DOCA Driver APIs

Reference Manual

Table of Contents

Chapter 1. Change Log

This chapter list changes in API that were introduced to the library.

1.3.0

- Field Groups, GPU Groups, and field watches created with a handle returned from dcgmConnect() are now cleaned up upon disconnect. dcgmConnect_v2() can be used to get the old behavior of objects persisting after disconnect.
- dcgmConnect_v2() was added as a method for specifying additional connection options when connecting to the host engine.
- dcgmUnwatchFields() was added as a method of unwatching fields that were previously watched with dcgmWatchFields()
- dcgmActionValidate_v2() was added to be able to pass more parameters to the DCGM GPU Diagnostic.
- dcgmDiagResponse_t was increased from v2 to v3. See dcgmDiagResponse_v3 for details

1.2.3

- No API changes in this version.

1.1.1

- dcgmGetAllSupportedDevices() was added as a method to get DCGM-supported GPU Ids. dcgmGetAllDevices() can still be used to get all GPU Ids in the system.

1.0.0

- Initial Release.

Chapter 2. Modules

Here is a list of all modules:

- Host API
- Device API
- FlexIO Device Error
- FlexIO Queue Access

2.1. Host API

FlexIO host API.

typedef void (flexio_func_arg_pack_fn_t)

Callback function to pack the arguments for a function.

This function is called internally from the FlexIO runtime upon user making a call (e.g., flexio_process_call). It packs the arguments for a user function into the argument buffer provided in `argbuf`. The argument list can be arbitrarily long and is represented by `ap`. The correct usage of this function requires the caller to initialize the list using `va_start`.

flexio_status flexio_app_create (const char *name, void *elf, size_t elf_size, flexio_app **app)

Create a container for a FlexIO App.

Parameters

name

- Name to assign the app (used in discovery later). length of name should be up to FLEXIO_MAX_NAME_LEN bytes.

elf

- pointer to the ELF.

elf_size
- size of the elf.

app

Returns

flexio status value.

Description

This function creates a named app with a given ELF. The ELF pointer provided is copied into the library. It is called from within the constructor generated by the compiler.

flexio_status flexio_app_destroy (flexio_app *app)

Destroy a flexio app.

Parameters

app
- App that was created before.

Returns

flexio status value.

Description

This function destroys the state associated with the app and all registered functions. This function will free the internal elf buffer. It is called from within the destructor generated by the compiler.

flexio_status flexio_app_get_elf (flexio_app *app, uint64_t *bin_buff, size_t bin_size)

Retrieve ELF binary associated with application.

Parameters

app
- App that created before.

bin_buff
- Pointer to buffer to copy ELF binary.

bin_size
- Size of buffer pointed by bin_buff. If parameter is smaller than ELF binary size function will fail.

Returns

flexio status value.

Description

This function registers the function name, stub address with the runtime. Compiler calls this from within the constructor.

flexio_status flexio_app_get_list (flexio_appapp_list, uint32_t *num_apps)

Get a list of FlexIO Apps that are available.

Parameters

app_list
  - A list of apps that are available.
num_apps

Returns

flexio status value.

Description

This function returns a list of Flex IO apps that are loaded.

flexio_status flexio_app_list_free (flexio_app **apps_list)

Free the list of flexio apps.

Returns

flexio status value.

Description

This function frees the list of apps obtained from `flexio_app_get_list`.

flexio_status flexio_buf_dev_alloc (flexio_process *process, size_t buff_bsize, flexio_uintptr_t *dest_daddr_p)

Allocates a buffer on Flex IO heap memory.

Parameters

process
- A pointer to the Flex IO process context.

buff_bsize
- The size of the buffer to allocate.

dest_daddr_p
- A pointer to the Flex IO address, where the buffer was allocated.

Returns

flexio status value.

Description

This function allocates a buffer with the requested size on the Flex IO heap memory. On success - sets dest_daddr_p to the start address of the allocated buffer. On Failure - sets dest_daddr_p to 0x0.

flexio_status flexio_buf_dev_free (flexio_process *process, flexio_uintptr_t daddr)

Deallocates Flex IO heap memory buffer.

Parameters

process
- A pointer to the Flex IO process context.

daddr
- A pointer to an address of allocated memory on the Flex IO heap. Zero value is valid argument.

Returns

flexio status value.

Description

This function frees Flex IO heap memory buffer by address.

flexio_status flexio_buf_dev_memset (flexio_process *process, int value, size_t buff_bsize, flexio_uintptr_t dest_daddr)

Sets DPA heap memory buffer to a given value.

Parameters

process
- A pointer to the Flex IO process context.

value
- A value to set the DPA heap memory buffer to.

buff_bsize
- The size of the Flex IO heap memory buffer.

dest_daddr
- Flex IO heap memory buffer address to set.

Returns

flexio status value.

flexio_status flexio_copy_from_host (flexio_process *process, void *src_haddr, size_t buff_bsize, flexio_uintptr_t *dest_daddr_p)

Copy from host memory to Flex IO heap memory buffer.

Parameters

process
- A pointer to the Flex IO process context.

src_haddr
- An address of the buffer on the host memory.

buff_bsize
- The size of the buffer to copy.

dest_daddr_p
- A pointer to the Flex IO address, where the buffer was copied to.

Returns

flexio status value.

Description

This function copies data from a buffer on the host memory to the Flex IO memory. The function allocates memory on the device heap which dest_address points to. It is the caller responsibility to deallocate this memory when it is no longer used.

flexio_status flexio_cq_create (flexio_process *process, ibv_context *ibv_ctx, const flexio_cq_attr *fattr, flexio_cq **cq)

Creates a Flex IO CQ.

Parameters

process
- A pointer to the Flex IO process.

ibv_ctx
- A pointer to an IBV device context (might be different than process'). If NULL - process' will be used.

fattr
- A pointer to the CQ attributes struct.

cq
- A pointer to the created CQ context pointer.

Returns

flexio status value.

Description

This function creates a Flex IO CQ.

flexio_status flexio_cq_destroy (flexio_cq *cq)

Destroys a Flex IO CQ.

Parameters

cq
- A pointer to a CQ context.

Returns

flexio status value.

Description

This function destroys a Flex IO CQ.

flexio_status flexio_cq_modify_moderation (flexio_cq *cq, uint16_t max_count, uint16_t period, uint16_t mode)

Modifies a Flex IO CQ moderation configuration.

Parameters

cq
- A pointer to a CQ context.

max_count
- CQ moderation max count value.

period
- CQ moderation period value.

mode
- CQ moderation mode value.

Returns

flexio status value.

flexio_status flexio_cq_query_moderation (flexio_cq *cq, uint16_t *max_count, uint16_t *period, uint16_t *mode)

Queries a Flex IO CQ moderation configuration.

Parameters

cq
- A pointer to a CQ context.

max_count
- A pointer to the CQ moderation max count value.

period
- A pointer to the CQ moderation period value.

mode
- A pointer to the CQ moderation mode value.

Returns

flexio status value.

flexio_status flexio_device_mkey_create (flexio_process *process, flexio_mkey_attr *fattr, flexio_mkey **mkey)

Creates an Mkey to the process device UMEM.

Parameters

process
  - A pointer to the Flex IO process context.

fattr
  - A pointer to a Flex IO MKey attribute struct.

mkey
  - A pointer to a pointer to the created MKey struct.

Returns

flexio status value.

Description

This function creates an MKey over the provided PD for the provided process device UMEM. The mkey_id will point to the field in the containing flexio_mkey object.

flexio_status flexio_device_mkey_destroy (flexio_mkey *mkey)

destroys an MKey object containing the given ID

Parameters

mkey
  - A pointer to the Flex IO MKey to destroy. NULL is a valid value.

Returns

flexio status value.

Description

This function destroys an Mkey object containing the given ID.

flexio_status flexio_emu_db_to_cq_map (ibv_context *ibv_ctx, uint32_t emu_dev_vhca_id, uint32_t emu_dev_queue_id, flexio_cq *cq, flexio_emu_db_to_cq_ctx **emu_db_to_cq_ctx)

Mapping emulated devices doorbell to CQ.

Parameters

ibv_ctx
- A pointer to a device context.

emu_dev_vhca_id
- An emulated device VHCA ID.

emu_dev_queue_id
- A queue of the emulated device.

cq
- A pointer to the Flex IO CQ.

emu_db_to_cq_ctx
- A pointer to the created emulated device doorbell to CQ context.

Returns

flexio status value.

Description

This function maps emulated device doorbell to Flex IO CQ. A doorbell from the emulated device queue will create CQE on the specified CQ.

flexio_status flexio_emu_db_to_cq_unmap (flexio_emu_db_to_cq_ctx *emu_db_to_cq_ctx)

Unmapping emulated devices doorbell to CQ.

Returns

flexio status value.

Description

This function unmaps emulated device queue doorbell to Flex IO CQ.

flexio_status flexio_emulated_device_msix_create (flexio_process *process, ibv_context *target_ibv_ctx, uint8_t direct_mode, uint32_t target_uar_id, uint16_t msix_id, flexio_emu_device_msix_map *map, flexio_msix **msix)

Create emulated device msix.

Parameters

process
- A pointer to the emulation manager Flex IO process.

target_ibv_ctx
- A pointer to a SF device context.

direct_mode
- a flag to indicate EQ is to be used directly by app.

target_uar_id
- uar page id to be used for emulated cq created, relevant when direct_mode is false.

msix_id
- The msix id.

map
- A pointer to msix to eq mapping.

msix
- A pointer to receive the flexio_msix handle.

Returns

flexio status value.

Description

This function creates emulated device msix infrastructure.

flexio_status flexio_emulated_device_msix_destroy (flexio_msix *msix)

destroy emulated device msix.

Parameters

msix
- A pointer to the flexio_msix handle structure.

Returns

flexio status value.

Description

This function destroy emulated device msix infrastructure.

flexio_status flexio_emulated_device_msix_map_create (flexio_process *process, uint32_t device_emulation_id, uint16_t max_msix, flexio_emu_device_msix_map **map)

Query mapping of EQs to MSI-X vectors.

Parameters

process
  - A pointer to the Flex IO process.

device_emulation_id

max_msix
  - maximal msix to query, should not exceed FLEXIO_MAX_EQ_TO_MSIX_MAP_PAIRS

map
  - A pointer to receive list of EQs to MSI-X vectors mapping.

Returns

flexio status value.

Description

This function gets emulated devices VHCA ID and return mapping of EQs to MSI-X vectors for this emulated device.

flexio_status flexio_emulated_device_msix_map_free (flexio_emu_device_msix_map *map)

Free list of EQs to MSI-X vectors mapping.

Parameters

map
  - A point to list of EQs to MSI-X vectors mapping.

Returns

flexio status value.

Description

This function frees a list of EQs to MSI-X vectors mapping.

flexio_err_display (flexio_process *process)

Print error data to stderr.

Parameters

process
  - A pointer to the Flex IO process.

Description

User should call this function, once he got information, that an unrecoverable error in DPA has happened.

Calling this function will reset error status.

flexio_err_handler_fd (flexio_process *process)

Get file descriptor for error handler.

Parameters

process
  - A pointer to the Flex IO process.

Returns

- file descriptor.

Description

User should get fd in order to monitor for nonrecoverable errors

User can poll all created processes, using select/poll/epoll functions family.

flexio_err_status (flexio_process *process)

Check if unrecoverable error occurred.

Parameters

process
    - A pointer to the Flex IO process.

Returns

- nonzero value if error happen.

Description

User can call this function many times. Error status will be reset only after calling flexio_err_display();

It is suggested to check error status after every negotiation with DPA and periodically later. Alternative way - to poll file descriptor got from flexio_err_handler_fd() in order to get events about error

flexio_status flexio_event_handler_create (flexio_process *process, flexio_event_handler_attr *fattr, const flexio_window *window, const flexio_outbox *outbox, flexio_event_handler **event_handler_ptr)

Creates a Flex IO event handler.

Parameters

process
    - A pointer to the Flex IO process.
fattr
    - A pointer to the event handler attributes struct.
window
outbox
event_handler_ptr
    - A pointer to the created event handler context pointer.

Returns

flexio status value.

Description

This function creates a Flex IO event handler for an existing Flex IO process.

flexio_status flexio_event_handler_destroy (flexio_event_handler *event_handler)

Destroys a Flex IO event handler.

Parameters

event_handler

- A pointer to an event handler context.

Returns

flexio status value.

Description

This function destroys a Flex IO event handler.

flexio_status flexio_event_handler_run (flexio_event_handler *event_handler, uint64_t user_arg)

Run a Flex IO event handler.

Parameters

event_handler

- A pointer to an event handler context.

user_arg

- A 64 bit argument for the event handler's thread.

Returns

flexio status value.

Description

This function makes a Flex IO event handler start running.

flexio_status flexio_func_get_register_info (flexio_app *app, flexio_func_t *host_stub_func_addr, uint32_t *pup, char *dev_func_name, char *dev_unpack_func_name, size_t func_name_len, size_t *argbuf_size, flexio_func_arg_pack_fn_t **host_pack_func, flexio_uintptr_t *dev_func_addr, flexio_uintptr_t *dev_unpack_func_addr)

Obtain info for previously registered function.

Parameters

app

- FlexIO app.

host_stub_func_addr

- Known host stub func addr.

pup

- Whether function has been registered with pack/unpack support (0: No, 1:Yes).

dev_func_name

- Name of device function.

dev_unpack_func_name

- Name of unpack routine on device, NA if pup==0.

func_name_len

- Size of function name len allocated.

argbuf_size

- Size of argument buffer, NA if pup==0.

host_pack_func

- Function pointer to host packing routine, NA if pup==0.

dev_func_addr

- address of device function.

dev_unpack_func_addr

- address of device unpack function.

Returns

flexio status value.

Description

This function is used to obtain info about a previously registered function. It is used to compose higher-level libraries on top of DPACC / FlexIO interface. It is not intended to be used directly by the user.

The caller must ensure that the string pointers have been allocated and are at least `FLEXIO_MAX_NAME_LEN` long to ensure that the call doesn't fail due to insufficient space for the function name.

flexio_status flexio_func_pup_register (flexio_app *app, const char *dev_func_name, const char *dev_unpack_func_name, flexio_func_t *host_stub_func_addr, size_t argbuf_size, flexio_func_arg_pack_fn_t *host_pack_func)

Register a function name at application start.

Parameters

app
- App that created before.

dev_func_name
- The device function name (entry point).

dev_unpack_func_name
- The device wrapper function that unpacks the argument buffer.

host_stub_func_addr
- The host stub function that is used by the application to reference the device function.

argbuf_size
- Size of the argument buffer required by this function.

host_pack_func
- Host callback function that packs the arguments.

Returns

flexio status value.

Description

This function registers the function name, stub address with the runtime. It is called from within the constructor generated by the compiler.

flexio_status flexio_func_register (flexio_app *app, const char *dev_func_name, flexio_func_t **out_func)

Register a function to be used later.

Parameters

app
- previously created flexio app.

dev_func_name
- name of flexio function on device that will be called.

out_func
- opaque handle to use with flexio_process_call(), flexio_event_handler_create(), ...

Returns

flexio status value.

Description

This function is intended to be called directly by user in the situation where they don't desire pack/unpack support that is typically done by the compiler interface.

It is the user's responsibility to ensure that a function that was registered for RPC has the type: flexio_dev_rpc_handler_t and the function registered for event handler is: flexio_dev_event_handler_t. The runtime will not provide any type checking. A mismatched call, such as using the out_func of an Event handler for flexio_process_call() will result in undefined behavior.

flexio_status flexio_host2dev_memcpy (flexio_process *process, void *src_haddr, size_t buff_bsize, flexio_uintptr_t dest_daddr)

Copy from host memory to a pre-allocted Flex IO heap memory buffer.

Parameters

process
- A pointer to the Flex IO process context.

src_haddr
- An address of the buffer on the host memory.

buff_bsize
- The size of the buffer to copy.

dest_daddr
- Flex IO heap memory buffer address to copy to.

Returns

flexio status value.

Description

This function copies data from a buffer on the host memory to a buffer on the Flex IO heap memory.

flexio_status flexio_outbox_create (flexio_process *process, ibv_context *other_ctx, flexio_uar *uar, flexio_outbox **outbox)

Creates a Flex IO outbox.

Parameters

process
: - A pointer to the Flex IO process.

other_ctx
: - An IBV context for creating the PRM object (Different than process's on multi GVMI case, NULL or same as process's otherwise).

uar
: - A Flex IO UAR struct created for the outbox's IBV device.

outbox
: - A pointer to the created outbox context pointer.

Returns

flexio status value.

Description

This function Creates a Flex IO outbox for the given process.

flexio_status flexio_outbox_destroy (flexio_outbox *outbox)

Destroys a Flex IO outbox.

Parameters

outbox
: - A pointer to a outbox context.

Returns

flexio status value.

Description

This function destroys a Flex IO outbox.

flexio_status flexio_print_destroy (flexio_process *process)

Destroys a flexio print environment.

Parameters

process
  - A pointer to the Flex IO process.

Returns

flexio status value.

Description

This function destroys and releases all resources, allocated for process printing needs by flexio_print_init().

flexio_status flexio_print_flush (flexio_process *process)

Flush print buffer in case of asynchronous print.

Parameters

process
  - A pointer to the Flex IO process.

Returns

flexio status value.

Description

All data from print buffer will be flushed to file, definded in flexio_print_init().

In case of synchronous print this functions does nothing. This function allocates resources to support printing from Flex IO to HOST.

flexio_status flexio_print_init (flexio_process *process, flexio_uar *flexio_uar, size_t data_bsize, FILE *out, int is_async, pthread_t *ppthread)

Create environment to support print from DPA.

Parameters

process
- A pointer to the Flex IO process.

flexio_uar
- A pointer to a Flex IO UAR object created by caller for device side

data_bsize
- size of buffer, used for data transfer from Flex IO to HOST MUST be power of two and be at least 2Kb

out
- file to save data from Flex IO. Use stdout if you want receive data on HOST's console

is_async
- select between sync(recommended)/async modes

ppthread
- A pointer to receive phread ID of created thread. May be NULL if user doesn't need it.

Returns

flexio status value.

Description

This function allocates resources to support printing from Flex IO to HOST.

Print works in synchronous or asynchronous modes. In case of synchronous mode dedicated thread started to receive data and print it immediately. In case of asynchronous mode all print buffer will be flushed by flexio_print_flush(). Buffer can be overrun.

This function doesn't have "destroy" pair. All printing infrastructure will be closed and resources will be released from flexio_process_destroy() function.

flexio_status flexio_process_call (flexio_process *process, flexio_func_t *host_func, uint64_t *func_ret, ...)

Calls a Flex IO process.

Parameters

process
- A pointer to the Flex IO process to run.

host_func
- The host stub function that is used by the application to reference the device function.

func_ret
- A pointer to the ELF function return value.

Returns

flexio status value.

flexio_status flexio_process_create (ibv_context *ibv_ctx, flexio_app *app, const flexio_process_attr *process_attr, flexio_process **process_ptr)

Create a new Flex IO process.

Parameters

ibv_ctx
- A pointer to a device context.

app
- Device side application handle.

process_attr
- Optional, process attributes for create. Can be NULL.

process_ptr
- A pointer to the created process pointer.

Returns

flexio status value.

Description

This function creates a new Flex IO process with requested image.

flexio_status flexio_process_destroy (flexio_process *process)

Destroys a new Flex IO process.

Parameters

process
 - A pointer to a process.

Returns

flexio status value.

Description

This function destroys a new Flex IO process.

flexio_status flexio_process_error_handler_set (flexio_process *process, const char *handler)

Set the Flexio process error handler.

Parameters

process
 - A pointer to a process

handler
 - a C string of the function name in the dpa elf file image passed to flexio_process_create

Returns

flexio status value.

Description

This function sets the Flexio process error handler. The error handler must be set after the process is created, and before the first thread is created.

flexio_status flexio_qp_create (flexio_process *process, ibv_context *ibv_ctx, flexio_qp_attr *qp_fattr, flexio_qp **qp_ptr)

Creates a Flex IO QP.

Parameters

process
- A pointer to the Flex IO process.

ibv_ctx
- A pointer to an IBV device context (might be different than process'). If NULL - process' will be used.

qp_fattr
- A pointer to the QP attributes struct.

qp_ptr
- A pointer to the created QP context pointer.

Returns

flexio status value.

Description

This function creates a Flex IO QP.

flexio_status flexio_qp_destroy (flexio_qp *qp)

Destroys a Flex IO QP.

Parameters

qp
- A pointer to the QP context.

Returns

flexio status value.

Description

This function destroys a Flex IO QP.

flexio_status flexio_qp_modify (flexio_qp *qp, flexio_qp_attr *fattr, flexio_qp_attr_opt_param_mask *mask)

Modify Flex IO QP.

Parameters

qp
- A pointer to the QP context.

fattr
- A pointer to the QP attributes struct that will also define the QP connection.

mask
- A pointer to the optional QP attributes mask.

Returns

flexio status value.

Description

This function modifies Flex IO QP and transition it between states. At the end of the procedure Flex IO QP would have moved from it's current state to to next state, given in the fattr, if the move is a legal transition in the QP's state machine.

flexio_status flexio_rq_create (flexio_process *process, ibv_context *ibv_ctx, uint32_t cq_num, const flexio_wq_attr *fattr, flexio_rq **flexio_rq_ptr)

Creates a Flex IO RQ.

Parameters

process
- A pointer to the Flex IO process.

ibv_ctx
- A pointer to an IBV device context (might be different than process'). If NULL - process' will be used.

cq_num
- A CQ number.

fattr
- A pointer to the RQ attributes struct.

flexio_rq_ptr

Returns

flexio status value.

Description

This function creates a Flex IO RQ.

flexio_status flexio_rq_destroy (flexio_rq *flexio_rq)

Destroys a Flex IO RQ.

Returns

flexio status value.

Description

This function destroys a Flex IO RQ.

flexio_status flexio_sq_create (flexio_process *process, ibv_context *ibv_ctx, uint32_t cq_num, const flexio_wq_attr *fattr, flexio_sq **flexio_sq_ptr)

Creates a Flex IO SQ.

Parameters

process

- A pointer to the Flex IO process.

ibv_ctx

- A pointer to an IBV device context (might be different than process'). If NULL - process' will be used.

cq_num

- A CQ number (can be Flex IO or host CQ).

fattr

- A pointer to the SQ attributes struct.

flexio_sq_ptr

Returns

flexio status value.

Description

This function creates a Flex IO SQ.

flexio_status flexio_sq_destroy (flexio_sq *flexio_sq)

Destroys a Flex IO SQ.

Returns

flexio status value.

Description

This function destroys a Flex IO SQ.

flexio_status flexio_uar_create (flexio_process *process, mlx5dv_devx_uar *devx_uar, flexio_uar **flexio_uar)

Creates a Flex IO UAR object.

Parameters

process
- A pointer to the Flex IO process context.

devx_uar
- A pointer to a DevX UAR struct.

flexio_uar
- A pointer to a pointer to the created Flex IO UAR struct.

Returns

flexio status value.

Description

This function creates a Flex IO UAR object from a DevX UAR object.

flexio_status flexio_uar_destroy (flexio_uar *uar)

destroys a Flex IO UAR object

Parameters

uar

- A pointer to the Flex IO UAR to destroy.

Returns

flexio status value.

Description

This function destroys a Flex IO UAR object.

flexio_status flexio_window_create (flexio_process *process, ibv_pd *pd, flexio_window **window)

Creates a Flex IO window.

Parameters

process

- A pointer to the Flex IO process.

pd

- A pointer to a protection domain struct to the memory the window should access.

window

- A pointer to the created window context pointer.

Returns

flexio status value.

Description

This function Creates a Flex IO window for the given process.

flexio_status flexio_window_destroy (flexio_window *window)

Destroys a Flex IO window.

Parameters

window

- A pointer to a window context.

Returns

flexio status value.

Description

This function destroys a Flex IO window.

2.2. Device API

FlexIO device API.

typedef uint64_t (flexio_dev_arg_unpack_func_t)

Unpack the arguments and call the user function.

This callback function is used at runtime to unpack the arguments from the call on Host and then call the function on DPA. This function is called internally from flexio dev.

typedef void (flexio_dev_event_handler_t)

event handler callback function type.

Defines an event handler callback function. On handler function end, need to call flexio_dev_finish() instead of a regular return statement, in order to properly release resources back to the OS.

typedef void (flexio_dev_poll_thread_t)

poll thread handler callback function type.

Defines a poll thread handler callback function.

typedef uint64_t (flexio_dev_rpc_handler_t)

RPC handler callback function type.

Defines an RPC handler callback function.

flexio_dev_finish (void)

Exit flexio process (no errors).

Description

This function releases resources back to OS and returns '0x40' in dpa_process_status

flexio_dev_get_thread_ctx (flexio_dev_thread_ctx **dtctx)

Request thread context.

Parameters

dtctx

- A pointer to a pointer of flexio_dev_thread_ctx structure.

Returns

0 on success negative value on failure.

Description

This function requests the thread context. Should be called for every start of thread.

uint32_t flexio_dev_get_thread_id (flexio_dev_thread_ctx *dtctx)

Get thread ID from thread context.

Parameters

dtctx

- A pointer to a flexio_dev_thread_ctx structure.

Returns

thread ID value.

Description

This function queries a thread context for its thread ID (from thread metadata).

flexio_uintptr_t flexio_dev_get_thread_local_storage (flexio_dev_thread_ctx *dtctx)

Get thread local storage address from thread context.

Parameters

dtctx
- A pointer to a flexio_dev_thread_ctx structure.

Returns

thread local storage value.

Description

This function queries a thread context for its thread local storage (from thread metadata).

flexio_dev_status_t flexio_dev_outbox_config (flexio_dev_thread_ctx *dtctx, uint16_t outbox_config_id)

Config thread outbox object.

Parameters

dtctx
- A pointer to flexio_dev_thread_ctx structure.

outbox_config_id
- The outbox object config id.

Returns

flexio_dev_status_t.

Description

This function updates the thread outbox object of the given thread context.

flexio_dev_print (const char *format, ...)

Same as a regular printf but with protection from simultaneous print from different threads.

Returns

- same as from regular printf.

Description

[in] - same as for regular printf.

flexio_dev_puts (flexio_dev_thread_ctx *dtctx, char *str)

Put a string to printing queue.

Parameters

dtctx
 - A pointer to a pointer of flexio_dev_thread_ctx structure.

str
 - A pointer to string

Returns

length of printed string.

Description

This function puts a string to host printing queue. This queue has been serviced by host application. Would have no effect, if the host application didn't configure printing environment. In order to initalize/configure printing envoirment - On HOST side - after flexio_process_create, flexio_print_init should be called. On DEV side - before using flexio_dev_puts, the thread context is needed, therefore flexio_dev_get_thread_ctx should be called before.

flexio_dev_reschedule (void)

Exit from a thread, leave process active.

Description

This function releases resources back to OS. For the next DUAR the thread will restart from the beginning.

flexio_dev_status_t flexio_dev_window_config (flexio_dev_thread_ctx *dtctx, uint16_t window_config_id, uint32_t mkey)

Config thread window object.

Parameters

dtctx

- A pointer to a flexio_dev_thread_ctx structure.

window_config_id

- The window object id.

mkey

- mkey object.

Returns

flexio_dev_status_t.

Description

This function updates the thread window object of the given thread context.

flexio_dev_status_t flexio_dev_window_copy_from_host (flexio_dev_thread_ctx *dtctx, void *daddr, uint64_t haddr, uint32_t size)

Copy a buffer from host memory to device memory.

Parameters

dtctx

- A pointer to a flexio_dev_thread_ctx structure.

daddr

- A pointer to the device memory buffer.

haddr

- A pointer to the host memory allocated buffer.

size

- Number of bytes to copy.

Returns

flexio_dev_status_t.

Description

This function copies specified number of bytes from host memory to device memory. UNSUPPORTED at this time.

flexio_dev_status_t flexio_dev_window_copy_to_host (flexio_dev_thread_ctx *dtctx, uint64_t haddr, const void *daddr, uint32_t size)

Copy a buffer from device memory to host memory.

Parameters

dtctx

- A pointer to a flexio_dev_thread_ctx structure.

haddr

- A pointer to the host memory allocated buffer.

daddr

- A pointer to the device memory buffer.

size

- Number of bytes to copy.

Returns

flexio_dev_status_t.

Description

This function copies specified number of bytes from device memory to host memory.

flexio_dev_status_t flexio_dev_window_mkey_config (flexio_dev_thread_ctx *dtctx, uint32_t mkey)

Config thread window mkey object.

Parameters

dtctx

- A pointer to a flexio_dev_thread_ctx structure.

mkey

- mkey object.

Returns

flexio_dev_status_t.

Description

This function updates the thread window mkey object of the given thread context.

flexio_dev_status_t flexio_dev_window_ptr_acquire (flexio_dev_thread_ctx *dtctx, uint64_t haddr, flexio_uintptr_t **daddr)

Generate device address from host allocated memory.

Parameters

dtctx
- A pointer to a flexio_dev_thread_ctx structure.

haddr
- Host allocated address.

daddr
- A pointer to write the device generated matching address.

Returns

flexio_dev_status_t.

Description

This function generates a memory address to be used by device to access host side memory, according to already create window object. from a host allocated address.

#define spin_init __atomic_exchange_n(&((lock)->locked), 0, __ATOMIC_SEQ_CST)

Initialize a spinlock mechanism.

Initialize a spinlock mechanism, must be called before use.

#define spin_lock do { \ uint32_t __old_val; \ do { \ __old_val = 0; \ } while (!(__atomic_compare_exchange_n(&((lock)->locked), &__old_val, 1, 0, \ __ATOMIC_SEQ_CST, __ATOMIC_SEQ_CST))); \ } while (0)

Lock a spinlock mechanism.

Lock a spinlock mechanism.

#define spin_unlock __atomic_exchange_n(&((lock)->locked), 0, __ATOMIC_SEQ_CST)

Unlock a spinlock mechanism.

Unlock a spinlock mechanism.

2.3. FlexIO Device Error

FlexIO Device Error definitions.

__attribute__ ((__noreturn__))

Exit the process and return a user (fatal) error code.

Description

Error codes returned to the host in the dpa_process_status field of the DPA_PROCESS object are defined as follows: 0: OK 1-63: RTOS or Firmware errors 64-127: Flexio-SDK errors 129-255: User defined

uint64_t flexio_dev_get_and_rst_errno (flexio_dev_thread_ctx *dtctx)

Get and Reset thread error flag (errno) of recoverable (non fatal) error.

Parameters

dtctx

- A pointer to a flexio_dev_thread_ctx structure.

Returns

- void.

uint64_t flexio_dev_get_errno (flexio_dev_thread_ctx *dtctx)

Get thread error flag (errno) of recoverable (non fatal) error.

Parameters

dtctx

- A pointer to a flexio_dev_thread_ctx structure.

Returns

thread error code.

Description

This function queries an errno field from thread context.

flexio_dev_rst_errno (flexio_dev_thread_ctx *dtctx)

Reset thread error flag (errno) of recoverable (non fatal) error.

Parameters

dtctx

- A pointer to a flexio_dev_thread_ctx structure.

2.4. FlexIO Queue Access

FlexIO queue access definitions.

flexio_dev_status_t flexio_dev_cq_arm (flexio_dev_thread_ctx *dtctx, uint32_t ci, uint32_t qnum)

Arm CQ function.

Parameters

dtctx

- A pointer to a pointer of flexio_dev_thread_ctx structure.

ci
- Current CQ consumer index.

qnum
- Number of the CQ to arm.

Returns

flexio_dev_status_t.

Description

Moves a CQ to 'armed' state. This means that next CQE created for this CQ will result in an EQE on the relevant EQ.

uint32_t flexio_dev_cqe_get_byte_cnt (flexio_dev_cqe64 *cqe)

Get byte count field from CQE function.

Parameters

cqe
- CQE to parse.

Returns

uint32_t - Byte count field value of the CQE.

Description

Parse a CQE for its byte count field.

uint8_t flexio_dev_cqe_get_opcode (flexio_dev_cqe64 *cqe)

Get the opcode field from CQE function.

Parameters

cqe
- CQE to parse.

Returns

uint8_t - Opcode field value of the CQE.

uint8_t flexio_dev_cqe_get_owner (flexio_dev_cqe64 *cqe)

Get owner field from CQE function.

Parameters

cqe
- CQE to parse.

Returns

uint8_t - Owner field value of the CQE.

Description

Parse a CQE for its owner field.

uint32_t flexio_dev_cqe_get_qpn (flexio_dev_cqe64 *cqe)

Get QP number field from CQE function.

Parameters

cqe
- CQE to parse.

Returns

uint32_t - QP number field value of the CQE.

Description

Parse a CQE for its QP number field.

uint16_t flexio_dev_cqe_get_wqe_counter (flexio_dev_cqe64 *cqe)

Get WQE counter filed from CQE function.

Parameters

cqe
- CQE to parse.

Returns

uint16_t - WQE counter field value of the CQE.

Description

Parse a CQE for its WQE counter field.

flexio_dev_status_t flexio_dev_db_ctx_arm (flexio_dev_thread_ctx *dtctx, uint32_t cqn, uint32_t emu_ctx_id)

arm the emulation context

Parameters

dtctx
- A pointer to a pointer of flexio_dev_thread_ctx structure.

cqn
- CQ number provided by host.

emu_ctx_id
- Emulation context ID, provided by a call on the host to flexio_emu_db_to_cq_ctx_get_id.

flexio_dev_status_t flexio_dev_db_ctx_force_trigger (flexio_dev_thread_ctx *dtctx, uint32_t cqn, uint32_t emu_ctx_id)

force trigger of emulation context

Parameters

dtctx

cqn
- CQ number provided by host.

emu_ctx_id
- Emulation context ID, provided by a call on the host to flexio_emu_db_to_cq_ctx_get_id.

flexio_dev_status_t flexio_dev_dbr_cq_set_ci (uint32_t *cq_dbr, uint32_t ci)

Set consumer index value for a CQ function.

Parameters

cq_dbr
- A pointer to the CQ's doorbell record address.

ci
- The consumer index value to update.

Returns

flexio_dev_status_t.

Description

Writes an updated consumer index number to a CQ's doorbell record

flexio_dev_status_t flexio_dev_dbr_rq_inc_pi (uint32_t *rq_dbr)

Increment producer index of an RQ by 1 function.

Parameters

rq_dbr
- A pointer to the CQ's doorbell record address.

Returns

flexio_dev_status_t.

Description

Mark a WQE for reuse by incrementing the relevant RQ producer index by 1

flexio_dev_status_t flexio_dev_dbr_sq_set_pi (uint32_t *sq_dbr, uint32_t pi)

Set producer index value for an SQ function.

Parameters

sq_dbr
- A pointer to the SQ's doorbell record address.

pi
- The producer index value to update.

Returns

flexio_dev_status_t.

Description

Writes an updated producer index number to an SQ's doorbell record

flexio_dev_status_t flexio_dev_eq_update_ci (flexio_dev_thread_ctx *dtctx, uint32_t ci, uint32_t qnum)

Update an EQ consumer index function.

Parameters

dtctx
- A pointer to a pointer of flexio_dev_thread_ctx structure.

ci
- Current EQ consumer index.

qnum
- Number of the EQ to update.

Returns

flexio_dev_status_t.

Description

Updates the consumer index of an EQ after handling an EQE.

uint32_t flexio_dev_eqe_get_cqn (flexio_dev_eqe *eqe)

Get CQ number field from EQE function.

Parameters

eqe

- EQE to parse.

Returns

uint32_t - CQ number field value of the EQE.

Description

Parse an EQE for its CQ number field.

uint8_t flexio_dev_eqe_get_owner (flexio_dev_eqe *eqe)

Get owner field from EQE function.

Parameters

eqe

- EQE to parse.

Returns

uint32_t - owner field value of the EQE.

Description

Parse an EQE for its owner field.

flexio_dev_status_t flexio_dev_msix_send (flexio_dev_thread_ctx *dtctx, uint32_t cqn)

Send msix on the cq linked to the msix eq.

Parameters

dtctx

- A pointer to a flexio_dev_thread_ctx structure.

cqn
- CQ number to trigger db on. Trigger is done via currently configured outbox, this can be changed with outbox config API according to CQ.

Returns

flexio_dev_status_t.

Description

This function trigger msix on the given cq.

flexio_dev_status_t flexio_dev_qp_sq_ring_db (flexio_dev_thread_ctx *dtctx, uint16_t pi, uint32_t qnum)

QP/SQ ring doorbell function.

Parameters

dtctx
- A pointer to a pointer of flexio_dev_thread_ctx structure.

pi
- Current queue producer index.

qnum
- Number of the queue to update.

Returns

flexio_dev_status_t.

Description

Rings the doorbell of a QP or SQ in order to alert the HW of pending work.

void *flexio_dev_rwqe_get_addr (flexio_dev_wqe_data_seg *rwqe)

Get address field from receive WQE function.

Parameters

rwqe
- WQE to parse.

Returns

void* - Address field value of the receive WQE.

Description

Parse a receive WQE for its address field.

flexio_dev_status_t flexio_dev_swqe_seg_atomic_set (flexio_dev_sqe_seg *swqe, uint64_t swap_or_add_data, uint64_t compare_data)

Fill out an Atomic send queue wqe segment function.

Parameters

swqe

- Send WQE segment to fill.

swap_or_add_data

- The data that will be swapped in or the data that will be added.

compare_data

- The data that will be compared with. Unused in fetch & add operation.

Returns

flexio_dev_status_t.

Description

Fill the fields of a send WQE segment (2 DWORDs) with Atomic segment information. This segment can service a compare & swap or fetch & add operation.

flexio_dev_status_t flexio_dev_swqe_seg_ctrl_set (flexio_dev_sqe_seg *swqe, uint32_t sq_pi, uint32_t sq_number, uint32_t ce, flexio_ctrl_seg_t ctrl_seg_type)

Fill out a control send queue wqe segment function.

Parameters

swqe

- Send WQE segment to fill.

sq_pi
- Producer index of the send WQE.

sq_number
- SQ number that holds the WQE.

ce
- wanted CQ policy for CQEs. Value is taken from cq_ce_mode enum.

ctrl_seg_type
- Type of control segment.

Returns

flexio_dev_status_t.

Description

Fill the fields of a send WQE segment (4 DWORDs) with control segment information. This should always be the 1st segment of the WQE.

flexio_dev_status_t flexio_dev_swqe_seg_data_set (flexio_dev_sqe_seg *swqe, uint32_t data_sz, uint32_t lkey, uint64_t data_addr)

Fill out a data send queue wqe segment function.

Parameters

swqe
- Send WQE segment to fill.

data_sz
- Size of the data.

lkey
- Local memory access key for the data operation.

data_addr
- Address of the data for the data operation.

Returns

flexio_dev_status_t.

Description

Fill the fields of a send WQE segment (4 DWORDs) with data segment information.

flexio_dev_status_t flexio_dev_swqe_seg_eth_set (flexio_dev_sqe_seg *swqe, uint16_t cs_swp_flags, uint16_t mss, uint16_t inline_hdr_bsz, uint8_t inline_hdrs)

Fill out an ethernet send queue wqe segment function.

Parameters

swqe
- Send WQE segment to fill.

cs_swp_flags
- Flags for checksum and swap, see PRM section 8.9.4.2, Send WQE Construction Summary.

mss
- Maximum Segment Size - For LSO WQEs - the number of bytes in the TCP payload to be transmitted in each packet. Must be 0 on non LSO WQEs.

inline_hdr_bsz
- Length of inlined packet headers in bytes. This includes the headers in the inline_data segment as well.

inline_hdrs
- First 2 bytes of the inlined packet headers.

Returns

flexio_dev_status_t.

Description

Fill the fields of a send WQE segment (4 DWORDs) with Ethernet segment information.

flexio_dev_status_t flexio_dev_swqe_seg_rdma_set (flexio_dev_sqe_seg *swqe, uint32_t rkey, uint64_t data_addr)

Fill out an RDMA send queue wqe segment function.

Parameters

swqe
- Send WQE segment to fill.

rkey
- Remote memory access key for the RDMA operation.

data_addr

- Address of the data for the RDMA operation.

Returns

flexio_dev_status_t.

Description

Fill the fields of a send WQE segment (4 DWORDs) with RDMA segment information.

NVIDIA Corporation | 2788 San Tomas Expressway, Santa Clara, CA 95051
http://www.nvidia.com