dpe/dp/include/ezdp_queue.h File Reference

Functions

- static __always_inline bool ezdp_init_ring (ezdp_ring_t *ring, struct ezdp_ring_cfg *ring_config)
- *Initialize ring.* static __always_inline bool ezdp_ring_empty (ezdp_ring_t *ring)
- *Check if ring is empty.* static __always_inline bool ezdp_ring_full (ezdp_ring_t *ring)
- *Check if array_queue is full.* static __always_inline uint32_t ezdp_ring_length (ezdp_ring_t *ring)
- *Return the number of entries in ring.* static __always_inline bool ezdp_enqueue_ring (ezdp_ring_t *ring, void __cmem *data, uint32_t size, char __cmem *work_area_ptr, uint32_t work_area_size, uint32_t flags)
- *Insert new entry into ring.* static __always_inline bool ezdp_dequeue_ring (ezdp_ring_t *ring, void __cmem *data, uint32_t size, char __cmem *work_area_ptr, uint32_t work_area_size, uint32_t flags)
- *Remove a head entry from ring.* static __always_inline bool ezdp_init_list (ezdp_list_t *list, struct ezdp_list_cfg *list_config, char __cmem *work_area_ptr, uint32_t work_area_size)
- *Initialize list.* static __always_inline bool ezdp_list_empty (ezdp_list_t *list)
- *Check if a list is empty.* static __always_inline bool ezdp_enqueue_list (ezdp_list_t *list, void *data, uint8_t size, char __cmem *work_area_ptr, uint32_t work_area_size, uint32_t flags)
- *Insert a new entry into a list.* static __always_inline bool ezdp_dequeue_list (ezdp_list_t *list, void *data, uint8_t size, char __cmem *work_area_ptr, uint32_t work_area_size, uint32_t flags)
- *Remove a head entry from a list.* static __always_inline bool ezdp_peek_list (ezdp_list_t *list, void *data, uint8_t size, char __cmem *work_area_ptr, uint32_t work_area_size, uint32_t flags)
- *Peek at head entry of a list.* static __always_inline bool ezdp_destroy_list (ezdp_list_t *list)

Destroy a list.

Function Documentation

static __always_inline bool ezdp_init_ring (ezdp_ring_t * *ring*, struct ezdp_ring_cfg * *ring_config*) [static]

Initialize ring.

Parameters:

[out] *ring* - pointer to ring object
[in] *ring_config* - ring configuration

Note:

ring_config->control_addr address must be in resolution of 16 bytes

Returns:

true- success false - failure

static __always_inline bool ezdp_ring_empty (ezdp_ring_t * *ring*) [static]

Check if ring is empty.

Parameters:

[in] *ring* - pointer to ring object

Returns:

true - queue is empty false - queue is not empty

static __always_inline bool ezdp_ring_full (ezdp_ring_t * *ring*) [static]

Check if array_queue is full.

Parameters:

[in] *ring* - pointer to ring object

Returns:

true - ring is full, false - ring is not full

static __always_inline uint32_t ezdp_ring_length (ezdp_ring_t * *ring*) [static]

Return the number of entries in ring.

Parameters:

[in] *ring* - pointer to ring object

Returns:

uint32_t

static __always_inline bool ezdp_enqueue_ring (ezdp_ring_t * *ring*, void __cmem * *data*, uint32_t *size*, char __cmem * *work_area_ptr*, uint32_t *work_area_size*, uint32_t *flags*) [static]

Insert new entry into ring.

Parameters:

[in] *ring* - pointer to ring object
[in] *data* - entry data
[in] *size* - entry data size
[in,out] *work_area_ptr* - work area pointer (temporary memory on CMEM to be used by the function). The size of the temporary memory is determined by EZDP_RING_WORK_AREA_SIZE
[in] *work_area_size* - size of work area pointer
[in] *flags* - execution flags. Bitwise OR of zero or more flags out of the following:
EZDP_MEMORY_FLAG_OVERWRITE - override ECC (do not merge)

Returns:

true - data inserted to queue successfully false - unable to insert data to queue

static __always_inline bool ezdp_dequeue_ring (ezdp_ring_t * *ring*, void __cmem * *data*, uint32_t *size*, char __cmem * *work_area_ptr*, uint32_t *work_area_size*, uint32_t *flags*) [static]

Remove a head entry from ring.

Parameters:

[in] *ring* - pointer ring object
[out] *data* - entry data
[in] *size* - entry data size
[in,out] *work_area_ptr* - work area pointer (temporary memory on CMEM to be used by the function). The size of the temporary memory is determined by EZDP_RING_WORK_AREA_SIZE
[in] *work_area_size* - size of work area pointer
[in] *flags* - execution flags. Bitwise OR of zero or more flags out of the following:
EZDP_MEMORY_FLAG_OVERWRITE - override ECC (do not merge)

Returns:

true - return & remove entry from queue successfully false - unable to return & remove entry from queue

static __always_inline bool ezdp_init_list (ezdp_list_t * *list*, struct ezdp_list_cfg * *list_config*, char __cmem * *work_area_ptr*, uint32_t *work_area_size*) [static]

Initialize list.

Parameters:

[out] *list* - a pointer to list object to initialize
[in] *list_config* - configuration information
[in,out] *work_area_ptr* - work area pointer (temporary memory on CMEM to be used by the function). The size of the temporary memory is determined by EZDP_LIST_WORK_AREA_SIZE
[in] *work_area_size* - size of work area pointer

Returns:

true - success initialization

static __always_inline bool ezdp_list_empty (ezdp_list_t * *list*) [static]

Check if a list is empty.

Parameters:

[in] *list* - pointer to list object

Returns:

true - list is empty false - list is not empty

static __always_inline bool ezdp_enqueue_list (ezdp_list_t * *list*, void * *data*, uint8_t *size*, char __cmem * *work_area_ptr*, uint32_t *work_area_size*, uint32_t *flags*) [static]

Insert a new entry into a list.

Parameters:

[in] *list* - pointer to list object
[in] *data* - data to enqueue
[in] *size* - size of data to enqueue (limited to 12B)
[in,out] *work_area_ptr* - work area pointer (temporary memory on CMEM to be used by the function). The size of the temporary memory is determined by EZDP_LIST_WORK_AREA_SIZE
[in] *work_area_size* - size of work area pointer
[in] *flags* - execution flags. Bitwise OR of zero or more flags out of the following:
EZDP_MEMORY_FLAG_OVERWRITE - override ECC (do not merge)

Returns:

true - data inserted to queue successfully false - unable to insert data to queue

static __always_inline bool ezdp_dequeue_list (ezdp_list_t * *list*, void * *data*, uint8_t *size*, char __cmem * *work_area_ptr*, uint32_t *work_area_size*, uint32_t *flags*) [static]

Remove a head entry from a list.

Parameters:

[in] *list* - pointer to the list object
[out] *data* - data to dequeue
[in] *size* - size of data to dequeue (limited to 12B)

[in,out] *work_area_ptr* - work area pointer (temporary memory on CMEM to be used by the function). The size of the temporary memory is determined by EZDP_LIST_WORK_AREA_SIZE
[in] *work_area_size* - size of work area pointer
[in] *flags* - execution flags. Bitwise OR of zero or more flags out of the following:
EZDP_MEMORY_FLAG_OVERWRITE - override ECC (do not merge)

Returns:

true - return & remove entry from queue successfully false - unable to return & remove entry from queue.

static __always_inline bool ezdp_peek_list (ezdp_list_t * *list*, void * *data*, uint8_t *size*, char __cmem * *work_area_ptr*, uint32_t *work_area_size*, uint32_t *flags*) `[static]`

Peek at head entry of a list.

Parameters:

[in] *list* - pointer to the list object
[out] *data* - data on peeked head
[in] *size* - size of data to peek (limited to 12B)
[in,out] *work_area_ptr* - work area pointer (temporary memory on CMEM to be used by the function). The size of the temporary memory is determined by EZDP_LIST_WORK_AREA_SIZE
[in] *work_area_size* - size of work area pointer
[in] *flags* - execution flags. Bitwise OR of zero or more flags out of the following:
EZDP_MEMORY_FLAG_OVERWRITE - override ECC (do not merge)

Returns:

true - return & remove entry from list successfully false - unable to return & remove entry from queue.

static __always_inline bool ezdp_destroy_list (ezdp_list_t * *list*) `[static]`

Destroy a list.

Parameters:

[out] *list* - pointer to the list object

Returns:

true - operation succeeded. false - queue is not empty,

dpe/dp/include/ezdp_queue_defs.h File Reference

Data Structures

- struct ezdp_ring_cfg
- *ring (array queue) configuration data structure* struct ezdp_list_cfg

list queue configuration data structure Defines

- #define EZDP_RING_WORK_AREA_SIZE sizeof(struct ezdp_ring_working_area)
- *Work area minimal required size definitions.* #define EZDP_LIST_WORK_AREA_SIZE sizeof(struct ezdp_list_working_area)

Typedefs

- typedef struct ezdp_ring ezdp_ring_t
- typedef struct ezdp_list ezdp_list_t

Define Documentation

#define EZDP_RING_WORK_AREA_SIZE sizeof(struct ezdp_ring_working_area)

Work area minimal required size definitions.

#define EZDP_LIST_WORK_AREA_SIZE sizeof(struct ezdp_list_working_area)

Typedef Documentation

typedef struct ezdp_ring ezdp_ring_t

typedef struct ezdp_list ezdp_list_t

dpe/dp/include/ezdp_search.h File Reference

Functions

- static __always_inline uint32_t ezdp_init_table_struct_desc (uint32_t struct_id, ezdp_table_struct_desc_t *table_struct_desc, char __cmem *work_area_ptr, uint32_t work_area_size)
- *Initialize the structure descriptor for a table structure.* static __always_inline uint32_t ezdp_validate_table_struct_desc (ezdp_table_struct_desc_t *table_struct_desc, uint32_t entry_size)
- *Validate the table structure parameters.* static __always_inline uint32_t ezdp_lookup_table_entry (ezdp_table_struct_desc_t *struct_desc, uint32_t key, void __cmem *entry_ptr, uint32_t entry_ptr_size, uint32_t flags)
- *Lookup an entry in a table structure.* static __always_inline uint32_t ezdp_add_table_entry (ezdp_table_struct_desc_t *struct_desc, uint32_t key, void __cmem *entry_ptr, uint32_t entry_ptr_size, uint32_t flags, char __cmem *work_area_ptr, uint32_t work_area_size)
- *Add an entry in a table structure.* static __always_inline uint32_t ezdp_modify_table_entry (ezdp_table_struct_desc_t *struct_desc, uint32_t key, void __cmem *entry_ptr, uint32_t entry_ptr_size, uint32_t flags, char __cmem *work_area_ptr, uint32_t work_area_size)
- *Modify an existing entry in a table structure.* static __always_inline uint32_t ezdp_update_table_entry (ezdp_table_struct_desc_t *struct_desc, uint32_t key, void __cmem *entry_ptr, uint32_t entry_ptr_size, uint32_t flags, char __cmem *work_area_ptr, uint32_t work_area_size)
- *Update an entry in a table structure.* static __always_inline uint32_t ezdp_delete_table_entry (ezdp_table_struct_desc_t *struct_desc, uint32_t key, uint32_t flags, char __cmem *work_area_ptr, uint32_t work_area_size)
- *Delete an entry from a table structure.* static __always_inline uint32_t ezdp_init_hash_struct_desc (uint32_t struct_id, ezdp_hash_struct_desc_t *hash_struct_desc, char __cmem *work_area_ptr, uint32_t work_area_size)
- *Initialize the structure descriptor for a hash structure.* static __always_inline uint32_t ezdp_validate_hash_struct_desc (ezdp_hash_struct_desc_t *hash_struct_desc, bool single_cycle, uint32_t key_size, uint32_t result_size, uint32_t entry_size)
- *Validate the hash structure parameters.* static __always_inline uint32_t ezdp_lookup_hash_entry (ezdp_hash_struct_desc_t *struct_desc, void __cmem *key_ptr, uint32_t key_ptr_size, void **result_ptr, uint32_t *result_ptr_size, uint32_t flags, char __cmem *work_area_ptr, uint32_t work_area_size)
- *Lookup an entry in a hash structure.* static __always_inline uint32_t ezdp_add_hash_entry (ezdp_hash_struct_desc_t *struct_desc, void __cmem *key_ptr, uint32_t key_ptr_size, void __cmem *result_ptr, uint32_t result_ptr_size, uint32_t flags, char __cmem *work_area_ptr, uint32_t work_area_size)
- *Add an entry in a hash structure.* static __always_inline uint32_t ezdp_modify_hash_entry (ezdp_hash_struct_desc_t *struct_desc, void __cmem *key_ptr, uint32_t key_ptr_size, void __cmem *result_ptr, uint32_t result_ptr_size, uint32_t flags, char __cmem *work_area_ptr, uint32_t work_area_size)
- *Modify an existing entry in a hash structure.* static __always_inline uint32_t ezdp_update_hash_entry (ezdp_hash_struct_desc_t *struct_desc, void __cmem *key_ptr, uint32_t key_ptr_size, void __cmem *result_ptr, uint32_t result_ptr_size, uint32_t flags, char __cmem *work_area_ptr, uint32_t work_area_size)
- *Update an entry in a hash structure.* static __always_inline uint32_t ezdp_delete_hash_entry (ezdp_hash_struct_desc_t *struct_desc, void __cmem *key_ptr, uint32_t key_ptr_size, uint32_t flags, char __cmem *work_area_ptr, uint32_t work_area_size)
- *Delete an entry from a hash structure.* static __always_inline uint32_t ezdp_scan_hash_slot (ezdp_hash_struct_desc_t *struct_desc, uint32_t slot_num, ezdp_scan_entry_cb scan_cb, uintptr_t user_data, uint32_t flags, char __cmem *work_area_ptr, uint32_t work_area_size)
- *Scan a hash slot.* static __always_inline void ezdp_get_hash_entry_key (uint32_t key_size, uint32_t result_size, uint32_t entry_size, void *entry_ptr, void *key_ptr)
- *Get hash key from entry.* static __always_inline uint32_t ezdp_init_ultra_ip_struct_desc (uint32_t struct_id, ezdp_ultra_ip_struct_desc_t __cmem *uip_struct_desc, char __cmem *work_area_ptr, uint32_t work_area_size)
- *Initialize the structure descriptor for a UltraIP structure.* static __always_inline uint32_t ezdp_validate_ultra_ip_struct_desc (ezdp_ultra_ip_struct_desc_t *uip_struct_desc, uint32_t key_size)
- *Validate the UltraIP structure parameters.* static __always_inline uint32_t ezdp_lookup_ultra_ip_entry (ezdp_ultra_ip_struct_desc_t *struct_desc, void __cmem *key_ptr, uint32_t key_size, void *result_ptr, uint32_t result_size, char __cmem *work_area_ptr, uint32_t work_area_size)

- *Lookup an entry in an UltraIP structure.* static __always_inline ezdp_lookup_int_tcam_retval_t ezdp_lookup_int_tcam (uint32_t side, uint32_t lookup_profile, void *__cmem key_ptr, uint32_t key_size, struct ezdp_lookup_int_tcam_result *__cmem result_ptr)
- *Lookup an entry in an internal TCAM.* static __always_inline void ezdp_lookup_int_tcam_async (uint32_t side, uint32_t lookup_profile, void *__cmem key_ptr, uint32_t key_size, struct ezdp_lookup_int_tcam_result *__cmem result_ptr)
- *Non blocking version of ezdp_lookup_int_tcam().* static __always_inline ezdp_lookup_ext_tcam_retval_t ezdp_lookup_ext_tcam (uint32_t side, uint32_t lookup_profile, void *__cmem key_ptr, uint32_t key_size, char *__cmem result_ptr, uint32_t result_len)
- *Lookup an entry in an external TCAM.* static __always_inline void ezdp_lookup_ext_tcam_async (uint32_t side, uint32_t lookup_profile, void *__cmem key_ptr, uint32_t key_size, char *__cmem result_ptr, uint32_t result_len)
- *Non blocking version of ezdp_lookup_ext_tcam().* static __always_inline uint32_t ezdp_init_alg_tcam_struct_desc (uint32_t struct_id, ezdp_alg_tcam_struct_desc_t *alg_tcam_struct_desc, char __cmem *work_area_ptr, uint32_t work_area_size)
- *Initialize the structure descriptor for an algorithmic TCAM structure.* static __always_inline uint32_t ezdp_validate_alg_tcam_struct_desc (ezdp_alg_tcam_struct_desc_t *alg_tcam_struct_desc, uint32_t key_size)
- *Validate the algorithmic TCAM structure parameters.* static __always_inline uint32_t ezdp_lookup_alg_tcam (ezdp_alg_tcam_struct_desc_t *struct_desc, uint8_t __cmem *key_ptr, uint32_t key_size, uint32_t *priority_ptr, char __cmem *work_area_ptr, uint32_t work_area_size)

Lookup an entry in an algorithmic TCAM structure.

Function Documentation

static __always_inline uint32_t ezdp_init_table_struct_desc (uint32_t *struct_id*, ezdp_table_struct_desc_t * *table_struct_desc*, char __cmem * *work_area_ptr*, uint32_t *work_area_size*) [static]

Initialize the structure descriptor for a table structure.

Parameters:

[in] *struct_id* - search struct id
[out] *table_struct_desc* - table struct descriptor
[in,out] *work_area_ptr* - work area pointer (temporary memory on CMEM to be used by the function). The size of the temporary memory is determined by EZDP_TABLE_LOW_LEVEL_WORK_AREA_SIZE or EZDP_TABLE_HIGH_LEVEL_WORK_AREA_SIZE(entry_size).
[in] *work_area_size* - size of work area pointer

Returns:

0 (success), EINVAL (failure) use ezdp_get_err_msg() API to get the detail error message of the failure

static __always_inline uint32_t ezdp_validate_table_struct_desc (ezdp_table_struct_desc_t * *table_struct_desc*, uint32_t *entry_size*) [static]

Validate the table structure parameters.

Check that table structure descriptor ,initialized by ezdp_init_table_struct_desc, is match the parameters of the dp application

Parameters:

[in] *table_struct_desc* - table struct descriptor
[in] *entry_size* - size of the table entry (result)

Returns:

0 (success), EINVAL (invalid argument) use ezdp_get_err_msg() API to get the detail error message of the failure

static __always_inline uint32_t ezdp_lookup_table_entry (ezdp_table_struct_desc_t * *struct_desc*, uint32_t *key*, void __cmem * *entry_ptr*, uint32_t *entry_ptr_size*, uint32_t *flags*) [static]

Lookup an entry in a table structure.

The lookup result is written to CMEM. Retry if there is memory error

Parameters:

[in] *struct_desc* - table structure description
[in] *key* - index into table
[out] *entry_ptr* - pointer (in CMEM) to write table entry (result)
[in] *entry_ptr_size* - size of the table entry (result)
[in] *flags* - flags

Note:

When the entry is not found (no match), entry_ptr is not valid.

Returns:

0 (found), ENOENT (not found), EIO (memory error)

static __always_inline uint32_t ezdp_add_table_entry (ezdp_table_struct_desc_t * *struct_desc*, uint32_t *key*, void __cmem * *entry_ptr*, uint32_t *entry_ptr_size*, uint32_t *flags*, char __cmem * *work_area_ptr*, uint32_t *work_area_size*) [static]

Add an entry in a table structure.

The function fail if entry with the same key already exist. The procedure: Lock table line, lookup entry, handle memory error (if no flag is set), if not found add entry, unlock.

Parameters:

[in] *struct_desc* - table structure description
[in] *key* - index into table
[out] *entry_ptr* - pointer to updated result (in CMEM)
[in] *entry_ptr_size* - size of the table entry (result)
[in] *flags* - flags {EZDP_OPPORTUNISTIC, EZDP_UNCONDITIONAL}
[in,out] *work_area_ptr* - work area pointer (temporary memory on CMEM to be used by the function). The size of the temporary memory is determined by EZDP_TABLE_HIGH_LEVEL_WORK_AREA_SIZE(entry_size).
[in] *work_area_size* - size of work area pointer

Note:

When unconditional flag is ON: add entry without lock and without lookup. When opportunistic flag is ON: if already locked or memory error, will return with success. When opportunistic flag is OFF: unlimited memory error handling and waiting for lock. if found already, will return with failure.

Returns:

0 (success), EEXIST (already exist), EIO (memory error) In debug mode use ezdp_get_err_msg() API to get the detail error message of the failure

static __always_inline uint32_t ezdp_modify_table_entry (ezdp_table_struct_desc_t * *struct_desc*, uint32_t *key*, void __cmem * *entry_ptr*, uint32_t *entry_ptr_size*, uint32_t *flags*, char __cmem * *work_area_ptr*, uint32_t *work_area_size*) [static]

Modify an existing entry in a table structure.

The function fail if entry doesn't exist. The procedure: Lock table line, lookup entry, handle memory error (if no flag is set), if found modify entry, unlock.

Parameters:

[in] *struct_desc* - table structure description
[in] *key* - index into table
[out] *entry_ptr* - pointer to updated result (in CMEM)

[in] *entry_ptr_size* - size of the table entry (result)
[in] *flags* - flags {EZDP_OPPORTUNISTIC, EZDP_UNCONDITIONAL}
[in,out] *work_area_ptr* - work area pointer (temporary memory on CMEM to be used by the function). The size of the temporary memory is determined by EZDP_TABLE_HIGH_LEVEL_WORK_AREA_SIZE(entry_size).
[in] *work_area_size* - size of work area pointer

Note:

When unconditional flag is ON: modify entry without lock and without lookup. When opportunistic flag is ON: if already locked or memory error, will return with success. When opportunistic flag is OFF: unlimited memory error handling and waiting for lock. if not found, will return with failure.

Returns:

0 (success), ENOENT (not found), EIO (memory error) In debug mode use ezdp_get_err_msg() API to get the detail error message of the failure

static __always_inline uint32_t ezdp_update_table_entry (ezdp_table_struct_desc_t * *struct_desc*, uint32_t *key*, void __cmem * *entry_ptr*, uint32_t *entry_ptr_size*, uint32_t *flags*, char __cmem * *work_area_ptr*, uint32_t *work_area_size*) `[static]`

Update an entry in a table structure.

The function add entry if it doesn't exist and update it if entry with the same key was found. The procedure: Lock table line, lookup entry, handle memory error (if no flag is set), if found modify otherwise add entry, unlock.

Parameters:

[in] *struct_desc* - table structure description
[in] *key* - index into table
[out] *entry_ptr* - pointer to updated result (in CMEM)
[in] *entry_ptr_size* - size of the table entry (result)
[in] *flags* - flags {EZDP_OPPORTUNISTIC, EZDP_UNCONDITIONAL}
[in,out] *work_area_ptr* - work area pointer (temporary memory on CMEM to be used by the function). The size of the temporary memory is determined by EZDP_TABLE_HIGH_LEVEL_WORK_AREA_SIZE(entry_size).
[in] *work_area_size* - size of work area pointer

Note:

When unconditional flag is ON: update entry without lock and without lookup. When opportunistic flag is ON: if already locked or memory error, will return with success. When opportunistic flag is OFF: unlimited memory error handling and waiting for lock.

Returns:

0 (success), EIO (memory error) In debug mode use ezdp_get_err_msg() API to get the detail error message of the failure

static __always_inline uint32_t ezdp_delete_table_entry (ezdp_table_struct_desc_t * *struct_desc*, uint32_t *key*, uint32_t *flags*, char __cmem * *work_area_ptr*, uint32_t *work_area_size*) `[static]`

Delete an entry from a table structure.

The function fail if entry doesn't exist. The procedure: Lock table line, lookup entry, handle memory error (if no flag is set), if found delete entry, unlock.

Parameters:

[in] *struct_desc* - table structure description
[in] *key* - index into table
[in] *flags* - flags {EZDP_OPPORTUNISTIC, EZDP_UNCONDITIONAL}
[in,out] *work_area_ptr* - work area pointer (temporary memory on CMEM to be used by the function). The size of the temporary memory is determined by EZDP_TABLE_HIGH_LEVEL_WORK_AREA_SIZE(entry_size).

[in] *work_area_size* - size of work area pointer

Note:

When unconditional flag is ON: delete entry without lock and without lookup. When opportunistic flag is ON: if already locked or memory error, will return with success. When opportunistic flag is OFF: unlimited memory error handling and waiting for lock. if not found, will return with failure.

Returns:

0 (success), ENOENT (not found), EIO (memory error) In debug mode use ezdp_get_err_msg() API to get the detail error message of the failure

static __always_inline uint32_t ezdp_init_hash_struct_desc (uint32_t *struct_id*, ezdp_hash_struct_desc_t * *hash_struct_desc*, char __cmem * *work_area_ptr*, uint32_t *work_area_size*) `[static]`

Initialize the structure descriptor for a hash structure.

Parameters:

[in] *struct_id* - search struct id
[out] *hash_struct_desc* - hash struct descriptor
[in] *work_area_ptr* - work area pointer (temporary memory on CMEM to be used by the function). The size of the temporary memory is determined by EZDP_HASH_LOW_LEVEL_WORK_AREA_SIZE or EZDP_HASH_HIGH_LEVEL_WORK_AREA_SIZE(result_size, key_size)
[in] *work_area_size* - size of work area pointer

Returns:

0 (success), EINVAL (failure) In debug mode use ezdp_get_err_msg() API to get the detail error message of the failure

static __always_inline uint32_t ezdp_validate_hash_struct_desc (ezdp_hash_struct_desc_t * *hash_struct_desc*, bool *single_cycle*, uint32_t *key_size*, uint32_t *result_size*, uint32_t *entry_size*) `[static]`

Validate the hash structure parameters.

Check that hash structure descriptor ,initialized by ezdp_init_hash_struct_desc, is match the parameters of the dp application

Parameters:

[in] *hash_struct_desc* - hash struct descriptor
[in] *single_cycle* - single-cycle or non-single-cycle hash
[in] *key_size* - size of the key
[in] *result_size* - size of the result (for best performance should be a multiple of 4)
[in] *entry_size* - size of the entry

Returns:

0 (success), EINVAL (invalid argument) use ezdp_get_err_msg() API to get the detail error message of the failure

static __always_inline uint32_t ezdp_lookup_hash_entry (ezdp_hash_struct_desc_t * *struct_desc*, void __cmem * *key_ptr*, uint32_t *key_ptr_size*, void ** *result_ptr*, uint32_t * *result_ptr_size*, uint32_t *flags*, char __cmem * *work_area_ptr*, uint32_t *work_area_size*) `[static]`

Lookup an entry in a hash structure.

Handle memory error. The lookup result is written to CMEM. WARNING! For best performance result_size should be a multiple of 4

Parameters:

[in] *struct_desc* - hash struct descriptor

[in] *key_ptr* - pointer to key (in CMEM)
[in] *key_ptr_size* - key size
[out] *result_ptr* - pointer to returned result on work area
[out] *result_ptr_size* - returned result size
[in] *flags* - flags
[in,out] *work_area_ptr* - work area pointer (temporary memory on CMEM to be used by the function). The size of the temporary memory is determined by
EZDP_HASH_HIGH_LEVEL_WORK_AREA_SIZE(result_size, key_size)
[in] *work_area_size* - size of work area pointer

Note:

When the entry is not found (no match), result_ptr is not valid.
result_ptr point to the data in work area. If you want to reuse the work area but still want to have the result you have to copy the data from work area to other place.

Returns:

0 (found), ENOENT (not found), EIO (memory error)

static __always_inline uint32_t ezdp_add_hash_entry (ezdp_hash_struct_desc_t * *struct_desc*, void __cmem * *key_ptr*, uint32_t *key_ptr_size*, void __cmem * *result_ptr*, uint32_t *result_ptr_size*, uint32_t *flags*, char __cmem * *work_area_ptr*, uint32_t *work_area_size*) `[static]`

Add an entry in a hash structure.

The function fail if entry already exist. The procedure: Lock table line, lookup entry, handle memory error (if no flag is set), if not found add entry, unlock.

Parameters:

[in] *struct_desc* - hash struct descriptor
[in] *key_ptr* - pointer to key (in CMEM)
[in] *key_ptr_size* - key size
[in] *result_ptr* - pointer to updated result (in CMEM)
[in] *result_ptr_size* - result size
[in] *flags* - flags {EZDP_OPPORTUNISTIC}
[out] *work_area_ptr* - work area pointer (temporary memory on CMEM to be used by the function). The size of the temporary memory is determined by
EZDP_HASH_HIGH_LEVEL_WORK_AREA_SIZE(result_size, key_size)
[in] *work_area_size* - size of work area pointer

Note:

When opportunistic flag is ON: if already locked or memory error or hash is full, will return with success.
When opportunistic flag is OFF: unlimited memory error handling and waiting for lock. if found already or hash is full, will return with failure.

Returns:

0 (success), EEXIST (already exist), ENOMEM (hash is full), EIO (memory error) In debug mode use ezdp_get_err_msg() API to get the detail error message of the failure

static __always_inline uint32_t ezdp_modify_hash_entry (ezdp_hash_struct_desc_t * *struct_desc*, void __cmem * *key_ptr*, uint32_t *key_ptr_size*, void __cmem * *result_ptr*, uint32_t *result_ptr_size*, uint32_t *flags*, char __cmem * *work_area_ptr*, uint32_t *work_area_size*) `[static]`

Modify an existing entry in a hash structure.

The function fail if entry doesn't exist. The procedure: Lock table line, lookup entry, handle memory error (if no flag is set), if found modify entry, unlock.

Parameters:

[in] *struct_desc* - hash struct descriptor
[in] *key_ptr* - pointer to key (in CMEM)

[in] *key_ptr_size* - key size
[in] *result_ptr* - pointer to updated result (in CMEM)
[in] *result_ptr_size* - result size
[in] *flags* - flags {EZDP_OPPORTUNISTIC}
[in,out] *work_area_ptr* - work area pointer (temporary memory on CMEM to be used by the function). The size of the temporary memory is determined by
EZDP_HASH_HIGH_LEVEL_WORK_AREA_SIZE(result_size, key_size)
[in] *work_area_size* - size of work area pointer

Note:

When opportunistic flag is ON: if already locked or memory error, will return with success. When opportunistic flag is OFF: unlimited memory error handling and waiting for lock. if not found, will return with failure.

Returns:

0 (success), ENOENT (not found), EIO (memory error) In debug mode use ezdp_get_err_msg() API to get the detail error message of the failure

static __always_inline uint32_t ezdp_update_hash_entry (ezdp_hash_struct_desc_t * *struct_desc*, void __cmem * *key_ptr*, uint32_t *key_ptr_size*, void __cmem * *result_ptr*, uint32_t *result_ptr_size*, uint32_t *flags*, char __cmem * *work_area_ptr*, uint32_t *work_area_size*) `[static]`

Update an entry in a hash structure.

The function add entry if it doesn't exist or modify if entry found The procedure: Lock table line, lookup entry, handle memory error (if no flag is set), if found modify entry otherwise add entry, unlock.

Parameters:

[in] *struct_desc* - hash struct descriptor
[in] *key_ptr* - pointer to key (in CMEM)
[in] *key_ptr_size* - key size
[in] *result_ptr* - pointer to updated result (in CMEM)
[in] *result_ptr_size* - result size
[in] *flags* - flags {EZDP_OPPORTUNISTIC}
[in,out] *work_area_ptr* - work area pointer (temporary memory on CMEM to be used by the function). The size of the temporary memory is determined by
EZDP_HASH_HIGH_LEVEL_WORK_AREA_SIZE(result_size, key_size)
[in] *work_area_size* - size of work area pointer

Note:

When opportunistic flag is ON: if already locked or memory error or hash is full, will return with success. When opportunistic flag is OFF: unlimited memory error handling and waiting for lock. if hash is full, will return with failure.

Returns:

0 (success), ENOMEM (hash is full), EIO (memory error) In debug mode use ezdp_get_err_msg() API to get the detail error message of the failure

static __always_inline uint32_t ezdp_delete_hash_entry (ezdp_hash_struct_desc_t * *struct_desc*, void __cmem * *key_ptr*, uint32_t *key_ptr_size*, uint32_t *flags*, char __cmem * *work_area_ptr*, uint32_t *work_area_size*) `[static]`

Delete an entry from a hash structure.

The function fail if entry doesn't exist. The procedure: Lock table line, lookup entry, handle memory error (if no flag is set), if found delete entry, unlock.

Parameters:

[in] *struct_desc* - hash struct descriptor
[in] *key_ptr* - pointer to key (in CMEM)

[in] *key_ptr_size* - key size
[in] *flags* - flags {EZDP_OPPORTUNISTIC}
[in,out] *work_area_ptr* - work area pointer (temporary memory on CMEM to be used by the function). The size of the temporary memory is determined by
EZDP_HASH_HIGH_LEVEL_WORK_AREA_SIZE(result_size, key_size)
[in] *work_area_size* - size of work area pointer

Note:

When opportunistic flag is ON: if already locked or memory error, will return with success. When opportunistic flag is OFF: unlimited memory error handling and waiting for lock.

Returns:

0 (success), ENOENT (not found), EIO (memory error) In debug mode use ezdp_get_err_msg() API to get the detail error message of the failure

static __always_inline uint32_t ezdp_scan_hash_slot (ezdp_hash_struct_desc_t * *struct_desc*, uint32_t *slot_num*, ezdp_scan_entry_cb *scan_cb*, uintptr_t *user_data*, uint32_t *flags*, char __cmem * *work_area_ptr*, uint32_t *work_area_size*) `[static]`

Scan a hash slot.

Lock hash slot, go over all the entries, handle memory error (if EZDP_OPPORTUNISTIC flag is not set), apply scan_cb on each entry and act according to the returned answer, compress (ifEZDP_COMPRESS set), unlock.

Parameters:

[in] *struct_desc* - hash struct descriptor
[in] *slot_num* - slot number to scan
[in] *scan_cb* - function to call on each entry in slot
[in] *user_data* - additional data provided by the user for scan_cb
[in] *flags* - flags {EZDP_OPPORTUNISTIC, EZDP_COMPRESS}
[out] *work_area_ptr* - work area pointer (temporary memory on CMEM to be used by the function). The size of the temporary memory is determined by
EZDP_HASH_HIGH_LEVEL_WORK_AREA_SIZE(result_size, key_size)
[in] *work_area_size* - size of work area pointer

Note:

When opportunistic flag is ON: if already locked or memory error, will return with success. When opportunistic flag is OFF: unlimited memory error handling and waiting for lock.

Returns:

0 (success), EIO (memory error) In debug mode use ezdp_get_err_msg() API to get the detail error message of the failure

static __always_inline void ezdp_get_hash_entry_key (uint32_t *key_size*, uint32_t *result_size*, uint32_t *entry_size*, void * *entry_ptr*, void * *key_ptr*) `[static]`

Get hash key from entry.

The key of the hash entry is copied to the key_ptr.

Parameters:

[in] *key_size* - size of the key
[in] *result_size* - size of the result
[in] *entry_size* - size of the entry
[in] *entry_ptr* - pointer to entry for internal use
[out] *key_ptr* - pointer to return key

Returns:

void

static __always_inline uint32_t ezdp_init_ultra_ip_struct_desc (uint32_t *struct_id*, ezdp_ultra_ip_struct_desc_t __cmem * *uip_struct_desc*, char __cmem * *work_area_ptr*, uint32_t *work_area_size*) `[static]`

Initialize the structure descriptor for a UltraIP structure.

Parameters:

[in] *struct_id* - search struct id.
[out] *uip_struct_desc* - UltraIP struct descriptor
[in,out] *work_area_ptr* - work area pointer (temporary memory on CMEM to be used by the function). The size of the temporary memory is determined by EZDP_ULTRA_IP_WORK_AREA_SIZE.
[in] *work_area_size* - size of work area pointer

Returns:

0 (success), EINVAL (failure) use ezdp_get_err_msg() API to get the detail error message of the failure

static __always_inline uint32_t ezdp_validate_ultra_ip_struct_desc (ezdp_ultra_ip_struct_desc_t * *uip_struct_desc*, uint32_t *key_size*) `[static]`

Validate the UltraIP structure parameters.

Check that UltraIP structure descriptor, initialized by ezdp_init_ultra_ip_struct_desc, is match the parameters of the dp application

Parameters:

[in] *uip_struct_desc* - UltraIP struct descriptor
[in] *key_size* - size of the key

Returns:

0 (success), EINVAL (invalid argument) use ezdp_get_err_msg() API to get the detail error message of the failure

static __always_inline uint32_t ezdp_lookup_ultra_ip_entry (ezdp_ultra_ip_struct_desc_t * *struct_desc*, void __cmem * *key_ptr*, uint32_t *key_size*, void * *result_ptr*, uint32_t *result_size*, char __cmem * *work_area_ptr*, uint32_t *work_area_size*) `[static]`

Lookup an entry in an UltraIP structure.

This function also handle memory error.

Parameters:

[in] *struct_desc* - main table summarized address descriptor
[in] *key_ptr* - pointer to key (in CMEM)
[in] *key_size* - size of key
[out] *result_ptr* - pointer to write the result
[in] *result_size* - size of result
[in,out] *work_area_ptr* - pointer to work area (temporary memory in CMEM to be used by the function). The size of the temporary memory is determined by EZDP_ULTRA_IP_WORK_AREA_SIZE
[in] *work_area_size* - size of work area pointer

Returns:

0 (found), ENOENT (not found), EIO (memory error)

static __always_inline ezdp_lookup_int_tcam_retval_t ezdp_lookup_int_tcam (uint32_t *side*, uint32_t *lookup_profile*, void *__cmem *key_ptr*, uint32_t *key_size*, struct ezdp_lookup_int_tcam_result *__cmem *result_ptr*) `[static]`

Lookup an entry in an internal TCAM.

The lookup is according to the TCAM profile. The lookup result is written to CMEM. In addition, the first 4 bytes of the lookup result are returned.

Parameters:

[in] *side* - NPS-400 has two internal TCAM engines. This field select which one to use (the iTCAM on side 0 or on side 1).
[in] *lookup_profile* - lookup profile id. select the search definition, which defines up to 4 profiles
[in] *key_ptr* - pointer to key (in CMEM)
[in] *key_size* - size of the key; limited to 10, 20, 30, 40, 50, 60, 70, 80 bytes
[out] *result_ptr* - pointer to result (in CMEM)

Returns:

ezdp_lookup_int_tcam_retval_t - according to ezdp_lookup_int_tcam_retval

static __always_inline void ezdp_lookup_int_tcam_async (uint32_t *side*, uint32_t *lookup_profile*, void *__cmem *key_ptr*, uint32_t *key_size*, struct ezdp_lookup_int_tcam_result *__cmem *result_ptr*) `[static]`

Non blocking version of ezdp_lookup_int_tcam().

Parameters:

[in] *side* - NPS-400 has two internal TCAM engines. This field select which one to use (the iTCAM on side 0 or on side 1).
[in] *lookup_profile* - lookup profile id. select the search definition, which defines up to 4 profiles
[in] *key_ptr* - pointer to key (in CMEM)
[in] *key_size* - size of the key; limited to 10, 20, 30, 40, 50, 60, 70, 80 bytes
[out] *result_ptr* - pointer to result (in CMEM)

Note:

Call ezdp_sync() to wait for the operation to complete, confirming that the lookup result is ready in CMEM.

Returns:

none

static __always_inline ezdp_lookup_ext_tcam_retval_t ezdp_lookup_ext_tcam (uint32_t *side*, uint32_t *lookup_profile*, void *__cmem *key_ptr*, uint32_t *key_size*, char *__cmem *result_ptr*, uint32_t *result_len*) `[static]`

Lookup an entry in an external TCAM.

The lookup is according to the TCAM profile The lookup result is written to CMEM. RFLAGS are returned.

Parameters:

[in] *side* - NPS-400 supports two external TCAM engines. This field select which one to use (the eTCAM on side 0 or on side 1).
[in] *lookup_profile* - lookup profile id. select the search definition profile
[in] *key_ptr* - pointer to key (in CMEM)
[in] *key_size* - size of the key; limited to 8, 10, 16, 20, 24, 30, 32, 40 and 80 bytes
[out] *result_ptr* - pointer to result (in CMEM) which consists of up to 6 result elements, based on lookup_profile definition. a result element may be one of the following:
ezdp_lookup_ext_tcam_index_result_element ezdp_lookup_ext_tcam_index_4B_data_result_element ezdp_lookup_ext_tcam_index_8B_data_result_element ezdp_lookup_ext_tcam_index_16B_data_result_element ezdp_lookup_ext_tcam_index_32B_data_result_element ezdp_lookup_ext_tcam_4B_data_result_element ezdp_lookup_ext_tcam_8B_data_result_element ezdp_lookup_ext_tcam_16B_data_result_element ezdp_lookup_ext_tcam_32B_data_result_element
[in] *result_len* - the lookup result length limit in bytes; 16, 32, 48 and 64 bytes

Returns:

ezdp_lookup_ext_tcam_retval_t - according to ezdp_lookup_ext_tcam_retval

static __always_inline void ezdp_lookup_ext_tcam_async (uint32_t *side*, uint32_t *lookup_profile*, void *__cmem *key_ptr*, uint32_t *key_size*, char *__cmem *result_ptr*, uint32_t *result_len*) `[static]`

Non blocking version of ezdp_lookup_ext_tcam().

Parameters:

[in] *side* - NPS-400 supports two external TCAM engines. This field select which one to use (the eTCAM on side 0 or on side 1).
[in] *lookup_profile* - lookup profile id. select the search definition profile
[in] *key_ptr* - pointer to key (in CMEM)
[in] *key_size* - size of the key; limited to 8, 10, 16, 20, 24, 30, 32, 40 and 80 bytes
[out] *result_ptr* - pointer to result (in CMEM) which consists of up to 6 result elements, based on lookup_profile definition. a result element may be one of the following:
ezdp_lookup_ext_tcam_index_result_element ezdp_lookup_ext_tcam_index_4B_data_result_element
ezdp_lookup_ext_tcam_index_8B_data_result_element
ezdp_lookup_ext_tcam_index_16B_data_result_element
ezdp_lookup_ext_tcam_index_32B_data_result_element ezdp_lookup_ext_tcam_4B_data_result_element
ezdp_lookup_ext_tcam_8B_data_result_element ezdp_lookup_ext_tcam_16B_data_result_element
ezdp_lookup_ext_tcam_32B_data_result_element
[in] *result_len* - the lookup result length limit in bytes; 16, 32, 48 and 64 bytes

Returns:

void

static __always_inline uint32_t ezdp_init_alg_tcam_struct_desc (uint32_t *struct_id*, ezdp_alg_tcam_struct_desc_t * *alg_tcam_struct_desc*, char __cmem * *work_area_ptr*, uint32_t *work_area_size*) `[static]`

Initialize the structure descriptor for an algorithmic TCAM structure.

Parameters:

[in] *struct_id* - search struct id.
[out] *alg_tcam_struct_desc* - algorithmic TCAM struct descriptor
[in] *work_area_ptr* - pointer to work area (temporary memory in CMEM to be used by the function). The size of the temporary memory is determined by EZDP_ALG_TCAM_WORK_AREA_SIZE
[in] *work_area_size* - size of work area pointer

Returns:

0 (success), EINVAL (failure). use ezdp_get_err_msg() API to get the detail error message of the failure

static __always_inline uint32_t ezdp_validate_alg_tcam_struct_desc (ezdp_alg_tcam_struct_desc_t * *alg_tcam_struct_desc*, uint32_t *key_size*) `[static]`

Validate the algorithmic TCAM structure parameters.

Check that alg TCAM structure descriptor, initialized by ezdp_init_alg_tcam_struct_desc, is match the parameters of the dp application

Parameters:

[in] *alg_tcam_struct_desc* - algorithmic TCAM struct descriptor
[in] *key_size* - size of the key

Returns:

0 (success), EINVAL (invalid argument) use ezdp_get_err_msg() API to get the detail error message of the failure

static __always_inline uint32_t ezdp_lookup_alg_tcam (ezdp_alg_tcam_struct_desc_t * *struct_desc*, uint8_t __cmem * *key_ptr*, uint32_t *key_size*, uint32_t * *priority_ptr*, char __cmem * *work_area_ptr*, uint32_t *work_area_size*) [static]

Lookup an entry in an algorithmic TCAM structure.

Handle memory error.

Parameters:

[in] *struct_desc* - algo tcam struct descriptor
[in] *key_ptr* - pointer to the key in CMEM
[in] *key_size* - size of the key
[out] *priority_ptr* - pointer to returned priority
[in] *work_area_ptr* - pointer to work area (temporary memory in CMEM to be used by the function). The size of the temporary memory is determined by EZDP_ALG_TCAM_WORK_AREA_SIZE
[in] *work_area_size* - size of work area pointer

Returns:

errno: 0 (found), ENOENT (not found), EIO (memory error).

dpe/dp/include/ezdp_search_defs.h File Reference

Data Structures

- struct ezdp_lookup_retval
- *Lookup return value.* struct ezdp_lookup_int_tcam_standard_result
- *Lookup internal tcam standard result.* struct ezdp_lookup_int_tcam_4B_data_result
- *Lookup internal tcam 4 byte associated data result.* struct ezdp_lookup_int_tcam_8B_data_result
- *Lookup internal tcam 8 byte associated data result.* struct ezdp_lookup_int_tcam_12B_data_result
- *Lookup internal tcam 12 byte associated data result.* struct ezdp_lookup_int_tcam_16B_data_result
- *Lookup internal tcam 16 byte associated data result.* struct ezdp_lookup_ext_tcam_retval
- *Lookup external tcam return value.* struct ezdp_lookup_ext_tcam_index_result_element
- *Lookup external tcam index result element.* struct ezdp_lookup_ext_tcam_index_4B_data_result_element
- *Lookup external tcam index result with 4 Byte associated data.* struct ezdp_lookup_ext_tcam_index_8B_data_result_element
- *Lookup external tcam index result with 8 Byte associated data.* struct ezdp_lookup_ext_tcam_index_16B_data_result_element
- *Lookup external tcam index result with 16 Byte associated data.* struct ezdp_lookup_ext_tcam_index_32B_data_result_element
- *Lookup external tcam index result with 32 Byte associated data.* struct ezdp_lookup_ext_tcam_4B_data_result_element
- *Lookup external tcam 4 Byte associated data only result.* struct ezdp_lookup_ext_tcam_8B_data_result_element
- *Lookup external tcam 8 Byte associated data only result.* struct ezdp_lookup_ext_tcam_16B_data_result_element
- *Lookup external tcam 16 Byte associated data only result.* struct ezdp_lookup_ext_tcam_32B_data_result_element
- *Lookup external tcam 32 Byte associated data only result.* struct ezdp_lookup_int_tcam_result
- *Lookup ITCAM result definition.* struct ezdp_lookup_int_tcam_retval

Lookup ITCAM retval definition. Defines

- #define EZDP_LOOKUP_VERSION_MAJOR 2
- #define EZDP_LOOKUP_VERSION_MINOR 1
- #define EZDP_ALG_TCAM_MAX_KEY_SIZE 128
- #define EZDP_PAD_ALG_TCAM_WORKING_AREA(key_size)
- #define EZDP_PAD_HASH_ENTRY(result_size, key_size) uint8_t __pad[_EZDP_HASH_CALC_ENTRY_PADDING_SIZE(result_size, key_size)]
- #define EZDP_PAD_HASH_WORKING_AREA(result_size, key_size) uint8_t __pad[_EZDP_LOOKUP_HASH_CALC_ENTRY_SIZE(result_size, key_size) + sizeof(ezdp_hash_op_ctx_t)]
- #define EZDP_TABLE_LOW_LEVEL_WORK_AREA_SIZE _EZDP_TABLE_LOW_LEVEL_WORK_AREA_SIZE
- *Work area minimal required size definitions.* #define EZDP_TABLE_HIGH_LEVEL_WORK_AREA_SIZE(entry_size) _EZDP_TABLE_HIGH_LEVEL_WORK_AREA_SIZE(entry_size)
- #define EZDP_HASH_LOW_LEVEL_WORK_AREA_SIZE _EZDP_HASH_LOW_LEVEL_WORK_AREA_SIZE
- #define EZDP_HASH_HIGH_LEVEL_WORK_AREA_SIZE(result_size, key_size) _EZDP_HASH_HIGH_LEVEL_WORK_AREA_SIZE(result_size, key_size)
- #define EZDP_ULTRA_IP_WORK_AREA_SIZE _EZDP_ULTRA_IP_WORK_AREA_SIZE
- #define EZDP_ALG_TCAM_WORK_AREA_SIZE(max_key_size) _EZDP_ALG_TCAM_WORK_AREA_SIZE(max_key_size)
- #define EZDP_LOOKUP_PARITY_BITS_SIZE 3
- #define EZDP_LOOKUP_RESERVED_BITS_SIZE 1
- #define EZDP_LOOKUP_RETVAL_DATA_SIZE 28
- #define EZDP_LOOKUP_RETVAL_DATA_OFFSET 0

- #define EZDP_LOOKUP_RETVAL_MATCH_SIZE 1
- #define EZDP_LOOKUP_RETVAL_MATCH_OFFSET 28
- #define EZDP_LOOKUP_RETVAL_MATCH_MASK (1 << EZDP_LOOKUP_RETVAL_MATCH_OFFSET)
- #define EZDP_LOOKUP_RETVAL_SUCCESS_SIZE 1
- #define EZDP_LOOKUP_RETVAL_SUCCESS_OFFSET 29
- #define EZDP_LOOKUP_RETVAL_SUCCESS_MASK (1 << EZDP_LOOKUP_RETVAL_SUCCESS_OFFSET)
- #define EZDP_LOOKUP_RETVAL_INFO_SIZE 1
- #define EZDP_LOOKUP_RETVAL_INFO_OFFSET 30
- #define EZDP_LOOKUP_RETVAL_INFO_MASK (1 << EZDP_LOOKUP_RETVAL_INFO_OFFSET)
- #define EZDP_LOOKUP_RETVAL_MEM_ERROR_SIZE 1
- #define EZDP_LOOKUP_RETVAL_MEM_ERROR_OFFSET 31
- #define EZDP_LOOKUP_RETVAL_MEM_ERROR_MASK (1 << EZDP_LOOKUP_RETVAL_MEM_ERROR_OFFSET)
- #define EZDP_LOOKUP_INT_TCAM_STANDARD_RESULT_INDEX_SIZE 15
- #define EZDP_LOOKUP_INT_TCAM_STANDARD_RESULT_INDEX_OFFSET 0
- #define EZDP_LOOKUP_INT_TCAM_STANDARD_RESULT_RESERVED0_15_SIZE 16
- #define EZDP_LOOKUP_INT_TCAM_STANDARD_RESULT_RESERVED0_15_OFFSET 15
- #define EZDP_LOOKUP_INT_TCAM_STANDARD_RESULT_MATCH_SIZE 1
- #define EZDP_LOOKUP_INT_TCAM_STANDARD_RESULT_MATCH_OFFSET 31
- #define EZDP_LOOKUP_INT_TCAM_STANDARD_RESULT_MATCH_MASK (1 << EZDP_LOOKUP_INT_TCAM_STANDARD_RESULT_MATCH_OFFSET)
- #define EZDP_LOOKUP_INT_TCAM_4B_DATA_RESULT_DATA_SIZE 31
- #define EZDP_LOOKUP_INT_TCAM_4B_DATA_RESULT_DATA_OFFSET 0
- #define EZDP_LOOKUP_INT_TCAM_4B_DATA_RESULT_MATCH_SIZE 1
- #define EZDP_LOOKUP_INT_TCAM_4B_DATA_RESULT_MATCH_OFFSET 31
- #define EZDP_LOOKUP_INT_TCAM_4B_DATA_RESULT_MATCH_MASK (1 << EZDP_LOOKUP_INT_TCAM_4B_DATA_RESULT_MATCH_OFFSET)
- #define EZDP_LOOKUP_INT_TCAM_8B_DATA_RESULT_DATA0_SIZE 31
- #define EZDP_LOOKUP_INT_TCAM_8B_DATA_RESULT_DATA0_OFFSET 0
- #define EZDP_LOOKUP_INT_TCAM_8B_DATA_RESULT_DATA0_WORD_SELECT 0
- #define EZDP_LOOKUP_INT_TCAM_8B_DATA_RESULT_DATA0_WORD_OFFSET 0
- #define EZDP_LOOKUP_INT_TCAM_8B_DATA_RESULT_MATCH_SIZE 1
- #define EZDP_LOOKUP_INT_TCAM_8B_DATA_RESULT_MATCH_OFFSET 31
- #define EZDP_LOOKUP_INT_TCAM_8B_DATA_RESULT_MATCH_WORD_SELECT 0
- #define EZDP_LOOKUP_INT_TCAM_8B_DATA_RESULT_MATCH_WORD_OFFSET 31
- #define EZDP_LOOKUP_INT_TCAM_8B_DATA_RESULT_MATCH_MASK (1 << EZDP_LOOKUP_INT_TCAM_8B_DATA_RESULT_MATCH_WORD_OFFSET)
- #define EZDP_LOOKUP_INT_TCAM_8B_DATA_RESULT_DATA1_SIZE 32
- #define EZDP_LOOKUP_INT_TCAM_8B_DATA_RESULT_DATA1_OFFSET 32
- #define EZDP_LOOKUP_INT_TCAM_8B_DATA_RESULT_DATA1_WORD_SELECT 1
- #define EZDP_LOOKUP_INT_TCAM_8B_DATA_RESULT_DATA1_WORD_OFFSET 0
- #define EZDP_LOOKUP_INT_TCAM_8B_DATA_RESULT_WORD_COUNT 2
- #define EZDP_LOOKUP_INT_TCAM_12B_DATA_RESULT_DATA0_SIZE 31
- #define EZDP_LOOKUP_INT_TCAM_12B_DATA_RESULT_DATA0_OFFSET 0
- #define EZDP_LOOKUP_INT_TCAM_12B_DATA_RESULT_DATA0_WORD_SELECT 0
- #define EZDP_LOOKUP_INT_TCAM_12B_DATA_RESULT_DATA0_WORD_OFFSET 0
- #define EZDP_LOOKUP_INT_TCAM_12B_DATA_RESULT_MATCH_SIZE 1
- #define EZDP_LOOKUP_INT_TCAM_12B_DATA_RESULT_MATCH_OFFSET 31
- #define EZDP_LOOKUP_INT_TCAM_12B_DATA_RESULT_MATCH_WORD_SELECT 0
- #define EZDP_LOOKUP_INT_TCAM_12B_DATA_RESULT_MATCH_WORD_OFFSET 31
- #define EZDP_LOOKUP_INT_TCAM_12B_DATA_RESULT_MATCH_MASK (1 << EZDP_LOOKUP_INT_TCAM_12B_DATA_RESULT_MATCH_WORD_OFFSET)
- #define EZDP_LOOKUP_INT_TCAM_12B_DATA_RESULT_DATA1_SIZE 32
- #define EZDP_LOOKUP_INT_TCAM_12B_DATA_RESULT_DATA1_OFFSET 32
- #define EZDP_LOOKUP_INT_TCAM_12B_DATA_RESULT_DATA1_WORD_SELECT 1

- #define EZDP_LOOKUP_INT_TCAM_12B_DATA_RESULT_DATA1_WORD_OFFSET 0
- #define EZDP_LOOKUP_INT_TCAM_12B_DATA_RESULT_DATA2_SIZE 32
- #define EZDP_LOOKUP_INT_TCAM_12B_DATA_RESULT_DATA2_OFFSET 64
- #define EZDP_LOOKUP_INT_TCAM_12B_DATA_RESULT_DATA2_WORD_SELECT 2
- #define EZDP_LOOKUP_INT_TCAM_12B_DATA_RESULT_DATA2_WORD_OFFSET 0
- #define EZDP_LOOKUP_INT_TCAM_12B_DATA_RESULT_WORD_COUNT 3
- #define EZDP_LOOKUP_INT_TCAM_16B_DATA_RESULT_DATA0_SIZE 31
- #define EZDP_LOOKUP_INT_TCAM_16B_DATA_RESULT_DATA0_OFFSET 0
- #define EZDP_LOOKUP_INT_TCAM_16B_DATA_RESULT_DATA0_WORD_SELECT 0
- #define EZDP_LOOKUP_INT_TCAM_16B_DATA_RESULT_DATA0_WORD_OFFSET 0
- #define EZDP_LOOKUP_INT_TCAM_16B_DATA_RESULT_MATCH_SIZE 1
- #define EZDP_LOOKUP_INT_TCAM_16B_DATA_RESULT_MATCH_OFFSET 31
- #define EZDP_LOOKUP_INT_TCAM_16B_DATA_RESULT_MATCH_WORD_SELECT 0
- #define EZDP_LOOKUP_INT_TCAM_16B_DATA_RESULT_MATCH_WORD_OFFSET 31
- #define EZDP_LOOKUP_INT_TCAM_16B_DATA_RESULT_MATCH_MASK (1 << EZDP_LOOKUP_INT_TCAM_16B_DATA_RESULT_MATCH_WORD_OFFSET)
- #define EZDP_LOOKUP_INT_TCAM_16B_DATA_RESULT_DATA1_SIZE 32
- #define EZDP_LOOKUP_INT_TCAM_16B_DATA_RESULT_DATA1_OFFSET 32
- #define EZDP_LOOKUP_INT_TCAM_16B_DATA_RESULT_DATA1_WORD_SELECT 1
- #define EZDP_LOOKUP_INT_TCAM_16B_DATA_RESULT_DATA1_WORD_OFFSET 0
- #define EZDP_LOOKUP_INT_TCAM_16B_DATA_RESULT_DATA2_SIZE 32
- #define EZDP_LOOKUP_INT_TCAM_16B_DATA_RESULT_DATA2_OFFSET 64
- #define EZDP_LOOKUP_INT_TCAM_16B_DATA_RESULT_DATA2_WORD_SELECT 2
- #define EZDP_LOOKUP_INT_TCAM_16B_DATA_RESULT_DATA2_WORD_OFFSET 0
- #define EZDP_LOOKUP_INT_TCAM_16B_DATA_RESULT_DATA3_SIZE 32
- #define EZDP_LOOKUP_INT_TCAM_16B_DATA_RESULT_DATA3_OFFSET 96
- #define EZDP_LOOKUP_INT_TCAM_16B_DATA_RESULT_DATA3_WORD_SELECT 3
- #define EZDP_LOOKUP_INT_TCAM_16B_DATA_RESULT_DATA3_WORD_OFFSET 0
- #define EZDP_LOOKUP_INT_TCAM_16B_DATA_RESULT_WORD_COUNT 4
- #define EZDP_LOOKUP_EXT_TCAM_RETVAL_NO_CONTEXT_MATCH_ERROR_SIZE 1
- #define EZDP_LOOKUP_EXT_TCAM_RETVAL_NO_CONTEXT_MATCH_ERROR_OFFSET 0
- #define EZDP_LOOKUP_EXT_TCAM_RETVAL_NO_CONTEXT_MATCH_ERROR_MASK (1 << EZDP_LOOKUP_EXT_TCAM_RETVAL_NO_CONTEXT_MATCH_ERROR_OFFSET)
- #define EZDP_LOOKUP_EXT_TCAM_RETVAL_MAC_ERROR_SIZE 1
- #define EZDP_LOOKUP_EXT_TCAM_RETVAL_MAC_ERROR_OFFSET 1
- #define EZDP_LOOKUP_EXT_TCAM_RETVAL_MAC_ERROR_MASK (1 << EZDP_LOOKUP_EXT_TCAM_RETVAL_MAC_ERROR_OFFSET)
- #define EZDP_LOOKUP_EXT_TCAM_RETVAL_DEVICE_ERROR_SIZE 1
- #define EZDP_LOOKUP_EXT_TCAM_RETVAL_DEVICE_ERROR_OFFSET 2
- #define EZDP_LOOKUP_EXT_TCAM_RETVAL_DEVICE_ERROR_MASK (1 << EZDP_LOOKUP_EXT_TCAM_RETVAL_DEVICE_ERROR_OFFSET)
- #define EZDP_LOOKUP_EXT_TCAM_RETVAL_TIME_OUT_ERROR_SIZE 1
- #define EZDP_LOOKUP_EXT_TCAM_RETVAL_TIME_OUT_ERROR_OFFSET 3
- #define EZDP_LOOKUP_EXT_TCAM_RETVAL_TIME_OUT_ERROR_MASK (1 << EZDP_LOOKUP_EXT_TCAM_RETVAL_TIME_OUT_ERROR_OFFSET)
- #define EZDP_LOOKUP_EXT_TCAM_RETVAL_ANY_MATCH_SIZE 1
- #define EZDP_LOOKUP_EXT_TCAM_RETVAL_ANY_MATCH_OFFSET 4
- #define EZDP_LOOKUP_EXT_TCAM_RETVAL_ANY_MATCH_MASK (1 << EZDP_LOOKUP_EXT_TCAM_RETVAL_ANY_MATCH_OFFSET)
- #define EZDP_LOOKUP_EXT_TCAM_RETVAL_MULTI_MATCH_SIZE 1
- #define EZDP_LOOKUP_EXT_TCAM_RETVAL_MULTI_MATCH_OFFSET 5
- #define EZDP_LOOKUP_EXT_TCAM_RETVAL_MULTI_MATCH_MASK (1 << EZDP_LOOKUP_EXT_TCAM_RETVAL_MULTI_MATCH_OFFSET)
- #define EZDP_LOOKUP_EXT_TCAM_RETVAL_TRUNCATED_SIZE 1
- #define EZDP_LOOKUP_EXT_TCAM_RETVAL_TRUNCATED_OFFSET 6
- #define EZDP_LOOKUP_EXT_TCAM_RETVAL_TRUNCATED_MASK (1 << EZDP_LOOKUP_EXT_TCAM_RETVAL_TRUNCATED_OFFSET)

- #define EZDP_LOOKUP_EXT_TCAM_RETVAL_LOOKUP_ERROR_SIZE 1
- #define EZDP_LOOKUP_EXT_TCAM_RETVAL_LOOKUP_ERROR_OFFSET 7
- #define EZDP_LOOKUP_EXT_TCAM_RETVAL_LOOKUP_ERROR_MASK (1 << EZDP_LOOKUP_EXT_TCAM_RETVAL_LOOKUP_ERROR_OFFSET)
- #define EZDP_LOOKUP_EXT_TCAM_RETVAL_RESERVED_BIT8_31_SIZE 24
- #define EZDP_LOOKUP_EXT_TCAM_RETVAL_RESERVED_BIT8_31_OFFSET 8
- #define EZDP_LOOKUP_EXT_TCAM_INDEX_RESULT_ELEMENT_INDEX_SIZE 21
- #define EZDP_LOOKUP_EXT_TCAM_INDEX_RESULT_ELEMENT_INDEX_OFFSET 0
- #define EZDP_LOOKUP_EXT_TCAM_INDEX_RESULT_ELEMENT_DEVICE_ID_SIZE 2
- #define EZDP_LOOKUP_EXT_TCAM_INDEX_RESULT_ELEMENT_DEVICE_ID_OFFSET 21
- #define EZDP_LOOKUP_EXT_TCAM_INDEX_RESULT_ELEMENT_RESERVED23_SIZE 1
- #define EZDP_LOOKUP_EXT_TCAM_INDEX_RESULT_ELEMENT_RESERVED23_OFFSET 23
- #define EZDP_LOOKUP_EXT_TCAM_INDEX_RESULT_ELEMENT_ANY_MATCH_SIZE 1
- #define EZDP_LOOKUP_EXT_TCAM_INDEX_RESULT_ELEMENT_ANY_MATCH_OFFSET 24
- #define EZDP_LOOKUP_EXT_TCAM_INDEX_RESULT_ELEMENT_ANY_MATCH_MASK (1 << EZDP_LOOKUP_EXT_TCAM_INDEX_RESULT_ELEMENT_ANY_MATCH_OFFSET)
- #define EZDP_LOOKUP_EXT_TCAM_INDEX_RESULT_ELEMENT_TRUNCATED_SIZE 1
- #define EZDP_LOOKUP_EXT_TCAM_INDEX_RESULT_ELEMENT_TRUNCATED_OFFSET 25
- #define EZDP_LOOKUP_EXT_TCAM_INDEX_RESULT_ELEMENT_TRUNCATED_MASK (1 << EZDP_LOOKUP_EXT_TCAM_INDEX_RESULT_ELEMENT_TRUNCATED_OFFSET)
- #define EZDP_LOOKUP_EXT_TCAM_INDEX_RESULT_ELEMENT_LOOKUP_ERROR_SIZE 1
- #define EZDP_LOOKUP_EXT_TCAM_INDEX_RESULT_ELEMENT_LOOKUP_ERROR_OFFSET 26
- #define EZDP_LOOKUP_EXT_TCAM_INDEX_RESULT_ELEMENT_LOOKUP_ERROR_MASK (1 << EZDP_LOOKUP_EXT_TCAM_INDEX_RESULT_ELEMENT_LOOKUP_ERROR_OFFSET)
- #define EZDP_LOOKUP_EXT_TCAM_INDEX_RESULT_ELEMENT_TYPE_SIZE 3
- #define EZDP_LOOKUP_EXT_TCAM_INDEX_RESULT_ELEMENT_TYPE_OFFSET 27
- #define EZDP_LOOKUP_EXT_TCAM_INDEX_RESULT_ELEMENT_MATCH_SIZE 1
- #define EZDP_LOOKUP_EXT_TCAM_INDEX_RESULT_ELEMENT_MATCH_OFFSET 30
- #define EZDP_LOOKUP_EXT_TCAM_INDEX_RESULT_ELEMENT_MATCH_MASK (1 << EZDP_LOOKUP_EXT_TCAM_INDEX_RESULT_ELEMENT_MATCH_OFFSET)
- #define EZDP_LOOKUP_EXT_TCAM_INDEX_RESULT_ELEMENT_VALID_SIZE 1
- #define EZDP_LOOKUP_EXT_TCAM_INDEX_RESULT_ELEMENT_VALID_OFFSET 31
- #define EZDP_LOOKUP_EXT_TCAM_INDEX_RESULT_ELEMENT_VALID_MASK (1 << EZDP_LOOKUP_EXT_TCAM_INDEX_RESULT_ELEMENT_VALID_OFFSET)
- #define EZDP_LOOKUP_EXT_TCAM_INDEX_4B_DATA_RESULT_ELEMENT_INDEX_SIZE 21
- #define EZDP_LOOKUP_EXT_TCAM_INDEX_4B_DATA_RESULT_ELEMENT_INDEX_OFFSET 0
- #define EZDP_LOOKUP_EXT_TCAM_INDEX_4B_DATA_RESULT_ELEMENT_INDEX_WORD_SELECT 0
- #define EZDP_LOOKUP_EXT_TCAM_INDEX_4B_DATA_RESULT_ELEMENT_INDEX_WORD_OFFSET 0
- #define EZDP_LOOKUP_EXT_TCAM_INDEX_4B_DATA_RESULT_ELEMENT_DEVICE_ID_SIZE 2
- #define EZDP_LOOKUP_EXT_TCAM_INDEX_4B_DATA_RESULT_ELEMENT_DEVICE_ID_OFFSET 21
- #define EZDP_LOOKUP_EXT_TCAM_INDEX_4B_DATA_RESULT_ELEMENT_DEVICE_ID_WORD_SELECT 0
- #define EZDP_LOOKUP_EXT_TCAM_INDEX_4B_DATA_RESULT_ELEMENT_DEVICE_ID_WORD_OFFSET 21
- #define EZDP_LOOKUP_EXT_TCAM_INDEX_4B_DATA_RESULT_ELEMENT_RESERVED23_24_SIZE 2
- #define EZDP_LOOKUP_EXT_TCAM_INDEX_4B_DATA_RESULT_ELEMENT_RESERVED23_24_OFFSET 23
- #define EZDP_LOOKUP_EXT_TCAM_INDEX_4B_DATA_RESULT_ELEMENT_TRUNCATED_SIZE 1
- #define EZDP_LOOKUP_EXT_TCAM_INDEX_4B_DATA_RESULT_ELEMENT_TRUNCATED_OFFSET 25

- #define EZDP_LOOKUP_EXT_TCAM_INDEX_4B_DATA_RESULT_ELEMENT_TRUNCATED_WORD_SELECT 0
- #define EZDP_LOOKUP_EXT_TCAM_INDEX_4B_DATA_RESULT_ELEMENT_TRUNCATED_WORD_OFFSET 25
- #define EZDP_LOOKUP_EXT_TCAM_INDEX_4B_DATA_RESULT_ELEMENT_TRUNCATED_MASK (1 << EZDP_LOOKUP_EXT_TCAM_INDEX_4B_DATA_RESULT_ELEMENT_TRUNCATED_WORD_OFFSET)
- #define EZDP_LOOKUP_EXT_TCAM_INDEX_4B_DATA_RESULT_ELEMENT_LOOKUP_ERROR_SIZE 1
- #define EZDP_LOOKUP_EXT_TCAM_INDEX_4B_DATA_RESULT_ELEMENT_LOOKUP_ERROR_OFFSET 26
- #define EZDP_LOOKUP_EXT_TCAM_INDEX_4B_DATA_RESULT_ELEMENT_LOOKUP_ERROR_WORD_SELECT 0
- #define EZDP_LOOKUP_EXT_TCAM_INDEX_4B_DATA_RESULT_ELEMENT_LOOKUP_ERROR_WORD_OFFSET 26
- #define EZDP_LOOKUP_EXT_TCAM_INDEX_4B_DATA_RESULT_ELEMENT_LOOKUP_ERROR_MASK (1 << EZDP_LOOKUP_EXT_TCAM_INDEX_4B_DATA_RESULT_ELEMENT_LOOKUP_ERROR_WORD_OFFSET)
- #define EZDP_LOOKUP_EXT_TCAM_INDEX_4B_DATA_RESULT_ELEMENT_TYPE_SIZE 3
- #define EZDP_LOOKUP_EXT_TCAM_INDEX_4B_DATA_RESULT_ELEMENT_TYPE_OFFSET 27
- #define EZDP_LOOKUP_EXT_TCAM_INDEX_4B_DATA_RESULT_ELEMENT_TYPE_WORD_SELECT 0
- #define EZDP_LOOKUP_EXT_TCAM_INDEX_4B_DATA_RESULT_ELEMENT_TYPE_WORD_OFFSET 27
- #define EZDP_LOOKUP_EXT_TCAM_INDEX_4B_DATA_RESULT_ELEMENT_MATCH_SIZE 1
- #define EZDP_LOOKUP_EXT_TCAM_INDEX_4B_DATA_RESULT_ELEMENT_MATCH_OFFSET 30
- #define EZDP_LOOKUP_EXT_TCAM_INDEX_4B_DATA_RESULT_ELEMENT_MATCH_WORD_SELECT 0
- #define EZDP_LOOKUP_EXT_TCAM_INDEX_4B_DATA_RESULT_ELEMENT_MATCH_WORD_OFFSET 30
- #define EZDP_LOOKUP_EXT_TCAM_INDEX_4B_DATA_RESULT_ELEMENT_MATCH_MASK (1 << EZDP_LOOKUP_EXT_TCAM_INDEX_4B_DATA_RESULT_ELEMENT_MATCH_WORD_OFFSET)
- #define EZDP_LOOKUP_EXT_TCAM_INDEX_4B_DATA_RESULT_ELEMENT_VALID_SIZE 1
- #define EZDP_LOOKUP_EXT_TCAM_INDEX_4B_DATA_RESULT_ELEMENT_VALID_OFFSET 31
- #define EZDP_LOOKUP_EXT_TCAM_INDEX_4B_DATA_RESULT_ELEMENT_VALID_WORD_SELECT 0
- #define EZDP_LOOKUP_EXT_TCAM_INDEX_4B_DATA_RESULT_ELEMENT_VALID_WORD_OFFSET 31
- #define EZDP_LOOKUP_EXT_TCAM_INDEX_4B_DATA_RESULT_ELEMENT_VALID_MASK (1 << EZDP_LOOKUP_EXT_TCAM_INDEX_4B_DATA_RESULT_ELEMENT_VALID_WORD_OFFSET)
- #define EZDP_LOOKUP_EXT_TCAM_INDEX_4B_DATA_RESULT_ELEMENT_ASSOC_DATA_SIZE 32
- #define EZDP_LOOKUP_EXT_TCAM_INDEX_4B_DATA_RESULT_ELEMENT_ASSOC_DATA_OFFSET 32
- #define EZDP_LOOKUP_EXT_TCAM_INDEX_4B_DATA_RESULT_ELEMENT_ASSOC_DATA_COUNT 4
- #define EZDP_LOOKUP_EXT_TCAM_INDEX_4B_DATA_RESULT_ELEMENT_ASSOC_DATA_WORD_SELECT 1

- #define EZDP_LOOKUP_EXT_TCAM_INDEX_4B_DATA_RESULT_ELEMENT_ASSOC_DATA_WORD_OFFSET 0
- #define EZDP_LOOKUP_EXT_TCAM_INDEX_4B_DATA_RESULT_ELEMENT_WORD_COUNT 2
- #define EZDP_LOOKUP_EXT_TCAM_INDEX_8B_DATA_RESULT_ELEMENT_INDEX_SIZE 21
- #define EZDP_LOOKUP_EXT_TCAM_INDEX_8B_DATA_RESULT_ELEMENT_INDEX_OFFSET 0
- #define EZDP_LOOKUP_EXT_TCAM_INDEX_8B_DATA_RESULT_ELEMENT_INDEX_WORD_SELECT 0
- #define EZDP_LOOKUP_EXT_TCAM_INDEX_8B_DATA_RESULT_ELEMENT_INDEX_WORD_OFFSET 0
- #define EZDP_LOOKUP_EXT_TCAM_INDEX_8B_DATA_RESULT_ELEMENT_DEVICE_ID_SIZE 2
- #define EZDP_LOOKUP_EXT_TCAM_INDEX_8B_DATA_RESULT_ELEMENT_DEVICE_ID_OFFSET 21
- #define EZDP_LOOKUP_EXT_TCAM_INDEX_8B_DATA_RESULT_ELEMENT_DEVICE_ID_WORD_SELECT 0
- #define EZDP_LOOKUP_EXT_TCAM_INDEX_8B_DATA_RESULT_ELEMENT_DEVICE_ID_WORD_OFFSET 21
- #define EZDP_LOOKUP_EXT_TCAM_INDEX_8B_DATA_RESULT_ELEMENT_RESERVED23_24_SIZE 2
- #define EZDP_LOOKUP_EXT_TCAM_INDEX_8B_DATA_RESULT_ELEMENT_RESERVED23_24_OFFSET 23
- #define EZDP_LOOKUP_EXT_TCAM_INDEX_8B_DATA_RESULT_ELEMENT_TRUNCATED_SIZE 1
- #define EZDP_LOOKUP_EXT_TCAM_INDEX_8B_DATA_RESULT_ELEMENT_TRUNCATED_OFFSET 25
- #define EZDP_LOOKUP_EXT_TCAM_INDEX_8B_DATA_RESULT_ELEMENT_TRUNCATED_WORD_SELECT 0
- #define EZDP_LOOKUP_EXT_TCAM_INDEX_8B_DATA_RESULT_ELEMENT_TRUNCATED_WORD_OFFSET 25
- #define EZDP_LOOKUP_EXT_TCAM_INDEX_8B_DATA_RESULT_ELEMENT_TRUNCATED_MASK (1 << EZDP_LOOKUP_EXT_TCAM_INDEX_8B_DATA_RESULT_ELEMENT_TRUNCATED_WORD_OFFSET)
- #define EZDP_LOOKUP_EXT_TCAM_INDEX_8B_DATA_RESULT_ELEMENT_LOOKUP_ERROR_SIZE 1
- #define EZDP_LOOKUP_EXT_TCAM_INDEX_8B_DATA_RESULT_ELEMENT_LOOKUP_ERROR_OFFSET 26
- #define EZDP_LOOKUP_EXT_TCAM_INDEX_8B_DATA_RESULT_ELEMENT_LOOKUP_ERROR_WORD_SELECT 0
- #define EZDP_LOOKUP_EXT_TCAM_INDEX_8B_DATA_RESULT_ELEMENT_LOOKUP_ERROR_WORD_OFFSET 26
- #define EZDP_LOOKUP_EXT_TCAM_INDEX_8B_DATA_RESULT_ELEMENT_LOOKUP_ERROR_MASK (1 << EZDP_LOOKUP_EXT_TCAM_INDEX_8B_DATA_RESULT_ELEMENT_LOOKUP_ERROR_WORD_OFFSET)
- #define EZDP_LOOKUP_EXT_TCAM_INDEX_8B_DATA_RESULT_ELEMENT_TYPE_SIZE 3
- #define EZDP_LOOKUP_EXT_TCAM_INDEX_8B_DATA_RESULT_ELEMENT_TYPE_OFFSET 27
- #define EZDP_LOOKUP_EXT_TCAM_INDEX_8B_DATA_RESULT_ELEMENT_TYPE_WORD_SELECT 0
- #define EZDP_LOOKUP_EXT_TCAM_INDEX_8B_DATA_RESULT_ELEMENT_TYPE_WORD_OFFSET 27
- #define EZDP_LOOKUP_EXT_TCAM_INDEX_8B_DATA_RESULT_ELEMENT_MATCH_SIZE 1

- #define EZDP_LOOKUP_EXT_TCAM_INDEX_8B_DATA_RESULT_ELEMENT_MATCH_OFFSET 30
- #define EZDP_LOOKUP_EXT_TCAM_INDEX_8B_DATA_RESULT_ELEMENT_MATCH_WORD_SELECT 0
- #define EZDP_LOOKUP_EXT_TCAM_INDEX_8B_DATA_RESULT_ELEMENT_MATCH_WORD_OFFSET 30
- #define EZDP_LOOKUP_EXT_TCAM_INDEX_8B_DATA_RESULT_ELEMENT_MATCH_MASK (1 << EZDP_LOOKUP_EXT_TCAM_INDEX_8B_DATA_RESULT_ELEMENT_MATCH_WORD_OFFSET)
- #define EZDP_LOOKUP_EXT_TCAM_INDEX_8B_DATA_RESULT_ELEMENT_VALID_SIZE 1
- #define EZDP_LOOKUP_EXT_TCAM_INDEX_8B_DATA_RESULT_ELEMENT_VALID_OFFSET 31
- #define EZDP_LOOKUP_EXT_TCAM_INDEX_8B_DATA_RESULT_ELEMENT_VALID_WORD_SELECT 0
- #define EZDP_LOOKUP_EXT_TCAM_INDEX_8B_DATA_RESULT_ELEMENT_VALID_WORD_OFFSET 31
- #define EZDP_LOOKUP_EXT_TCAM_INDEX_8B_DATA_RESULT_ELEMENT_VALID_MASK (1 << EZDP_LOOKUP_EXT_TCAM_INDEX_8B_DATA_RESULT_ELEMENT_VALID_WORD_OFFSET)
- #define EZDP_LOOKUP_EXT_TCAM_INDEX_8B_DATA_RESULT_ELEMENT_ASSOC_DATA_SIZE 64
- #define EZDP_LOOKUP_EXT_TCAM_INDEX_8B_DATA_RESULT_ELEMENT_ASSOC_DATA_OFFSET 32
- #define EZDP_LOOKUP_EXT_TCAM_INDEX_8B_DATA_RESULT_ELEMENT_ASSOC_DATA_COUNT 8
- #define EZDP_LOOKUP_EXT_TCAM_INDEX_8B_DATA_RESULT_ELEMENT_ASSOC_DATA_WORD_SELECT 1
- #define EZDP_LOOKUP_EXT_TCAM_INDEX_8B_DATA_RESULT_ELEMENT_ASSOC_DATA_WORD_OFFSET 0
- #define EZDP_LOOKUP_EXT_TCAM_INDEX_8B_DATA_RESULT_ELEMENT_WORD_COUNT 3
- #define EZDP_LOOKUP_EXT_TCAM_INDEX_16B_DATA_RESULT_ELEMENT_INDEX_SIZE 21
- #define EZDP_LOOKUP_EXT_TCAM_INDEX_16B_DATA_RESULT_ELEMENT_INDEX_OFFSET 0
- #define EZDP_LOOKUP_EXT_TCAM_INDEX_16B_DATA_RESULT_ELEMENT_INDEX_WORD_SELECT 0
- #define EZDP_LOOKUP_EXT_TCAM_INDEX_16B_DATA_RESULT_ELEMENT_INDEX_WORD_OFFSET 0
- #define EZDP_LOOKUP_EXT_TCAM_INDEX_16B_DATA_RESULT_ELEMENT_DEVICE_ID_SIZE 2
- #define EZDP_LOOKUP_EXT_TCAM_INDEX_16B_DATA_RESULT_ELEMENT_DEVICE_ID_OFFSET 21
- #define EZDP_LOOKUP_EXT_TCAM_INDEX_16B_DATA_RESULT_ELEMENT_DEVICE_ID_WORD_SELECT 0
- #define EZDP_LOOKUP_EXT_TCAM_INDEX_16B_DATA_RESULT_ELEMENT_DEVICE_ID_WORD_OFFSET 21
- #define EZDP_LOOKUP_EXT_TCAM_INDEX_16B_DATA_RESULT_ELEMENT_RESERVED23_24_SIZE 2
- #define EZDP_LOOKUP_EXT_TCAM_INDEX_16B_DATA_RESULT_ELEMENT_RESERVED23_24_OFFSET 23
- #define EZDP_LOOKUP_EXT_TCAM_INDEX_16B_DATA_RESULT_ELEMENT_TRUNCATED_SIZE 1
- #define EZDP_LOOKUP_EXT_TCAM_INDEX_16B_DATA_RESULT_ELEMENT_TRUNCATED_OFFSET 25
- #define EZDP_LOOKUP_EXT_TCAM_INDEX_16B_DATA_RESULT_ELEMENT_TRUNCATED_WORD_SELECT 0
- #define EZDP_LOOKUP_EXT_TCAM_INDEX_16B_DATA_RESULT_ELEMENT_TRUNCATED_WORD_OFFSET 25
- #define EZDP_LOOKUP_EXT_TCAM_INDEX_16B_DATA_RESULT_ELEMENT_TRUNCATED_MASK (1 <<

EZDP_LOOKUP_EXT_TCAM_INDEX_16B_DATA_RESULT_ELEMENT_TRUNCATED_WORD_OFFSET)

- #define EZDP_LOOKUP_EXT_TCAM_INDEX_16B_DATA_RESULT_ELEMENT_LOOKUP_ERROR_SIZE 1
- #define EZDP_LOOKUP_EXT_TCAM_INDEX_16B_DATA_RESULT_ELEMENT_LOOKUP_ERROR_OFFSET 26
- #define EZDP_LOOKUP_EXT_TCAM_INDEX_16B_DATA_RESULT_ELEMENT_LOOKUP_ERROR_WORD_SELECT 0
- #define EZDP_LOOKUP_EXT_TCAM_INDEX_16B_DATA_RESULT_ELEMENT_LOOKUP_ERROR_WORD_OFFSET 26
- #define EZDP_LOOKUP_EXT_TCAM_INDEX_16B_DATA_RESULT_ELEMENT_LOOKUP_ERROR_MASK (1 << EZDP_LOOKUP_EXT_TCAM_INDEX_16B_DATA_RESULT_ELEMENT_LOOKUP_ERROR_WORD_OFFSET)
- #define EZDP_LOOKUP_EXT_TCAM_INDEX_16B_DATA_RESULT_ELEMENT_TYPE_SIZE 3
- #define EZDP_LOOKUP_EXT_TCAM_INDEX_16B_DATA_RESULT_ELEMENT_TYPE_OFFSET 27
- #define EZDP_LOOKUP_EXT_TCAM_INDEX_16B_DATA_RESULT_ELEMENT_TYPE_WORD_SELECT 0
- #define EZDP_LOOKUP_EXT_TCAM_INDEX_16B_DATA_RESULT_ELEMENT_TYPE_WORD_OFFSET 27
- #define EZDP_LOOKUP_EXT_TCAM_INDEX_16B_DATA_RESULT_ELEMENT_MATCH_SIZE 1
- #define EZDP_LOOKUP_EXT_TCAM_INDEX_16B_DATA_RESULT_ELEMENT_MATCH_OFFSET 30
- #define EZDP_LOOKUP_EXT_TCAM_INDEX_16B_DATA_RESULT_ELEMENT_MATCH_WORD_SELECT 0
- #define EZDP_LOOKUP_EXT_TCAM_INDEX_16B_DATA_RESULT_ELEMENT_MATCH_WORD_OFFSET 30
- #define EZDP_LOOKUP_EXT_TCAM_INDEX_16B_DATA_RESULT_ELEMENT_MATCH_MASK (1 << EZDP_LOOKUP_EXT_TCAM_INDEX_16B_DATA_RESULT_ELEMENT_MATCH_WORD_OFFSET)
- #define EZDP_LOOKUP_EXT_TCAM_INDEX_16B_DATA_RESULT_ELEMENT_VALID_SIZE 1
- #define EZDP_LOOKUP_EXT_TCAM_INDEX_16B_DATA_RESULT_ELEMENT_VALID_OFFSET 31
- #define EZDP_LOOKUP_EXT_TCAM_INDEX_16B_DATA_RESULT_ELEMENT_VALID_WORD_SELECT 0
- #define EZDP_LOOKUP_EXT_TCAM_INDEX_16B_DATA_RESULT_ELEMENT_VALID_WORD_OFFSET 31
- #define EZDP_LOOKUP_EXT_TCAM_INDEX_16B_DATA_RESULT_ELEMENT_VALID_MASK (1 << EZDP_LOOKUP_EXT_TCAM_INDEX_16B_DATA_RESULT_ELEMENT_VALID_WORD_OFFSET)
- #define EZDP_LOOKUP_EXT_TCAM_INDEX_16B_DATA_RESULT_ELEMENT_ASSOC_DATA_SIZE 128
- #define EZDP_LOOKUP_EXT_TCAM_INDEX_16B_DATA_RESULT_ELEMENT_ASSOC_DATA_OFFSET 32
- #define EZDP_LOOKUP_EXT_TCAM_INDEX_16B_DATA_RESULT_ELEMENT_ASSOC_DATA_COUNT 16
- #define EZDP_LOOKUP_EXT_TCAM_INDEX_16B_DATA_RESULT_ELEMENT_ASSOC_DATA_WORD_SELECT 1
- #define EZDP_LOOKUP_EXT_TCAM_INDEX_16B_DATA_RESULT_ELEMENT_ASSOC_DATA_WORD_OFFSET 0
- #define EZDP_LOOKUP_EXT_TCAM_INDEX_16B_DATA_RESULT_ELEMENT_WORD_COUNT 5
- #define EZDP_LOOKUP_EXT_TCAM_INDEX_32B_DATA_RESULT_ELEMENT_INDEX_SIZE 21
- #define EZDP_LOOKUP_EXT_TCAM_INDEX_32B_DATA_RESULT_ELEMENT_INDEX_OFFSET 0
- #define EZDP_LOOKUP_EXT_TCAM_INDEX_32B_DATA_RESULT_ELEMENT_INDEX_WORD_SELECT 0
- #define EZDP_LOOKUP_EXT_TCAM_INDEX_32B_DATA_RESULT_ELEMENT_INDEX_WORD_OFFSET 0

- #define EZDP_LOOKUP_EXT_TCAM_INDEX_32B_DATA_RESULT_ELEMENT_DEVICE_ID_SIZE 2
- #define EZDP_LOOKUP_EXT_TCAM_INDEX_32B_DATA_RESULT_ELEMENT_DEVICE_ID_OFFSET 21
- #define EZDP_LOOKUP_EXT_TCAM_INDEX_32B_DATA_RESULT_ELEMENT_DEVICE_ID_WORD_SELECT 0
- #define EZDP_LOOKUP_EXT_TCAM_INDEX_32B_DATA_RESULT_ELEMENT_DEVICE_ID_WORD_OFFSET 21
- #define EZDP_LOOKUP_EXT_TCAM_INDEX_32B_DATA_RESULT_ELEMENT_RESERVED23_24_SIZE 2
- #define EZDP_LOOKUP_EXT_TCAM_INDEX_32B_DATA_RESULT_ELEMENT_RESERVED23_24_OFFSET 23
- #define EZDP_LOOKUP_EXT_TCAM_INDEX_32B_DATA_RESULT_ELEMENT_TRUNCATED_SIZE 1
- #define EZDP_LOOKUP_EXT_TCAM_INDEX_32B_DATA_RESULT_ELEMENT_TRUNCATED_OFFSET 25
- #define EZDP_LOOKUP_EXT_TCAM_INDEX_32B_DATA_RESULT_ELEMENT_TRUNCATED_WORD_SELECT 0
- #define EZDP_LOOKUP_EXT_TCAM_INDEX_32B_DATA_RESULT_ELEMENT_TRUNCATED_WORD_OFFSET 25
- #define EZDP_LOOKUP_EXT_TCAM_INDEX_32B_DATA_RESULT_ELEMENT_TRUNCATED_MASK (1 << EZDP_LOOKUP_EXT_TCAM_INDEX_32B_DATA_RESULT_ELEMENT_TRUNCATED_WORD_OFFSET)
- #define EZDP_LOOKUP_EXT_TCAM_INDEX_32B_DATA_RESULT_ELEMENT_LOOKUP_ERROR_SIZE 1
- #define EZDP_LOOKUP_EXT_TCAM_INDEX_32B_DATA_RESULT_ELEMENT_LOOKUP_ERROR_OFFSET 26
- #define EZDP_LOOKUP_EXT_TCAM_INDEX_32B_DATA_RESULT_ELEMENT_LOOKUP_ERROR_WORD_SELECT 0
- #define EZDP_LOOKUP_EXT_TCAM_INDEX_32B_DATA_RESULT_ELEMENT_LOOKUP_ERROR_WORD_OFFSET 26
- #define EZDP_LOOKUP_EXT_TCAM_INDEX_32B_DATA_RESULT_ELEMENT_LOOKUP_ERROR_MASK (1 << EZDP_LOOKUP_EXT_TCAM_INDEX_32B_DATA_RESULT_ELEMENT_LOOKUP_ERROR_WORD_OFFSET)
- #define EZDP_LOOKUP_EXT_TCAM_INDEX_32B_DATA_RESULT_ELEMENT_TYPE_SIZE 3
- #define EZDP_LOOKUP_EXT_TCAM_INDEX_32B_DATA_RESULT_ELEMENT_TYPE_OFFSET 27
- #define EZDP_LOOKUP_EXT_TCAM_INDEX_32B_DATA_RESULT_ELEMENT_TYPE_WORD_SELECT 0
- #define EZDP_LOOKUP_EXT_TCAM_INDEX_32B_DATA_RESULT_ELEMENT_TYPE_WORD_OFFSET 27
- #define EZDP_LOOKUP_EXT_TCAM_INDEX_32B_DATA_RESULT_ELEMENT_MATCH_SIZE 1
- #define EZDP_LOOKUP_EXT_TCAM_INDEX_32B_DATA_RESULT_ELEMENT_MATCH_OFFSET 30
- #define EZDP_LOOKUP_EXT_TCAM_INDEX_32B_DATA_RESULT_ELEMENT_MATCH_WORD_SELECT 0
- #define EZDP_LOOKUP_EXT_TCAM_INDEX_32B_DATA_RESULT_ELEMENT_MATCH_WORD_OFFSET 30
- #define EZDP_LOOKUP_EXT_TCAM_INDEX_32B_DATA_RESULT_ELEMENT_MATCH_MASK (1 << EZDP_LOOKUP_EXT_TCAM_INDEX_32B_DATA_RESULT_ELEMENT_MATCH_WORD_OFFSET)
- #define EZDP_LOOKUP_EXT_TCAM_INDEX_32B_DATA_RESULT_ELEMENT_VALID_SIZE 1
- #define EZDP_LOOKUP_EXT_TCAM_INDEX_32B_DATA_RESULT_ELEMENT_VALID_OFFSET 31

- #define EZDP_LOOKUP_EXT_TCAM_INDEX_32B_DATA_RESULT_ELEMENT_VALID_WORD_SELECT 0
- #define EZDP_LOOKUP_EXT_TCAM_INDEX_32B_DATA_RESULT_ELEMENT_VALID_WORD_OFFSET 31
- #define EZDP_LOOKUP_EXT_TCAM_INDEX_32B_DATA_RESULT_ELEMENT_VALID_MASK (1 << EZDP_LOOKUP_EXT_TCAM_INDEX_32B_DATA_RESULT_ELEMENT_VALID_WORD_OFFSET)
- #define EZDP_LOOKUP_EXT_TCAM_INDEX_32B_DATA_RESULT_ELEMENT_ASSOC_DATA_SIZE 256
- #define EZDP_LOOKUP_EXT_TCAM_INDEX_32B_DATA_RESULT_ELEMENT_ASSOC_DATA_OFFSET 32
- #define EZDP_LOOKUP_EXT_TCAM_INDEX_32B_DATA_RESULT_ELEMENT_ASSOC_DATA_COUNT 32
- #define EZDP_LOOKUP_EXT_TCAM_INDEX_32B_DATA_RESULT_ELEMENT_ASSOC_DATA_WORD_SELECT 1
- #define EZDP_LOOKUP_EXT_TCAM_INDEX_32B_DATA_RESULT_ELEMENT_ASSOC_DATA_WORD_OFFSET 0
- #define EZDP_LOOKUP_EXT_TCAM_INDEX_32B_DATA_RESULT_ELEMENT_WORD_COUNT 9
- #define EZDP_LOOKUP_EXT_TCAM_4B_DATA_RESULT_ELEMENT_ASSOC_DATA_SIZE 24
- #define EZDP_LOOKUP_EXT_TCAM_4B_DATA_RESULT_ELEMENT_ASSOC_DATA_OFFSET 0
- #define EZDP_LOOKUP_EXT_TCAM_4B_DATA_RESULT_ELEMENT_ASSOC_DATA_COUNT 3
- #define EZDP_LOOKUP_EXT_TCAM_4B_DATA_RESULT_ELEMENT_RESERVED24_SIZE 1
- #define EZDP_LOOKUP_EXT_TCAM_4B_DATA_RESULT_ELEMENT_RESERVED24_OFFSET 24
- #define EZDP_LOOKUP_EXT_TCAM_4B_DATA_RESULT_ELEMENT_TRUNCATED_SIZE 1
- #define EZDP_LOOKUP_EXT_TCAM_4B_DATA_RESULT_ELEMENT_TRUNCATED_OFFSET 25
- #define EZDP_LOOKUP_EXT_TCAM_4B_DATA_RESULT_ELEMENT_TRUNCATED_MASK (1 << EZDP_LOOKUP_EXT_TCAM_4B_DATA_RESULT_ELEMENT_TRUNCATED_OFFSET)
- #define EZDP_LOOKUP_EXT_TCAM_4B_DATA_RESULT_ELEMENT_LOOKUP_ERROR_SIZE 1
- #define EZDP_LOOKUP_EXT_TCAM_4B_DATA_RESULT_ELEMENT_LOOKUP_ERROR_OFFSET 26
- #define EZDP_LOOKUP_EXT_TCAM_4B_DATA_RESULT_ELEMENT_LOOKUP_ERROR_MASK (1 << EZDP_LOOKUP_EXT_TCAM_4B_DATA_RESULT_ELEMENT_LOOKUP_ERROR_OFFSET)
- #define EZDP_LOOKUP_EXT_TCAM_4B_DATA_RESULT_ELEMENT_TYPE_SIZE 3
- #define EZDP_LOOKUP_EXT_TCAM_4B_DATA_RESULT_ELEMENT_TYPE_OFFSET 27
- #define EZDP_LOOKUP_EXT_TCAM_4B_DATA_RESULT_ELEMENT_MATCH_SIZE 1
- #define EZDP_LOOKUP_EXT_TCAM_4B_DATA_RESULT_ELEMENT_MATCH_OFFSET 30
- #define EZDP_LOOKUP_EXT_TCAM_4B_DATA_RESULT_ELEMENT_MATCH_MASK (1 << EZDP_LOOKUP_EXT_TCAM_4B_DATA_RESULT_ELEMENT_MATCH_OFFSET)
- #define EZDP_LOOKUP_EXT_TCAM_4B_DATA_RESULT_ELEMENT_VALID_SIZE 1
- #define EZDP_LOOKUP_EXT_TCAM_4B_DATA_RESULT_ELEMENT_VALID_OFFSET 31
- #define EZDP_LOOKUP_EXT_TCAM_4B_DATA_RESULT_ELEMENT_VALID_MASK (1 << EZDP_LOOKUP_EXT_TCAM_4B_DATA_RESULT_ELEMENT_VALID_OFFSET)
- #define EZDP_LOOKUP_EXT_TCAM_8B_DATA_RESULT_ELEMENT_ASSOC_DATA_SIZE 56
- #define EZDP_LOOKUP_EXT_TCAM_8B_DATA_RESULT_ELEMENT_ASSOC_DATA_OFFSET 0
- #define EZDP_LOOKUP_EXT_TCAM_8B_DATA_RESULT_ELEMENT_ASSOC_DATA_COUNT 7
- #define EZDP_LOOKUP_EXT_TCAM_8B_DATA_RESULT_ELEMENT_ASSOC_DATA_WORD_SELECT 0
- #define EZDP_LOOKUP_EXT_TCAM_8B_DATA_RESULT_ELEMENT_ASSOC_DATA_WORD_OFFSET 0
- #define EZDP_LOOKUP_EXT_TCAM_8B_DATA_RESULT_ELEMENT_RESERVED24_SIZE 1
- #define EZDP_LOOKUP_EXT_TCAM_8B_DATA_RESULT_ELEMENT_RESERVED24_OFFSET 56
- #define EZDP_LOOKUP_EXT_TCAM_8B_DATA_RESULT_ELEMENT_TRUNCATED_SIZE 1
- #define EZDP_LOOKUP_EXT_TCAM_8B_DATA_RESULT_ELEMENT_TRUNCATED_OFFSET 57
- #define EZDP_LOOKUP_EXT_TCAM_8B_DATA_RESULT_ELEMENT_TRUNCATED_WORD_SELECT 1
- #define EZDP_LOOKUP_EXT_TCAM_8B_DATA_RESULT_ELEMENT_TRUNCATED_WORD_OFFSET 25

- #define EZDP_LOOKUP_EXT_TCAM_8B_DATA_RESULT_ELEMENT_TRUNCATED_MASK (1 << EZDP_LOOKUP_EXT_TCAM_8B_DATA_RESULT_ELEMENT_TRUNCATED_WORD_OFFSET)
- #define EZDP_LOOKUP_EXT_TCAM_8B_DATA_RESULT_ELEMENT_LOOKUP_ERROR_SIZE 1
- #define EZDP_LOOKUP_EXT_TCAM_8B_DATA_RESULT_ELEMENT_LOOKUP_ERROR_OFFSET 58
- #define EZDP_LOOKUP_EXT_TCAM_8B_DATA_RESULT_ELEMENT_LOOKUP_ERROR_WORD_SELECT 1
- #define EZDP_LOOKUP_EXT_TCAM_8B_DATA_RESULT_ELEMENT_LOOKUP_ERROR_WORD_OFFSET 2 6
- #define EZDP_LOOKUP_EXT_TCAM_8B_DATA_RESULT_ELEMENT_LOOKUP_ERROR_MASK (1 << EZDP_LOOKUP_EXT_TCAM_8B_DATA_RESULT_ELEMENT_LOOKUP_ERROR_WORD_OFFSET)
- #define EZDP_LOOKUP_EXT_TCAM_8B_DATA_RESULT_ELEMENT_TYPE_SIZE 3
- #define EZDP_LOOKUP_EXT_TCAM_8B_DATA_RESULT_ELEMENT_TYPE_OFFSET 59
- #define EZDP_LOOKUP_EXT_TCAM_8B_DATA_RESULT_ELEMENT_TYPE_WORD_SELECT 1
- #define EZDP_LOOKUP_EXT_TCAM_8B_DATA_RESULT_ELEMENT_TYPE_WORD_OFFSET 27
- #define EZDP_LOOKUP_EXT_TCAM_8B_DATA_RESULT_ELEMENT_MATCH_SIZE 1
- #define EZDP_LOOKUP_EXT_TCAM_8B_DATA_RESULT_ELEMENT_MATCH_OFFSET 62
- #define EZDP_LOOKUP_EXT_TCAM_8B_DATA_RESULT_ELEMENT_MATCH_WORD_SELECT 1
- #define EZDP_LOOKUP_EXT_TCAM_8B_DATA_RESULT_ELEMENT_MATCH_WORD_OFFSET 30
- #define EZDP_LOOKUP_EXT_TCAM_8B_DATA_RESULT_ELEMENT_MATCH_MASK (1 << EZDP_LOOKUP_EXT_TCAM_8B_DATA_RESULT_ELEMENT_MATCH_WORD_OFFSET)
- #define EZDP_LOOKUP_EXT_TCAM_8B_DATA_RESULT_ELEMENT_VALID_SIZE 1
- #define EZDP_LOOKUP_EXT_TCAM_8B_DATA_RESULT_ELEMENT_VALID_OFFSET 63
- #define EZDP_LOOKUP_EXT_TCAM_8B_DATA_RESULT_ELEMENT_VALID_WORD_SELECT 1
- #define EZDP_LOOKUP_EXT_TCAM_8B_DATA_RESULT_ELEMENT_VALID_WORD_OFFSET 31
- #define EZDP_LOOKUP_EXT_TCAM_8B_DATA_RESULT_ELEMENT_VALID_MASK (1 << EZDP_LOOKUP_EXT_TCAM_8B_DATA_RESULT_ELEMENT_VALID_WORD_OFFSET)
- #define EZDP_LOOKUP_EXT_TCAM_8B_DATA_RESULT_ELEMENT_WORD_COUNT 2
- #define EZDP_LOOKUP_EXT_TCAM_16B_DATA_RESULT_ELEMENT_ASSOC_DATA_SIZE 120
- #define EZDP_LOOKUP_EXT_TCAM_16B_DATA_RESULT_ELEMENT_ASSOC_DATA_OFFSET 0
- #define EZDP_LOOKUP_EXT_TCAM_16B_DATA_RESULT_ELEMENT_ASSOC_DATA_COUNT 15
- #define EZDP_LOOKUP_EXT_TCAM_16B_DATA_RESULT_ELEMENT_ASSOC_DATA_WORD_SELECT 0
- #define EZDP_LOOKUP_EXT_TCAM_16B_DATA_RESULT_ELEMENT_ASSOC_DATA_WORD_OFFSET 0
- #define EZDP_LOOKUP_EXT_TCAM_16B_DATA_RESULT_ELEMENT_RESERVED24_SIZE 1
- #define EZDP_LOOKUP_EXT_TCAM_16B_DATA_RESULT_ELEMENT_RESERVED24_OFFSET 120
- #define EZDP_LOOKUP_EXT_TCAM_16B_DATA_RESULT_ELEMENT_TRUNCATED_SIZE 1
- #define EZDP_LOOKUP_EXT_TCAM_16B_DATA_RESULT_ELEMENT_TRUNCATED_OFFSET 121
- #define EZDP_LOOKUP_EXT_TCAM_16B_DATA_RESULT_ELEMENT_TRUNCATED_WORD_SELECT 3
- #define EZDP_LOOKUP_EXT_TCAM_16B_DATA_RESULT_ELEMENT_TRUNCATED_WORD_OFFSET 25
- #define EZDP_LOOKUP_EXT_TCAM_16B_DATA_RESULT_ELEMENT_TRUNCATED_MASK (1 << EZDP_LOOKUP_EXT_TCAM_16B_DATA_RESULT_ELEMENT_TRUNCATED_WORD_OFFSET)
- #define EZDP_LOOKUP_EXT_TCAM_16B_DATA_RESULT_ELEMENT_LOOKUP_ERROR_SIZE 1
- #define EZDP_LOOKUP_EXT_TCAM_16B_DATA_RESULT_ELEMENT_LOOKUP_ERROR_OFFSET 122
- #define EZDP_LOOKUP_EXT_TCAM_16B_DATA_RESULT_ELEMENT_LOOKUP_ERROR_WORD_SELECT 3
- #define EZDP_LOOKUP_EXT_TCAM_16B_DATA_RESULT_ELEMENT_LOOKUP_ERROR_WORD_OFFSET 26
- #define EZDP_LOOKUP_EXT_TCAM_16B_DATA_RESULT_ELEMENT_LOOKUP_ERROR_MASK (1 << EZDP_LOOKUP_EXT_TCAM_16B_DATA_RESULT_ELEMENT_LOOKUP_ERROR_WORD_OFFSET)

- #define EZDP_LOOKUP_EXT_TCAM_16B_DATA_RESULT_ELEMENT_TYPE_SIZE 3
- #define EZDP_LOOKUP_EXT_TCAM_16B_DATA_RESULT_ELEMENT_TYPE_OFFSET 123
- #define EZDP_LOOKUP_EXT_TCAM_16B_DATA_RESULT_ELEMENT_TYPE_WORD_SELECT 3
- #define EZDP_LOOKUP_EXT_TCAM_16B_DATA_RESULT_ELEMENT_TYPE_WORD_OFFSET 27
- #define EZDP_LOOKUP_EXT_TCAM_16B_DATA_RESULT_ELEMENT_MATCH_SIZE 1
- #define EZDP_LOOKUP_EXT_TCAM_16B_DATA_RESULT_ELEMENT_MATCH_OFFSET 126
- #define EZDP_LOOKUP_EXT_TCAM_16B_DATA_RESULT_ELEMENT_MATCH_WORD_SELECT 3
- #define EZDP_LOOKUP_EXT_TCAM_16B_DATA_RESULT_ELEMENT_MATCH_WORD_OFFSET 30
- #define EZDP_LOOKUP_EXT_TCAM_16B_DATA_RESULT_ELEMENT_MATCH_MASK (1 << EZDP_LOOKUP_EXT_TCAM_16B_DATA_RESULT_ELEMENT_MATCH_WORD_OFFSET)
- #define EZDP_LOOKUP_EXT_TCAM_16B_DATA_RESULT_ELEMENT_VALID_SIZE 1
- #define EZDP_LOOKUP_EXT_TCAM_16B_DATA_RESULT_ELEMENT_VALID_OFFSET 127
- #define EZDP_LOOKUP_EXT_TCAM_16B_DATA_RESULT_ELEMENT_VALID_WORD_SELECT 3
- #define EZDP_LOOKUP_EXT_TCAM_16B_DATA_RESULT_ELEMENT_VALID_WORD_OFFSET 31
- #define EZDP_LOOKUP_EXT_TCAM_16B_DATA_RESULT_ELEMENT_VALID_MASK (1 << EZDP_LOOKUP_EXT_TCAM_16B_DATA_RESULT_ELEMENT_VALID_WORD_OFFSET)
- #define EZDP_LOOKUP_EXT_TCAM_16B_DATA_RESULT_ELEMENT_WORD_COUNT 4
- #define EZDP_LOOKUP_EXT_TCAM_32B_DATA_RESULT_ELEMENT_ASSOC_DATA_SIZE 248
- #define EZDP_LOOKUP_EXT_TCAM_32B_DATA_RESULT_ELEMENT_ASSOC_DATA_OFFSET 0
- #define EZDP_LOOKUP_EXT_TCAM_32B_DATA_RESULT_ELEMENT_ASSOC_DATA_COUNT 31
- #define EZDP_LOOKUP_EXT_TCAM_32B_DATA_RESULT_ELEMENT_ASSOC_DATA_WORD_SELECT 0
- #define EZDP_LOOKUP_EXT_TCAM_32B_DATA_RESULT_ELEMENT_ASSOC_DATA_WORD_OFFSET 0
- #define EZDP_LOOKUP_EXT_TCAM_32B_DATA_RESULT_ELEMENT_RESERVED24_SIZE 1
- #define EZDP_LOOKUP_EXT_TCAM_32B_DATA_RESULT_ELEMENT_RESERVED24_OFFSET 248
- #define EZDP_LOOKUP_EXT_TCAM_32B_DATA_RESULT_ELEMENT_TRUNCATED_SIZE 1
- #define EZDP_LOOKUP_EXT_TCAM_32B_DATA_RESULT_ELEMENT_TRUNCATED_OFFSET 249
- #define EZDP_LOOKUP_EXT_TCAM_32B_DATA_RESULT_ELEMENT_TRUNCATED_WORD_SELECT 7
- #define EZDP_LOOKUP_EXT_TCAM_32B_DATA_RESULT_ELEMENT_TRUNCATED_WORD_OFFSET 25
- #define EZDP_LOOKUP_EXT_TCAM_32B_DATA_RESULT_ELEMENT_TRUNCATED_MASK (1 << EZDP_LOOKUP_EXT_TCAM_32B_DATA_RESULT_ELEMENT_TRUNCATED_WORD_OFFSET)
- #define EZDP_LOOKUP_EXT_TCAM_32B_DATA_RESULT_ELEMENT_LOOKUP_ERROR_SIZE 1
- #define EZDP_LOOKUP_EXT_TCAM_32B_DATA_RESULT_ELEMENT_LOOKUP_ERROR_OFFSET 250
- #define EZDP_LOOKUP_EXT_TCAM_32B_DATA_RESULT_ELEMENT_LOOKUP_ERROR_WORD_SELECT 7
- #define EZDP_LOOKUP_EXT_TCAM_32B_DATA_RESULT_ELEMENT_LOOKUP_ERROR_WORD_OFFSET 26
- #define EZDP_LOOKUP_EXT_TCAM_32B_DATA_RESULT_ELEMENT_LOOKUP_ERROR_MASK (1 << EZDP_LOOKUP_EXT_TCAM_32B_DATA_RESULT_ELEMENT_LOOKUP_ERROR_WORD_OFFSET)
- #define EZDP_LOOKUP_EXT_TCAM_32B_DATA_RESULT_ELEMENT_TYPE_SIZE 3
- #define EZDP_LOOKUP_EXT_TCAM_32B_DATA_RESULT_ELEMENT_TYPE_OFFSET 251
- #define EZDP_LOOKUP_EXT_TCAM_32B_DATA_RESULT_ELEMENT_TYPE_WORD_SELECT 7
- #define EZDP_LOOKUP_EXT_TCAM_32B_DATA_RESULT_ELEMENT_TYPE_WORD_OFFSET 27
- #define EZDP_LOOKUP_EXT_TCAM_32B_DATA_RESULT_ELEMENT_MATCH_SIZE 1
- #define EZDP_LOOKUP_EXT_TCAM_32B_DATA_RESULT_ELEMENT_MATCH_OFFSET 254
- #define EZDP_LOOKUP_EXT_TCAM_32B_DATA_RESULT_ELEMENT_MATCH_WORD_SELECT 7
- #define EZDP_LOOKUP_EXT_TCAM_32B_DATA_RESULT_ELEMENT_MATCH_WORD_OFFSET 30
- #define EZDP_LOOKUP_EXT_TCAM_32B_DATA_RESULT_ELEMENT_MATCH_MASK (1 << EZDP_LOOKUP_EXT_TCAM_32B_DATA_RESULT_ELEMENT_MATCH_WORD_OFFSET)
- #define EZDP_LOOKUP_EXT_TCAM_32B_DATA_RESULT_ELEMENT_VALID_SIZE 1
- #define EZDP_LOOKUP_EXT_TCAM_32B_DATA_RESULT_ELEMENT_VALID_OFFSET 255

- #define EZDP_LOOKUP_EXT_TCAM_32B_DATA_RESULT_ELEMENT_VALID_WORD_SELECT 7
- #define EZDP_LOOKUP_EXT_TCAM_32B_DATA_RESULT_ELEMENT_VALID_WORD_OFFSET 31
- #define EZDP_LOOKUP_EXT_TCAM_32B_DATA_RESULT_ELEMENT_VALID_MASK (1 << EZDP_LOOKUP_EXT_TCAM_32B_DATA_RESULT_ELEMENT_VALID_WORD_OFFSET)
- #define EZDP_LOOKUP_EXT_TCAM_32B_DATA_RESULT_ELEMENT_WORD_COUNT 8
- #define EZDP_LOOKUP_INT_TCAM_STANDARD_RESULT_MAX_NUM 4

Typedefs

- typedef uint32_t ezdp_hashed_key_t
- typedef uint32_t ezdp_lookup_retval_t
- typedef uint32_t ezdp_lookup_int_tcam_standard_result_t
- typedef uint32_t ezdp_lookup_int_tcam_4B_data_result_t
- typedef uint32_t ezdp_lookup_ext_tcam_retval_t
- typedef uint32_t ezdp_lookup_ext_tcam_index_result_element_t
- typedef uint32_t ezdp_lookup_ext_tcam_4B_data_result_element_t
- typedef struct ezdp_table_struct_desc ezdp_table_struct_desc_t
- typedef struct ezdp_hash_struct_desc ezdp_hash_struct_desc_t
- typedef struct ezdp_alg_tcam_struct_desc ezdp_alg_tcam_struct_desc_t
- typedef ezdp_sum_addr_table_desc_t ezdp_search_base_addr_t
- typedef uint32_t ezdp_lookup_int_tcam_retval_t

Enumerations

- enum ezdp_ext_tcam_result_element_type { EZDP_INDEX = 0x0, EZDP_INDEX_4B_DATA = 0x1, EZDP_INDEX_8B_DATA = 0x2, EZDP_INDEX_16B_DATA = 0x3, EZDP_INDEX_32B_DATA = 0x4, EZDP_USER_DEFINED_ASSOC_DATA1 = 0x5, EZDP_USER_DEFINED_ASSOC_DATA2 = 0x6, EZDP_USER_DEFINED_ASSOC_DATA3 = 0x7 }
  ext tcam result element type possible values.
- enum ezdp_search_hash_flags { EZDP_NONE = 0x0, EZDP_OPPORTUNISTIC = 0x1, EZDP_COMPRESS = 0x2, EZDP_UNCONDITIONAL = 0x4 }
  search hash flags.
- enum ezdp_scan_hash_slot_action { EZDP_ACCEPT_ENTRY = 0x0, EZDP_DELETE_ENTRY = 0x1, EZDP_UPDATE_ENTRY = 0x2 }
  scan hash slot callback function possible return values.

Variables

- enum ezdp_scan_hash_slot_action(* ezdp_scan_entry_cb)(void __cmem *result_ptr, struct ezdp_hash_struct_desc *hash_desc, uintptr_t user_data)

Define Documentation

#define EZDP_LOOKUP_VERSION_MAJOR 2

#define EZDP_LOOKUP_VERSION_MINOR 1

#define EZDP_ALG_TCAM_MAX_KEY_SIZE 128

#define EZDP_PAD_ALG_TCAM_WORKING_AREA(key_size)

```
Value:union \

{ \

uint8_t
acl_line[EZDP_ALG_TCAM_WORKING_AREA_ACL_LINE_COUNT]; \

struct ezdp_driver_algtcam_main_table_line main_table_line; \
```

```
}; \
\
union \
{ \
struct ezdp_lookup_int_tcam_result          tcam_result; \
struct ezdp_sum_addr                        memcmp_sd; \
uint8_t
memcmp_array[_EZDP_ALG_TCAM_WORKING_AREA_CALC_PADDING_SIZE(key_size)]; \
};
```

#define EZDP_PAD_HASH_ENTRY(result_size, key_size) uint8_t __pad[_EZDP_HASH_CALC_ENTRY_PADDING_SIZE(result_size, key_size)]

#define EZDP_PAD_HASH_WORKING_AREA(result_size, key_size) uint8_t __pad[_EZDP_LOOKUP_HASH_CALC_ENTRY_SIZE(result_size, key_size) + sizeof(ezdp_hash_op_ctx_t)]

#define EZDP_TABLE_LOW_LEVEL_WORK_AREA_SIZE _EZDP_TABLE_LOW_LEVEL_WORK_AREA_SIZE

Work area minimal required size definitions.

#define EZDP_TABLE_HIGH_LEVEL_WORK_AREA_SIZE(entry_size) _EZDP_TABLE_HIGH_LEVEL_WORK_AREA_SIZE(entry_size)

#define EZDP_HASH_LOW_LEVEL_WORK_AREA_SIZE _EZDP_HASH_LOW_LEVEL_WORK_AREA_SIZE

#define EZDP_HASH_HIGH_LEVEL_WORK_AREA_SIZE(result_size, key_size) _EZDP_HASH_HIGH_LEVEL_WORK_AREA_SIZE(result_size, key_size)

#define EZDP_ULTRA_IP_WORK_AREA_SIZE _EZDP_ULTRA_IP_WORK_AREA_SIZE

#define EZDP_ALG_TCAM_WORK_AREA_SIZE(max_key_size) _EZDP_ALG_TCAM_WORK_AREA_SIZE(max_key_size)

#define EZDP_LOOKUP_PARITY_BITS_SIZE 3

#define EZDP_LOOKUP_RESERVED_BITS_SIZE 1

#define EZDP_LOOKUP_RETVAL_DATA_SIZE 28

#define EZDP_LOOKUP_RETVAL_DATA_OFFSET 0

#define EZDP_LOOKUP_RETVAL_MATCH_SIZE 1

#define EZDP_LOOKUP_RETVAL_MATCH_OFFSET 28

#define EZDP_LOOKUP_RETVAL_MATCH_MASK (1 << EZDP_LOOKUP_RETVAL_MATCH_OFFSET)

#define EZDP_LOOKUP_RETVAL_SUCCESS_SIZE 1

#define EZDP_LOOKUP_RETVAL_SUCCESS_OFFSET 29

#define EZDP_LOOKUP_RETVAL_SUCCESS_MASK (1 << EZDP_LOOKUP_RETVAL_SUCCESS_OFFSET)

#define EZDP_LOOKUP_RETVAL_INFO_SIZE 1

#define EZDP_LOOKUP_RETVAL_INFO_OFFSET 30

#define EZDP_LOOKUP_RETVAL_INFO_MASK (1 << EZDP_LOOKUP_RETVAL_INFO_OFFSET)

#define EZDP_LOOKUP_RETVAL_MEM_ERROR_SIZE 1

#define EZDP_LOOKUP_RETVAL_MEM_ERROR_OFFSET 31

#define EZDP_LOOKUP_RETVAL_MEM_ERROR_MASK (1 << EZDP_LOOKUP_RETVAL_MEM_ERROR_OFFSET)

#define EZDP_LOOKUP_INT_TCAM_STANDARD_RESULT_INDEX_SIZE 15

#define EZDP_LOOKUP_INT_TCAM_STANDARD_RESULT_INDEX_OFFSET 0

```
#define EZDP_LOOKUP_INT_TCAM_STANDARD_RESULT_RESERVED0_15_SIZE 16

#define EZDP_LOOKUP_INT_TCAM_STANDARD_RESULT_RESERVED0_15_OFFSET 15

#define EZDP_LOOKUP_INT_TCAM_STANDARD_RESULT_MATCH_SIZE 1

#define EZDP_LOOKUP_INT_TCAM_STANDARD_RESULT_MATCH_OFFSET 31

#define EZDP_LOOKUP_INT_TCAM_STANDARD_RESULT_MATCH_MASK (1 <<
EZDP_LOOKUP_INT_TCAM_STANDARD_RESULT_MATCH_OFFSET)

#define EZDP_LOOKUP_INT_TCAM_4B_DATA_RESULT_DATA_SIZE 31

#define EZDP_LOOKUP_INT_TCAM_4B_DATA_RESULT_DATA_OFFSET 0

#define EZDP_LOOKUP_INT_TCAM_4B_DATA_RESULT_MATCH_SIZE 1

#define EZDP_LOOKUP_INT_TCAM_4B_DATA_RESULT_MATCH_OFFSET 31

#define EZDP_LOOKUP_INT_TCAM_4B_DATA_RESULT_MATCH_MASK (1 <<
EZDP_LOOKUP_INT_TCAM_4B_DATA_RESULT_MATCH_OFFSET)

#define EZDP_LOOKUP_INT_TCAM_8B_DATA_RESULT_DATA0_SIZE 31

#define EZDP_LOOKUP_INT_TCAM_8B_DATA_RESULT_DATA0_OFFSET 0

#define EZDP_LOOKUP_INT_TCAM_8B_DATA_RESULT_DATA0_WORD_SELECT 0

#define EZDP_LOOKUP_INT_TCAM_8B_DATA_RESULT_DATA0_WORD_OFFSET 0

#define EZDP_LOOKUP_INT_TCAM_8B_DATA_RESULT_MATCH_SIZE 1

#define EZDP_LOOKUP_INT_TCAM_8B_DATA_RESULT_MATCH_OFFSET 31

#define EZDP_LOOKUP_INT_TCAM_8B_DATA_RESULT_MATCH_WORD_SELECT 0

#define EZDP_LOOKUP_INT_TCAM_8B_DATA_RESULT_MATCH_WORD_OFFSET 31

#define EZDP_LOOKUP_INT_TCAM_8B_DATA_RESULT_MATCH_MASK (1 <<
EZDP_LOOKUP_INT_TCAM_8B_DATA_RESULT_MATCH_WORD_OFFSET)

#define EZDP_LOOKUP_INT_TCAM_8B_DATA_RESULT_DATA1_SIZE 32

#define EZDP_LOOKUP_INT_TCAM_8B_DATA_RESULT_DATA1_OFFSET 32

#define EZDP_LOOKUP_INT_TCAM_8B_DATA_RESULT_DATA1_WORD_SELECT 1

#define EZDP_LOOKUP_INT_TCAM_8B_DATA_RESULT_DATA1_WORD_OFFSET 0

#define EZDP_LOOKUP_INT_TCAM_8B_DATA_RESULT_WORD_COUNT 2

#define EZDP_LOOKUP_INT_TCAM_12B_DATA_RESULT_DATA0_SIZE 31
```

#define EZDP_LOOKUP_INT_TCAM_12B_DATA_RESULT_DATA0_OFFSET 0

#define EZDP_LOOKUP_INT_TCAM_12B_DATA_RESULT_DATA0_WORD_SELECT 0

#define EZDP_LOOKUP_INT_TCAM_12B_DATA_RESULT_DATA0_WORD_OFFSET 0

#define EZDP_LOOKUP_INT_TCAM_12B_DATA_RESULT_MATCH_SIZE 1

#define EZDP_LOOKUP_INT_TCAM_12B_DATA_RESULT_MATCH_OFFSET 31

#define EZDP_LOOKUP_INT_TCAM_12B_DATA_RESULT_MATCH_WORD_SELECT 0

#define EZDP_LOOKUP_INT_TCAM_12B_DATA_RESULT_MATCH_WORD_OFFSET 31

#define EZDP_LOOKUP_INT_TCAM_12B_DATA_RESULT_MATCH_MASK (1 << EZDP_LOOKUP_INT_TCAM_12B_DATA_RESULT_MATCH_WORD_OFFSET)

#define EZDP_LOOKUP_INT_TCAM_12B_DATA_RESULT_DATA1_SIZE 32

#define EZDP_LOOKUP_INT_TCAM_12B_DATA_RESULT_DATA1_OFFSET 32

#define EZDP_LOOKUP_INT_TCAM_12B_DATA_RESULT_DATA1_WORD_SELECT 1

#define EZDP_LOOKUP_INT_TCAM_12B_DATA_RESULT_DATA1_WORD_OFFSET 0

#define EZDP_LOOKUP_INT_TCAM_12B_DATA_RESULT_DATA2_SIZE 32

#define EZDP_LOOKUP_INT_TCAM_12B_DATA_RESULT_DATA2_OFFSET 64

#define EZDP_LOOKUP_INT_TCAM_12B_DATA_RESULT_DATA2_WORD_SELECT 2

#define EZDP_LOOKUP_INT_TCAM_12B_DATA_RESULT_DATA2_WORD_OFFSET 0

#define EZDP_LOOKUP_INT_TCAM_12B_DATA_RESULT_WORD_COUNT 3

#define EZDP_LOOKUP_INT_TCAM_16B_DATA_RESULT_DATA0_SIZE 31

#define EZDP_LOOKUP_INT_TCAM_16B_DATA_RESULT_DATA0_OFFSET 0

#define EZDP_LOOKUP_INT_TCAM_16B_DATA_RESULT_DATA0_WORD_SELECT 0

#define EZDP_LOOKUP_INT_TCAM_16B_DATA_RESULT_DATA0_WORD_OFFSET 0

#define EZDP_LOOKUP_INT_TCAM_16B_DATA_RESULT_MATCH_SIZE 1

#define EZDP_LOOKUP_INT_TCAM_16B_DATA_RESULT_MATCH_OFFSET 31

#define EZDP_LOOKUP_INT_TCAM_16B_DATA_RESULT_MATCH_WORD_SELECT 0

#define EZDP_LOOKUP_INT_TCAM_16B_DATA_RESULT_MATCH_WORD_OFFSET 31

#define EZDP_LOOKUP_INT_TCAM_16B_DATA_RESULT_MATCH_MASK (1 << EZDP_LOOKUP_INT_TCAM_16B_DATA_RESULT_MATCH_WORD_OFFSET)

```
#define EZDP_LOOKUP_INT_TCAM_16B_DATA_RESULT_DATA1_SIZE 32

#define EZDP_LOOKUP_INT_TCAM_16B_DATA_RESULT_DATA1_OFFSET 32

#define EZDP_LOOKUP_INT_TCAM_16B_DATA_RESULT_DATA1_WORD_SELECT 1

#define EZDP_LOOKUP_INT_TCAM_16B_DATA_RESULT_DATA1_WORD_OFFSET 0

#define EZDP_LOOKUP_INT_TCAM_16B_DATA_RESULT_DATA2_SIZE 32

#define EZDP_LOOKUP_INT_TCAM_16B_DATA_RESULT_DATA2_OFFSET 64

#define EZDP_LOOKUP_INT_TCAM_16B_DATA_RESULT_DATA2_WORD_SELECT 2

#define EZDP_LOOKUP_INT_TCAM_16B_DATA_RESULT_DATA2_WORD_OFFSET 0

#define EZDP_LOOKUP_INT_TCAM_16B_DATA_RESULT_DATA3_SIZE 32

#define EZDP_LOOKUP_INT_TCAM_16B_DATA_RESULT_DATA3_OFFSET 96

#define EZDP_LOOKUP_INT_TCAM_16B_DATA_RESULT_DATA3_WORD_SELECT 3

#define EZDP_LOOKUP_INT_TCAM_16B_DATA_RESULT_DATA3_WORD_OFFSET 0

#define EZDP_LOOKUP_INT_TCAM_16B_DATA_RESULT_WORD_COUNT 4

#define EZDP_LOOKUP_EXT_TCAM_RETVAL_NO_CONTEXT_MATCH_ERROR_SIZE 1

#define EZDP_LOOKUP_EXT_TCAM_RETVAL_NO_CONTEXT_MATCH_ERROR_OFFSET 0

#define EZDP_LOOKUP_EXT_TCAM_RETVAL_NO_CONTEXT_MATCH_ERROR_MASK (1 <<
EZDP_LOOKUP_EXT_TCAM_RETVAL_NO_CONTEXT_MATCH_ERROR_OFFSET)

#define EZDP_LOOKUP_EXT_TCAM_RETVAL_MAC_ERROR_SIZE 1

#define EZDP_LOOKUP_EXT_TCAM_RETVAL_MAC_ERROR_OFFSET 1

#define EZDP_LOOKUP_EXT_TCAM_RETVAL_MAC_ERROR_MASK (1 <<
EZDP_LOOKUP_EXT_TCAM_RETVAL_MAC_ERROR_OFFSET)

#define EZDP_LOOKUP_EXT_TCAM_RETVAL_DEVICE_ERROR_SIZE 1

#define EZDP_LOOKUP_EXT_TCAM_RETVAL_DEVICE_ERROR_OFFSET 2

#define EZDP_LOOKUP_EXT_TCAM_RETVAL_DEVICE_ERROR_MASK (1 <<
EZDP_LOOKUP_EXT_TCAM_RETVAL_DEVICE_ERROR_OFFSET)

#define EZDP_LOOKUP_EXT_TCAM_RETVAL_TIME_OUT_ERROR_SIZE 1

#define EZDP_LOOKUP_EXT_TCAM_RETVAL_TIME_OUT_ERROR_OFFSET 3

#define EZDP_LOOKUP_EXT_TCAM_RETVAL_TIME_OUT_ERROR_MASK (1 <<
EZDP_LOOKUP_EXT_TCAM_RETVAL_TIME_OUT_ERROR_OFFSET)
```

```
#define EZDP_LOOKUP_EXT_TCAM_RETVAL_ANY_MATCH_SIZE 1

#define EZDP_LOOKUP_EXT_TCAM_RETVAL_ANY_MATCH_OFFSET 4

#define EZDP_LOOKUP_EXT_TCAM_RETVAL_ANY_MATCH_MASK (1 <<
EZDP_LOOKUP_EXT_TCAM_RETVAL_ANY_MATCH_OFFSET)

#define EZDP_LOOKUP_EXT_TCAM_RETVAL_MULTI_MATCH_SIZE 1

#define EZDP_LOOKUP_EXT_TCAM_RETVAL_MULTI_MATCH_OFFSET 5

#define EZDP_LOOKUP_EXT_TCAM_RETVAL_MULTI_MATCH_MASK (1 <<
EZDP_LOOKUP_EXT_TCAM_RETVAL_MULTI_MATCH_OFFSET)

#define EZDP_LOOKUP_EXT_TCAM_RETVAL_TRUNCATED_SIZE 1

#define EZDP_LOOKUP_EXT_TCAM_RETVAL_TRUNCATED_OFFSET 6

#define EZDP_LOOKUP_EXT_TCAM_RETVAL_TRUNCATED_MASK (1 <<
EZDP_LOOKUP_EXT_TCAM_RETVAL_TRUNCATED_OFFSET)

#define EZDP_LOOKUP_EXT_TCAM_RETVAL_LOOKUP_ERROR_SIZE 1

#define EZDP_LOOKUP_EXT_TCAM_RETVAL_LOOKUP_ERROR_OFFSET 7

#define EZDP_LOOKUP_EXT_TCAM_RETVAL_LOOKUP_ERROR_MASK (1 <<
EZDP_LOOKUP_EXT_TCAM_RETVAL_LOOKUP_ERROR_OFFSET)

#define EZDP_LOOKUP_EXT_TCAM_RETVAL_RESERVED_BIT8_31_SIZE 24

#define EZDP_LOOKUP_EXT_TCAM_RETVAL_RESERVED_BIT8_31_OFFSET 8

#define EZDP_LOOKUP_EXT_TCAM_INDEX_RESULT_ELEMENT_INDEX_SIZE 21

#define EZDP_LOOKUP_EXT_TCAM_INDEX_RESULT_ELEMENT_INDEX_OFFSET 0

#define EZDP_LOOKUP_EXT_TCAM_INDEX_RESULT_ELEMENT_DEVICE_ID_SIZE 2

#define EZDP_LOOKUP_EXT_TCAM_INDEX_RESULT_ELEMENT_DEVICE_ID_OFFSET 21

#define EZDP_LOOKUP_EXT_TCAM_INDEX_RESULT_ELEMENT_RESERVED23_SIZE 1

#define EZDP_LOOKUP_EXT_TCAM_INDEX_RESULT_ELEMENT_RESERVED23_OFFSET 23

#define EZDP_LOOKUP_EXT_TCAM_INDEX_RESULT_ELEMENT_ANY_MATCH_SIZE 1

#define EZDP_LOOKUP_EXT_TCAM_INDEX_RESULT_ELEMENT_ANY_MATCH_OFFSET 24

#define EZDP_LOOKUP_EXT_TCAM_INDEX_RESULT_ELEMENT_ANY_MATCH_MASK (1 <<
EZDP_LOOKUP_EXT_TCAM_INDEX_RESULT_ELEMENT_ANY_MATCH_OFFSET)

#define EZDP_LOOKUP_EXT_TCAM_INDEX_RESULT_ELEMENT_TRUNCATED_SIZE 1
```

#define EZDP_LOOKUP_EXT_TCAM_INDEX_RESULT_ELEMENT_TRUNCATED_OFFSET 25

#define EZDP_LOOKUP_EXT_TCAM_INDEX_RESULT_ELEMENT_TRUNCATED_MASK (1 << EZDP_LOOKUP_EXT_TCAM_INDEX_RESULT_ELEMENT_TRUNCATED_OFFSET)

#define EZDP_LOOKUP_EXT_TCAM_INDEX_RESULT_ELEMENT_LOOKUP_ERROR_SIZE 1

#define EZDP_LOOKUP_EXT_TCAM_INDEX_RESULT_ELEMENT_LOOKUP_ERROR_OFFSET 26

#define EZDP_LOOKUP_EXT_TCAM_INDEX_RESULT_ELEMENT_LOOKUP_ERROR_MASK (1 << EZDP_LOOKUP_EXT_TCAM_INDEX_RESULT_ELEMENT_LOOKUP_ERROR_OFFSET)

#define EZDP_LOOKUP_EXT_TCAM_INDEX_RESULT_ELEMENT_TYPE_SIZE 3

#define EZDP_LOOKUP_EXT_TCAM_INDEX_RESULT_ELEMENT_TYPE_OFFSET 27

#define EZDP_LOOKUP_EXT_TCAM_INDEX_RESULT_ELEMENT_MATCH_SIZE 1

#define EZDP_LOOKUP_EXT_TCAM_INDEX_RESULT_ELEMENT_MATCH_OFFSET 30

#define EZDP_LOOKUP_EXT_TCAM_INDEX_RESULT_ELEMENT_MATCH_MASK (1 << EZDP_LOOKUP_EXT_TCAM_INDEX_RESULT_ELEMENT_MATCH_OFFSET)

#define EZDP_LOOKUP_EXT_TCAM_INDEX_RESULT_ELEMENT_VALID_SIZE 1

#define EZDP_LOOKUP_EXT_TCAM_INDEX_RESULT_ELEMENT_VALID_OFFSET 31

#define EZDP_LOOKUP_EXT_TCAM_INDEX_RESULT_ELEMENT_VALID_MASK (1 << EZDP_LOOKUP_EXT_TCAM_INDEX_RESULT_ELEMENT_VALID_OFFSET)

#define EZDP_LOOKUP_EXT_TCAM_INDEX_4B_DATA_RESULT_ELEMENT_INDEX_SIZE 21

#define EZDP_LOOKUP_EXT_TCAM_INDEX_4B_DATA_RESULT_ELEMENT_INDEX_OFFSET 0

#define EZDP_LOOKUP_EXT_TCAM_INDEX_4B_DATA_RESULT_ELEMENT_INDEX_WORD_SELECT 0

#define EZDP_LOOKUP_EXT_TCAM_INDEX_4B_DATA_RESULT_ELEMENT_INDEX_WORD_OFFSET 0

#define EZDP_LOOKUP_EXT_TCAM_INDEX_4B_DATA_RESULT_ELEMENT_DEVICE_ID_SIZE 2

#define EZDP_LOOKUP_EXT_TCAM_INDEX_4B_DATA_RESULT_ELEMENT_DEVICE_ID_OFFSET 21

#define EZDP_LOOKUP_EXT_TCAM_INDEX_4B_DATA_RESULT_ELEMENT_DEVICE_ID_WORD_SELECT 0

#define EZDP_LOOKUP_EXT_TCAM_INDEX_4B_DATA_RESULT_ELEMENT_DEVICE_ID_WORD_OFFSET 21

#define EZDP_LOOKUP_EXT_TCAM_INDEX_4B_DATA_RESULT_ELEMENT_RESERVED23_24_SIZE 2

#define EZDP_LOOKUP_EXT_TCAM_INDEX_4B_DATA_RESULT_ELEMENT_RESERVED23_24_OFFSET 23

#define EZDP_LOOKUP_EXT_TCAM_INDEX_4B_DATA_RESULT_ELEMENT_TRUNCATED_SIZE 1

#define EZDP_LOOKUP_EXT_TCAM_INDEX_4B_DATA_RESULT_ELEMENT_TRUNCATED_OFFSET 25

#define EZDP_LOOKUP_EXT_TCAM_INDEX_4B_DATA_RESULT_ELEMENT_TRUNCATED_WORD_SELECT 0

#define EZDP_LOOKUP_EXT_TCAM_INDEX_4B_DATA_RESULT_ELEMENT_TRUNCATED_WORD_OFFSET 25

#define EZDP_LOOKUP_EXT_TCAM_INDEX_4B_DATA_RESULT_ELEMENT_TRUNCATED_MASK (1 << EZDP_LOOKUP_EXT_TCAM_INDEX_4B_DATA_RESULT_ELEMENT_TRUNCATED_WORD_OFFSET)

#define EZDP_LOOKUP_EXT_TCAM_INDEX_4B_DATA_RESULT_ELEMENT_LOOKUP_ERROR_SIZE 1

#define EZDP_LOOKUP_EXT_TCAM_INDEX_4B_DATA_RESULT_ELEMENT_LOOKUP_ERROR_OFFSET 26

#define EZDP_LOOKUP_EXT_TCAM_INDEX_4B_DATA_RESULT_ELEMENT_LOOKUP_ERROR_WORD_SELECT 0

#define EZDP_LOOKUP_EXT_TCAM_INDEX_4B_DATA_RESULT_ELEMENT_LOOKUP_ERROR_WORD_OFFSET 26

#define EZDP_LOOKUP_EXT_TCAM_INDEX_4B_DATA_RESULT_ELEMENT_LOOKUP_ERROR_MASK (1 << EZDP_LOOKUP_EXT_TCAM_INDEX_4B_DATA_RESULT_ELEMENT_LOOKUP_ERROR_WORD_OFFSET)

#define EZDP_LOOKUP_EXT_TCAM_INDEX_4B_DATA_RESULT_ELEMENT_TYPE_SIZE 3

#define EZDP_LOOKUP_EXT_TCAM_INDEX_4B_DATA_RESULT_ELEMENT_TYPE_OFFSET 27

#define EZDP_LOOKUP_EXT_TCAM_INDEX_4B_DATA_RESULT_ELEMENT_TYPE_WORD_SELECT 0

#define EZDP_LOOKUP_EXT_TCAM_INDEX_4B_DATA_RESULT_ELEMENT_TYPE_WORD_OFFSET 27

```
#define EZDP_LOOKUP_EXT_TCAM_INDEX_4B_DATA_RESULT_ELEMENT_MATCH_SIZE 1

#define EZDP_LOOKUP_EXT_TCAM_INDEX_4B_DATA_RESULT_ELEMENT_MATCH_OFFSET 30

#define
EZDP_LOOKUP_EXT_TCAM_INDEX_4B_DATA_RESULT_ELEMENT_MATCH_WORD_SELECT 0

#define
EZDP_LOOKUP_EXT_TCAM_INDEX_4B_DATA_RESULT_ELEMENT_MATCH_WORD_OFFSET 30

#define EZDP_LOOKUP_EXT_TCAM_INDEX_4B_DATA_RESULT_ELEMENT_MATCH_MASK (1 <<
EZDP_LOOKUP_EXT_TCAM_INDEX_4B_DATA_RESULT_ELEMENT_MATCH_WORD_OFFSET)

#define EZDP_LOOKUP_EXT_TCAM_INDEX_4B_DATA_RESULT_ELEMENT_VALID_SIZE 1

#define EZDP_LOOKUP_EXT_TCAM_INDEX_4B_DATA_RESULT_ELEMENT_VALID_OFFSET 31

#define
EZDP_LOOKUP_EXT_TCAM_INDEX_4B_DATA_RESULT_ELEMENT_VALID_WORD_SELECT 0

#define
EZDP_LOOKUP_EXT_TCAM_INDEX_4B_DATA_RESULT_ELEMENT_VALID_WORD_OFFSET 31

#define EZDP_LOOKUP_EXT_TCAM_INDEX_4B_DATA_RESULT_ELEMENT_VALID_MASK (1 <<
EZDP_LOOKUP_EXT_TCAM_INDEX_4B_DATA_RESULT_ELEMENT_VALID_WORD_OFFSET)

#define
EZDP_LOOKUP_EXT_TCAM_INDEX_4B_DATA_RESULT_ELEMENT_ASSOC_DATA_SIZE 32

#define
EZDP_LOOKUP_EXT_TCAM_INDEX_4B_DATA_RESULT_ELEMENT_ASSOC_DATA_OFFSET 32

#define
EZDP_LOOKUP_EXT_TCAM_INDEX_4B_DATA_RESULT_ELEMENT_ASSOC_DATA_COUNT 4

#define
EZDP_LOOKUP_EXT_TCAM_INDEX_4B_DATA_RESULT_ELEMENT_ASSOC_DATA_WORD_SELE
CT 1

#define
EZDP_LOOKUP_EXT_TCAM_INDEX_4B_DATA_RESULT_ELEMENT_ASSOC_DATA_WORD_OFFS
ET 0

#define EZDP_LOOKUP_EXT_TCAM_INDEX_4B_DATA_RESULT_ELEMENT_WORD_COUNT 2

#define EZDP_LOOKUP_EXT_TCAM_INDEX_8B_DATA_RESULT_ELEMENT_INDEX_SIZE 21

#define EZDP_LOOKUP_EXT_TCAM_INDEX_8B_DATA_RESULT_ELEMENT_INDEX_OFFSET 0

#define
EZDP_LOOKUP_EXT_TCAM_INDEX_8B_DATA_RESULT_ELEMENT_INDEX_WORD_SELECT 0

#define
EZDP_LOOKUP_EXT_TCAM_INDEX_8B_DATA_RESULT_ELEMENT_INDEX_WORD_OFFSET 0
```

#define EZDP_LOOKUP_EXT_TCAM_INDEX_8B_DATA_RESULT_ELEMENT_DEVICE_ID_SIZE 2

#define EZDP_LOOKUP_EXT_TCAM_INDEX_8B_DATA_RESULT_ELEMENT_DEVICE_ID_OFFSET 21

#define EZDP_LOOKUP_EXT_TCAM_INDEX_8B_DATA_RESULT_ELEMENT_DEVICE_ID_WORD_SELECT 0

#define EZDP_LOOKUP_EXT_TCAM_INDEX_8B_DATA_RESULT_ELEMENT_DEVICE_ID_WORD_OFFSET 21

#define EZDP_LOOKUP_EXT_TCAM_INDEX_8B_DATA_RESULT_ELEMENT_RESERVED23_24_SIZE 2

#define EZDP_LOOKUP_EXT_TCAM_INDEX_8B_DATA_RESULT_ELEMENT_RESERVED23_24_OFFSET 23

#define EZDP_LOOKUP_EXT_TCAM_INDEX_8B_DATA_RESULT_ELEMENT_TRUNCATED_SIZE 1

#define EZDP_LOOKUP_EXT_TCAM_INDEX_8B_DATA_RESULT_ELEMENT_TRUNCATED_OFFSET 25

#define EZDP_LOOKUP_EXT_TCAM_INDEX_8B_DATA_RESULT_ELEMENT_TRUNCATED_WORD_SELECT 0

#define EZDP_LOOKUP_EXT_TCAM_INDEX_8B_DATA_RESULT_ELEMENT_TRUNCATED_WORD_OFFSET 25

#define EZDP_LOOKUP_EXT_TCAM_INDEX_8B_DATA_RESULT_ELEMENT_TRUNCATED_MASK (1 << EZDP_LOOKUP_EXT_TCAM_INDEX_8B_DATA_RESULT_ELEMENT_TRUNCATED_WORD_OFFSET)

#define EZDP_LOOKUP_EXT_TCAM_INDEX_8B_DATA_RESULT_ELEMENT_LOOKUP_ERROR_SIZE 1

#define EZDP_LOOKUP_EXT_TCAM_INDEX_8B_DATA_RESULT_ELEMENT_LOOKUP_ERROR_OFFSET 26

#define EZDP_LOOKUP_EXT_TCAM_INDEX_8B_DATA_RESULT_ELEMENT_LOOKUP_ERROR_WORD_SELECT 0

#define EZDP_LOOKUP_EXT_TCAM_INDEX_8B_DATA_RESULT_ELEMENT_LOOKUP_ERROR_WORD_OFFSET 26

#define EZDP_LOOKUP_EXT_TCAM_INDEX_8B_DATA_RESULT_ELEMENT_LOOKUP_ERROR_MASK (1 <<

EZDP_LOOKUP_EXT_TCAM_INDEX_8B_DATA_RESULT_ELEMENT_LOOKUP_ERROR_WORD_OFFSET)

#define EZDP_LOOKUP_EXT_TCAM_INDEX_8B_DATA_RESULT_ELEMENT_TYPE_SIZE 3

#define EZDP_LOOKUP_EXT_TCAM_INDEX_8B_DATA_RESULT_ELEMENT_TYPE_OFFSET 27

#define EZDP_LOOKUP_EXT_TCAM_INDEX_8B_DATA_RESULT_ELEMENT_TYPE_WORD_SELECT 0

#define EZDP_LOOKUP_EXT_TCAM_INDEX_8B_DATA_RESULT_ELEMENT_TYPE_WORD_OFFSET 27

#define EZDP_LOOKUP_EXT_TCAM_INDEX_8B_DATA_RESULT_ELEMENT_MATCH_SIZE 1

#define EZDP_LOOKUP_EXT_TCAM_INDEX_8B_DATA_RESULT_ELEMENT_MATCH_OFFSET 30

#define EZDP_LOOKUP_EXT_TCAM_INDEX_8B_DATA_RESULT_ELEMENT_MATCH_WORD_SELECT 0

#define EZDP_LOOKUP_EXT_TCAM_INDEX_8B_DATA_RESULT_ELEMENT_MATCH_WORD_OFFSET 30

#define EZDP_LOOKUP_EXT_TCAM_INDEX_8B_DATA_RESULT_ELEMENT_MATCH_MASK (1 << EZDP_LOOKUP_EXT_TCAM_INDEX_8B_DATA_RESULT_ELEMENT_MATCH_WORD_OFFSET)

#define EZDP_LOOKUP_EXT_TCAM_INDEX_8B_DATA_RESULT_ELEMENT_VALID_SIZE 1

#define EZDP_LOOKUP_EXT_TCAM_INDEX_8B_DATA_RESULT_ELEMENT_VALID_OFFSET 31

#define EZDP_LOOKUP_EXT_TCAM_INDEX_8B_DATA_RESULT_ELEMENT_VALID_WORD_SELECT 0

#define EZDP_LOOKUP_EXT_TCAM_INDEX_8B_DATA_RESULT_ELEMENT_VALID_WORD_OFFSET 31

#define EZDP_LOOKUP_EXT_TCAM_INDEX_8B_DATA_RESULT_ELEMENT_VALID_MASK (1 << EZDP_LOOKUP_EXT_TCAM_INDEX_8B_DATA_RESULT_ELEMENT_VALID_WORD_OFFSET)

#define EZDP_LOOKUP_EXT_TCAM_INDEX_8B_DATA_RESULT_ELEMENT_ASSOC_DATA_SIZE 64

#define EZDP_LOOKUP_EXT_TCAM_INDEX_8B_DATA_RESULT_ELEMENT_ASSOC_DATA_OFFSET 32

#define EZDP_LOOKUP_EXT_TCAM_INDEX_8B_DATA_RESULT_ELEMENT_ASSOC_DATA_COUNT 8

#define EZDP_LOOKUP_EXT_TCAM_INDEX_8B_DATA_RESULT_ELEMENT_ASSOC_DATA_WORD_SELECT 1

#define EZDP_LOOKUP_EXT_TCAM_INDEX_8B_DATA_RESULT_ELEMENT_ASSOC_DATA_WORD_OFFSET 0

#define EZDP_LOOKUP_EXT_TCAM_INDEX_8B_DATA_RESULT_ELEMENT_WORD_COUNT 3

#define EZDP_LOOKUP_EXT_TCAM_INDEX_16B_DATA_RESULT_ELEMENT_INDEX_SIZE 21

#define EZDP_LOOKUP_EXT_TCAM_INDEX_16B_DATA_RESULT_ELEMENT_INDEX_OFFSET 0

#define EZDP_LOOKUP_EXT_TCAM_INDEX_16B_DATA_RESULT_ELEMENT_INDEX_WORD_SELECT 0

#define EZDP_LOOKUP_EXT_TCAM_INDEX_16B_DATA_RESULT_ELEMENT_INDEX_WORD_OFFSET 0

#define EZDP_LOOKUP_EXT_TCAM_INDEX_16B_DATA_RESULT_ELEMENT_DEVICE_ID_SIZE 2

#define EZDP_LOOKUP_EXT_TCAM_INDEX_16B_DATA_RESULT_ELEMENT_DEVICE_ID_OFFSET 21

#define EZDP_LOOKUP_EXT_TCAM_INDEX_16B_DATA_RESULT_ELEMENT_DEVICE_ID_WORD_SELECT 0

#define EZDP_LOOKUP_EXT_TCAM_INDEX_16B_DATA_RESULT_ELEMENT_DEVICE_ID_WORD_OFFSET 21

#define EZDP_LOOKUP_EXT_TCAM_INDEX_16B_DATA_RESULT_ELEMENT_RESERVED23_24_SIZE 2

#define EZDP_LOOKUP_EXT_TCAM_INDEX_16B_DATA_RESULT_ELEMENT_RESERVED23_24_OFFSET 23

#define EZDP_LOOKUP_EXT_TCAM_INDEX_16B_DATA_RESULT_ELEMENT_TRUNCATED_SIZE 1

#define EZDP_LOOKUP_EXT_TCAM_INDEX_16B_DATA_RESULT_ELEMENT_TRUNCATED_OFFSET 25

#define EZDP_LOOKUP_EXT_TCAM_INDEX_16B_DATA_RESULT_ELEMENT_TRUNCATED_WORD_SELECT 0

#define EZDP_LOOKUP_EXT_TCAM_INDEX_16B_DATA_RESULT_ELEMENT_TRUNCATED_WORD_OFFSET 25

#define EZDP_LOOKUP_EXT_TCAM_INDEX_16B_DATA_RESULT_ELEMENT_TRUNCATED_MASK (1 << EZDP_LOOKUP_EXT_TCAM_INDEX_16B_DATA_RESULT_ELEMENT_TRUNCATED_WORD_OFFSET)

#define EZDP_LOOKUP_EXT_TCAM_INDEX_16B_DATA_RESULT_ELEMENT_LOOKUP_ERROR_SIZE 1

```
#define
EZDP_LOOKUP_EXT_TCAM_INDEX_16B_DATA_RESULT_ELEMENT_LOOKUP_ERROR_OFFSET
 26

#define
EZDP_LOOKUP_EXT_TCAM_INDEX_16B_DATA_RESULT_ELEMENT_LOOKUP_ERROR_WORD_S
ELECT 0

#define
EZDP_LOOKUP_EXT_TCAM_INDEX_16B_DATA_RESULT_ELEMENT_LOOKUP_ERROR_WORD_O
FFSET 26

#define
EZDP_LOOKUP_EXT_TCAM_INDEX_16B_DATA_RESULT_ELEMENT_LOOKUP_ERROR_MASK (
1 <<
EZDP_LOOKUP_EXT_TCAM_INDEX_16B_DATA_RESULT_ELEMENT_LOOKUP_ERROR_WORD_O
FFSET)

#define EZDP_LOOKUP_EXT_TCAM_INDEX_16B_DATA_RESULT_ELEMENT_TYPE_SIZE 3

#define EZDP_LOOKUP_EXT_TCAM_INDEX_16B_DATA_RESULT_ELEMENT_TYPE_OFFSET 27

#define
EZDP_LOOKUP_EXT_TCAM_INDEX_16B_DATA_RESULT_ELEMENT_TYPE_WORD_SELECT 0

#define
EZDP_LOOKUP_EXT_TCAM_INDEX_16B_DATA_RESULT_ELEMENT_TYPE_WORD_OFFSET 27

#define EZDP_LOOKUP_EXT_TCAM_INDEX_16B_DATA_RESULT_ELEMENT_MATCH_SIZE 1

#define
EZDP_LOOKUP_EXT_TCAM_INDEX_16B_DATA_RESULT_ELEMENT_MATCH_OFFSET 30

#define
EZDP_LOOKUP_EXT_TCAM_INDEX_16B_DATA_RESULT_ELEMENT_MATCH_WORD_SELECT 0

#define
EZDP_LOOKUP_EXT_TCAM_INDEX_16B_DATA_RESULT_ELEMENT_MATCH_WORD_OFFSET 3
0

#define EZDP_LOOKUP_EXT_TCAM_INDEX_16B_DATA_RESULT_ELEMENT_MATCH_MASK (1
<<
EZDP_LOOKUP_EXT_TCAM_INDEX_16B_DATA_RESULT_ELEMENT_MATCH_WORD_OFFSET)

#define EZDP_LOOKUP_EXT_TCAM_INDEX_16B_DATA_RESULT_ELEMENT_VALID_SIZE 1

#define EZDP_LOOKUP_EXT_TCAM_INDEX_16B_DATA_RESULT_ELEMENT_VALID_OFFSET 31

#define
EZDP_LOOKUP_EXT_TCAM_INDEX_16B_DATA_RESULT_ELEMENT_VALID_WORD_SELECT 0

#define
EZDP_LOOKUP_EXT_TCAM_INDEX_16B_DATA_RESULT_ELEMENT_VALID_WORD_OFFSET 31
```

#define EZDP_LOOKUP_EXT_TCAM_INDEX_16B_DATA_RESULT_ELEMENT_VALID_MASK (1 << EZDP_LOOKUP_EXT_TCAM_INDEX_16B_DATA_RESULT_ELEMENT_VALID_WORD_OFFSET)

#define EZDP_LOOKUP_EXT_TCAM_INDEX_16B_DATA_RESULT_ELEMENT_ASSOC_DATA_SIZE 128

#define EZDP_LOOKUP_EXT_TCAM_INDEX_16B_DATA_RESULT_ELEMENT_ASSOC_DATA_OFFSET 32

#define EZDP_LOOKUP_EXT_TCAM_INDEX_16B_DATA_RESULT_ELEMENT_ASSOC_DATA_COUNT 16

#define EZDP_LOOKUP_EXT_TCAM_INDEX_16B_DATA_RESULT_ELEMENT_ASSOC_DATA_WORD_SELECT 1

#define EZDP_LOOKUP_EXT_TCAM_INDEX_16B_DATA_RESULT_ELEMENT_ASSOC_DATA_WORD_OFFSET 0

#define EZDP_LOOKUP_EXT_TCAM_INDEX_16B_DATA_RESULT_ELEMENT_WORD_COUNT 5

#define EZDP_LOOKUP_EXT_TCAM_INDEX_32B_DATA_RESULT_ELEMENT_INDEX_SIZE 21

#define EZDP_LOOKUP_EXT_TCAM_INDEX_32B_DATA_RESULT_ELEMENT_INDEX_OFFSET 0

#define EZDP_LOOKUP_EXT_TCAM_INDEX_32B_DATA_RESULT_ELEMENT_INDEX_WORD_SELECT 0

#define EZDP_LOOKUP_EXT_TCAM_INDEX_32B_DATA_RESULT_ELEMENT_INDEX_WORD_OFFSET 0

#define EZDP_LOOKUP_EXT_TCAM_INDEX_32B_DATA_RESULT_ELEMENT_DEVICE_ID_SIZE 2

#define EZDP_LOOKUP_EXT_TCAM_INDEX_32B_DATA_RESULT_ELEMENT_DEVICE_ID_OFFSET 21

#define EZDP_LOOKUP_EXT_TCAM_INDEX_32B_DATA_RESULT_ELEMENT_DEVICE_ID_WORD_SELECT 0

#define EZDP_LOOKUP_EXT_TCAM_INDEX_32B_DATA_RESULT_ELEMENT_DEVICE_ID_WORD_OFFSET 21

#define EZDP_LOOKUP_EXT_TCAM_INDEX_32B_DATA_RESULT_ELEMENT_RESERVED23_24_SIZE 2

#define EZDP_LOOKUP_EXT_TCAM_INDEX_32B_DATA_RESULT_ELEMENT_RESERVED23_24_OFFSET 23

#define EZDP_LOOKUP_EXT_TCAM_INDEX_32B_DATA_RESULT_ELEMENT_TRUNCATED_SIZE 1

```
#define
EZDP_LOOKUP_EXT_TCAM_INDEX_32B_DATA_RESULT_ELEMENT_TRUNCATED_OFFSET 25

#define
EZDP_LOOKUP_EXT_TCAM_INDEX_32B_DATA_RESULT_ELEMENT_TRUNCATED_WORD_SELE
CT 0

#define
EZDP_LOOKUP_EXT_TCAM_INDEX_32B_DATA_RESULT_ELEMENT_TRUNCATED_WORD_OFFS
ET 25

#define
EZDP_LOOKUP_EXT_TCAM_INDEX_32B_DATA_RESULT_ELEMENT_TRUNCATED_MASK (1 <<
EZDP_LOOKUP_EXT_TCAM_INDEX_32B_DATA_RESULT_ELEMENT_TRUNCATED_WORD_OFFS
ET)

#define
EZDP_LOOKUP_EXT_TCAM_INDEX_32B_DATA_RESULT_ELEMENT_LOOKUP_ERROR_SIZE 1

#define
EZDP_LOOKUP_EXT_TCAM_INDEX_32B_DATA_RESULT_ELEMENT_LOOKUP_ERROR_OFFSET
 26

#define
EZDP_LOOKUP_EXT_TCAM_INDEX_32B_DATA_RESULT_ELEMENT_LOOKUP_ERROR_WORD_S
ELECT 0

#define
EZDP_LOOKUP_EXT_TCAM_INDEX_32B_DATA_RESULT_ELEMENT_LOOKUP_ERROR_WORD_O
FFSET 26

#define
EZDP_LOOKUP_EXT_TCAM_INDEX_32B_DATA_RESULT_ELEMENT_LOOKUP_ERROR_MASK (
1 <<
EZDP_LOOKUP_EXT_TCAM_INDEX_32B_DATA_RESULT_ELEMENT_LOOKUP_ERROR_WORD_O
FFSET)

#define EZDP_LOOKUP_EXT_TCAM_INDEX_32B_DATA_RESULT_ELEMENT_TYPE_SIZE 3

#define EZDP_LOOKUP_EXT_TCAM_INDEX_32B_DATA_RESULT_ELEMENT_TYPE_OFFSET 27

#define
EZDP_LOOKUP_EXT_TCAM_INDEX_32B_DATA_RESULT_ELEMENT_TYPE_WORD_SELECT 0

#define
EZDP_LOOKUP_EXT_TCAM_INDEX_32B_DATA_RESULT_ELEMENT_TYPE_WORD_OFFSET 27

#define EZDP_LOOKUP_EXT_TCAM_INDEX_32B_DATA_RESULT_ELEMENT_MATCH_SIZE 1

#define
EZDP_LOOKUP_EXT_TCAM_INDEX_32B_DATA_RESULT_ELEMENT_MATCH_OFFSET 30

#define
EZDP_LOOKUP_EXT_TCAM_INDEX_32B_DATA_RESULT_ELEMENT_MATCH_WORD_SELECT 0
```

```
#define
EZDP_LOOKUP_EXT_TCAM_INDEX_32B_DATA_RESULT_ELEMENT_MATCH_WORD_OFFSET 3
0

#define EZDP_LOOKUP_EXT_TCAM_INDEX_32B_DATA_RESULT_ELEMENT_MATCH_MASK (1
<<
EZDP_LOOKUP_EXT_TCAM_INDEX_32B_DATA_RESULT_ELEMENT_MATCH_WORD_OFFSET)

#define EZDP_LOOKUP_EXT_TCAM_INDEX_32B_DATA_RESULT_ELEMENT_VALID_SIZE 1

#define EZDP_LOOKUP_EXT_TCAM_INDEX_32B_DATA_RESULT_ELEMENT_VALID_OFFSET 31

#define
EZDP_LOOKUP_EXT_TCAM_INDEX_32B_DATA_RESULT_ELEMENT_VALID_WORD_SELECT 0

#define
EZDP_LOOKUP_EXT_TCAM_INDEX_32B_DATA_RESULT_ELEMENT_VALID_WORD_OFFSET 31

#define EZDP_LOOKUP_EXT_TCAM_INDEX_32B_DATA_RESULT_ELEMENT_VALID_MASK (1 <<
EZDP_LOOKUP_EXT_TCAM_INDEX_32B_DATA_RESULT_ELEMENT_VALID_WORD_OFFSET)

#define
EZDP_LOOKUP_EXT_TCAM_INDEX_32B_DATA_RESULT_ELEMENT_ASSOC_DATA_SIZE 256

#define
EZDP_LOOKUP_EXT_TCAM_INDEX_32B_DATA_RESULT_ELEMENT_ASSOC_DATA_OFFSET 32

#define
EZDP_LOOKUP_EXT_TCAM_INDEX_32B_DATA_RESULT_ELEMENT_ASSOC_DATA_COUNT 32

#define
EZDP_LOOKUP_EXT_TCAM_INDEX_32B_DATA_RESULT_ELEMENT_ASSOC_DATA_WORD_SEL
ECT 1

#define
EZDP_LOOKUP_EXT_TCAM_INDEX_32B_DATA_RESULT_ELEMENT_ASSOC_DATA_WORD_OFF
SET 0

#define EZDP_LOOKUP_EXT_TCAM_INDEX_32B_DATA_RESULT_ELEMENT_WORD_COUNT 9

#define EZDP_LOOKUP_EXT_TCAM_4B_DATA_RESULT_ELEMENT_ASSOC_DATA_SIZE 24

#define EZDP_LOOKUP_EXT_TCAM_4B_DATA_RESULT_ELEMENT_ASSOC_DATA_OFFSET 0

#define EZDP_LOOKUP_EXT_TCAM_4B_DATA_RESULT_ELEMENT_ASSOC_DATA_COUNT 3

#define EZDP_LOOKUP_EXT_TCAM_4B_DATA_RESULT_ELEMENT_RESERVED24_SIZE 1

#define EZDP_LOOKUP_EXT_TCAM_4B_DATA_RESULT_ELEMENT_RESERVED24_OFFSET 24

#define EZDP_LOOKUP_EXT_TCAM_4B_DATA_RESULT_ELEMENT_TRUNCATED_SIZE 1

#define EZDP_LOOKUP_EXT_TCAM_4B_DATA_RESULT_ELEMENT_TRUNCATED_OFFSET 25
```

```
#define EZDP_LOOKUP_EXT_TCAM_4B_DATA_RESULT_ELEMENT_TRUNCATED_MASK (1 << EZDP_LOOKUP_EXT_TCAM_4B_DATA_RESULT_ELEMENT_TRUNCATED_OFFSET)

#define EZDP_LOOKUP_EXT_TCAM_4B_DATA_RESULT_ELEMENT_LOOKUP_ERROR_SIZE 1

#define EZDP_LOOKUP_EXT_TCAM_4B_DATA_RESULT_ELEMENT_LOOKUP_ERROR_OFFSET 26

#define EZDP_LOOKUP_EXT_TCAM_4B_DATA_RESULT_ELEMENT_LOOKUP_ERROR_MASK (1 << EZDP_LOOKUP_EXT_TCAM_4B_DATA_RESULT_ELEMENT_LOOKUP_ERROR_OFFSET)

#define EZDP_LOOKUP_EXT_TCAM_4B_DATA_RESULT_ELEMENT_TYPE_SIZE 3

#define EZDP_LOOKUP_EXT_TCAM_4B_DATA_RESULT_ELEMENT_TYPE_OFFSET 27

#define EZDP_LOOKUP_EXT_TCAM_4B_DATA_RESULT_ELEMENT_MATCH_SIZE 1

#define EZDP_LOOKUP_EXT_TCAM_4B_DATA_RESULT_ELEMENT_MATCH_OFFSET 30

#define EZDP_LOOKUP_EXT_TCAM_4B_DATA_RESULT_ELEMENT_MATCH_MASK (1 << EZDP_LOOKUP_EXT_TCAM_4B_DATA_RESULT_ELEMENT_MATCH_OFFSET)

#define EZDP_LOOKUP_EXT_TCAM_4B_DATA_RESULT_ELEMENT_VALID_SIZE 1

#define EZDP_LOOKUP_EXT_TCAM_4B_DATA_RESULT_ELEMENT_VALID_OFFSET 31

#define EZDP_LOOKUP_EXT_TCAM_4B_DATA_RESULT_ELEMENT_VALID_MASK (1 << EZDP_LOOKUP_EXT_TCAM_4B_DATA_RESULT_ELEMENT_VALID_OFFSET)

#define EZDP_LOOKUP_EXT_TCAM_8B_DATA_RESULT_ELEMENT_ASSOC_DATA_SIZE 56

#define EZDP_LOOKUP_EXT_TCAM_8B_DATA_RESULT_ELEMENT_ASSOC_DATA_OFFSET 0

#define EZDP_LOOKUP_EXT_TCAM_8B_DATA_RESULT_ELEMENT_ASSOC_DATA_COUNT 7

#define EZDP_LOOKUP_EXT_TCAM_8B_DATA_RESULT_ELEMENT_ASSOC_DATA_WORD_SELECT 0

#define EZDP_LOOKUP_EXT_TCAM_8B_DATA_RESULT_ELEMENT_ASSOC_DATA_WORD_OFFSET 0

#define EZDP_LOOKUP_EXT_TCAM_8B_DATA_RESULT_ELEMENT_RESERVED24_SIZE 1

#define EZDP_LOOKUP_EXT_TCAM_8B_DATA_RESULT_ELEMENT_RESERVED24_OFFSET 56

#define EZDP_LOOKUP_EXT_TCAM_8B_DATA_RESULT_ELEMENT_TRUNCATED_SIZE 1

#define EZDP_LOOKUP_EXT_TCAM_8B_DATA_RESULT_ELEMENT_TRUNCATED_OFFSET 57

#define EZDP_LOOKUP_EXT_TCAM_8B_DATA_RESULT_ELEMENT_TRUNCATED_WORD_SELECT 1

#define EZDP_LOOKUP_EXT_TCAM_8B_DATA_RESULT_ELEMENT_TRUNCATED_WORD_OFFSET 25
```

```
#define EZDP_LOOKUP_EXT_TCAM_8B_DATA_RESULT_ELEMENT_TRUNCATED_MASK (1 <<
EZDP_LOOKUP_EXT_TCAM_8B_DATA_RESULT_ELEMENT_TRUNCATED_WORD_OFFSET)

#define EZDP_LOOKUP_EXT_TCAM_8B_DATA_RESULT_ELEMENT_LOOKUP_ERROR_SIZE 1

#define
EZDP_LOOKUP_EXT_TCAM_8B_DATA_RESULT_ELEMENT_LOOKUP_ERROR_OFFSET 58

#define
EZDP_LOOKUP_EXT_TCAM_8B_DATA_RESULT_ELEMENT_LOOKUP_ERROR_WORD_SELECT
1

#define
EZDP_LOOKUP_EXT_TCAM_8B_DATA_RESULT_ELEMENT_LOOKUP_ERROR_WORD_OFFSET
26

#define EZDP_LOOKUP_EXT_TCAM_8B_DATA_RESULT_ELEMENT_LOOKUP_ERROR_MASK (1
<<
EZDP_LOOKUP_EXT_TCAM_8B_DATA_RESULT_ELEMENT_LOOKUP_ERROR_WORD_OFFSET)

#define EZDP_LOOKUP_EXT_TCAM_8B_DATA_RESULT_ELEMENT_TYPE_SIZE 3

#define EZDP_LOOKUP_EXT_TCAM_8B_DATA_RESULT_ELEMENT_TYPE_OFFSET 59

#define EZDP_LOOKUP_EXT_TCAM_8B_DATA_RESULT_ELEMENT_TYPE_WORD_SELECT 1

#define EZDP_LOOKUP_EXT_TCAM_8B_DATA_RESULT_ELEMENT_TYPE_WORD_OFFSET 27

#define EZDP_LOOKUP_EXT_TCAM_8B_DATA_RESULT_ELEMENT_MATCH_SIZE 1

#define EZDP_LOOKUP_EXT_TCAM_8B_DATA_RESULT_ELEMENT_MATCH_OFFSET 62

#define EZDP_LOOKUP_EXT_TCAM_8B_DATA_RESULT_ELEMENT_MATCH_WORD_SELECT 1

#define EZDP_LOOKUP_EXT_TCAM_8B_DATA_RESULT_ELEMENT_MATCH_WORD_OFFSET 30

#define EZDP_LOOKUP_EXT_TCAM_8B_DATA_RESULT_ELEMENT_MATCH_MASK (1 <<
EZDP_LOOKUP_EXT_TCAM_8B_DATA_RESULT_ELEMENT_MATCH_WORD_OFFSET)

#define EZDP_LOOKUP_EXT_TCAM_8B_DATA_RESULT_ELEMENT_VALID_SIZE 1

#define EZDP_LOOKUP_EXT_TCAM_8B_DATA_RESULT_ELEMENT_VALID_OFFSET 63

#define EZDP_LOOKUP_EXT_TCAM_8B_DATA_RESULT_ELEMENT_VALID_WORD_SELECT 1

#define EZDP_LOOKUP_EXT_TCAM_8B_DATA_RESULT_ELEMENT_VALID_WORD_OFFSET 31

#define EZDP_LOOKUP_EXT_TCAM_8B_DATA_RESULT_ELEMENT_VALID_MASK (1 <<
EZDP_LOOKUP_EXT_TCAM_8B_DATA_RESULT_ELEMENT_VALID_WORD_OFFSET)

#define EZDP_LOOKUP_EXT_TCAM_8B_DATA_RESULT_ELEMENT_WORD_COUNT 2

#define EZDP_LOOKUP_EXT_TCAM_16B_DATA_RESULT_ELEMENT_ASSOC_DATA_SIZE 120
```

```
#define EZDP_LOOKUP_EXT_TCAM_16B_DATA_RESULT_ELEMENT_ASSOC_DATA_OFFSET 0

#define EZDP_LOOKUP_EXT_TCAM_16B_DATA_RESULT_ELEMENT_ASSOC_DATA_COUNT 15

#define
EZDP_LOOKUP_EXT_TCAM_16B_DATA_RESULT_ELEMENT_ASSOC_DATA_WORD_SELECT 0

#define
EZDP_LOOKUP_EXT_TCAM_16B_DATA_RESULT_ELEMENT_ASSOC_DATA_WORD_OFFSET 0

#define EZDP_LOOKUP_EXT_TCAM_16B_DATA_RESULT_ELEMENT_RESERVED24_SIZE 1

#define
EZDP_LOOKUP_EXT_TCAM_16B_DATA_RESULT_ELEMENT_RESERVED24_OFFSET 120

#define EZDP_LOOKUP_EXT_TCAM_16B_DATA_RESULT_ELEMENT_TRUNCATED_SIZE 1

#define EZDP_LOOKUP_EXT_TCAM_16B_DATA_RESULT_ELEMENT_TRUNCATED_OFFSET 121

#define
EZDP_LOOKUP_EXT_TCAM_16B_DATA_RESULT_ELEMENT_TRUNCATED_WORD_SELECT 3

#define
EZDP_LOOKUP_EXT_TCAM_16B_DATA_RESULT_ELEMENT_TRUNCATED_WORD_OFFSET 25

#define EZDP_LOOKUP_EXT_TCAM_16B_DATA_RESULT_ELEMENT_TRUNCATED_MASK (1 <<
EZDP_LOOKUP_EXT_TCAM_16B_DATA_RESULT_ELEMENT_TRUNCATED_WORD_OFFSET)

#define EZDP_LOOKUP_EXT_TCAM_16B_DATA_RESULT_ELEMENT_LOOKUP_ERROR_SIZE 1

#define
EZDP_LOOKUP_EXT_TCAM_16B_DATA_RESULT_ELEMENT_LOOKUP_ERROR_OFFSET 122

#define
EZDP_LOOKUP_EXT_TCAM_16B_DATA_RESULT_ELEMENT_LOOKUP_ERROR_WORD_SELECT
 3

#define
EZDP_LOOKUP_EXT_TCAM_16B_DATA_RESULT_ELEMENT_LOOKUP_ERROR_WORD_OFFSET
 26

#define
EZDP_LOOKUP_EXT_TCAM_16B_DATA_RESULT_ELEMENT_LOOKUP_ERROR_MASK (1 <<
EZDP_LOOKUP_EXT_TCAM_16B_DATA_RESULT_ELEMENT_LOOKUP_ERROR_WORD_OFFSET)

#define EZDP_LOOKUP_EXT_TCAM_16B_DATA_RESULT_ELEMENT_TYPE_SIZE 3

#define EZDP_LOOKUP_EXT_TCAM_16B_DATA_RESULT_ELEMENT_TYPE_OFFSET 123

#define EZDP_LOOKUP_EXT_TCAM_16B_DATA_RESULT_ELEMENT_TYPE_WORD_SELECT 3

#define EZDP_LOOKUP_EXT_TCAM_16B_DATA_RESULT_ELEMENT_TYPE_WORD_OFFSET 27

#define EZDP_LOOKUP_EXT_TCAM_16B_DATA_RESULT_ELEMENT_MATCH_SIZE 1
```

#define EZDP_LOOKUP_EXT_TCAM_16B_DATA_RESULT_ELEMENT_MATCH_OFFSET 126

#define EZDP_LOOKUP_EXT_TCAM_16B_DATA_RESULT_ELEMENT_MATCH_WORD_SELECT 3

#define EZDP_LOOKUP_EXT_TCAM_16B_DATA_RESULT_ELEMENT_MATCH_WORD_OFFSET 30

#define EZDP_LOOKUP_EXT_TCAM_16B_DATA_RESULT_ELEMENT_MATCH_MASK (1 << EZDP_LOOKUP_EXT_TCAM_16B_DATA_RESULT_ELEMENT_MATCH_WORD_OFFSET)

#define EZDP_LOOKUP_EXT_TCAM_16B_DATA_RESULT_ELEMENT_VALID_SIZE 1

#define EZDP_LOOKUP_EXT_TCAM_16B_DATA_RESULT_ELEMENT_VALID_OFFSET 127

#define EZDP_LOOKUP_EXT_TCAM_16B_DATA_RESULT_ELEMENT_VALID_WORD_SELECT 3

#define EZDP_LOOKUP_EXT_TCAM_16B_DATA_RESULT_ELEMENT_VALID_WORD_OFFSET 31

#define EZDP_LOOKUP_EXT_TCAM_16B_DATA_RESULT_ELEMENT_VALID_MASK (1 << EZDP_LOOKUP_EXT_TCAM_16B_DATA_RESULT_ELEMENT_VALID_WORD_OFFSET)

#define EZDP_LOOKUP_EXT_TCAM_16B_DATA_RESULT_ELEMENT_WORD_COUNT 4

#define EZDP_LOOKUP_EXT_TCAM_32B_DATA_RESULT_ELEMENT_ASSOC_DATA_SIZE 248

#define EZDP_LOOKUP_EXT_TCAM_32B_DATA_RESULT_ELEMENT_ASSOC_DATA_OFFSET 0

#define EZDP_LOOKUP_EXT_TCAM_32B_DATA_RESULT_ELEMENT_ASSOC_DATA_COUNT 31

#define EZDP_LOOKUP_EXT_TCAM_32B_DATA_RESULT_ELEMENT_ASSOC_DATA_WORD_SELECT 0

#define EZDP_LOOKUP_EXT_TCAM_32B_DATA_RESULT_ELEMENT_ASSOC_DATA_WORD_OFFSET 0

#define EZDP_LOOKUP_EXT_TCAM_32B_DATA_RESULT_ELEMENT_RESERVED24_SIZE 1

#define EZDP_LOOKUP_EXT_TCAM_32B_DATA_RESULT_ELEMENT_RESERVED24_OFFSET 248

#define EZDP_LOOKUP_EXT_TCAM_32B_DATA_RESULT_ELEMENT_TRUNCATED_SIZE 1

#define EZDP_LOOKUP_EXT_TCAM_32B_DATA_RESULT_ELEMENT_TRUNCATED_OFFSET 249

#define EZDP_LOOKUP_EXT_TCAM_32B_DATA_RESULT_ELEMENT_TRUNCATED_WORD_SELECT 7

#define EZDP_LOOKUP_EXT_TCAM_32B_DATA_RESULT_ELEMENT_TRUNCATED_WORD_OFFSET 25

#define EZDP_LOOKUP_EXT_TCAM_32B_DATA_RESULT_ELEMENT_TRUNCATED_MASK (1 << EZDP_LOOKUP_EXT_TCAM_32B_DATA_RESULT_ELEMENT_TRUNCATED_WORD_OFFSET)

#define EZDP_LOOKUP_EXT_TCAM_32B_DATA_RESULT_ELEMENT_LOOKUP_ERROR_SIZE 1

```
#define
EZDP_LOOKUP_EXT_TCAM_32B_DATA_RESULT_ELEMENT_LOOKUP_ERROR_OFFSET 250

#define
EZDP_LOOKUP_EXT_TCAM_32B_DATA_RESULT_ELEMENT_LOOKUP_ERROR_WORD_SELECT
 7

#define
EZDP_LOOKUP_EXT_TCAM_32B_DATA_RESULT_ELEMENT_LOOKUP_ERROR_WORD_OFFSET
 26

#define
EZDP_LOOKUP_EXT_TCAM_32B_DATA_RESULT_ELEMENT_LOOKUP_ERROR_MASK (1 <<
EZDP_LOOKUP_EXT_TCAM_32B_DATA_RESULT_ELEMENT_LOOKUP_ERROR_WORD_OFFSET)

#define EZDP_LOOKUP_EXT_TCAM_32B_DATA_RESULT_ELEMENT_TYPE_SIZE 3

#define EZDP_LOOKUP_EXT_TCAM_32B_DATA_RESULT_ELEMENT_TYPE_OFFSET 251

#define EZDP_LOOKUP_EXT_TCAM_32B_DATA_RESULT_ELEMENT_TYPE_WORD_SELECT 7

#define EZDP_LOOKUP_EXT_TCAM_32B_DATA_RESULT_ELEMENT_TYPE_WORD_OFFSET 27

#define EZDP_LOOKUP_EXT_TCAM_32B_DATA_RESULT_ELEMENT_MATCH_SIZE 1

#define EZDP_LOOKUP_EXT_TCAM_32B_DATA_RESULT_ELEMENT_MATCH_OFFSET 254

#define EZDP_LOOKUP_EXT_TCAM_32B_DATA_RESULT_ELEMENT_MATCH_WORD_SELECT 7

#define
EZDP_LOOKUP_EXT_TCAM_32B_DATA_RESULT_ELEMENT_MATCH_WORD_OFFSET 30

#define EZDP_LOOKUP_EXT_TCAM_32B_DATA_RESULT_ELEMENT_MATCH_MASK (1 <<
EZDP_LOOKUP_EXT_TCAM_32B_DATA_RESULT_ELEMENT_MATCH_WORD_OFFSET)

#define EZDP_LOOKUP_EXT_TCAM_32B_DATA_RESULT_ELEMENT_VALID_SIZE 1

#define EZDP_LOOKUP_EXT_TCAM_32B_DATA_RESULT_ELEMENT_VALID_OFFSET 255

#define EZDP_LOOKUP_EXT_TCAM_32B_DATA_RESULT_ELEMENT_VALID_WORD_SELECT 7

#define EZDP_LOOKUP_EXT_TCAM_32B_DATA_RESULT_ELEMENT_VALID_WORD_OFFSET 31

#define EZDP_LOOKUP_EXT_TCAM_32B_DATA_RESULT_ELEMENT_VALID_MASK (1 <<
EZDP_LOOKUP_EXT_TCAM_32B_DATA_RESULT_ELEMENT_VALID_WORD_OFFSET)

#define EZDP_LOOKUP_EXT_TCAM_32B_DATA_RESULT_ELEMENT_WORD_COUNT 8

#define EZDP_LOOKUP_INT_TCAM_STANDARD_RESULT_MAX_NUM 4
```

Typedef Documentation

typedef uint32_t ezdp_hashed_key_t

typedef uint32_t ezdp_lookup_retval_t

typedef uint32_t ezdp_lookup_int_tcam_standard_result_t

typedef uint32_t ezdp_lookup_int_tcam_4B_data_result_t

typedef uint32_t ezdp_lookup_ext_tcam_retval_t

typedef uint32_t ezdp_lookup_ext_tcam_index_result_element_t

typedef uint32_t ezdp_lookup_ext_tcam_4B_data_result_element_t

typedef struct ezdp_table_struct_desc ezdp_table_struct_desc_t

typedef struct ezdp_hash_struct_desc ezdp_hash_struct_desc_t

typedef struct ezdp_alg_tcam_struct_desc ezdp_alg_tcam_struct_desc_t

typedef ezdp_sum_addr_table_desc_t ezdp_search_base_addr_t

typedef uint32_t ezdp_lookup_int_tcam_retval_t

Enumeration Type Documentation

enum ezdp_ext_tcam_result_element_type

ext tcam result element type possible values.

Enumerator:

EZDP_INDEX Index only.

EZDP_INDEX_4B_DATA Index plus 4 Byte (32 bit) associated data.

EZDP_INDEX_8B_DATA Index plus 8 Byte (64 bit) associated data.

EZDP_INDEX_16B_DATA Index plus 16 Byte (128 bit) associated data.

EZDP_INDEX_32B_DATA Index plus 32 Byte (256 bit) associated data.

EZDP_USER_DEFINED_ASSOC_DATA1 User defined associated data 1.

EZDP_USER_DEFINED_ASSOC_DATA2 User defined associated data 2.

EZDP_USER_DEFINED_ASSOC_DATA3 User defined associated data 3.

enum ezdp_search_hash_flags

search hash flags.

Enumerator:

EZDP_NONE Default.

EZDP_OPPORTUNISTIC Opportunistic (no memory error handling or waiting for resources).

EZDP_COMPRESS Compress hash slot.

EZDP_UNCONDITIONAL Unconditional (perform operation with no lookup and no lock).

enum ezdp_scan_hash_slot_action

scan hash slot callback function possible return values.

Enumerator:

EZDP_ACCEPT_ENTRY Accept the entry.

EZDP_DELETE_ENTRY Delete the entry.

EZDP_UPDATE_ENTRY Update the entry.

Variable Documentation

enum ezdp_scan_hash_slot_action(* ezdp_scan_entry_cb)(void __cmem *result_ptr, struct ezdp_hash_struct_desc *hash_desc, uintptr_t user_data)

dpe/dp/include/ezdp_search_prm.h File Reference

Functions

- static __always_inline void ezdp_prm_lock_table_line (ezdp_table_struct_desc_t *table_struct_desc, uint32_t key)
- *Lock a table entry.* static __always_inline bool ezdp_prm_trylock_table_line (ezdp_table_struct_desc_t *table_struct_desc, uint32_t key)
- *Try to lock a table entry.* static __always_inline void ezdp_prm_unlock_table_line (ezdp_table_struct_desc_t *table_struct_desc, uint32_t key)
- *Release a table entry lock.* static __always_inline ezdp_search_base_addr_t ezdp_prm_get_table_base_addr (ezdp_table_struct_desc_t *table_struct_desc)
- *Get search base address (used for lookup) from table struct descriptor.* static __always_inline ezdp_lookup_retval_t ezdp_prm_lookup_table_entry (ezdp_search_base_addr_t table_base_addr, uint32_t entry_size, uint32_t key, void *__cmem entry_ptr)
- *Lookup an entry in a table structure.* static __always_inline void ezdp_prm_update_table_entry (ezdp_table_struct_desc_t *struct_desc, uint32_t entry_size, uint32_t key, void *__cmem entry_ptr, char *__cmem work_area_ptr, uint32_t work_area_size)
- *Add a new entry in a table structure.* static __always_inline void ezdp_prm_delete_table_entry (ezdp_table_struct_desc_t *struct_desc, uint32_t entry_size, uint32_t key, char *__cmem work_area_ptr, uint32_t work_area_size)
- *Delete an entry from a table structure.* static __always_inline ezdp_hashed_key_t ezdp_prm_hash_key32 (uint32_t data, uint32_t key_offset, uint32_t key_size)
- *Calculate hash value for keys of up to 32 bits.* static __always_inline ezdp_hashed_key_t ezdp_prm_hash_key64 (uint64_t data, uint32_t key_offset, uint32_t key_size)
- *Calculate hash value for keys of up to 64 bits.* static __always_inline ezdp_hashed_key_t ezdp_prm_hash_bulk_key (uint8_t __cmem *key_ptr, uint32_t key_size)
- *Calculate hash value for keys > 8 bytes.* static __always_inline void ezdp_prm_lock_hash_slot (ezdp_hash_struct_desc_t *struct_desc, ezdp_hashed_key_t hashed_key)
- *Lock a hash slot according to the hashed key.* static __always_inline bool ezdp_prm_trylock_hash_slot (ezdp_hash_struct_desc_t *struct_desc, ezdp_hashed_key_t hashed_key)
- *Try to lock a hash slot according to the hashed key.* static __always_inline void ezdp_prm_unlock_hash_slot (ezdp_hash_struct_desc_t *struct_desc, ezdp_hashed_key_t hashed_key)
- *Release a hash slot lock.* static __always_inline ezdp_search_base_addr_t ezdp_prm_get_hash_base_addr (ezdp_hash_struct_desc_t *struct_desc)
- *Get search base address (used for lookup) from hash struct descriptor.* static __always_inline ezdp_lookup_retval_t ezdp_prm_lookup_hash_entry (ezdp_search_base_addr_t hash_base_addr, bool single_cycle, uint32_t key_size, uint32_t result_size, uint32_t entry_size, uint32_t hashed_key, void __cmem *key_ptr, void __cmem *entry_ptr, char __cmem *work_area_ptr, uint32_t work_area_size)
- *Lookup an entry in a hash structure.* static __always_inline ezdp_lookup_retval_t ezdp_prm_locate_hash_entry (ezdp_search_base_addr_t hash_base_addr, bool single_cycle, uint32_t key_size, uint32_t result_size, uint32_t entry_size, ezdp_hashed_key_t hashed_key, void __cmem *key_ptr, void __cmem *entry_ptr, ezdp_hash_op_ctx_t __cmem *op_ctx)
- *Lookup an entry location in a hash structure.* static __always_inline uint32_t ezdp_prm_add_hash_entry (ezdp_hash_struct_desc_t *struct_desc, bool single_cycle, uint32_t key_size, uint32_t result_size, uint32_t entry_size, ezdp_hashed_key_t hashed_key, void __cmem *key_ptr, void __cmem *result_ptr, ezdp_hash_op_ctx_t __cmem *op_ctx, void __cmem *entry_ptr)
- *Add a new entry in a hash structure.* static __always_inline void ezdp_prm_modify_hash_entry (ezdp_hash_struct_desc_t *struct_desc, bool single_cycle, uint32_t key_size, uint32_t result_size, uint32_t entry_size, ezdp_hash_op_ctx_t __cmem *op_ctx, void __cmem *entry_ptr)
- *Modify an existing entry in a hash structure.* static __always_inline void ezdp_prm_delete_hash_entry (ezdp_hash_struct_desc_t *struct_desc, bool single_cycle, uint32_t entry_size, ezdp_hash_op_ctx_t __cmem *op_ctx)
- *Delete an existing entry from a hash structure.* static __always_inline ezdp_lookup_retval_t ezdp_prm_get_hash_first_entry (ezdp_hash_struct_desc_t *hash_desc, bool single_cycle, uint32_t entry_size, uint32_t hash_index, void __cmem *entry_ptr, ezdp_hash_op_ctx_t __cmem *op_ctx)

- *Get first entry of a hash slot.* static __always_inline ezdp_lookup_retval_t ezdp_prm_get_hash_next_entry (ezdp_hash_struct_desc_t *hash_desc, uint32_t entry_size, void __cmem *entry_ptr, ezdp_hash_op_ctx_t __cmem *op_ctx)
- *Get next entry of a hash slot.* static __always_inline void ezdp_prm_compress_hash_entry (struct ezdp_hash_struct_desc *hash_desc, bool single_cycle, uint32_t entry_size, ezdp_hash_op_ctx_t __cmem *op_ctx, void __cmem *entry_ptr)
- *Compress a hash entry with other entries in the same hash slot, if possible.* static __always_inline ezdp_search_base_addr_t ezdp_prm_get_ultra_ip_base_addr (ezdp_ultra_ip_struct_desc_t *struct_desc)
- *Get search base address (used for lookup) from UltraIP struct descriptor.* static __always_inline ezdp_lookup_retval_t ezdp_prm_lookup_ultra_ip_entry (ezdp_search_base_addr_t uip_base_addr, void __cmem *key_ptr, uint32_t key_size, char __cmem *work_area_ptr, uint32_t work_area_size)
- *Lookup an entry in an UltraIP structure.* static __always_inline ezdp_lookup_retval_t ezdp_prm_lookup_alg_tcam (struct ezdp_alg_tcam_struct_desc *struct_desc, uint8_t __cmem *key_ptr, uint32_t key_size, void __cmem *work_area_ptr, uint32_t work_area_size)

Lookup an entry in an algorithmic TCAM structure.

Function Documentation

static __always_inline void ezdp_prm_lock_table_line (ezdp_table_struct_desc_t * *table_struct_desc*, uint32_t *key*) [static]

Lock a table entry.

Calling thread shall acquire the lock if it is not held by another thread. Otherwise, the thread shall spin until the lock becomes available

Parameters:

[in] *table_struct_desc* - table structure description
[in] *key* - index into table

Returns:

none

static __always_inline bool ezdp_prm_trylock_table_line (ezdp_table_struct_desc_t * *table_struct_desc*, uint32_t *key*) [static]

Try to lock a table entry.

Calling thread shall acquire the lock if it is not held by another thread.

Parameters:

[in] *table_struct_desc* - table structure description
[in] *key* - index into table

Returns:

true: locked, false: did not succeed to lock

static __always_inline void ezdp_prm_unlock_table_line (ezdp_table_struct_desc_t * *table_struct_desc*, uint32_t *key*) [static]

Release a table entry lock.

Parameters:

[in] *table_struct_desc* - table structure description
[in] *key* - index into table

Returns:

none

static __always_inline ezdp_search_base_addr_t ezdp_prm_get_table_base_addr (ezdp_table_struct_desc_t * *table_struct_desc*) [static]

Get search base address (used for lookup) from table struct descriptor.

Parameters:

[in] *table_struct_desc* - table structure description

Returns:

ezdp_prm_table_lookup_struct_desc_t

static __always_inline ezdp_lookup_retval_t ezdp_prm_lookup_table_entry (ezdp_search_base_addr_t *table_base_addr*, uint32_t *entry_size*, uint32_t *key*, void *__cmem *entry_ptr*) [static]

Lookup an entry in a table structure.

The lookup result is written to CMEM.

Parameters:

[in] *table_base_addr* - lookup table structure description
[in] *entry_size* - size of the table entry (result)
[in] *key* - index into table
[out] *entry_ptr* - pointer (in CMEM) to write table entry (result)

Note:

Does not handle memory error

Returns:

ezdp_lookup_retval_t - lookup result as defined in ezdp_lookup_retval

static __always_inline void ezdp_prm_update_table_entry (ezdp_table_struct_desc_t * *struct_desc*, uint32_t *entry_size*, uint32_t *key*, void *__cmem *entry_ptr*, char *__cmem *work_area_ptr*, uint32_t *work_area_size*) [static]

Add a new entry in a table structure.

Override the existing entry.

Parameters:

[in] *struct_desc* - table structure description can be derived from ezdp_get_table_struct_desc API, based on struct ID
[in] *key* - key
[in] *entry_size* - size of the table entry (result)
[in] *entry_ptr* - pointer (in CMEM) to updated entry (result)
[in] *work_area_ptr* - work area pointer (temporary memory on CMEM to be used by the function). The size of the temporary memory is determined by EZDP_TABLE_LOW_LEVEL_WORK_AREA_SIZE
[in] *work_area_size* - size of work area pointer

Returns:

void

static __always_inline void ezdp_prm_delete_table_entry (ezdp_table_struct_desc_t * *struct_desc*, uint32_t *entry_size*, uint32_t *key*, char *__cmem *work_area_ptr*, uint32_t *work_area_size*) [static]

Delete an entry from a table structure.

Parameters:

[in] *struct_desc* - table structure description can be derived from ezdp_get_table_struct_desc API, based on struct ID
[in] *key* - key
[in] *entry_size* - size of the table entry (result)
[in] *work_area_ptr* - work area pointer (temporary memory on CMEM to be used by the function). The size of the temporary memory is determined by EZDP_TABLE_LOW_LEVEL_WORK_AREA_SIZE
[in] *work_area_size* - size of work area pointer

Returns:

None

static __always_inline ezdp_hashed_key_t ezdp_prm_hash_key32 (uint32_t *data*, uint32_t *key_offset*, uint32_t *key_size*) [static]

Calculate hash value for keys of up to 32 bits.

Parameters:

[in] *data* - contains the key to calculate hash from
[in] *key_offset* - offset in bytes to key in data 0 <= key_offset <= 3
[in] *key_size* - size in bytes of the key to hash 1 <= key_size <= 4

Returns:

the calculated hash of the key

static __always_inline ezdp_hashed_key_t ezdp_prm_hash_key64 (uint64_t *data*, uint32_t *key_offset*, uint32_t *key_size*) [static]

Calculate hash value for keys of up to 64 bits.

Parameters:

[in] *data* - contains the key to calculate hash from
[in] *key_offset* - offset in bytes to key in data 0 <= key_offset <= 3
[in] *key_size* - size in bytes of the key to hash 1 <= key_size <= 8

Returns:

the calculated hash of the key

static __always_inline ezdp_hashed_key_t ezdp_prm_hash_bulk_key (uint8_t __cmem * *key_ptr*, uint32_t *key_size*) [static]

Calculate hash value for keys > 8 bytes.

Parameters:

[in] *key_ptr* - pointer to key in cmem
[in] *key_size* - size in bytes of the key to hash (values 9..64)

Returns:

the calculated hash of the key

static __always_inline void ezdp_prm_lock_hash_slot (ezdp_hash_struct_desc_t * *struct_desc*, ezdp_hashed_key_t *hashed_key*) [static]

Lock a hash slot according to the hashed key.

Calling thread shall acquire the lock if it is not held by another thread. Otherwise, the thread shall spin until the lock becomes available

Parameters:

[in] *struct_desc* - hash struct descriptor
[in] *hashed_key* - the hash of the key

Returns:

none

static __always_inline bool ezdp_prm_trylock_hash_slot (ezdp_hash_struct_desc_t * *struct_desc*, ezdp_hashed_key_t *hashed_key*) `[static]`

Try to lock a hash slot according to the hashed key.

Calling thread shall acquire the lock if it is not held by another thread.

Parameters:

[in] *struct_desc* - hash struct descriptor
[in] *hashed_key* - the hash of the key

Returns:

true: locked, false: did not succeed to lock

static __always_inline void ezdp_prm_unlock_hash_slot (ezdp_hash_struct_desc_t * *struct_desc*, ezdp_hashed_key_t *hashed_key*) `[static]`

Release a hash slot lock.

Parameters:

[in] *struct_desc* - hash struct descriptor
[in] *hashed_key* - the hash of the key

Returns:

none

static __always_inline ezdp_search_base_addr_t ezdp_prm_get_hash_base_addr (ezdp_hash_struct_desc_t * *struct_desc*) `[static]`

Get search base address (used for lookup) from hash struct descriptor.

Parameters:

[in] *struct_desc* - hash structure description

Returns:

ezdp_prm_hash_lookup_struct_desc_t

static __always_inline ezdp_lookup_retval_t ezdp_prm_lookup_hash_entry (ezdp_search_base_addr_t *hash_base_addr*, bool *single_cycle*, uint32_t *key_size*, uint32_t *result_size*, uint32_t *entry_size*, uint32_t *hashed_key*, void __cmem * *key_ptr*, void __cmem * *entry_ptr*, char __cmem * *work_area_ptr*, uint32_t *work_area_size*) `[static]`

Lookup an entry in a hash structure.

The lookup result is written to CMEM.

Parameters:

[in] *hash_base_addr* - lookup hash struct descriptor

[in] *single_cycle* - single-cycle or non-single-cycle hash
[in] *key_size* - size of the key
[in] *result_size* - size of the result (for best performance should be a multiple of 4)
[in] *entry_size* - size of the entry
[in] *hashed_key* - the hash of the key
[in] *key_ptr* - pointer to key (in CMEM)
[out] *entry_ptr* - pointer (in CMEM) to return entry
[in] *work_area_ptr* - work area pointer (temporary memory on CMEM to be used by the function). The size of the temporary memory is determined by EZDP_HASH_LOW_LEVEL_WORK_AREA_SIZE
[in] *work_area_size* - size of work area pointer

Note:

This function does not handle memory error. When the entry is not found (no match), entry_ptr is not valid. For best performance result_size should be a multiple of 4

Returns:

ezdp_lookup_retval_t - lookup result as defined in ezdp_lookup_retval

static __always_inline ezdp_lookup_retval_t ezdp_prm_locate_hash_entry (ezdp_search_base_addr_t *hash_base_addr*, bool *single_cycle*, uint32_t *key_size*, uint32_t *result_size*, uint32_t *entry_size*, ezdp_hashed_key_t *hashed_key*, void __cmem * *key_ptr*, void __cmem * *entry_ptr*, ezdp_hash_op_ctx_t __cmem * *op_ctx*) `[static]`

Lookup an entry location in a hash structure.

The lookup result is written to CMEM. The state of the lookup is stored for future update operations. In addition, the operation context is returned, to be used later to perform add/modify/delete.

Parameters:

[in] *hash_base_addr* - lookup hash struct descriptor
[in] *single_cycle* - single-cycle or non-single-cycle hash
[in] *key_size* - size of the key
[in] *result_size* - size of the result (for best performance should be a multiple of 4)
[in] *entry_size* - size of the entry
[in] *hashed_key* - the hash of the key
[in] *key_ptr* - pointer to key (in CMEM)
[out] *entry_ptr* - pointer (in CMEM) to return entry
[in] *op_ctx* - operation context status (in CMEM)

Note:

This function does not handle memory error and multi thread synchronization. When the entry is not found (no match), entry_ptr is not valid. For best performance result_size should be a multiple of 4

Returns:

ezdp_lookup_retval_t - lookup result as defined in ezdp_lookup_retval

static __always_inline uint32_t ezdp_prm_add_hash_entry (ezdp_hash_struct_desc_t * *struct_desc*, bool *single_cycle*, uint32_t *key_size*, uint32_t *result_size*, uint32_t *entry_size*, ezdp_hashed_key_t *hashed_key*, void __cmem * *key_ptr*, void __cmem * *result_ptr*, ezdp_hash_op_ctx_t __cmem * *op_ctx*, void __cmem * *entry_ptr*) `[static]`

Add a new entry in a hash structure.

Parameters:

[in] *struct_desc* - hash struct descriptor
[in] *single_cycle* - single-cycle or non-single-cycle hash
[in] *key_size* - size of the key
[in] *result_size* - size of the result (for best performance should be a multiple of 4)
[in] *entry_size* - size of the entry
[in] *hashed_key* - the hash of the key

[in] *key_ptr* - pointer to key (in CMEM)
[in] *result_ptr* - pointer to updated result (in CMEM)
[in] *op_ctx* - operation context status (in CMEM)
[in] *entry_ptr* - temporary entry size buffer on CMEM

Note:

Not thread safe. Must be used with ezdp_lock_hash() before retrieving op_ctx.

Returns:

0 (success), ENOMEM (hash is full) use ezdp_get_err_msg() API to get the detail error message of the failure

static __always_inline void ezdp_prm_modify_hash_entry (ezdp_hash_struct_desc_t * *struct_desc*, bool *single_cycle*, uint32_t *key_size*, uint32_t *result_size*, uint32_t *entry_size*, ezdp_hash_op_ctx_t __cmem * *op_ctx*, void __cmem * *entry_ptr*) `[static]`

Modify an existing entry in a hash structure.

Parameters:

[in] *struct_desc* - hash struct descriptor
[in] *single_cycle* - single-cycle or non-single-cycle hash
[in] *key_size* - size of the key
[in] *result_size* - size of the result (for best performance should be a multiple of 4)
[in] *entry_size* - size of the entry
[in] *op_ctx* - operation context status (in CMEM)
[in] *entry_ptr* - temporary entry size buffer on CMEM to be used by the function

Note:

Not thread safe. Must be used with ezdp_lock_hash() before retrieving op_ctx.

Returns:

none

static __always_inline void ezdp_prm_delete_hash_entry (ezdp_hash_struct_desc_t * *struct_desc*, bool *single_cycle*, uint32_t *entry_size*, ezdp_hash_op_ctx_t __cmem * *op_ctx*) `[static]`

Delete an existing entry from a hash structure.

Parameters:

[in] *struct_desc* - hash struct descriptor
[in] *single_cycle* - single-cycle or non-single-cycle hash
[in] *entry_size* - size of the entry
[in] *op_ctx* - operation context status (in CMEM)

Note:

Not thread safe. Must be used with ezdp_lock_hash() before retrieving op_ctx.

Returns:

None

static __always_inline ezdp_lookup_retval_t ezdp_prm_get_hash_first_entry (ezdp_hash_struct_desc_t * *hash_desc*, bool *single_cycle*, uint32_t *entry_size*, uint32_t *hash_index*, void __cmem * *entry_ptr*, ezdp_hash_op_ctx_t __cmem * *op_ctx*) `[static]`

Get first entry of a hash slot.

Parameters:

[in] *hash_desc* - hash descriptor for the search table
[in] *single_cycle* - single-cycle or non-single-cycle hash
[in] *entry_size* - size of the entry
[in] *hash_index* - hash index (line/slot) number to scan
[out] *entry_ptr* - pointer (in CMEM) to return entry
[in] *op_ctx* - operation context status (in CMEM)

Note:

When hash index is empty (first entry not found), entry_ptr is not valid. Not thread safe. Must be used with ezdp_lock_hash() before retrieving op_ctx.

Returns:

ezdp_lookup_retval_t - lookup result as defined in ezdp_lookup_retval

static __always_inline ezdp_lookup_retval_t ezdp_prm_get_hash_next_entry (ezdp_hash_struct_desc_t * *hash_desc*, uint32_t *entry_size*, void __cmem * *entry_ptr*, ezdp_hash_op_ctx_t __cmem * *op_ctx*) `[static]`

Get next entry of a hash slot.

Parameters:

[in] *hash_desc* - hash descriptor for the search table
[in] *entry_size* - size of the entry
[out] *entry_ptr* - pointer (in CMEM) to return entry
[in] *op_ctx* - operation context status (in CMEM)

Note:

When next entry is not found, entry_ptr is not valid. Not thread safe. Must be used with ezdp_lock_hash() before retrieving op_ctx.

Returns:

ezdp_lookup_retval_t - lookup result as defined in ezdp_lookup_retval

static __always_inline void ezdp_prm_compress_hash_entry (struct ezdp_hash_struct_desc * *hash_desc*, bool *single_cycle*, uint32_t *entry_size*, ezdp_hash_op_ctx_t __cmem * *op_ctx*, void __cmem * *entry_ptr*) `[static]`

Compress a hash entry with other entries in the same hash slot, if possible.

Parameters:

[in] *hash_desc* - hash descriptor for the search table
[in] *single_cycle* - single-cycle or non-single-cycle hash
[in] *entry_size* - size of the entry
[in] *op_ctx* - operation context status (in CMEM)
[in] *entry_ptr* - pointer (in CMEM) to entry for internal use

Note:

Not thread safe. Must be used with ezdp_lock_hash() before retrieving op_ctx.

Returns:

void

static __always_inline ezdp_search_base_addr_t ezdp_prm_get_ultra_ip_base_addr (ezdp_ultra_ip_struct_desc_t * *struct_desc*) `[static]`

Get search base address (used for lookup) from UltraIP struct descriptor.

Parameters:

[in] *struct_desc* - ultra ip structure description

Returns:

ezdp_prm_ultra_ip_lookup_struct_desc_t

static __always_inline ezdp_lookup_retval_t ezdp_prm_lookup_ultra_ip_entry (ezdp_search_base_addr_t *uip_base_addr*, void __cmem * *key_ptr*, uint32_t *key_size*, char __cmem * *work_area_ptr*, uint32_t *work_area_size*) [static]

Lookup an entry in an UltraIP structure.

Parameters:

[in] *uip_base_addr* - lookup table summarized address descriptor
[in] *key_ptr* - pointer to key (in CMEM)
[in] *key_size* - size of key
[in] *work_area_ptr* - pointer to work area (temporary memory in CMEM to be used by the function). The size of the temporary memory is determined by EZDP_ULTRA_IP_WORK_AREA_SIZE
[in] *work_area_size* - size of work area pointer

Note:

Does not handle memory error.

Returns:

ezdp_lookup_retval_t - lookup result as defined in ezdp_lookup_retval ezdp_lookup_retval.data is 28bit data saved in found ultra ip entry

static __always_inline ezdp_lookup_retval_t ezdp_prm_lookup_alg_tcam (struct ezdp_alg_tcam_struct_desc * *struct_desc*, uint8_t __cmem * *key_ptr*, uint32_t *key_size*, void __cmem * *work_area_ptr*, uint32_t *work_area_size*) [static]

Lookup an entry in an algorithmic TCAM structure.

Parameters:

[in] *struct_desc* - algo tcam struct descriptor
[in] *key_ptr* - pointer to the key in CMEM
[in] *key_size* - size of the key
[in] *work_area_ptr* - pointer to work area (temporary memory in CMEM to be used by the function). The size of the temporary memory is determined by EZDP_ALG_TCAM_WORK_AREA_SIZE
[in] *work_area_size* - size of work area pointer

Note:

Does not handle memory error.

Returns:

ezdp_lookup_retval_t - lookup result as defined in ezdp_lookup_retval ezdp_lookup_retval.data is 28bit priority

dpe/dp/include/ezdp_security.h File Reference

Functions

- static __always_inline void ezdp_encrypt (ezdp_security_handle_t sec_handle, void __cmem *plain_data_ptr, void __cmem *encr_data_ptr, uint32_t size)
- *Encrypt a data segment.* static __always_inline void ezdp_encrypt_async (ezdp_security_handle_t sec_handle, void __cmem *plain_data_ptr, void __cmem *encr_data_ptr, uint32_t size)
- *Non blocking version of ezdp_encrypt().* static __always_inline void ezdp_decrypt (ezdp_security_handle_t sec_handle, void __cmem *encr_data_ptr, void __cmem *plain_data_ptr, uint32_t size)
- *Decrypt a data segment.* static __always_inline void ezdp_decrypt_async (ezdp_security_handle_t sec_handle, void __cmem *encr_data_ptr, void __cmem *plain_data_ptr, uint32_t size)
- *Non blocking version of ezdp_decrypt().* static __always_inline void ezdp_mac_calculation (ezdp_security_handle_t sec_handle, void __cmem *data_ptr, const bool init, const bool last, uint32_t size)
- *Calculate the message authentication code (MAC) on a data segment.* static __always_inline void ezdp_mac_calculation_async (ezdp_security_handle_t sec_handle, void __cmem *data_ptr, const bool init, const bool last, uint32_t size)
- *Non blocking version of ezdp_mac_calculation().* static __always_inline void ezdp_start_hmac_calculation (ezdp_security_handle_t sec_handle, void __cmem *key_ptr, uint32_t size)
- *Start a hash-based message authentication code (MAC) calculation.* static __always_inline void ezdp_start_hmac_calculation_async (ezdp_security_handle_t sec_handle, void __cmem *key_ptr, uint32_t size)
- *Non blocking version of ezdp_start_hmac_calculation().* static __always_inline void ezdp_end_hmac_calculation (ezdp_security_handle_t sec_handle, void __cmem *key_ptr, uint32_t size)
- *Complete a hash-based message authentication code (MAC) calculation.* static __always_inline void ezdp_end_hmac_calculation_async (ezdp_security_handle_t sec_handle, void __cmem *key_ptr, uint32_t size)
- *Non blocking version of ezdp_end_hmac_calculation().* static __always_inline void ezdp_generate_security_initial_vector (ezdp_security_handle_t sec_handle, void __cmem *data_ptr, const bool init, uint32_t size)
- *Generate security initial vector.* static __always_inline void ezdp_generate_security_initial_vector_async (ezdp_security_handle_t sec_handle, void __cmem *data_ptr, const bool init, uint32_t size)
- *Non blocking version of ezdp_generate_security_initial_vector().* static __always_inline void ezdp_end_gcm_mac_calculation (ezdp_security_handle_t sec_handle, void __cmem *key_ptr, uint32_t size)
- *Complete a GCM hash-based message authentication code (MAC) calculation.* static __always_inline void ezdp_end_gcm_mac_calculation_async (ezdp_security_handle_t sec_handle, void __cmem *key_ptr, uint32_t size)
- *Non blocking version of ezdp_end_gcm_mac_calculation().* static __always_inline void ezdp_expand_security_key (ezdp_security_handle_t sec_handle)
- *Expands the key in the security context memory.* static __always_inline void ezdp_expand_security_key_async (ezdp_security_handle_t sec_handle)
- *Non blocking version of ezdp_expand_security_key().* static __always_inline void ezdp_write_security_state (ezdp_security_handle_t sec_handle, void __cmem *sec_state_ptr, uint32_t sec_state_size)
- *Copy the security state data from CMEM to the security context memory.* static __always_inline void ezdp_write_security_state_async (ezdp_security_handle_t sec_handle, void __cmem *sec_state_ptr, uint32_t sec_state_size)
- *Non blocking version of ezdp_write_security_state().* static __always_inline void ezdp_read_security_state (ezdp_security_handle_t sec_handle, void __cmem *sec_state_ptr, uint32_t sec_state_size)
- *Copy the security state data from the security context memory to CMEM.* static __always_inline void ezdp_read_security_state_async (ezdp_security_handle_t sec_handle, void __cmem *sec_state_ptr, uint32_t sec_state_size)
- *Non blocking version of ezdp_read_security_state().* static __always_inline uint32_t ezdp_security_state_size (ezdp_security_handle_t sec_handle)
- *Return the state size.* static __always_inline void ezdp_write_security_key (ezdp_security_handle_t sec_handle, void __cmem *key_ptr, uint32_t key_size)
- *Copy the security key from CMEM to the security context memory.* static __always_inline void ezdp_write_security_key_async (ezdp_security_handle_t sec_handle, void __cmem *key_ptr, uint32_t key_size)
- *Non blocking version of ezdp_write_security_key().* static __always_inline void ezdp_read_security_key (ezdp_security_handle_t sec_handle, void __cmem *key_ptr, uint32_t key_size)

- *Copy the security key from the security context memory to CMEM.* static __always_inline void ezdp_read_security_key_async (ezdp_security_handle_t sec_handle, void __cmem *key_ptr, uint32_t key_size)
- *Non blocking version of ezdp_read_security_key().* static __always_inline uint32_t ezdp_security_key_size (ezdp_security_handle_t sec_handle)
- *Return the key size.* static __always_inline void ezdp_write_security_mac (ezdp_security_handle_t sec_handle, void __cmem *mac_ptr, uint32_t mac_size)
- *Copy the security MAC from CMEM to the security context memory.* static __always_inline void ezdp_write_security_mac_async (ezdp_security_handle_t sec_handle, void __cmem *mac_ptr, uint32_t mac_size)
- *Non blocking version of ezdp_write_security_mac().* static __always_inline void ezdp_read_security_mac (ezdp_security_handle_t sec_handle, void __cmem *mac_ptr, uint32_t mac_size)
- *Copy the security MAC from the security context memory to CMEM.* static __always_inline void ezdp_read_security_mac_async (ezdp_security_handle_t sec_handle, void __cmem *mac_ptr, uint32_t mac_size)
- *Non blocking version of ezdp_read_security_mac().* static __always_inline uint32_t ezdp_security_mac_size (ezdp_security_handle_t sec_handle)
- *Return the MAC size.* static __always_inline void ezdp_write_security_initial_vector (ezdp_security_handle_t sec_handle, void __cmem *iv_ptr, uint32_t iv_size)
- *Copy the security initial vector from CMEM to the security context memory.* static __always_inline void ezdp_write_security_initial_vector_async (ezdp_security_handle_t sec_handle, void __cmem *iv_ptr, uint32_t iv_size)
- *Non blocking version of ezdp_write_security_initial_vector().* static __always_inline void ezdp_read_security_initial_vector (ezdp_security_handle_t sec_handle, void __cmem *iv_ptr, uint32_t iv_size)
- *Copy the security initial vector from the security context memory to CMEM.* static __always_inline void ezdp_read_security_initial_vector_async (ezdp_security_handle_t sec_handle, void __cmem *iv_ptr, uint32_t iv_size)
- *Non blocking version of ezdp_read_security_initial_vector().* static __always_inline uint32_t ezdp_security_initial_vector_size (ezdp_security_handle_t sec_handle)
- *Return the initial vector size.* static __always_inline void ezdp_write_security_context (ezdp_security_handle_t sec_handle, void __cmem *cntx_ptr, uint32_t cntx_size)
- *Copy the security context from CMEM to the security context memory.* static __always_inline void ezdp_write_security_context_async (ezdp_security_handle_t sec_handle, void __cmem *cntx_ptr, uint32_t cntx_size)
- *Non blocking version of ezdp_write_security_context().* static __always_inline void ezdp_read_security_context (ezdp_security_handle_t sec_handle, void __cmem *cntx_ptr, uint32_t cntx_size)
- *Copy the security context from the security context memory to CMEM.* static __always_inline void ezdp_read_security_context_async (ezdp_security_handle_t sec_handle, void __cmem *cntx_ptr, uint32_t cntx_size)
- *Non blocking version of ezdp_read_security_context().* static __always_inline uint32_t ezdp_security_block_size (ezdp_security_handle_t sec_handle)

Return the algorithmic engine minimal block size.

Function Documentation

static __always_inline void ezdp_encrypt (ezdp_security_handle_t *sec_handle*, void __cmem * *plain_data_ptr*, void __cmem * *encr_data_ptr*, uint32_t *size*) [static]

Encrypt a data segment.

The encryption is performed based on the algorithm configured in the security handle. The encrypted data is written back to CMEM and the initial vector is updated in the security context memory. In addition, for authenticated encryption algorithms such as AES_GCM and AES_CCM, the message digest is also updated in the security context memory.

Parameters:

[in] *sec_handle* - security handle
[in] *plain_data_ptr*- pointer to source (plain) data in CMEM to be encrypted
[out] *encr_data_ptr*- pointer to destination in CMEM to write encrypted data to

[in] *size* - number of bytes to encrypt

Note:

- The size must be a multiple of X as defined per algorithm.

Returns:

none

static __always_inline void ezdp_encrypt_async (ezdp_security_handle_t *sec_handle*, void __cmem * *plain_data_ptr*, void __cmem * *encr_data_ptr*, uint32_t *size*) [static]

Non blocking version of ezdp_encrypt().

Parameters:

[in] *sec_handle* - security handle
[in] *plain_data_ptr*- pointer to source (plain) data in CMEM to be encrypted
[out] *encr_data_ptr*- pointer to destination in CMEM to write encrypted data to
[in] *size* - number of bytes to encrypt

Note:

- Call ezdp_sync() to wait for the operation to complete, confirming that the encrypted data is ready in CMEM.

Returns:

none

static __always_inline void ezdp_decrypt (ezdp_security_handle_t *sec_handle*, void __cmem * *encr_data_ptr*, void __cmem * *plain_data_ptr*, uint32_t *size*) [static]

Decrypt a data segment.

The decryption is performed based on the algorithm configured in the security handle. The decrypted data is written back to CMEM and the initial vector is updated in the security context memory. In addition, for authenticated encryption algorithms such as AES_GCM and AES_CCM, the message digest is also updated in the security context memory.

Parameters:

[in] *sec_handle* - security handle
[in] *encr_data_ptr* - pointer to source (encrypted) data in CMEM to be decrypted
[out] *plain_data_ptr*- pointer to destination in CMEM to write decrypted data to
[in] *size* - number of bytes to decrypt

Note:

- The size must be a multiple of X as defined per algorithm.

Returns:

none

static __always_inline void ezdp_decrypt_async (ezdp_security_handle_t *sec_handle*, void __cmem * *encr_data_ptr*, void __cmem * *plain_data_ptr*, uint32_t *size*) [static]

Non blocking version of ezdp_decrypt().

Parameters:

[in] *sec_handle* - security handle
[in] *encr_data_ptr* - pointer to source (encrypted) data in CMEM to be decrypted
[out] *plain_data_ptr*- pointer to destination in CMEM to write decrypted data to
[in] *size* - number of bytes to decrypt

Note:

- Call ezdp_sync() to wait for the operation to complete, confirming that the decrypted data is ready in CMEM.

Returns:

none

static __always_inline void ezdp_mac_calculation (ezdp_security_handle_t *sec_handle*, void __cmem * *data_ptr*, const bool *init*, const bool *last*, uint32_t *size*) [static]

Calculate the message authentication code (MAC) on a data segment.

The MAC calculation is done based on the algorithm configured in the security handle. The message digest is updated in the security context memory.

Parameters:

[in] *sec_handle* - security handle
[in] *data_ptr* - pointer to source data in CMEM
[in] *init* - indicates first segment
[in] *last* - indicates last segment
[in] *size* - number of bytes to calculate MAC on

Returns:

none

static __always_inline void ezdp_mac_calculation_async (ezdp_security_handle_t *sec_handle*, void __cmem * *data_ptr*, const bool *init*, const bool *last*, uint32_t *size*) [static]

Non blocking version of ezdp_mac_calculation().

Parameters:

[in] *sec_handle* - security handle
[in] *data_ptr* - pointer to source data in CMEM
[in] *init* - indicates first segment
[in] *last* - indicates last segment
[in] *size* - number of bytes to calculate MAC on

Note:

- Call ezdp_sync() to wait for the operation to complete, confirming that the data was read from CMEM and the message digest is updated in the security context memory.

Returns:

none

static __always_inline void ezdp_start_hmac_calculation (ezdp_security_handle_t *sec_handle*, void __cmem * *key_ptr*, uint32_t *size*) [static]

Start a hash-based message authentication code (MAC) calculation.

This operation pads the data (encryption key) if required, XORs it with 0x36 (aka i_pad) and then calculates the message authentication code (MAC) on the result. The MAC calculation is performed based on the algorithm configured in the security handle. The message digest is updated in the security context memory.

Parameters:

[in] *sec_handle* - security handle
[in] *key_ptr* - pointer to the encryption key in CMEM
[in] *size* - number of bytes in the encryption key

Note:

- Must be called with encryption key as data, as defined in HMAC standards.

Returns:

none

static __always_inline void ezdp_start_hmac_calculation_async (ezdp_security_handle_t *sec_handle*, void __cmem * *key_ptr*, uint32_t *size*) [static]

Non blocking version of ezdp_start_hmac_calculation().

Parameters:

[in] *sec_handle* - security handle
[in] *key_ptr* - pointer to the encryption key in CMEM
[in] *size* - number of bytes in the encryption key

Note:

- Call ezdp_sync() to wait for the operation to complete, confirming that the data was read from CMEM and the message digest is updated in the security context memory.

Returns:

none

static __always_inline void ezdp_end_hmac_calculation (ezdp_security_handle_t *sec_handle*, void __cmem * *key_ptr*, uint32_t *size*) [static]

Complete a hash-based message authentication code (MAC) calculation.

This operation pads the data (encryption key) if required, XORs it with 0x5c (aka o_pa) and then calculates the message authentication code (MAC) on the result concatenated with the message digest in the security context memory. The MAC calculation is performed based on the algorithm configured in the security handle. The message digest is updated in the security context memory.

Parameters:

[in] *sec_handle* - security handle
[in] *key_ptr* - pointer to the encryption key in CMEM
[in] *size* - number of bytes in the encryption key

Note:

- Must be called with encryption key as data, as defined in HMAC standards.

Returns:

none

static __always_inline void ezdp_end_hmac_calculation_async (ezdp_security_handle_t *sec_handle*, void __cmem * *key_ptr*, uint32_t *size*) [static]

Non blocking version of ezdp_end_hmac_calculation().

Parameters:

[in] *sec_handle* - security handle
[in] *key_ptr* - pointer to the encryption key in CMEM
[in] *size* - number of bytes in the encryption key

Note:

- Call ezdp_sync() to wait for the operation to complete, confirming that the data was read from CMEM and the message digest is updated in the security context memory.

Returns:

none

static __always_inline void ezdp_generate_security_initial_vector (ezdp_security_handle_t *sec_handle*, void __cmem * *data_ptr*, const bool *init*, uint32_t *size*) [static]

Generate security initial vector.

The calculation of the initial vector is done based on the algorithm configured in the security handle. The initial vector is updated in the security context memory. For GCM algorithm, this function generates J0, which is later used as the initial vector.

Parameters:

[in] *sec_handle* - security handle
[in] *data_ptr* - pointer to source data in CMEM
[in] *init* - indicates first segment
[in] *size* - number of bytes to calculate MAC on

Note:

Applicable to EZDP_AES_GCM_128_ALG EZDP_AES_GCM_192_ALG EZDP_AES_GCM_256_ALG algorithms only

Returns:

none

static __always_inline void ezdp_generate_security_initial_vector_async (ezdp_security_handle_t *sec_handle*, void __cmem * *data_ptr*, const bool *init*, uint32_t *size*) [static]

Non blocking version of ezdp_generate_security_initial_vector().

Parameters:

[in] *sec_handle* - security handle
[in] *data_ptr* - pointer to source data in CMEM
[in] *init* - indicates first segment
[in] *size* - number of bytes to calculate MAC on

Note:

- Applicable to EZDP_AES_GCM_128_ALG EZDP_AES_GCM_192_ALG EZDP_AES_GCM_256_ALG algorithms only

- Call ezdp_sync() to wait for the operation to complete, confirming that the data was read from CMEM and the initial vector is updated in the security context memory.

Returns:

none

static __always_inline void ezdp_end_gcm_mac_calculation (ezdp_security_handle_t *sec_handle*, void __cmem * *key_ptr*, uint32_t *size*) [static]

Complete a GCM hash-based message authentication code (MAC) calculation.

Authentication code (MAC) on the result concatenated with the message digest in the security context memory. The MAC calculation is performed based on the GCM algorithm. The message digest is updated in the security context memory.

Parameters:

[in] *sec_handle* - security handle
[in] *key_ptr* - pointer to the encryption key in CMEM
[in] *size* - number of bytes in the encryption key

Returns:
none

static __always_inline void ezdp_end_gcm_mac_calculation_async (ezdp_security_handle_t *sec_handle*, void __cmem * *key_ptr*, uint32_t *size*) [static]

Non blocking version of ezdp_end_gcm_mac_calculation().

Parameters:
[in] *sec_handle* - security handle
[in] *key_ptr* - pointer to the encryption key in CMEM
[in] *size* - number of bytes in the encryption key

Note:
- Call ezdp_sync() to wait for the operation to complete, confirming that the data was read from CMEM and the message digest is updated in the security context memory.

Returns:
none

static __always_inline void ezdp_expand_security_key (ezdp_security_handle_t *sec_handle*) [static]

Expands the key in the security context memory.

Reads the security key in the security context memory, expands it and stores it back to the security context memory. This operation should be performed once after loading a non-expanded key to the security context memory.

Parameters:
[in] *sec_handle* - security handle

Note:
- Relevant for decryption algorithms.

Returns:
none

static __always_inline void ezdp_expand_security_key_async (ezdp_security_handle_t *sec_handle*) [static]

Non blocking version of ezdp_expand_security_key().

Parameters:
[in] *sec_handle* - security handle

Note:
- Call ezdp_sync() to wait for the operation to complete, confirming that the data is ready in the security context memory.

Returns:
none

static __always_inline void ezdp_write_security_state (ezdp_security_handle_t *sec_handle*, void __cmem * *sec_state_ptr*, uint32_t *sec_state_size*) [static]

Copy the security state data from CMEM to the security context memory.

Parameters:

[in] *sec_handle* - security handle
[in] *sec_state_ptr* - pointer to the security state data in CMEM
[in] *sec_state_size* - the size of the security sec_state_ptr

Note:

- The size of the security state is according to the the algorithm type, see ezdp_sec_state_size enum.

Returns:

none

static __always_inline void ezdp_write_security_state_async (ezdp_security_handle_t *sec_handle*, void __cmem * *sec_state_ptr*, uint32_t *sec_state_size*) [static]

Non blocking version of ezdp_write_security_state().

Parameters:

[in] *sec_handle* - security handle
[in] *sec_state_ptr* - pointer to the security state data in CMEM
[in] *sec_state_size* - the size of the security sec_state_ptr

Note:

- Call ezdp_sync() to wait for the operation to complete, confirming that the data was written into the security context memory.

Returns:

none

static __always_inline void ezdp_read_security_state (ezdp_security_handle_t *sec_handle*, void __cmem * *sec_state_ptr*, uint32_t *sec_state_size*) [static]

Copy the security state data from the security context memory to CMEM.

Parameters:

[in] *sec_handle* - security handle
[out] *sec_state_ptr*- pointer in CMEM to write the security state data to
[in] *sec_state_size* - the size of the security sec_state_ptr

Note:

- The size of the security state is according to the the algorithm type, see ezdp_sec_state_size enum.

Returns:

none

static __always_inline void ezdp_read_security_state_async (ezdp_security_handle_t *sec_handle*, void __cmem * *sec_state_ptr*, uint32_t *sec_state_size*) [static]

Non blocking version of ezdp_read_security_state().

Parameters:

[in] *sec_handle* - security handle
[out] *sec_state_ptr*- pointer in CMEM to write the security state data to
[in] *sec_state_size* - the size of the security sec_state_ptr

Note:

- Call ezdp_sync() to wait for the operation to complete, confirming that the data is ready in CMEM.

Returns:

none

static __always_inline uint32_t ezdp_security_state_size (ezdp_security_handle_t *sec_handle*) [static]

Return the state size.

The size of the state is according to alg type configured in security handle

Parameters:

[in] *sec_handle* - security handle, only alg_type is used

Returns:

0 - error, state size

static __always_inline void ezdp_write_security_key (ezdp_security_handle_t *sec_handle*, void __cmem * *key_ptr*, uint32_t *key_size*) [static]

Copy the security key from CMEM to the security context memory.

Parameters:

[in] *sec_handle* - security handle
[in] *key_ptr* - pointer to the security key in CMEM
[in] *key_size* - the size of the key_ptr

Note:

- The size of the security key is according to algorithm type, see ezdp_sec_key_size enum.

Returns:

none

static __always_inline void ezdp_write_security_key_async (ezdp_security_handle_t *sec_handle*, void __cmem * *key_ptr*, uint32_t *key_size*) [static]

Non blocking version of ezdp_write_security_key().

Parameters:

[in] *sec_handle* - security handle
[in] *key_ptr* - pointer to the security key in CMEM
[in] *key_size* - the size of the key_ptr

Note:

- Call ezdp_sync() to wait for the operation to complete, confirming that the data was written to the security context memory.

Returns:

none

static __always_inline void ezdp_read_security_key (ezdp_security_handle_t *sec_handle*, void __cmem * *key_ptr*, uint32_t *key_size*) [static]

Copy the security key from the security context memory to CMEM.

Parameters:

[in] *sec_handle* - security handle
[out] *key_ptr* - pointer in CMEM to write the security key to
[in] *key_size* - the size of the key_ptr

Note:

- The size of the security key is derived from algorithm type, see ezdp_sec_key_size enum.

Returns:

none

static __always_inline void ezdp_read_security_key_async (ezdp_security_handle_t *sec_handle*, void __cmem * *key_ptr*, uint32_t *key_size*) [static]

Non blocking version of ezdp_read_security_key().

Parameters:

[in] *sec_handle* - security handle
[out] *key_ptr* - pointer in CMEM to write the security key to
[in] *key_size* - the size of the key_ptr

Note:

- Call ezdp_sync() to wait for the operation to complete, confirming that the data is ready in CMEM.

Returns:

none

static __always_inline uint32_t ezdp_security_key_size (ezdp_security_handle_t *sec_handle*) [static]

Return the key size.

The size of the key is according to alg type configured in security handle

Parameters:

[in] *sec_handle* - security handle, only alg_type is used

Returns:

0 - error, key size

static __always_inline void ezdp_write_security_mac (ezdp_security_handle_t *sec_handle*, void __cmem * *mac_ptr*, uint32_t *mac_size*) [static]

Copy the security MAC from CMEM to the security context memory.

Parameters:

[in] *sec_handle* - security handle
[in] *mac_ptr* - pointer to the security MAC in CMEM
[in] *mac_size* - size of the mac_ptr

Note:

- The size of the security mac is according to algorithm type, see ezdp_sec_mac_size enum.

Returns:

none

static __always_inline void ezdp_write_security_mac_async (ezdp_security_handle_t *sec_handle*, void __cmem * *mac_ptr*, uint32_t *mac_size*) [static]

Non blocking version of ezdp_write_security_mac().

Parameters:

[in] *sec_handle* - security handle
[in] *mac_ptr* - pointer to the security MAC in CMEM
[in] *mac_size* - size of the mac_ptr

Note:

- Call ezdp_sync() to wait for the operation to complete, confirming that the data was written to the security context memory.

Returns:

none

static __always_inline void ezdp_read_security_mac (ezdp_security_handle_t *sec_handle*, void __cmem * *mac_ptr*, uint32_t *mac_size*) [static]

Copy the security MAC from the security context memory to CMEM.

Parameters:

[in] *sec_handle* - security handle
[out] *mac_ptr* - pointer in CMEM to write the security MAC to
[in] *mac_size* - size of the mac_ptr

Note:

- The size of the security MAC is derived from algorithm type, see ezdp_sec_mac_size enum.

Returns:

none

static __always_inline void ezdp_read_security_mac_async (ezdp_security_handle_t *sec_handle*, void __cmem * *mac_ptr*, uint32_t *mac_size*) [static]

Non blocking version of ezdp_read_security_mac().

Parameters:

[in] *sec_handle* - security handle
[out] *mac_ptr* - pointer in CMEM to write the security MAC to
[in] *mac_size* - size of the mac_ptr

Note:

- Call ezdp_sync() to wait for the operation to complete, confirming that the data is ready in CMEM.

Returns:

none

static __always_inline uint32_t ezdp_security_mac_size (ezdp_security_handle_t *sec_handle*) [static]

Return the MAC size.

The size of the MAC is according to alg type configured in security handle

Parameters:

[in] *sec_handle* - security handle, only alg_type is used

Returns:

0 - error, MAC size

static __always_inline void ezdp_write_security_initial_vector (ezdp_security_handle_t *sec_handle*, void __cmem * *iv_ptr*, uint32_t *iv_size*) [static]

Copy the security initial vector from CMEM to the security context memory.

Parameters:

[in] *sec_handle* - security handle
[in] *iv_ptr* - pointer to the security initial vector in CMEM
[in] *iv_size* - size of the iv_ptr

Note:

- The size of the security initial vector is according to algorithm type, see ezdp_security_initial vector_size enum.

Returns:

none

static __always_inline void ezdp_write_security_initial_vector_async (ezdp_security_handle_t *sec_handle*, void __cmem * *iv_ptr*, uint32_t *iv_size*) [static]

Non blocking version of ezdp_write_security_initial_vector().

Parameters:

[in] *sec_handle* - security handle
[in] *iv_ptr* - pointer to the security initial vector in CMEM
[in] *iv_size* - size of the iv_ptr

Note:

- Call ezdp_sync() to wait for the operation to complete, confirming that the data was written to the security context memory.

Returns:

none

static __always_inline void ezdp_read_security_initial_vector (ezdp_security_handle_t *sec_handle*, void __cmem * *iv_ptr*, uint32_t *iv_size*) [static]

Copy the security initial vector from the security context memory to CMEM.

Parameters:

[in] *sec_handle* - security handle
[out] *iv_ptr* - pointer in CMEM to write the security initial vector to
[in] *iv_size* - size of the iv_ptr

Note:

- The size of the security initial vector is derived from algorithm type, see ezdp_sec_mac_size enum.

Returns:

none

static __always_inline void ezdp_read_security_initial_vector_async (ezdp_security_handle_t *sec_handle*, void __cmem * *iv_ptr*, uint32_t *iv_size*) [static]

Non blocking version of ezdp_read_security_initial_vector().

Parameters:

[in] *sec_handle* - security handle
[out] *iv_ptr* - pointer in CMEM to write the security initial vector to
[in] *iv_size* - size of the iv_ptr

Note:

- Call ezdp_sync() to wait for the operation to complete, confirming that the data is ready in CMEM.

Returns:

none

static __always_inline uint32_t ezdp_security_initial_vector_size (ezdp_security_handle_t *sec_handle*) [static]

Return the initial vector size.

The size of the initial vector is according to alg type configured in security handle

Parameters:

[in] *sec_handle* - security handle, only alg_type is used

Returns:

0 - error, initial vector size

static __always_inline void ezdp_write_security_context (ezdp_security_handle_t *sec_handle*, void __cmem * *cntx_ptr*, uint32_t *cntx_size*) [static]

Copy the security context from CMEM to the security context memory.

Parameters:

[in] *sec_handle* - security handle
[in] *cntx_ptr* - pointer to the security context data in CMEM
[in] *cntx_size* - size of the cntx_ptr

Note:

- The size of the security context is 64 bytes

Returns:

none

static __always_inline void ezdp_write_security_context_async (ezdp_security_handle_t *sec_handle*, void __cmem * *cntx_ptr*, uint32_t *cntx_size*) [static]

Non blocking version of ezdp_write_security_context().

Parameters:

[in] *sec_handle* - security handle
[in] *cntx_ptr* - pointer to context data on CMEM
[in] *cntx_size* - size of the cntx_ptr

Note:

- Call ezdp_sync() to wait for the operation to complete, confirming that the data was written to the security context memory.

Returns:

none

static __always_inline void ezdp_read_security_context (ezdp_security_handle_t *sec_handle*, void __cmem * *cntx_ptr*, uint32_t *cntx_size*) [static]

Copy the security context from the security context memory to CMEM.

Parameters:

[in] *sec_handle* - security handle
[out] *cntx_ptr* - pointer in CMEM to write the context data to
[in] *cntx_size* - size of the cntx_ptr

Note:

- The size of the security context is 64 bytes

Returns:

none

static __always_inline void ezdp_read_security_context_async (ezdp_security_handle_t *sec_handle*, void __cmem * *cntx_ptr*, uint32_t *cntx_size*) [static]

Non blocking version of ezdp_read_security_context().

Parameters:

[in] *sec_handle* - security handle
[out] *cntx_ptr* - pointer CMEM to write the context data to
[in] *cntx_size* - size of the cntx_ptr

Note:

- Call ezdp_sync() to wait for the operation to complete, confirming that the data is ready in CMEM.

Returns:

none

static __always_inline uint32_t ezdp_security_block_size (ezdp_security_handle_t *sec_handle*) [static]

Return the algorithmic engine minimal block size.

The size of the block is according to alg type configured in security handle

Parameters:

[in] *sec_handle* - security handle, only alg_type is used

Returns:

0 - error, block size

dpe/dp/include/ezdp_security_defs.h File Reference

Data Structures

- struct ezdp_security_handle

security handle configuration data structure Defines

- #define EZDP_SECURITY_CONTEXT_SIZE 80
- *The size of the security context.* #define EZDP_SECURITY_CLUSTER_MAX_CONTEXTS 128
- *The number of security contexts in cluster.* #define EZDP_SECURITY_HANDLE_CONTEXT_ID_SIZE 8
- #define EZDP_SECURITY_HANDLE_CONTEXT_ID_OFFSET 0
- #define EZDP_SECURITY_HANDLE_RESERVED8_15_SIZE 8
- #define EZDP_SECURITY_HANDLE_RESERVED8_15_OFFSET 8
- #define EZDP_SECURITY_HANDLE_ALG_TYPE_SIZE 8
- #define EZDP_SECURITY_HANDLE_ALG_TYPE_OFFSET 16
- #define EZDP_SECURITY_HANDLE_RESERVED24_31_SIZE 8
- #define EZDP_SECURITY_HANDLE_RESERVED24_31_OFFSET 24

Typedefs

- typedef uint32_t ezdp_security_handle_t

Enumerations

- enum ezdp_sec_alg { EZDP_DES_CBC_ALG = EZASM_SECURITY_ALGORITHM_TYPE_DES_CBC,
  EZDP_3DES2_CBC_ALG = EZASM_SECURITY_ALGORITHM_TYPE_3DES2_CBC,
  EZDP_3DES3_CBC_ALG = EZASM_SECURITY_ALGORITHM_TYPE_3DES3_CBC,
  EZDP_DES_CFB_ALG = EZASM_SECURITY_ALGORITHM_TYPE_DES_CFB,
  EZDP_3DES2_CFB_ALG = EZASM_SECURITY_ALGORITHM_TYPE_3DES2_CFB,
  EZDP_3DES3_CFB_ALG = EZASM_SECURITY_ALGORITHM_TYPE_3DES3_CFB,
  EZDP_DES_OFB_ALG = EZASM_SECURITY_ALGORITHM_TYPE_DES_OFB,
  EZDP_3DES2_OFB_ALG = EZASM_SECURITY_ALGORITHM_TYPE_3DES2_OFB,
  EZDP_3DES3_OFB_ALG = EZASM_SECURITY_ALGORITHM_TYPE_3DES3_OFB,
  EZDP_DES_CTR_ALG = EZASM_SECURITY_ALGORITHM_TYPE_DES_CTR,
  EZDP_3DES2_CTR_ALG = EZASM_SECURITY_ALGORITHM_TYPE_3DES2_CTR,
  EZDP_3DES3_CTR_ALG = EZASM_SECURITY_ALGORITHM_TYPE_3DES3_CTR,
  EZDP_DES_ECB_ALG = EZASM_SECURITY_ALGORITHM_TYPE_DES_ECB,
  EZDP_3DES2_ECB_ALG = EZASM_SECURITY_ALGORITHM_TYPE_3DES2_ECB,
  EZDP_3DES3_ECB_ALG = EZASM_SECURITY_ALGORITHM_TYPE_3DES3_ECB,
  EZDP_AES_CBC_128_ALG = EZASM_SECURITY_ALGORITHM_TYPE_AES_CBC_128,
  EZDP_AES_CBC_192_ALG = EZASM_SECURITY_ALGORITHM_TYPE_AES_CBC_192,
  EZDP_AES_CBC_256_ALG = EZASM_SECURITY_ALGORITHM_TYPE_AES_CBC_256,
  EZDP_AES_CFB_128_ALG = EZASM_SECURITY_ALGORITHM_TYPE_AES_CFB_128,
  EZDP_AES_CFB_192_ALG = EZASM_SECURITY_ALGORITHM_TYPE_AES_CFB_192,
  EZDP_AES_CFB_256_ALG = EZASM_SECURITY_ALGORITHM_TYPE_AES_CFB_256,
  EZDP_AES_OFB_128_ALG = EZASM_SECURITY_ALGORITHM_TYPE_AES_OFB_128,
  EZDP_AES_OFB_192_ALG = EZASM_SECURITY_ALGORITHM_TYPE_AES_OFB_192,
  EZDP_AES_OFB_256_ALG = EZASM_SECURITY_ALGORITHM_TYPE_AES_OFB_256,
  EZDP_AES_CTR_128_ALG = EZASM_SECURITY_ALGORITHM_TYPE_AES_CTR_128,
  EZDP_AES_CTR_192_ALG = EZASM_SECURITY_ALGORITHM_TYPE_AES_CTR_192,
  EZDP_AES_CTR_256_ALG = EZASM_SECURITY_ALGORITHM_TYPE_AES_CTR_256,
  EZDP_AES_ECB_128_ALG = EZASM_SECURITY_ALGORITHM_TYPE_AES_ECB_128,
  EZDP_AES_ECB_192_ALG = EZASM_SECURITY_ALGORITHM_TYPE_AES_ECB_192,
  EZDP_AES_ECB_256_ALG = EZASM_SECURITY_ALGORITHM_TYPE_AES_ECB_256,
  EZDP_AES_CCM_128_ALG = EZASM_SECURITY_ALGORITHM_TYPE_AES_CCM_128,
  EZDP_AES_CCM_192_ALG = EZASM_SECURITY_ALGORITHM_TYPE_AES_CCM_192,
  EZDP_AES_CCM_256_ALG = EZASM_SECURITY_ALGORITHM_TYPE_AES_CCM_256,
  EZDP_AES_GCM_128_ALG = EZASM_SECURITY_ALGORITHM_TYPE_AES_GCM_128,
  EZDP_AES_GCM_192_ALG = EZASM_SECURITY_ALGORITHM_TYPE_AES_GCM_192,

EZDP_AES_GCM_256_ALG = EZASM_SECURITY_ALGORITHM_TYPE_AES_GCM_256, EZDP_MD5_ALG = EZASM_SECURITY_ALGORITHM_TYPE_MD5, EZDP_SHA1_ALG = EZASM_SECURITY_ALGORITHM_TYPE_SHA1, EZDP_GHASH_128_ALG = EZASM_SECURITY_ALGORITHM_TYPE_GHASH_128, EZDP_GHASH_192_ALG = EZASM_SECURITY_ALGORITHM_TYPE_GHASH_192, EZDP_GHASH_256_ALG = EZASM_SECURITY_ALGORITHM_TYPE_GHASH_256, EZDP_SHA2_224_ALG = EZASM_SECURITY_ALGORITHM_TYPE_SHA2_224, EZDP_SHA2_256_ALG = EZASM_SECURITY_ALGORITHM_TYPE_SHA2_256, EZDP_SHA2_384_ALG = EZASM_SECURITY_ALGORITHM_TYPE_SHA2_384, EZDP_SHA2_512_ALG = EZASM_SECURITY_ALGORITHM_TYPE_SHA2_512, EZDP_AES_CMAC_128_ALG = EZASM_SECURITY_ALGORITHM_TYPE_AES_CMAC_128, EZDP_AES_CMAC_192_ALG = EZASM_SECURITY_ALGORITHM_TYPE_AES_CMAC_192, EZDP_AES_CMAC_256_ALG = EZASM_SECURITY_ALGORITHM_TYPE_AES_CMAC_256, EZDP_AES_XCBC_MAC_128_ALG = EZASM_SECURITY_ALGORITHM_TYPE_AES_XCBC_MAC_128 }

security encryption types.

- enum ezdp_sec_key_size { EZDP_DES_XXX_KEY_SIZE = 8, EZDP_3DES2_XXX_KEY_SIZE = 16, EZDP_3DES3_XXX_KEY_SIZE = 24, EZDP_AES_XXX_128_KEY_SIZE = 16, EZDP_AES_XXX_192_KEY_SIZE = 24, EZDP_AES_XXX_256_KEY_SIZE = 32, EZDP_AES_XXX_MAC_128_KEY_SIZE = 16, EZDP_AES_XXX_MAC_192_KEY_SIZE = 24, EZDP_AES_XXX_MAC_256_KEY_SIZE = 32, EZDP_GHASH_128_KEY_SIZE = 16, EZDP_GHASH_192_KEY_SIZE = 24, EZDP_GHASH_256_KEY_SIZE = 32 }

  security key size possible values.
- enum ezdp_sec_initial_vector_size { EZDP_DES_IV_SIZE = 8, EZDP_3DES_IV_SIZE = 8, EZDP_AES_IV_SIZE = 16, EZDP_XXX_MAC_IV_SIZE = 16 }

  security key size possible values.
- enum ezdp_sec_state_size { EZDP_DES_STATE_SIZE = 8, EZDP_AES_STATE_SIZE = 16, EZDP_AES_CCM_STATE_SIZE = 48, EZDP_AES_GCM_STATE_SIZE = 48, EZDP_AES_CMAC_XXX_STATE_SIZE = 32, EZDP_AES_XCBC_MAC_STATE_SIZE = 48, EZDP_MD5_STATE_SIZE = 32, EZDP_SHA1_STATE_SIZE = 36, EZDP_GHASH_STATE_SIZE = 32, EZDP_GHASH_XXX_STATE_SIZE = 32, EZDP_SHA2_224_STATE_SIZE = 48, EZDP_SHA2_256_STATE_SIZE = 48, EZDP_SHA2_384_STATE_SIZE = 80, EZDP_SHA2_512_STATE_SIZE = 80 }

  security algorithm type possible values.
- enum ezdp_sec_mac_size { EZDP_AES_CCM_MAC_SIZE = 16, EZDP_AES_GCM_MAC_SIZE = 16, EZDP_MD5_MAC_SIZE = 16, EZDP_SHA1_MAC_SIZE = 20, EZDP_SHA2_224_MAC_SIZE = 28, EZDP_SHA2_256_MAC_SIZE = 32, EZDP_SHA2_384_MAC_SIZE = 48, EZDP_SHA2_512_MAC_SIZE = 64, EZDP_AES_CMAC_XXX_MAC_SIZE = 16, EZDP_AES_XCBC_MAC_128_MAC_SIZE = 16, EZDP_GHASH_MAC_SIZE = 16 }

  security algorithm type possible values.
- enum ezdp_sec_block_size { EZDP_DES_BLOCK_SIZE = 8, EZDP_3DES_BLOCK_SIZE = 8, EZDP_AES_BLOCK_SIZE = 16, EZDP_GHASH_BLOCK_SIZE = 16, EZDP_MD5_BLOCK_SIZE = 64, EZDP_SHA1_BLOCK_SIZE = 64, EZDP_SHA2_224_BLOCK_SIZE = 64, EZDP_SHA2_256_BLOCK_SIZE = 64, EZDP_SHA2_384_BLOCK_SIZE = 128, EZDP_SHA2_512_BLOCK_SIZE = 128 }

  security block size possible values.

Define Documentation

#define EZDP_SECURITY_CONTEXT_SIZE 80

The size of the security context.

#define EZDP_SECURITY_CLUSTER_MAX_CONTEXTS 128

The number of security contexts in cluster.

#define EZDP_SECURITY_HANDLE_CONTEXT_ID_SIZE 8

#define EZDP_SECURITY_HANDLE_CONTEXT_ID_OFFSET 0

#define EZDP_SECURITY_HANDLE_RESERVED8_15_SIZE 8

#define EZDP_SECURITY_HANDLE_RESERVED8_15_OFFSET 8

#define EZDP_SECURITY_HANDLE_ALG_TYPE_SIZE 8

#define EZDP_SECURITY_HANDLE_ALG_TYPE_OFFSET 16

#define EZDP_SECURITY_HANDLE_RESERVED24_31_SIZE 8

#define EZDP_SECURITY_HANDLE_RESERVED24_31_OFFSET 24

Typedef Documentation

typedef uint32_t ezdp_security_handle_t

Enumeration Type Documentation

enum ezdp_sec_alg

security encryption types.

Enumerator:

EZDP_DES_CBC_ALG Type: Encryption, Algorithm: DES, Mode: CBC, Key: single key of 64 bits.

EZDP_3DES2_CBC_ALG Type: Encryption, Algorithm: 3DES, Mode: CBC, Key: 2 keys of 64 bits (K1=K3, K2).

EZDP_3DES3_CBC_ALG Type: Encryption, Algorithm: 3DES, Mode: CBC, Key: 3 keys of 64 bits (K1, K2, K3).

EZDP_DES_CFB_ALG Type: Encryption, Algorithm: DES, Mode: CFB, Key: single key of 64 bits.

EZDP_3DES2_CFB_ALG Type: Encryption, Algorithm: 3DES, Mode: CFB, Key: 2 keys of 64 bits (K1=K3, K2).

EZDP_3DES3_CFB_ALG Type: Encryption, Algorithm: 3DES, Mode: CFB, Key: 3 keys of 64 bits (K1, K2, K3).

EZDP_DES_OFB_ALG Type: Encryption, Algorithm: DES, Mode: OFB, Key: single key of 64 bits.

EZDP_3DES2_OFB_ALG Type: Encryption, Algorithm: 3DES, Mode: OFB, Key: 2 keys of 64 bits (K1=K3, K2).

EZDP_3DES3_OFB_ALG Type: Encryption, Algorithm: 3DES, Mode: OFB, Key: 3 keys of 64 bits (K1, K2, K3).

EZDP_DES_CTR_ALG Type: Encryption, Algorithm: DES, Mode: CTR, Key: single key of 64 bits.

EZDP_3DES2_CTR_ALG Type: Encryption, Algorithm: 3DES, Mode: CTR, Key: 2 keys of 64 bits (K1=K3, K2).

EZDP_3DES3_CTR_ALG Type: Encryption, Algorithm: 3DES, Mode: CTR, Key: 3 keys of 64 bits (K1, K2, K3).

EZDP_DES_ECB_ALG Type: Encryption, Algorithm: DES, Mode: ECB, Key: single key of 64 bits.

EZDP_3DES2_ECB_ALG Type: Encryption, Algorithm: 3DES, Mode: ECB, Key: 2 keys of 64 bits (K1=K3, K2).

EZDP_3DES3_ECB_ALG Type: Encryption, Algorithm: 3DES, Mode: ECB, Key: 3 keys of 64 bits (K1, K2, K3).

EZDP_AES_CBC_128_ALG Type: Encryption, Algorithm: AES, Mode: CBC, Key: 128 bits.

EZDP_AES_CBC_192_ALG Type: Encryption, Algorithm: AES, Mode: CBC, Key: 192 bits.

EZDP_AES_CBC_256_ALG Type: Encryption, Algorithm: AES, Mode: CBC, Key: 192 bits.

EZDP_AES_CFB_128_ALG Type: Encryption, Algorithm: AES, Mode: CFB, Key: 128 bits.

EZDP_AES_CFB_192_ALG Type: Encryption, Algorithm: AES, Mode: CFB, Key: 192 bits.

EZDP_AES_CFB_256_ALG Type: Encryption, Algorithm: AES, Mode: CFB, Key: 192 bits.

EZDP_AES_OFB_128_ALG Type: Encryption, Algorithm: AES, Mode: OFB, Key: 128 bits.

EZDP_AES_OFB_192_ALG Type: Encryption, Algorithm: AES, Mode: OFB, Key: 192 bits.

EZDP_AES_OFB_256_ALG Type: Encryption, Algorithm: AES, Mode: OFB, Key: 192 bits.

EZDP_AES_CTR_128_ALG Type: Encryption, Algorithm: AES, Mode: CTR, Key: 128 bits.

EZDP_AES_CTR_192_ALG Type: Encryption, Algorithm: AES, Mode: CTR, Key: 192 bits.

EZDP_AES_CTR_256_ALG Type: Encryption, Algorithm: AES, Mode: CTR, Key: 192 bits.

EZDP_AES_ECB_128_ALG Type: Encryption, Algorithm: AES, Mode: ECB, Key: 128 bits.

EZDP_AES_ECB_192_ALG Type: Encryption, Algorithm: AES, Mode: ECB, Key: 192 bits.

EZDP_AES_ECB_256_ALG Type: Encryption, Algorithm: AES, Mode: ECB, Key: 192 bits.

EZDP_AES_CCM_128_ALG Type: Encryption and Authentication, Algorithm: AES, Mode: CCM, Key: 128 bits.

EZDP_AES_CCM_192_ALG Type: Encryption and Authentication, Algorithm: AES, Mode: CCM, Key: 192 bits.

EZDP_AES_CCM_256_ALG Type: Encryption and Authentication, Algorithm: AES, Mode: CCM, Key: 192 bits.

EZDP_AES_GCM_128_ALG Type: Encryption and Authentication, Algorithm: AES, Mode: GCM, Key: 128 bits.

EZDP_AES_GCM_192_ALG Type: Encryption and Authentication, Algorithm: AES, Mode: GCM, Key: 192 bits.

EZDP_AES_GCM_256_ALG Type: Encryption and Authentication, Algorithm: AES, Mode: GCM, Key: 192 bits.

EZDP_MD5_ALG Type: Authentication, Algorithm: MD5.

EZDP_SHA1_ALG Type: Authentication, Algorithm: SHA1.

EZDP_GHASH_128_ALG Type: Authentication, Algorithm: GHASH Key: 128 bits.

EZDP_GHASH_192_ALG Type: Authentication, Algorithm: GHASH Key: 192 bits.

EZDP_GHASH_256_ALG Type: Authentication, Algorithm: GHASH Key: 256 bits.

EZDP_SHA2_224_ALG Type: Authentication, Algorithm: SHA2-224.

EZDP_SHA2_256_ALG Type: Authentication, Algorithm: SHA2-256.

EZDP_SHA2_384_ALG Type: Authentication, Algorithm: SHA2-384.

EZDP_SHA2_512_ALG Type: Authentication, Algorithm: SHA2-512.

EZDP_AES_CMAC_128_ALG Type: Authentication, Algorithm: AES, Mode: CMAC, Key: 128 bits.

EZDP_AES_CMAC_192_ALG Type: Authentication, Algorithm: AES, Mode: CMAC, Key: 192 bits.

EZDP_AES_CMAC_256_ALG Type: Authentication, Algorithm: AES, Mode: CMAC, Key: 192 bits.

EZDP_AES_XCBC_MAC_128_ALG Type: Authentication, Algorithm: AES, Mode: XCBC MAC, Key: 128 bits.

enum ezdp_sec_key_size

security key size possible values.

Enumerator:

EZDP_DES_XXX_KEY_SIZE Type: Encryption, Algorithm: DES - key size 8 bytes (single key of 64 bits).

EZDP_3DES2_XXX_KEY_SIZE Type: Encryption, Algorithm: 3DES - key size 16 bytes (2 keys of 64 bits (K1=K3, K2)).

EZDP_3DES3_XXX_KEY_SIZE Type: Encryption, Algorithm: 3DES - key size 24 bytes (3 keys of 64 bits (K1, K2, K3)).

EZDP_AES_XXX_128_KEY_SIZE Type: Encryption, Algorithm: AES - key size 16 bytes.

EZDP_AES_XXX_192_KEY_SIZE Type: Encryption, Algorithm: AES - key size 24 bytes.

EZDP_AES_XXX_256_KEY_SIZE Type: Encryption, Algorithm: AES - key size 32 bytes.

EZDP_AES_XXX_MAC_128_KEY_SIZE Type: Authentication, Algorithm: AES - key size 16 bytes.

EZDP_AES_XXX_MAC_192_KEY_SIZE Type: Authentication, Algorithm: AES, Mode - key size 24 bytes.

EZDP_AES_XXX_MAC_256_KEY_SIZE Type: Authentication, Algorithm: AES - key size 32 bytes.

EZDP_GHASH_128_KEY_SIZE Type: Encryption, Algorithm: GHASH - key size 16 bytes.

EZDP_GHASH_192_KEY_SIZE Type: Encryption, Algorithm: GHASH - key size 24 bytes.

EZDP_GHASH_256_KEY_SIZE Type: Encryption, Algorithm: GHASH - key size 32 bytes.

enum ezdp_sec_initial_vector_size

security key size possible values.

Enumerator:

EZDP_DES_IV_SIZE Type: Encryption, Algorithm: DES - Initial vector size 8 bytes.

EZDP_3DES_IV_SIZE Type: Encryption, Algorithm: 3DES - Initial vector size 8 bytes.

EZDP_AES_IV_SIZE Type: Encryption, Algorithm: AES - Initial vector size 16 bytes.

EZDP_XXX_MAC_IV_SIZE Type: Authentication, Algorithm: AES - - Initial vector size 16 bytes.

enum ezdp_sec_state_size

security algorithm type possible values.

Enumerator:

EZDP_DES_STATE_SIZE Type: Encryption, Algorithm: DES - state size 8 bytes.

EZDP_AES_STATE_SIZE Type: Encryption, Algorithm: AES - state size 16 bytes.

EZDP_AES_CCM_STATE_SIZE Type: Authentication, Algorithm: AES_CCM - state size 48 bytes.

EZDP_AES_GCM_STATE_SIZE Type: Authentication, Algorithm: AES_GCM - state size 48 bytes.

EZDP_AES_CMAC_XXX_STATE_SIZE Type: Authentication, Algorithm: AES_CMAC_XXX - state size 32 bytes.

EZDP_AES_XCBC_MAC_STATE_SIZE Type: Authentication, Algorithm: AES_XCBC_MAC_128 - state size 48 bytes.

EZDP_MD5_STATE_SIZE Type: Authentication, Algorithm: MD5 - state size 32 bytes.

EZDP_SHA1_STATE_SIZE Type: Authentication, Algorithm: SHA1 - state size 36 bytes.

EZDP_GHASH_STATE_SIZE Type: Authentication, Algorithm: GHASH - state size 32 bytes.

EZDP_GHASH_XXX_STATE_SIZE Type: Authentication, Algorithm: GHASH - state size 32 bytes.

EZDP_SHA2_224_STATE_SIZE Type: Authentication, Algorithm: SHA2-224 - state size 48 byts.

EZDP_SHA2_256_STATE_SIZE Type: Authentication, Algorithm: SHA2-256 - state size 48 bytes.

EZDP_SHA2_384_STATE_SIZE Type: Authentication, Algorithm: SHA2-384 - state size 80 bytes.

EZDP_SHA2_512_STATE_SIZE Type: Authentication, Algorithm: SHA2-512 - state size 80 bytes.

enum ezdp_sec_mac_size

security algorithm type possible values.

Enumerator:

EZDP_AES_CCM_MAC_SIZE Type: Authentication, Algorithm: AES_CCM - MAC size 16 bytes.

EZDP_AES_GCM_MAC_SIZE Type: Authentication, Algorithm: AES_GCM - MAC size 16 bytes.

EZDP_MD5_MAC_SIZE Type: Authentication, Algorithm: MD5 - MAC size 16 bytes.

EZDP_SHA1_MAC_SIZE Type: Authentication, Algorithm: SHA1 - MAC size 20 bytes.

EZDP_SHA2_224_MAC_SIZE Type: Authentication, Algorithm: SHA2-224 - MAC size 32 bytes.

EZDP_SHA2_256_MAC_SIZE Type: Authentication, Algorithm: SHA2-256 - MAC size 32 bytes.

EZDP_SHA2_384_MAC_SIZE Type: Authentication, Algorithm: SHA2-384 - MAC size 48 bytes.

EZDP_SHA2_512_MAC_SIZE Type: Authentication, Algorithm: SHA2-512 - MAC size 64 bytes.

EZDP_AES_CMAC_XXX_MAC_SIZE Type: Authentication, Algorithm: AES_CMAC_XXX - MAC size 16 bytes.

EZDP_AES_XCBC_MAC_128_MAC_SIZE Type: Authentication, Algorithm: AES_XCBC_MAC_128 - MAC size 16 bytes.

EZDP_GHASH_MAC_SIZE Type: Authentication, Algorithm: GHASH - MAC size 16 bytes.

enum ezdp_sec_block_size

security block size possible values.

Enumerator:

EZDP_DES_BLOCK_SIZE Type: Encryption, Algorithm: DES - Block size 8 bytes.

EZDP_3DES_BLOCK_SIZE Type: Encryption, Algorithm: 3DES - Block size 8 bytes.

EZDP_AES_BLOCK_SIZE Type: Encryption, Algorithm: AES - Block size 16 bytes.

EZDP_GHASH_BLOCK_SIZE Type: Encryption, Algorithm: GHASH - Block size 16 bytes.

EZDP_MD5_BLOCK_SIZE Type: Authentication, Algorithm: MD5 - Block size 64 bytes.

EZDP_SHA1_BLOCK_SIZE Type: Authentication, Algorithm: SHA1 - Block size 64 bytes.

EZDP_SHA2_224_BLOCK_SIZE Type: Authentication, Algorithm: SHA2-224 - Block size 64 bytes.

EZDP_SHA2_256_BLOCK_SIZE Type: Authentication, Algorithm: SHA2-256 - Block size 64 bytes.

EZDP_SHA2_384_BLOCK_SIZE Type: Authentication, Algorithm: SHA2-384 - Block size 128 bytes.

EZDP_SHA2_512_BLOCK_SIZE Type: Authentication, Algorithm: SHA2-512 - Block size 128 bytes.

dpe/dp/include/ezdp_string.h File Reference

Functions

- static __always_inline void * ezdp_mem_copy (void *dst, const void *src, uint32_t size)
- *Copy a block of memory.* static __always_inline void * ezdp_mem_set (void *dst, int val, uint32_t size)
- *Set a block of memory to the specified value.* static uint32_t ezdp_mem_cmp (uint8_t __cmem *ptr1, uint8_t __cmem *ptr2, uint32_t size)
- *Compare two blocks of memory in CMEM.* static __always_inline uint32_t ezdp_mem_cmp_byte_skip (uint8_t __cmem *ptr1, uint8_t __cmem *ptr2, uint32_t size, uint32_t ptr1_offset)

Compare two blocks of memory in CMEM, skipping intermediate bytes.

Function Documentation

static __always_inline void* ezdp_mem_copy (void * *dst*, const void * *src*, uint32_t *size*) [static]

Copy a block of memory.

Parameters:

[in] *dst* - copy destination
[in] *src* - copy source
[in] *size* - size in bytes to copy

Note:

May not work properly if dst and src overlap.

Returns:

pointer to dst

static __always_inline void* ezdp_mem_set (void * *dst*, int *val*, uint32_t *size*) [static]

Set a block of memory to the specified value.

Parameters:

[in] *dst* - copy destination
[in] *val* - set value
[in] *size* - size in bytes

Returns:

pointer to dst

static uint32_t ezdp_mem_cmp (uint8_t __cmem * *ptr1*, uint8_t __cmem * *ptr2*, uint32_t *size*) [inline, static]

Compare two blocks of memory in CMEM.

Parameters:

[in] *ptr1* - pointer to first block of memory in CMEM
[in] *ptr2* - pointer to second block of memory in CMEM
[in] *size* - number of bytes to compare

Returns:

uint32_t - 0 if memory is identical

static __always_inline uint32_t ezdp_mem_cmp_byte_skip (uint8_t __cmem * *ptr1*, uint8_t __cmem * *ptr2*, uint32_t *size*, uint32_t *ptr1_offset*) `[static]`

Compare two blocks of memory in CMEM, skipping intermediate bytes.

Ignores a single byte every byte_skip_cycle_size bytes This can be used, for example, to compare data that is ECC protected in-band using a single byte of protection data for each 16 or 32 bytes.

Parameters:

[in] *ptr1* - pointer to first block of memory in CMEM
[in] *ptr2* - pointer to second block of memory in CMEM
[in] *size* - number of bytes to compare
[in] *ptr1_offset* - offset in ptr1 to begin from

Returns:

uint32_t - 0 if memory is identical

dpe/dp/include/ezdp_time.h File Reference

Functions

- static __always_inline uint32_t ezdp_get_system_tick (uint64_t __cmem *core_cycles)
- *Get system tick (in core cycles).* static __always_inline void ezdp_get_system_tick_async (uint64_t __cmem *core_cycles)
- *Non blocking version of ezdp_get_system_tick().* static __always_inline uint32_t ezdp_get_real_time_clock (struct ezdp_rtc __cmem *real_time_clock)
- *Get real time clock.* static __always_inline void ezdp_get_real_time_clock_async (struct ezdp_rtc __cmem *real_time_clock)

Non blocking version of ezdp_get_real_time_clock().

Function Documentation

static __always_inline uint32_t ezdp_get_system_tick (uint64_t __cmem * *core_cycles*) [static]

Get system tick (in core cycles).

Parameters:
[out] *core_cycles* - pointer to CMEM to write the current system tick into

Returns:
uint32_t - 32 LSB of the system tick

static __always_inline void ezdp_get_system_tick_async (uint64_t __cmem * *core_cycles*) [static]

Non blocking version of ezdp_get_system_tick().

Parameters:
[out] *core_cycles* - pointer to CMEM to write the current system tick into

Note:
Call ezdp_sync() to wait for the operation to complete, confirming that the result is ready in CMEM.

Returns:
none

static __always_inline uint32_t ezdp_get_real_time_clock (struct ezdp_rtc __cmem * *real_time_clock*) [static]

Get real time clock.

Parameters:
[out] *real_time_clock* - pointer to CMEM to write the real time clock into

Returns:
uint32_t - 32 LSB of the real time clock

static __always_inline void ezdp_get_real_time_clock_async (struct ezdp_rtc __cmem * *real_time_clock*) [static]

Non blocking version of ezdp_get_real_time_clock().

Parameters:

[out] *real_time_clock* - pointer to CMEM to write the real time clock into

Note:

Call ezdp_sync() to wait for the operation to complete, confirming that the result is ready in CMEM.

Returns:

none

dpe/dp/include/ezdp_time_defs.h File Reference

Data Structures

- struct ezdp_rtc

ezdp_rtc struct for ezdp Defines

- #define EZDP_RTC_SEC_SIZE 32
- #define EZDP_RTC_SEC_OFFSET 0
- #define EZDP_RTC_SEC_WORD_SELECT 0
- #define EZDP_RTC_SEC_WORD_OFFSET 0
- #define EZDP_RTC_NSEC_SIZE 32
- #define EZDP_RTC_NSEC_OFFSET 32
- #define EZDP_RTC_NSEC_WORD_SELECT 1
- #define EZDP_RTC_NSEC_WORD_OFFSET 0
- #define EZDP_RTC_WORD_COUNT 2

Define Documentation

#define EZDP_RTC_SEC_SIZE 32

#define EZDP_RTC_SEC_OFFSET 0

#define EZDP_RTC_SEC_WORD_SELECT 0

#define EZDP_RTC_SEC_WORD_OFFSET 0

#define EZDP_RTC_NSEC_SIZE 32

#define EZDP_RTC_NSEC_OFFSET 32

#define EZDP_RTC_NSEC_WORD_SELECT 1

#define EZDP_RTC_NSEC_WORD_OFFSET 0

#define EZDP_RTC_WORD_COUNT 2

dpe/dp/include/ezdp_version.h File Reference

Data Structures

- struct ezdp_version

version info data structure Defines

- #define EZDP_VERSION_MAX_STRING_LENGTH 1024
- #define EZDP_VERSION_STR(s) #s
- #define EZDP_VERSION_XSTR(s) EZDP_VERSION_STR(s)
- #define _SCM_REV_ local
- #define EZDP_VERSION_DEF_PROJECT_NAME "NPS-400"
- #define EZDP_VERSION_DEF_MAJOR_VER 1
- #define EZDP_VERSION_DEF_MINOR_VER 9
- #define EZDP_VERSION_DEF_VERSION_CHAR 't'
- #define EZDP_VERSION_DEF_VERSION_STRING ""
- #define EZDP_VERSION_DEF_PATCH_MAJOR_VER 0
- #define EZDP_VERSION_DEF_PATCH_MINOR_VER 0
- #define EZDP_VERSION_DEF_PATCH_MICRO_VER 0
- #define ezdp_version_get_string(version_info_, version_string_)

Define Documentation

#define EZDP_VERSION_MAX_STRING_LENGTH 1024

#define EZDP_VERSION_STR(s) #s

#define EZDP_VERSION_XSTR(s) EZDP_VERSION_STR(s)

#define _SCM_REV_ local

#define EZDP_VERSION_DEF_PROJECT_NAME "NPS-400"

#define EZDP_VERSION_DEF_MAJOR_VER 1

#define EZDP_VERSION_DEF_MINOR_VER 9

#define EZDP_VERSION_DEF_VERSION_CHAR 't'

#define EZDP_VERSION_DEF_VERSION_STRING ""

#define EZDP_VERSION_DEF_PATCH_MAJOR_VER 0

#define EZDP_VERSION_DEF_PATCH_MINOR_VER 0

#define EZDP_VERSION_DEF_PATCH_MICRO_VER 0

#define ezdp_version_get_string(version_info_, version_string_)

Index