Linux Software (DM75xx)

Generated by Doxygen 1.8.8

Thu Apr 28 2016 08:55:52

Contents

Chapter 1

Module Index

1.1 Modules

Here is a list of all modules:

Chapter 2

Data Structure Index

2.1 Data Structures

Here are the data structures with brief descriptions:

Chapter 3

File Index

3.1 File List

Here is a list of all documented files with brief descriptions:

Chapter 4

Module Documentation

4.1 DM75xx driver header file

Modules

- DM75xx driver enumerations
- DM75xx driver macros
- DM75xx driver structures
- DM75xx driver forward declarations
- DM75xx driver functions

4.1.1 Detailed Description

4.2 DM75xx driver enumerations

Typedefs

- typedef enum
  dm75xx_pci_region_access_dir dm75xx_pci_region_access_dir_t

  Standard PCI region access direction type.

Enumerations

- enum dm75xx_pci_region_access_dir { DM75xx_PCI_REGION_ACCESS_READ = 0, DM75xx_PCI_REG↩
  ION_ACCESS_WRITE }

  Direction of access to standard PCI region.

4.2.1 Detailed Description

4.2.2 Enumeration Type Documentation

4.2.2.1 enum **dm75xx_pci_region_access_dir**

Direction of access to standard PCI region.

Enumerator

DM75xx_PCI_REGION_ACCESS_READ	Read from the region
DM75xx_PCI_REGION_ACCESS_WRITE	Write to the region

Definition at line 54 of file dm75xx_driver.h.

4.3 DM75xx driver macros

Macros

- #define DM75xx_DEVICE_NAME_LENGTH 22

    Maximum number of characters in device's name.
- #define DM7520_PCI_DEVICE_ID 0x7520

    DM7520 PCI device ID.
- #define DM7540_PCI_DEVICE_ID 0x7540

    DM7540 PCI device ID.
- #define RTD_PCI_VENDOR_ID 0x1435

    RTD Embedded Technologies PCI vendor ID.
- #define DM75xx_PCI_REGIONS PCI_ROM_RESOURCE

    Number of standard PCI regions.
- #define DM75xx_DMA_CHANNELS 2

    Number of FIFO channels per device.
- #define DM75xx_MAX_DMA_BUFFER_SIZE 0x20000

    Maximum size in bytes of any DMA buffer.
- #define DM75xx_INT_QUEUE_SIZE 0x10

    Maximum size in entries of the interrupt status queue.

4.3.1 Detailed Description

DM75xx_Driver_Enumerations

4.3.2 Macro Definition Documentation

4.3.2.1 #define DM75xx_MAX_DMA_BUFFER_SIZE 0x20000

Maximum size in bytes of any DMA buffer.

Note

> Be aware that the probability of DMA buffer allocation failure increases as the buffer size increases.
> If this default value does not suit your needs, you can change it and then recompile the driver.
> The max buffer size is set to 128k to remain architecture independent. it is more than likely that much more than this can be allocated on an x86 system at one time.

Definition at line 149 of file dm75xx_driver.h.

4.4 DM75xx driver structures

Data Structures

- struct dm75xx_pci_region

  DM75xx PCI region descriptor. This structure holds information about one of a device's PCI memory regions.
- struct dm75xx_dma_chain_descriptor

  Dm75xx DMA chaining descriptor.
- struct dm75xx_dma_descriptor

  DM75xx DMA buffer descriptor. This structure holds allocation information for a single DMA buffer.
- struct dm75xx_device_descriptor

  DM75xx device descriptor. This structure holds information about a device needed by the kernel.

Typedefs

- typedef struct dm75xx_pci_region dm75xx_pci_region_t

  DM75xx PCI region descriptor type.
- typedef struct dm75xx_dma_chain_descriptor dm75xx_dma_chain_descriptor_t

  DM75xx DMA Chaining descriptor type.
- typedef struct dm75xx_dma_descriptor dm75xx_dma_descriptor_t

  DM75xx DMA buffer descriptor type.
- typedef struct dm75xx_device_descriptor dm75xx_device_descriptor_t

  DM75xx device descriptor type.

4.4.1 Detailed Description

DM75xx_Driver_Macros

4.5 DM75xx driver forward declarations

Variables

- static struct file_operations dm75xx_file_ops

  File operations supported by driver.

4.5.1 Detailed Description

DM75xx_Driver_Structures

4.6 DM75xx driver functions

Functions

- static void dm75xx_access_pci_region (const dm75xx_device_descriptor_t ∗dm75xx, dm75xx_pci_access↩_request_t ∗pci_request, dm75xx_pci_region_access_dir_t direction)

  Read from or write to one of the standard PCI regions.
- static int dm75xx_allocate_irq (dm75xx_device_descriptor_t ∗dm75xx, const struct pci_dev ∗pci_device)

  Allocate an interrupt line for a DM75xx device.
- static void dm75xx_enable_plx_interrupts (const dm75xx_device_descriptor_t ∗dm75xx, uint8_t enable)

  Enable PLX interrupts for the specified DM75xx Device.
- static void dm75xx_enable_plx_dma (const dm75xx_device_descriptor_t ∗dm75xx, dm75xx_dma_channel↩_t channel)

  Configure PLX Mode register for the specified DMA Channel.
- static void dm75xx_get_pci_master_status (dm75xx_device_descriptor_t ∗dm75xx, uint8_t ∗pci_master)

  Determine whether or not a device is PCI master capable.
- static void dm75xx_initialize_device_descriptor (dm75xx_device_descriptor_t ∗dm75xx)

  Initialize the device descriptor for the specified DM75xx device.
- static void dm75xx_initialize_hardware (const dm75xx_device_descriptor_t ∗dm75xx)

  Initialize the specified DM75xx device.
- INTERRUPT_HANDLER_TYPE dm75xx_interrupt_handler (int irq_number, void ∗device_id)

  DM75xx device interrupt handler.
- static long dm75xx_ioctl (struct file ∗file, unsigned int request_code, unsigned long ioctl_param)

  Process ioctl(2) system calls directed toward a DM75xx device file.
- static void dm75xx_board_reset (dm75xx_device_descriptor_t ∗dm75xx)

  Performs a reset of the board and device descriptor.
- static void dm75xx_interrupt_enable (dm75xx_device_descriptor_t ∗dm75xx, dm75xx_int_source_t source, uint8_t enable)

  Performs the actual enable/disable of the interrupt sources.
- static int dm75xx_interrupt_control (dm75xx_device_descriptor_t ∗dm75xx, unsigned long ioctl_param)

  Control the interrupts on the boards. This includes enabling, disabling and checking the enable/disable status of the interrupts.
- static int dm75xx_get_interrupt (dm75xx_device_descriptor_t ∗dm75xx, unsigned long ioctl_param)

  Returns the top entry from the interrupt status queue.
- static void dm75xx_put_interrupt (dm75xx_device_descriptor_t ∗dm75xx, uint32_t interrupt)

  Adds an interrupt to the interrupt status queue.
- static int dm75xx_service_dma_function (dm75xx_device_descriptor_t ∗dm75xx, unsigned long ioctl_param)

  Process user space DMA function requests.
- static int dm75xx_dma_abort (dm75xx_device_descriptor_t ∗dm75xx, dm75xx_dma_channel_t channel)

  Aborts any DMA transfers on the given channel.
- static int dm75xx_dma_alloc_buffer (dm75xx_device_descriptor_t ∗dm75xx, dm75xx_dma_channel_t channel)

  Allocates a coherent and consistent buffer for our DMA operations.
- static int dm75xx_dreq_init (dm75xx_device_descriptor_t ∗dm75xx, dm75xx_dma_channel_t channel, dm75xx_dma_request_t dreq)

  Performs some DMA initialization work based on the DREQ source.
- static int dm75xx_dma_initialize (dm75xx_device_descriptor_t ∗dm75xx, dm75xx_ioctl_argument_t ∗ioctl_↩argument)

  Initialize DMA for the specified channel and source for the DM75xx device.
- static void dm75xx_free_dma_mappings (dm75xx_device_descriptor_t ∗dm75xx, dm75xx_dma_channel_↩t channel)

  Free all coherent/consistent DMA mappings for the given DMA channel on the specified DM75xx device.

- int dm75xx_load (void)

  Perform all actions necessary to initialize the DM75xx driver and devices.
- static int dm75xx_modify_pci_region (dm75xx_device_descriptor_t ∗dm75xx, unsigned long ioctl_param)

  Read an unsigned value from one of a device's PCI regions, modify certain bits in the value, and then write it back to the region.
- static int dm75xx_open (struct inode ∗inode, struct file ∗file)

  Prepare a DM75xx device file to be opened and used.
- static unsigned int dm75xx_poll (struct file ∗file, struct poll_table_struct ∗poll_table)

  Determine whether or not a DM75xx device is readable. This function supports the poll(2) and select(2) system calls.
- static int dm75xx_probe_devices (uint32_t ∗device_count, dm75xx_device_descriptor_t ∗∗device_←
descriptors)

  Probe and set up all DM75xx devices.
- static int dm75xx_process_pci_regions (dm75xx_device_descriptor_t ∗dm75xx, const struct pci_dev ∗pci_←
device)

  For each of the standard PCI regions, get the region's base address and length from kernel PCI resource information set up at boot. Also, remap any memory-mapped region into the kernel's virtual address space.
- static int dm75xx_register_char_device (int ∗major)

  Register the DM75xx character device and request dynamic allocation of a character device major number.
- static int dm75xx_release (struct inode ∗inode, struct file ∗file)

  Do all processing necessary after the last reference to a DM75xx device file is released elsewhere in the kernel.
- static void dm75xx_release_resources (void)

  Release any resources allocated by the driver.
- void dm75xx_unload (void)

  Perform all actions necessary to deinitialize the DM75xx driver and devices.
- static int dm75xx_unregister_char_device (void)

  Unregister the DM75xx character device and free the character device major number.
- static int dm75xx_validate_device (const dm75xx_device_descriptor_t ∗dm75xx)

  Given what is assumed to be the address of a DM75xx device descriptor, make sure it corresponds to a valid DM75xx device descriptor.
- static int dm75xx_validate_pci_access (const dm75xx_device_descriptor_t ∗dm75xx, const dm75xx_pci_←
access_request_t ∗pci_request)

  Validate a user-space access to one of the device's PCI regions.
- static int dm75xx_get_fifo_size (dm75xx_device_descriptor_t ∗dm75xx, unsigned int ∗size)

  Measure the size of the fifo by filling it until it is half-full than doubling that value to get the size of the fifo.

4.6.1 Detailed Description

DM75xx_Driver_Forward_Declarations

4.6.2 Function Documentation

4.6.2.1 static void dm75xx_access_pci_region (const **dm75xx_device_descriptor_t** ∗ *dm75xx,* **dm75xx_pci_access_request_t** ∗ *pci_request,* **dm75xx_pci_region_access_dir_t** *direction*) `[static]`

Read from or write to one of the standard PCI regions.

Parameters

dm75xx	Address of device's DM75xx device descriptor.
pci_request	Address of access' PCI request descriptor.
direction	Direction of access to PCI region (read from or write to).

Warning

This function performs no validation on its arguments. All arguments are assumed correct.

4.6.2.2 static int dm75xx_allocate_irq (dm75xx_device_descriptor_t ∗ *dm75xx,* const struct pci_dev ∗ *pci_device*) [static]

Allocate an interrupt line for a DM75xx device.

Parameters

dm75xx	Address of device's DM75xx device descriptor.
pci_device	Address of kernel's PCI device structure for the current DM75xx device.

Return values

0	Success.
$<$	0

Failure.

The following values may be returned:

- `−EBUSY` The interrupt line is allocated to another device which requested it as unsharable; returned by request_irq().
- `−EINVAL` The interrupt line is not valid; returned by request_irq().
- `−EINVAL` No interrupt handler is to be associated with the requested interrupt line; returned by request_irq().
- `−ENOMEM` Memory for interrupt action descriptor could not be allocated; returned by request_irq().

Note

On failure, this function will clean up by releasing any resources allocated by the driver to this point.

4.6.2.3 static void dm75xx_board_reset (dm75xx_device_descriptor_t ∗ *dm75xx*) [static]

Performs a reset of the board and device descriptor.

Parameters

dm75xx	Address of the device's DM75xx device descriptor

4.6.2.4 static int dm75xx_dma_abort (dm75xx_device_descriptor_t ∗ *dm75xx,* dm75xx_dma_channel_t *channel*) [static]

Aborts any DMA transfers on the given channel.

Parameters

dm75xx	Address of the device's DM75xx device descriptor.
channel	The DMA channel on which to cancel any transfers.

Return values

0	Success
$<$	0

Failure.

The following values may be returned:

- `-EINVAL` The channel entered is invalid

4.6.2.5 static int dm75xx_dma_alloc_buffer (**dm75xx_device_descriptor_t** $*$ ***dm75xx,*** **dm75xx_dma_channel_t** ***channel***) `[static]`

Allocates a coherent and consistent buffer for our DMA operations.

Parameters

dm75xx	Address of the device's DM75xx device descriptor.
channel	The DMA channel for which to allocate a buffer.

Return values

0	Success
$<$	0

Failure.

The following values may be returned:

- `-ENOMEM` Failed to allocate DMA buffer

  Note

  The buffer allocated by this function will be mapped to user space.
  The pages allocated by this function will be reserved in 2.4 kernel. This is done to allow successful userspace mapping.

4.6.2.6 static int dm75xx_dma_initialize (**dm75xx_device_descriptor_t** $*$ ***dm75xx,*** **dm75xx_ioctl_argument_t** $*$ ***ioctl_argument***) `[static]`

Initialize DMA for the specified channel and source for the DM75xx device.

Parameters

dm75xx	Address of the device's DM75xx device descriptor.
ioctl_argument	Address of the kerne's ioctl() request structure.

Return values

0	Success
$<$	0

Failure.

The following values may be returned:

- `-EAGAIN` DMA has already been initialized.

- `-ENOMEM` Failed to allocate page for DMA chain descriptors.
- `-EOPNOTSUPP` The device is not PCI master capable.
- `-EINVAL` Invalid buffer size requested.
- `-EINVAL` Invalid DMA channel.
- `-EINVAL` Invalid DREQ source.

Please see the descriptions of dm75xx_dreq_init(), and dm75xx_dma_alloc_buffer() for information on other possible values returned in this case.

Note

When initializing DMA, this function: 1) allocates coherent/consistent DMA mappings, 2) allocates memory to store DMA buffer allocation information, 3) allocates memory to link DMA buffers into device's DMA buffer list, 4) links all DMA buffers into the device's DMA buffer list, 5) allocates memory to link DMA buffers in device's free DMA buffer list, and 6) links all DMA buffers into the device's free DMA buffer list.
Factors beyond the number and size of DMA buffers affect the probability of DMA buffer allocation failure. These factors include the number of processes on the system, how much system memory is already in use, and the presence of processes (such as the X server) which use a lot of memory.
System memory can be a scarce resource Every system entity needs some amount of memory. Memory is being allocated and released all the time.

4.6.2.7 static int dm75xx_dreq_init (**dm75xx_device_descriptor_t** ∗ *dm75xx*, **dm75xx_dma_channel_t** *channel*, **dm75xx_dma_request_t** *dreq*) `[static]`

Performs some DMA initialization work based on the DREQ source.

Parameters

dm75xx	Address of the device's DM75xx device descriptor.
channel	The DMA Channel to perform the initialization for.
dreq	The selected DREQ source.

Return values

0	Success
$<$	0

Failure.

The following values may be returned:

- `-EINVAL` Invalid DREQ source

4.6.2.8 static void dm75xx_enable_plx_dma (const **dm75xx_device_descriptor_t** ∗ *dm75xx*, **dm75xx_dma_channel_t** *channel*) `[static]`

Configure PLX Mode register for the specified DMA Channel.

Parameters

dm75xx	Address of the devices' DM75xx Device Descriptor.
channel	The DMA Channel to configure.

4.6.2.9 static void dm75xx_enable_plx_interrupts (const dm75xx_device_descriptor_t ∗ *dm75xx*, uint8_t *enable*) [static]

Enable PLX interrupts for the specified DM75xx Device.

Parameters

dm75xx	Address of the devices' DM75xx Device Descriptor.
enable	Flag indicating whether or not PLX interrupts should be enabled. A value of zero indicates disable and any other value indicates enable.

4.6.2.10 static void dm75xx_free_dma_mappings (dm75xx_device_descriptor_t ∗ *dm75xx*, dm75xx_dma_channel_t *channel*) [static]

Free all coherent/consistent DMA mappings for the given DMA channel on the specified DM75xx device.

Parameters

dm75xx	Address of the device's DM75xx device descriptor.
channel	The DMA channel for which to free the DMA mappings.

Note

This function also frees the memory allocated to manage the DMA buffer allocation information and the DMA buffer lists.

4.6.2.11 static int dm75xx_get_fifo_size (dm75xx_device_descriptor_t ∗ *dm75xx*, unsigned int ∗ *size*) [static]

Measure the size of the fifo by filling it until it is half-full than doubling that value to get the size of the fifo.

Parameters

dm75xx	Address of the device descriptor.
size	Address of the variable to store the size once it is found

Return values

0	Success.
$<$	0

Failure

4.6.2.12 static int dm75xx_get_interrupt (dm75xx_device_descriptor_t ∗ *dm75xx*, unsigned long *ioctl_param*) [static]

Returns the top entry from the interrupt status queue.

Parameters

dm75xx	Address of the device's DM75xx device descriptor.
ioctl_param	Third parameter given on ioctl() call. This is the user space address of the structure used to pass in the arguments.

Return values

0	Success
<	0

Failure.

The following values may be returned:

```
@arg \c
    -EFAULT     Copy to user failed
```

4.6.2.13 static void dm75xx_get_pci_master_status (dm75xx_device_descriptor_t ∗ *dm75xx,* uint8_t ∗ *pci_master*) [static]

Determine whether or not a device is PCI master capable.

Parameters

dm75xx	Address of device's DM75xx device descriptor.
pci_master	Address where pci master capable flag should be stored. Zero will be stored if the device is not PCI master capable. A non-zero value will be stored here if the device is PCI master capable.

Note

PCI Master capability is required for DMA operations.

4.6.2.14 static void dm75xx_initialize_device_descriptor (dm75xx_device_descriptor_t ∗ *dm75xx*) [static]

Initialize the device descriptor for the specified DM75xx device.

Parameters

dm75xx	Address of device's DM75xx device descriptor.

Note

When initializing the device descriptor, the driver will perform the following: 1) Reset interrupt tracking and status variables 2) Initialize wait queue 3) Reset DMA information

4.6.2.15 static void dm75xx_initialize_hardware (const dm75xx_device_descriptor_t ∗ *dm75xx*) [static]

Initialize the specified DM75xx device.

Parameters

dm75xx	Address of device's DM75xx device descriptor.

Note

When initializing a device, the driver will perform the following: 1) Hardware reset of the board 2) disables PLX PCI interrupts 3) disables PLX local interrupt input 4) disables PLX DMA channel 0/1 interrupts

4.6.2.16 static int dm75xx_interrupt_control (dm75xx_device_descriptor_t * *dm75xx,* unsigned long *ioctl_param*) [static]

Control the interrupts on the boards. This includes enabling, disabling and checking the enable/disable status of the interrupts.

Parameters

dm75xx	Address of the device's DM75xx device descriptor.
ioctl_param	Third parameter given on ioctl() call. This is the user space address of the structure used to pass in the arguments.

Return values

0	Success
<	0

Failure.

The following may be returned:

```
@arg \c
    -EFAULT     Copy from user failed

    -ENOSYS     Interrupt control function requested does not exist
```

4.6.2.17 static void dm75xx_interrupt_enable (dm75xx_device_descriptor_t ∗ *dm75xx*, dm75xx_int_source_t *source*, uint8_t *enable*) [static]

Performs the actual enable/disable of the interrupt sources.

Parameters

dm75xx	Address of the device's DM75xx device descriptor
source	A bit mask indicating which interrupt sources to enable/disable
enable	flag indicating if we are performing an enable or a disable

4.6.2.18 INTERRUPT_HANDLER_TYPE dm75xx_interrupt_handler (int *irq_number*, void ∗ *device_id*)

DM75xx device interrupt handler.

Parameters

irq_number	Interrupt line number.
device_id	Address of device's DM75xx device descriptor. This is set on request_irq() call.

Return values

IRQ_HANDLED	Interrupt successfully processed;
IRQ_NONE	Interrupt could not be processed;

4.6.2.19 static long dm75xx_ioctl (struct file ∗ *file*, unsigned int *request_code*, unsigned long *ioctl_param*) [static]

Process ioctl(2) system calls directed toward a DM75xx device file.

Parameters

file	Address of kernel's file descriptor for the device file.
request_code	The service being requested.
ioctl_param	Third parameter given on ioctl() call. Depending upon request_code, ioctl_param may or may not be used. Also based upon request_code, ioctl_param may be an actual value or may be an address. If the third parameter is not given on the ioctl() call, then ioctl_param has some undefined value.

Return values

0	Success.
$<$	0

Failure.

The following values may be returned:

- `-EINVAL` request_code is not valid.

Please see the descriptions of dm75xx_validate_device(), dm75xx_read_pci_region(), dm75xx_write_pci_region(), dm75xx_modify_pci_region(), dm75xx_service_dma_function(), dm75xx_get_interrupt(), dm75xx_interrupt_↩ control(), and dm75xx_board_reset() for information on other possible values returned in this case.

4.6.2.20 int dm75xx_load (void)

Perform all actions necessary to initialize the DM75xx driver and devices.

Return values

0	Success.
$<$	0

Failure.

The following values may be returned:

- `-ENOMEM` /proc entry creation failed.

Please see the descriptions of dm75xx_probe_devices() and dm75xx_register_char_device() for information on other possible values returned in this case.

Note

On failure, this function will clean up by releasing any resources allocated by the driver.
When loaded, the driver performs a board reset, disables PLX PCI interrupts, disables PLX local interrupt input, and disables PLX DMA channel 0/1 interrupts.

4.6.2.21 static int dm75xx_modify_pci_region (dm75xx_device_descriptor_t ∗ *dm75xx,* unsigned long *ioctl_param*) `[static]`

Read an unsigned value from one of a device's PCI regions, modify certain bits in the value, and then write it back to the region.

Parameters

dm75xx	Address of device's DM75xx device descriptor.
ioctl_param	Third parameter given on ioctl() call. This is the user space address of the structure used to pass in the arguments.

Return values

0	Success.
$<$	0

Failure.

The following values may be returned:

- `-EFAULT` ioctl_param is not a valid user address.

Please see the description of dm75xx_validate_pci_access() for information on other possible values returned in this case.

4.6.2.22 static int dm75xx_open (struct inode ∗ *inode,* struct file ∗ *file*) `[static]`

Prepare a DM75xx device file to be opened and used.

Parameters

inode	Address of kernel's inode descriptor for the device file.
file	Address of kernel's file descriptor for the device file.

Return values

0	Success.
<	0

Failure.

The following values may be returned:

- `−EBUSY` The device file is already open.
- `−ENODEV` The device's inode does not refer to a valid DM75xx device; 2.4 kernel only.

Note

When a device is opened, the driver disables & clears all device interrupts, enables PLX PCI interrupts, enables PLX local interrupt input, and enables PLX DMA channel 0/1 interrupts.

4.6.2.23 static unsigned int dm75xx_poll (struct file ∗ *file,* struct poll_table_struct ∗ *poll_table*) `[static]`

Determine whether or not a DM75xx device is readable. This function supports the poll(2) and select(2) system calls.

Parameters

file	Address of kernel's file descriptor for the device file.
poll_table	Address of kernel's poll table descriptor. This keeps track of all event queues on which the process can wait.

Return values

status	mask

Bit mask describing the status of the device.

The following bits may be set in the mask:

- `POLLPRI` will be set if the file descriptor contains an invalid device descriptor.
- `POLLIN` will be set if an interrupt occurred since the last time the interrupt status was read.
- `POLLRDNORM` will be set if an interrupt occurred since the last time the interrupt status was read.

Note

A DM75xx device is readable if and only if an interrupt just occurred on the device and a process has not yet obtained the interrupt status from it.
This function is used in the process of waiting until an interrupt occurs on a device.
This function can be executed before an interrupt occurs, which happens if something sends a signal to the process.

4.6.2.24 static int dm75xx_probe_devices (uint32_t * *device_count*, dm75xx_device_descriptor_t ** *device_descriptors*) [static]

Probe and set up all DM75xx devices.

Parameters

device_count	Address where DM75xx device count should be stored. The content of this this memory is undefined if the function fails.
device_↩ descriptors	Address where address of device descriptor memory should be stored. The content of this memory is undefined if the function fails.

Return values

0	Success.
$<$	0

Failure.

The following values may be returned:

- `-ENAMETOOLONG` Device name creation failed.
- `-ENODEV` No DM75xx devices found.
- `-ENOMEM` Device descriptor memory allocation failed.

Please see the descriptions of dm75xx_process_pci_regions(), dm75xx_allocate_irq() ... for information on other possible values returned in this case.

Note

If set up of any device fails, then all device set up fails.
This function allocates memory for the DM75xx device descriptors based upon the number of devices found.
On failure, this function will clean up by releasing any resources allocated by the driver to this point.

4.6.2.25 static int dm75xx_process_pci_regions (dm75xx_device_descriptor_t * *dm75xx*, const struct pci_dev * *pci_device*) [static]

For each of the standard PCI regions, get the region's base address and length from kernel PCI resource information set up at boot. Also, remap any memory-mapped region into the kernel's virtual address space.

Parameters

dm75xx	Address of device's DM75xx device descriptor.
pci_device	Address of kernel's PCI device structure for the current DM75xx device.

Return values

0	Success.
$<$	0

Failure.

The following values may be returned:

- `-EBUSY` I/O port or I/O memory range allocation failed.
- `-EIO` A region's resource flags are not valid.
- `-ENOMEM` Remapping a memory-mapped region into the kernel's virtual address space failed.

Note

Currently, only BAR0 through BAR2 are used. BAR0 is the memory-mapped PLX DMA register region. BAR1 is the I/O-mapped PLX DMA register region. BAR2 is the memory-mapped FPGA register region.
On failure, this function will clean up by releasing any resources allocated by the driver to this point.

4.6.2.26 static void dm75xx_put_interrupt (dm75xx_device_descriptor_t $*$ *dm75xx*, uint32_t *interrupt*) `[static]`

Adds an interrupt to the interrupt status queue.

Parameters

dm75xx	Address of the device's DM75xx device descriptor.
interrupt	Interrupt to add to the queue.

Return values

0	Success
$<$	0

Failure.

The following values may be returned:

```
@arg \c
    -EFAULT     Copy to user failed
```

4.6.2.27 static int dm75xx_register_char_device (int $*$ *major*) `[static]`

Register the DM75xx character device and request dynamic allocation of a character device major number.

Parameters

major	Address where character device major number should be stored.

Return values

0	Success.
$<$	0

Failure.

The following values may be returned:

- `-EBUSY` A character device major number could not be allocated; returned by alloc_chrdev_region().
- `-EBUSY` All character device major numbers are in use; returned by register_chrdev().
- `-ENOMEM` Memory allocation failed; returned by alloc_chrdev_region().

Note

This function hides the character device interface differences between 2.4 and 2.6 kernels.

4.6.2.28 static int dm75xx_release (struct inode $*$ *inode*, struct file $*$ *file*) `[static]`

Do all processing necessary after the last reference to a DM75xx device file is released elsewhere in the kernel.

Parameters

inode	Address of kernel's inode descriptor for the device file. Unused.
file	Address of kernel's file descriptor for the device file.

Return values

0	Success.
$<$	0

Failure. Please see the description of dm75xx_validate_device() for information on possible values returned in this case.

Note

When a device is released, the driver disables PLX PCI interrupts, disables PLX local interrupt input, and disables PLX DMA channel 0/1 interrupts.

4.6.2.29 static void dm75xx_release_resources (void) `[static]`

Release any resources allocated by the driver.

Note

This function is called both at module unload time and when the driver is cleaning up after some error occurred.

4.6.2.30 static int dm75xx_service_dma_function (dm75xx_device_descriptor_t ∗ *dm75xx,* unsigned long *ioctl_param*) `[static]`

Process user space DMA function requests.

Parameters

dm75xx	Address of the device's DM75xx device descriptor.
ioctl_param	Third parameters given on ioctl() call. This is the user space address of the structure used to pass in the arguments.

Return values

0	Success
$<$	0

Failure.

The following values may be returned:

- `-EFAULT` ioctl_param is not a valid user address
- `-EFAULT` DMA channel or source used to operate upon is not valid.
- `-ENOSYS` DMA function request is not valid

Please see the descriptions of dm75xx_dma_initialize(), and dm75xx_dma_abort() for information on other possible values returned in this case.

4.6.2.31 static int dm75xx_unregister_char_device (void) `[static]`

Unregister the DM75xx character device and free the character device major number.

Return values

0	Success.
$<$	0

Failure.

The following values may be returned:

- `-EINVAL` Character major number is not valid; returned by unregister_chrdev(); 2.4 kernel only.

-EINVAL Character major number has no file operations registered for it; returned by unregister_chrdev(); 2.4 kernel only.

-EINVAL Device name specified when character major number was registered does not match the name being unregistered; returned by unregister_chrdev(); 2.4 kernel only.

Note

This function hides the character device interface differences between 2.4 and 2.6 kernels.
This function does not fail on 2.6 kernels.

4.6.2.32 static int dm75xx_validate_device (const **dm75xx_device_descriptor_t** $*$ *dm75xx*) `[static]`

Given what is assumed to be the address of a DM75xx device descriptor, make sure it corresponds to a valid DM75xx device descriptor.

Parameters

dm75xx	Address of device descriptor to be verified.

Return values

0	Success.
$<$	0

Failure.

The following values may be returned:

- `-EBADFD` dm75xx is not a valid DM75xx device descriptor address.

4.6.2.33 static int dm75xx_validate_pci_access (const **dm75xx_device_descriptor_t** $*$ *dm75xx,* const **dm75xx_pci_access_request_t** $*$ *pci_request*) `[static]`

Validate a user-space access to one of the device's PCI regions.

Parameters

dm75xx	Address of the device's DM75xx device descriptor.
pci_request	Address of PCI region access request descriptor.

Return values

0	Success
$<$	0

Failure.

The following values may be returned:

- `-EINVAL` The PCI region is not vaild.

- `-EMSGSIZE` The access size is not valid.
- `-EOPNOTSUPP` The PCI region offset is valid but is not suitably aligned for the number of bytes to be accessed.
- `-ERANGE` The PCI region offset is not valid.

Note

This function accesses information in the device descriptor. Therefore, the device descriptor spin lock should be held when this function is called.

4.7 DM75xx ioctl header file

Modules

- DM75xx ioctl enumerations
- DM75xx ioctl structures
- DM75xx ioctl macros

4.7.1 Detailed Description

4.8 DM75xx ioctl enumerations

Typedefs

- typedef enum
  dm75xx_dma_manage_function dm75xx_dma_manage_function_t

  Functions supported by driver DMA management system.
- typedef enum
  dm75xx_int_control_function dm75xx_int_control_function_t

  Functions supported by driver interrupt control system.

Enumerations

- enum dm75xx_dma_manage_function { DM75xx_DMA_FUNCTION_INITIALIZE = 0, DM75xx_DMA_FUN↩
  CTION_ABORT }

  Functions supported by driver DMA management system.
- enum dm75xx_int_control_function { DM75xx_INT_CONTROL_ENABLE = 0, DM75xx_INT_CONTROL_D↩
  ISABLE, DM75xx_INT_CONTROL_CHECK }

  Functions supported by driver interrupt control system.

4.8.1 Detailed Description

4.8.2 Enumeration Type Documentation

4.8.2.1 enum **dm75xx_dma_manage_function**

Functions supported by driver DMA management system.

Enumerator

DM75xx_DMA_FUNCTION_INITIALIZE DMA initialization

DM75xx_DMA_FUNCTION_ABORT DMA abort

Definition at line 48 of file dm75xx_ioctl.h.

4.8.2.2 enum **dm75xx_int_control_function**

Functions supported by driver interrupt control system.

Enumerator

DM75xx_INT_CONTROL_ENABLE Enable Interrupts

DM75xx_INT_CONTROL_DISABLE Disable Interrupts

DM75xx_INT_CONTROL_CHECK Returns with a value indicating which interrupts are currently enabled. This value should then be checked with DM75xx_INTERRUPT_ACTIVE().

Definition at line 63 of file dm75xx_ioctl.h.

4.9 DM75xx ioctl structures

Data Structures

- struct dm75xx_ioctl_region_readwrite

  ioctl() request structure for read from or write to PCI region
- struct dm75xx_ioctl_region_modify

  ioctl() request structure for PCI region read/modify/write
- struct dm75xx_ioctl_dma_function

  ioctl() request structure for performing a DMA function
- struct dm75xx_ioctl_int_control

  ioctl() request structure for interrupt control.
- union dm75xx_ioctl_argument

  ioctl() request structure encapsulating all possible requests. This is what gets passed into the kernel from user space on the ioctl() call.

Typedefs

- typedef struct
  dm75xx_ioctl_region_readwrite dm75xx_ioctl_region_readwrite_t
- typedef struct
  dm75xx_ioctl_region_modify dm75xx_ioctl_region_modify_t

  ioctl() PCI region read/modify/write request descriptor type
- typedef struct
  dm75xx_ioctl_dma_function dm75xx_ioctl_dma_function_t

  ioctl() request structure for performing a DMA function type.
- typedef struct
  dm75xx_ioctl_int_control dm75xx_ioctl_int_control_t

  ioctl() request structure for interrupt control.
- typedef union dm75xx_ioctl_argument dm75xx_ioctl_argument_t

  ioctl() request descriptor type

4.9.1 Detailed Description

DM75xx_Ioctl_Enumerations

4.9.2 Typedef Documentation

4.9.2.1 typedef struct **dm75xx_ioctl_region_readwrite dm75xx_ioctl_region_readwrite_t**

typedef for the PCI region access request type

Definition at line 108 of file dm75xx_ioctl.h.

4.10 DM75xx ioctl macros

Macros

- #define DM75xx_IOCTL_MAGIC 'D'

  Unique 8-bit value used to generate unique ioctl() request codes.
- #define DM75xx_IOCTL_REQUEST_BASE 0x00

  First ioctl() request number.
- #define DM75xx_IOCTL_REGION_READ

  ioctl() request code for reading from a PCI region
- #define DM75xx_IOCTL_REGION_WRITE

  ioctl() request code for writing to a PCI region
- #define DM75xx_IOCTL_REGION_MODIFY

  ioctl() request code for PCI region read/modify/write
- #define DM75xx_IOCTL_DMA_FUNCTION

  ioctl() request code for DMA function
- #define DM75xx_IOCTL_WAKEUP

  ioctl() request code to wake up a sleeping driver function
- #define DM75xx_IOCTL_INT_STATUS

  ioctl() request code to get the interrupt status queue
- #define DM75xx_IOCTL_GET_FIFO_SIZE

  ioctl() request code to get the fifo size
- #define DM75xx_IOCTL_GET_BOARD_TYPE

  ioctl() request code to get the board type
- #define DM75xx_IOCTL_INT_CONTROL

  ioctl() request code to control interrupts
- #define DM75xx_IOCTL_RESET

  ioctl() request code to reset the board
- #define DM75xx_IOCTL_RESET_DMA_STATUS

  ioctl() request code to control DMA buffer status

4.10.1 Detailed Description

DM75xx_Ioctl_Structures

4.11 DM75xx kernel compatibility header file

4.12 DM75xx kernel compatibility interrupt handler macros

Macros

- #define INTERRUPT_HANDLER_TYPE static irqreturn_t

  Type returned by interrupt handler.

Typedefs

- typedef irqreturn_t(∗ dm75xx_handler_t)(int, void ∗)

  Type definition for interrupt handling function.

4.12.1 Detailed Description

DM75xx_Kernel_Module_Major_Minor_Number_Macros

4.13 kernel compatibility interrupt handler macros

Macros

- #define DM75XX_IOCTL .unlocked_ioctl

  In Kernel 2.6.35, .ioctl was replaced with .unlocked_ioctl.

4.13.1 Detailed Description

DM75xx_Kernel_Interrupt_Handler_Macros

4.14 DM75xx kernel compatibility device I/O memory access macros

Macros

- #define IO_MEMORY_READ8 ioread8

  Entity which reads an 8-bit value from device I/O memory.
- #define IO_MEMORY_READ16 ioread16

  Entity which reads a 16-bit value from device I/O memory.
- #define IO_MEMORY_READ32 ioread32

  Entity which reads a 32-bit value from device I/O memory.
- #define IO_MEMORY_WRITE8 iowrite8

  Entity which writes an 8-bit value to device I/O memory.
- #define IO_MEMORY_WRITE16 iowrite16

  Entity which writes a 16-bit value to device I/O memory.
- #define IO_MEMORY_WRITE32 iowrite32

  Entity which writes a 32-bit value to device I/O memory.

4.14.1 Detailed Description

DM75xx_Kernel_File_Ops_Struct_Macros

4.15 DM75xx user library header file

Modules

- DM75xx user library macros
- DM75xx user library type definitions
- DM75xx user library structures
- DM75xx user library functions

4.15.1 Detailed Description

4.16 DM75xx user library macros

Macros

- #define DM75xx_INTERRUPT_ACTIVE(status, source) (((status) & (source)) ? 0xFF : 0x00)

  Determine whether or not the specified interrupt source has occurred in your the user space ISR.
- #define DM75xx_ADC_ANALOG_DATA(data) (((int16_t) (data)) >> 3)

  This macro will return the sample portion of raw analog data.
- #define DM75xx_ADC_MARKERS(data) ((data) & 0x07)

  This macro will turn the data marker portion of raw analog data.
- #define DM75xx_DAC_PACK_DATA(data, mcbsp_bit, data_markers)

  This macro will assemble a package to be sent to the Digital to Analog FIFO.

4.16.1 Detailed Description

4.16.2 Macro Definition Documentation

4.16.2.1 #define DM75xx_ADC_ANALOG_DATA(*data*) (((int16_t) (data)) >> 3)

This macro will return the sample portion of raw analog data.

Parameters

data	The raw analog data

Returns

The 12 bit signed analog sample

Note

The value returned by this macro should be stored in an int16_t

Definition at line 94 of file dm75xx_library.h.

Referenced by main().

4.16.2.2 #define DM75xx_ADC_MARKERS(*data*) ((data) & 0x07)

This macro will turn the data marker portion of raw analog data.

Parameters

data	The raw analog data

Returns

The 3 marker bits

Note

The value returned by this macro should be stored in a uint8_t

Definition at line 112 of file dm75xx_library.h.

4.16.2.3 #define DM75xx_DAC_PACK_DATA(*data,* *mcbsp_bit,* *data_markers*)

Value:

```
((int16_t)((int16_t)(data) << 3) | \
        (mcbsp_bit & 0x0004) | \
        (data_markers & 0x0003))
```

This macro will assemble a package to be sent to the Digital to Analog FIFO.

Parameters

data	The 12 bit signed data to write to the Digital to Analog channel
mcbsp_bit	A bit designating which Digital to Analog channel will receive McBSP data
data_markers	The 2 data marker bits

Returns

The combined DAC Data in an int16_t

Definition at line 139 of file dm75xx_library.h.

Referenced by main().

4.16.2.4 #define DM75xx_INTERRUPT_ACTIVE(*status,* *source*) (((status) & (source)) ? 0xFF : 0x00)

Determine whether or not the specified interrupt source has occurred in your the user space ISR.

Parameters

status	Interrupt status to examine.
source	Interrupt source to determine state of.

Return values

0x00	The specified interrupt source is not pending.
0xFF	The specified interrupt source is pending.

Definition at line 75 of file dm75xx_library.h.

Referenced by ISR().

4.17 DM75xx user library type definitions

Typedefs

- typedef int DM75xx_Error

  DM75xx user library error code type.

4.17.1 Detailed Description

DM75xx_Library_Macros

4.18 DM75xx user library structures

Data Structures

- struct DM75xx_Board_Descriptor

  DM75xx board descriptor. This structure holds information about a device needed by the library.

Typedefs

- typedef struct
  DM75xx_Board_Descriptor DM75xx_Board_Descriptor

4.18.1 Detailed Description

DM75xx_Library_Types

4.18.2 Typedef Documentation

4.18.2.1 typedef struct **DM75xx_Board_Descriptor DM75xx_Board_Descriptor**

DM75xx board descriptor type

Definition at line 224 of file dm75xx_library.h.

4.19 DM75xx user library functions

Modules

- DM75xx user library board control functions
- DM75xx user library DMA functions
- DM75xx user library general functions
- DM75xx user library user timer/counter control
- DM75xx user library burst clock control
- DM75xx user library pacer clock control
- DM75xx user library channel gain table

- DM75xx_Error DM75xx_ADC_FIFO_Read (DM75xx_Board_Descriptor ∗handle, uint16_t ∗value)

  DM75xx_Library_ADC_Functions DM75xx user library analog to digital.
- DM75xx_Error DM75xx_ADC_Software_Sample (DM75xx_Board_Descriptor ∗handle)

  Analog to Digital Software Sample.
- DM75xx_Error DM75xx_ADC_Conv_Signal (DM75xx_Board_Descriptor ∗handle, dm75xx_adc_conv_↩ signal_t adc_conv_signal)

  Select the A/D Conversion Signal.
- DM75xx_Error DM75xx_ADC_SCNT_Source (DM75xx_Board_Descriptor ∗handle, dm75xx_adc_scnt_src↩ _t src)

  Select the A/D Sample Counter Source.
- DM75xx_Error DM75xx_ADC_About_Enable (DM75xx_Board_Descriptor ∗handle, uint16_t enable)

  Enable/Disable About Counter stop.
- DM75xx_Error DM75xx_ADC_Clear (DM75xx_Board_Descriptor ∗handle)

  Clear Analag to Digital FIFO.
- DM75xx_Error DM75xx_ADC_SCNT_Read (DM75xx_Board_Descriptor ∗handle, uint16_t ∗data)

  Read the value in the A/D Sample Counter.
- DM75xx_Error DM75xx_ADC_SCNT_Load (DM75xx_Board_Descriptor ∗handle, uint16_t data)

  Load a value into the A/D Sample Counter.

- DM75xx_Error DM75xx_DAC_Soft_Update (DM75xx_Board_Descriptor ∗handle, uint8_t dac)

  DM75xx_Library_DAC_Functions DM75xx user library digital to analog.
- DM75xx_Error DM75xx_DAC_Get_Update_Counter (DM75xx_Board_Descriptor ∗handle, dm75xx_dac_↩ channel_t dac, uint16_t ∗data)

  Get DAC update counter for a specified channel.
- DM75xx_Error DM75xx_DAC_Set_Update_Counter (DM75xx_Board_Descriptor ∗handle, dm75xx_dac_↩ channel_t dac, uint16_t data)

  Set the DAC update counter for a specified channel.
- DM75xx_Error DM75xx_DAC_Set_Range (DM75xx_Board_Descriptor ∗handle, dm75xx_dac_channel_↩ t dac, dm75xx_dac_range_t range)

  Set the DAC output range for a specified channel.
- DM75xx_Error DM75xx_DAC_Set_Update_Source (DM75xx_Board_Descriptor ∗handle, dm75xx_dac_↩ channel_t dac, dm75xx_dac_update_src_t src)

  Set the DAC Update Source for the specified channel.
- DM75xx_Error DM75xx_DAC_Set_Mode (DM75xx_Board_Descriptor ∗handle, dm75xx_dac_channel_t dac, dm75xx_dac_mode_t mode)

  Set the DAC mode for a specified channel.
- DM75xx_Error DM75xx_DAC_FIFO_Write (DM75xx_Board_Descriptor ∗handle, dm75xx_dac_channel_↩ t dac, uint16_t data)

  Write a value to the DAC FIFO of a specified channel.
- DM75xx_Error DM75xx_DAC_Set_Frequency (DM75xx_Board_Descriptor ∗handle, dm75xx_dac_freq_↩ t freq)

Set the primary slock frequency for DAC conversion.

- DM75xx_Error DM75xx_DAC_Set_Count (DM75xx_Board_Descriptor ∗handle, uint32_t count)

  Set the DAC Clock Count.
- DM75xx_Error DM75xx_DAC_Set_Rate (DM75xx_Board_Descriptor ∗handle, dm75xx_dac_freq_t freq, uint32_t rate, float ∗actualRate)

  Set the DAC conversion rate.
- DM75xx_Error DM75xx_DAC_Set_Clock_Stop (DM75xx_Board_Descriptor ∗handle, dm75xx_dac_clk_↩ stop_t stop)

  Set the DAC Clock Stop Value.
- DM75xx_Error DM75xx_DAC_Set_Clock_Start (DM75xx_Board_Descriptor ∗handle, dm75xx_dac_clk_↩ start_t start)

  Set the DAC Clock Start Value.
- DM75xx_Error DM75xx_DAC_Start (DM75xx_Board_Descriptor ∗handle)

  Causes a DAC Software Start.
- DM75xx_Error DM75xx_DAC_Stop (DM75xx_Board_Descriptor ∗handle)

  Causes a DAC Software Stop.
- DM75xx_Error DM75xx_DAC_Setup (DM75xx_Board_Descriptor ∗handle, dm75xx_dac_channel_t dac, dm75xx_dac_range_t range, dm75xx_dac_update_src_t src, dm75xx_dac_mode_t mode)

  Setup a DAC channel.
- DM75xx_Error DM75xx_DAC_Reset (DM75xx_Board_Descriptor ∗handle, dm75xx_dac_channel_t dac)

  Reset a DAC Fifo.
- DM75xx_Error DM75xx_DAC_Clear (DM75xx_Board_Descriptor ∗handle, dm75xx_dac_channel_t dac)

  Clear a DAC Fifo.
- DM75xx_Error DM75xx_DAC_Set_CLK_Mode (DM75xx_Board_Descriptor ∗handle, dm75xx_dac_clk_↩ mode_t clk_mode)

  Set DAC Clock Mode.

- DM75xx_Error DM75xx_HSDIN_Software_Sample (DM75xx_Board_Descriptor ∗handle)

  DM75xx_Library_HSDIN_Functions DM75xx user library high speed digital.
- DM75xx_Error DM75xx_HSDIN_Sample_Signal (DM75xx_Board_Descriptor ∗handle, dm75xx_hsdin_↩ signal_t signal)

  Set HighSpeed digital sampling signal.
- DM75xx_Error DM75xx_HSDIN_Clear (DM75xx_Board_Descriptor ∗handle)

  Clear High Speed Digital FIFO.
- DM75xx_Error DM75xx_HSDIN_FIFO_Read (DM75xx_Board_Descriptor ∗handle, uint16_t ∗data)

  Read value from High Speed Digital FIFO.

- DM75xx_Error DM75xx_SBUS_Set_Source (DM75xx_Board_Descriptor ∗handle, dm75xx_sbus_t sbus, dm75xx_sbus_src_t src)

  DM75xx_Library_SBUS_Functions DM75xx user library syncbus.
- DM75xx_Error DM75xx_SBUS_Enable (DM75xx_Board_Descriptor ∗handle, dm75xx_sbus_t sbus, uint16↩ _t enable)

  Enable/Disable Syncbus.

- DM75xx_Error DM75xx_ACNT_Get_Count (DM75xx_Board_Descriptor ∗handle, uint16_t ∗data)

  DM75xx_Library_ACNT_Functions DM75xx user library about counter.
- DM75xx_Error DM75xx_ACNT_Set_Count (DM75xx_Board_Descriptor ∗handle, uint16_t data)

  Set the About Counter value.

- DM75xx_Error DM75xx_DCNT_Get_Count (DM75xx_Board_Descriptor ∗handle, uint16_t ∗data)

  DM75xx_Library_DCNT_Functions DM75xx user library delay counter.
- DM75xx_Error DM75xx_DCNT_Set_Count (DM75xx_Board_Descriptor ∗handle, uint16_t data)

Set the Delay Counter value.

- DM75xx_Error DM75xx_DIO_Set_Port (DM75xx_Board_Descriptor ∗handle, dm75xx_dio_port_t port, uint8_t data)

  DM75xx_Library_DIO_Functions DM75xx user library digital input/output.
- DM75xx_Error DM75xx_DIO_Get_Port (DM75xx_Board_Descriptor ∗handle, dm75xx_dio_port_t port, uint8_t ∗data)

  Get the value from the specified Digital I/O Port.
- DM75xx_Error DM75xx_DIO_Get_Status (DM75xx_Board_Descriptor ∗handle, uint8_t ∗data)

  Get the Digital I/O Status byte.
- DM75xx_Error DM75xx_DIO_Clear_IRQ (DM75xx_Board_Descriptor ∗handle)

  Clear Digital I/O IRQ Status.
- DM75xx_Error DM75xx_DIO_Reset (DM75xx_Board_Descriptor ∗handle)

  Clear Digital I/O Chip.
- DM75xx_Error DM75xx_DIO_Set_Direction (DM75xx_Board_Descriptor ∗handle, dm75xx_dio_port_t port, uint8_t direction)

  Set the direction of the specified Digital I/O Port.
- DM75xx_Error DM75xx_DIO_Set_Mask (DM75xx_Board_Descriptor ∗handle, uint8_t mask)

  Set Digital I/O Port 0 Mask.
- DM75xx_Error DM75xx_DIO_Set_Compare (DM75xx_Board_Descriptor ∗handle, uint8_t compare)

  Set the compare register for Digital I/O Port 0.
- DM75xx_Error DM75xx_DIO_Get_Compare (DM75xx_Board_Descriptor ∗handle, uint8_t ∗compare)

  Get the compare register for Digital I/O Port 0.
- DM75xx_Error DM75xx_DIO_IRQ_Mode (DM75xx_Board_Descriptor ∗handle, dm75xx_dio_mode_t mode)

  Set the IRQ Mode for Digital I/O.
- DM75xx_Error DM75xx_DIO_Clock (DM75xx_Board_Descriptor ∗handle, dm75xx_dio_clk_t clock)

  Set the Digital I/O Sample Clock.
- DM75xx_Error DM75xx_DIO_Enable_IRQ (DM75xx_Board_Descriptor ∗handle, uint8_t enable)

  Enable/Disable Digital I/O Interrupts.

- DM75xx_Error DM75xx_UIO_Select (DM75xx_Board_Descriptor ∗handle, dm75xx_uio_channel_t channel, dm75xx_uio_source_t source)

  DM75xx_Library_UIO_Functions DM75xx user library user I/O.
- DM75xx_Error DM75xx_UIO_Read (DM75xx_Board_Descriptor ∗handle, uint32_t ∗data)

  Read the current status of the user I/O.
- DM75xx_Error DM75xx_UIO_Write (DM75xx_Board_Descriptor ∗handle, uint32_t data)

  Write the value of the user I/O.

- DM75xx_Error DM75xx_McBSP_ADC_FIFO (DM75xx_Board_Descriptor ∗handle, uint8_t enable)

  DM75xx_Library_McBSP_Functions DM75xx user library mcbsp.
- DM75xx_Error DM75xx_McBSP_DAC_FIFO (DM75xx_Board_Descriptor ∗handle, uint8_t enable)

  Enable/Disable D/A FIFO to DSP.

- DM75xx_Error DM75xx_ETRIG_Polarity_Select (DM75xx_Board_Descriptor ∗handle, dm75xx_ext_↩ polarity_t polarity)

  DM75xx_Library_EXT_Functions DM75xx user library external trigger/interrupt.
- DM75xx_Error DM75xx_EINT_Polarity_Select (DM75xx_Board_Descriptor ∗handle, dm75xx_ext_polarity_t polarity)

  Set the External Interrupt polarity.

- DM75xx_Error DM75xx_FIFO_Get_Status (DM75xx_Board_Descriptor ∗handle, uint16_t ∗fifo_status)

DM75xx_Library_STATUS_Functions DM75xx user library status.

- DM75xx_Error DM75xx_CLK_Get_Status (DM75xx_Board_Descriptor ∗handle, uint16_t ∗status)

  Get status of pacer/burst clocks.

- DM75xx_Error DM75xx_Calibrate (DM75xx_Board_Descriptor ∗handle, uint16_t dac1_value, uint16_↩
t dac2_value, dm75xx_dac_range_t dac1_range, dm75xx_dac_range_t dac2_range)

  DM75xx_Library_SDM7540_Functions DM75xx user library SDM7540 functions.
- DM75xx_Error DM75xx_DSP_CMD_Send (DM75xx_Board_Descriptor ∗handle, dm75xx_dsp_command_t command)

  Issue a command to the 7540 onboard DSP.
- DM75xx_Error DM75xx_DSP_CMD_Complete (DM75xx_Board_Descriptor ∗handle, uint8_t ∗data)

  Checks if the last command given to the DSP is finished.
- DM75xx_Error DM75xx_DSP_CMD_Status (DM75xx_Board_Descriptor ∗handle, dm75xx_dsp_command↩
_t command)

  Checks whether or not a command successfully completed on the DSP.
- DM75xx_Error DM75xx_ALGDIO_Get_Mask (DM75xx_Board_Descriptor ∗handle, dm75xx_algdio_mask_t ∗pin1, dm75xx_algdio_mask_t ∗pin2)

  Get the the mask of the Analog DIO.
- DM75xx_Error DM75xx_ALGDIO_Set_Mask (DM75xx_Board_Descriptor ∗handle, dm75xx_algdio_mask_t pin1, dm75xx_algdio_mask_t pin2)

  Set the Analog DIO Mask.
- DM75xx_Error DM75xx_ALGDIO_Get_Direction (DM75xx_Board_Descriptor ∗handle, dm75xx_algdio_↩
direction_t ∗pin1, dm75xx_algdio_direction_t ∗pin2)

  Get the Analog DIO Direction.
- DM75xx_Error DM75xx_ALGDIO_Set_Direction (DM75xx_Board_Descriptor ∗handle, dm75xx_algdio_↩
direction_t pin1, dm75xx_algdio_direction_t pin2)

  Set the Analog DIO Direction.
- DM75xx_Error DM75xx_ALGDIO_Set_Data (DM75xx_Board_Descriptor ∗handle, uint8_t pin1, uint8_t pin2)

  Set the Analog DIO pin values.
- DM75xx_Error DM75xx_ALGDIO_Get_Data (DM75xx_Board_Descriptor ∗handle, uint8_t ∗pin1, uint8_↩
t ∗pin2)

  Get the Analog DIO pin values.
- DM75xx_Error DM75xx_ALGDIO_Get_IRQ_Status (DM75xx_Board_Descriptor ∗handle, uint8_t ∗status)

  Get Analog DIO IRQ Status.
- DM75xx_Error DM75xx_Get_Temp (DM75xx_Board_Descriptor ∗handle, uint8_t ∗temp)

  Get the temperature from the board.

4.19.1 Detailed Description

DM75xx_Library_Structures

4.19.2 Function Documentation

4.19.2.1 DM75xx_Error DM75xx_ACNT_Get_Count (DM75xx_Board_Descriptor ∗ *handle,* uint16_t ∗ *data*)

DM75xx_Library_ACNT_Functions DM75xx user library about counter.

DM75xx_Library_SBUS_Functions

```
Get About Counter value
```

Parameters

handle	`Address of device's library board descriptor.`
data	Address of the variable to store the value.

Return values

0	Success.
-1	Failure. `Please see the ioctl(2) man page for information on possible values errno` may have in this case.

4.19.2.2 DM75xx_Error DM75xx_ACNT_Set_Count (DM75xx_Board_Descriptor ∗ *handle,* uint16_t *data*)

Set the About Counter value.

Parameters

handle	`Address of device's library board descriptor.`
data	Value at which to set the About Counter.

Return values

0	Success.
-1	Failure. `Please see the ioctl(2) man page for information on possible values errno` may have in this case.

Referenced by main().

4.19.2.3 DM75xx_Error DM75xx_ADC_About_Enable (DM75xx_Board_Descriptor ∗ *handle,* uint16_t *enable*)

Enable/Disable About Counter stop.

Parameters

handle	`Address of device's library board descriptor.`
enable	Enable/Disable.

Return values

0	Success.
-1	Failure. `Please see the ioctl(2) man page for information on possible values errno` may have in this case.

Referenced by main().

4.19.2.4 DM75xx_Error DM75xx_ADC_Clear (DM75xx_Board_Descriptor ∗ *handle*)

Clear Analag to Digital FIFO.

Parameters

handle	Address of device's library board descriptor.

Return values

0	Success.
-1	Failure.

Please see the ioctl(2) man page for information on possible values errno may have in this case.

Referenced by main().

4.19.2.5 DM75xx_Error DM75xx_ADC_Conv_Signal (DM75xx_Board_Descriptor ∗ *handle*, dm75xx_adc_conv_signal_t *adc_conv_signal*)

Select the A/D Conversion Signal.

Parameters

handle	`Address of device's library board descriptor.`
adc_conv_signal	The A/D conversion signal to select.

Return values

0	Success.
-1	Failure. `Please see the ioctl(2) man page for information on possible values errno` may have in this case.

Referenced by main().

4.19.2.6 DM75xx_Error DM75xx_ADC_FIFO_Read (DM75xx_Board_Descriptor ∗ *handle*, uint16_t ∗ *value*)

DM75xx_Library_ADC_Functions DM75xx user library analog to digital.

DM75xx_Library_CGT_Functions

```
Read a value from the A/D FIFO
```

Parameters

handle	`Address of device's library board descriptor.`
value	Address of the variable to store the value read.

Returns

0

Success

Returns

-1

Failure.

errno may be set as follows:

- `EINVAL` pointer was NULL.

Please see the close(2) man page for information on other possible values errno may have in this case.

Referenced by main().

4.19.2.7 DM75xx_Error DM75xx_ADC_SCNT_Load (DM75xx_Board_Descriptor ∗ *handle,* uint16_t *data*)

Load a value into the A/D Sample Counter.

Parameters

handle	Address of device's library board descriptor.
data	Value to load into the sample counter.

Return values

0	Success.
-1	Failure.

Please see the ioctl(2) man page for information on possible values errno may have in this case.

4.19.2.8 DM75xx_Error DM75xx_ADC_SCNT_Read (DM75xx_Board_Descriptor ∗ *handle,* uint16_t ∗ *data*)

Read the value in the A/D Sample Counter.

Parameters

handle	Address of device's library board descriptor.
data	Address of the variable to store the data.

Return values

0	Success.
-1	Failure.

Please see the ioctl(2) man page for information on possible values errno may have in this case.

4.19.2.9 DM75xx_Error DM75xx_ADC_SCNT_Source (DM75xx_Board_Descriptor ∗ *handle,* dm75xx_adc_scnt_src_t *src*)

Select the A/D Sample Counter Source.

Parameters

handle	`Address of device's library board descriptor.`

src	The selected sample counter source.

Return values

0	Success.
-1	Failure. `Please see the ioctl(2) man page for information on possible values errno` may have in this case.

4.19.2.10 DM75xx_Error DM75xx_ADC_Software_Sample (DM75xx_Board_Descriptor ∗ *handle*)

Analog to Digital Software Sample.

Parameters

handle	`Address of device's library board descriptor.`

Return values

0	Success.
-1	Failure. `Please see the ioctl(2) man page for information on possible values errno` may have in this case.

Referenced by main().

4.19.2.11 DM75xx_Error DM75xx_ALGDIO_Get_Data (DM75xx_Board_Descriptor ∗ *handle,* uint8_t ∗ *pin1,* uint8_t ∗ *pin2*)

Get the Analog DIO pin values.

Parameters

handle	Address of the device's library board descriptor.
pin1	Value at logic high (0xFF) or logic low (0x00).
pin2	Value at logic high (0xFF) or logic low (0x00).

Return values

0	Success.
-1	Failure.

4.19.2.12 DM75xx_Error DM75xx_ALGDIO_Get_Direction (DM75xx_Board_Descriptor ∗ *handle,* dm75xx_algdio_direction_t ∗ *pin1,* dm75xx_algdio_direction_t ∗ *pin2*)

Get the Analog DIO Direction.

Parameters

handle	Address of the device's library board descriptor.
pin1	Pin1 Direction 0 = Input, Positive vals = Output.
pin2	Pin2 Direction 0 = Input, Positive vals = Output.

Return values

0	Success.
-1	Failure.

4.19.2.13 DM75xx_Error DM75xx_ALGDIO_Get_IRQ_Status (DM75xx_Board_Descriptor ∗ *handle,* uint8_t ∗ *status*)

Get Analog DIO IRQ Status.

Parameters

handle	Address of the device's library board descriptor.
status	The IRQ status.

Return values

0	Success.
-1	Failure.

4.19.2.14 DM75xx_Error DM75xx_ALGDIO_Get_Mask (DM75xx_Board_Descriptor ∗ *handle,* dm75xx_algdio_mask_t ∗ *pin1,* dm75xx_algdio_mask_t ∗ *pin2*)

Get the the mask of the Analog DIO.

Parameters

handle	Address of the device's library board descriptor.
pin1	Pin1 mask enabled/disabled.
pin2	Pin2 mask enabled/disabled.

Return values

0	Success.
-1	Failure.

4.19.2.15 DM75xx_Error DM75xx_ALGDIO_Set_Data (DM75xx_Board_Descriptor ∗ *handle,* uint8_t *pin1,* uint8_t *pin2*)

Set the Analog DIO pin values.

Parameters

handle	Address of the device's library board descriptor.
pin1	Value at logic high (0xFF) or logic low (0x00).
pin2	Value at logic high (0xFF) or logic low (0x00).

Return values

0	Success.
-1	Failure.

Referenced by main().

4.19.2.16 DM75xx_Error DM75xx_ALGDIO_Set_Direction (DM75xx_Board_Descriptor ∗ *handle,* dm75xx_algdio_direction_t *pin1,* dm75xx_algdio_direction_t *pin2*)

Set the Analog DIO Direction.

Parameters

handle	Address of the device's library board descriptor.
pin1	Pin1 Direction 0 = Input, Positive vals = Output.
pin2	Pin2 Direction 0 = Input, Positive vals = Output.

Return values

0	Success.
-1	Failure.

Referenced by main().

4.19.2.17 DM75xx_Error DM75xx_ALGDIO_Set_Mask (DM75xx_Board_Descriptor ∗ *handle,* dm75xx_algdio_mask_t *pin1,* dm75xx_algdio_mask_t *pin2*)

Set the Analog DIO Mask.

Parameters

handle	Address of the device's library board descriptor.
pin1	Pin1 mask enabled/disabled.
pin2	Pin2 mask enabled/disabled.

Return values

0	Success.
-1	Failure.

Referenced by main().

4.19.2.18 DM75xx_Error DM75xx_Calibrate (DM75xx_Board_Descriptor ∗ *handle,* uint16_t *dac1_value,* uint16_t *dac2_value,* dm75xx_dac_range_t *dac1_range,* dm75xx_dac_range_t *dac2_range*)

DM75xx_Library_SDM7540_Functions DM75xx user library SDM7540 functions.

DM75xx_Library_STATUS_Functions

```
Calibrate an SDM7540/SDM8540.
```

Parameters

handle	Address of the device's library board descriptor.
dac1_value	Value to set on DAC1 after calibration.
dac2_value	Value to set on DAC2 after calibration.
dac1_range	The voltage range by which to calibrate dac1.
dac2_range	The voltage range by which to calibrate dac2.

Return values

0	Success.
-1	Failure.

Referenced by main().

4.19.2.19 DM75xx_Error DM75xx_CLK_Get_Status (DM75xx_Board_Descriptor ∗ *handle,* uint16_t ∗ *status*)

Get status of pacer/burst clocks.

Parameters

handle	Address of device's library board descriptor.
status	Variable in which to store the current status.

Return values

0	Success.
-1	Failure.

Please see the ioctl(2) man page for information on possible values errno may have in this case.

4.19.2.20 DM75xx_Error DM75xx_DAC_Clear (DM75xx_Board_Descriptor ∗ *handle,* dm75xx_dac_channel_t *dac*)

Clear a DAC Fifo.

Parameters

handle	`Address of device's library board descriptor.`
dac	The specified DAC channel.

Return values

0	Success.
-1	Failure. `Please see the ioctl(2) man page for information on possible values errno` may have in this case.

4.19.2.21 DM75xx_Error DM75xx_DAC_FIFO_Write (DM75xx_Board_Descriptor ∗ *handle,* dm75xx_dac_channel_t *dac,* uint16_t *data*)

Write a value to the DAC FIFO of a specified channel.

Parameters

handle	`Address of device's library board descriptor.`
dac	The specified DAC channel.
data	Value to write to the DAC FIFO.

Return values

0	Success.
-1	Failure. `Please see the ioctl(2) man page for information on possible values errno` may have in this case.

Referenced by main().

4.19.2.22 DM75xx_Error DM75xx_DAC_Get_Update_Counter (DM75xx_Board_Descriptor ∗ *handle,* dm75xx_dac_channel_t *dac,* uint16_t ∗ *data*)

Get DAC update counter for a specified channel.

Parameters

handle	`Address of device's library board descriptor.`
dac	The specific DAC channel.
data	Address of the variable to store the data.

Return values

0	Success.
-1	Failure. `Please see the ioctl(2) man page for information on possible values errno` may have in this case.

Referenced by main().

4.19.2.23 DM75xx_Error DM75xx_DAC_Reset (DM75xx_Board_Descriptor ∗ *handle,* dm75xx_dac_channel_t *dac*)

Reset a DAC Fifo.

Parameters

handle	`Address of device's library board descriptor.`
dac	The specified DAC channel.

Return values

0	Success.
-1	Failure. `Please see the ioctl(2) man page for information on possible values errno` may have in this case.

4.19.2.24 DM75xx_Error DM75xx_DAC_Set_CLK_Mode (DM75xx_Board_Descriptor ∗ *handle,* dm75xx_dac_clk_mode_t *clk_mode*)

Set DAC Clock Mode.

Parameters

handle	`Address of device's library board descriptor.`
clk_mode	The mode set the DAC Clock (Free Run or Start/Stop).

Return values

0	Success.
-1	Failure. `Please see the ioctl(2) man page for information on possible values errno` may have in this case.

Referenced by main().

4.19.2.25 DM75xx_Error DM75xx_DAC_Set_Clock_Start (DM75xx_Board_Descriptor ∗ *handle,* dm75xx_dac_clk_start_t *start*)

Set the DAC Clock Start Value.

Parameters

handle	`Address of device's library board descriptor.`
start	The selected clock start value to be written.

Return values

0	Success.
-1	Failure. `Please see the ioctl(2) man page for information on possible values errno` may have in this case.

Note

This function calls DM75xx_DAC_Set_Clock()

Referenced by main().

4.19.2.26 DM75xx_Error DM75xx_DAC_Set_Clock_Stop (DM75xx_Board_Descriptor ∗ *handle,* dm75xx_dac_clk_stop_t *stop*)

Set the DAC Clock Stop Value.

Parameters

handle	`Address of device's library board descriptor.`
stop	The selected clock stop value to be written.

Return values

0	Success.
-1	Failure. `Please see the ioctl(2) man page for information on possible values errno` may have in this case.

Note

This function calls DM75xx_DAC_Set_Clock()

Referenced by main().

4.19.2.27 DM75xx_Error DM75xx_DAC_Set_Count (DM75xx_Board_Descriptor ∗ *handle,* uint32_t *count*)

Set the DAC Clock Count.

Parameters

handle	`Address of device's library board descriptor.`
count	The value to which to set the DAC Clock Count

Return values

0	Success.
-1	Failure. `Please see the ioctl(2) man page for information on possible values errno` may have in this case.

Note

This function calls DM75xx_DAC_Set_Clock()

4.19.2.28 DM75xx_Error DM75xx_DAC_Set_Frequency (DM75xx_Board_Descriptor ∗ *handle,* dm75xx_dac_freq_t *freq*)

Set the primary slock frequency for DAC conversion.

Parameters

handle	`Address of device's library board descriptor.`
freq	The specified primary clock frequency.

Return values

0	Success.
-1	Failure. `Please see the ioctl(2) man page for information on possible values errno` may have in this case.

Referenced by main().

4.19.2.29 DM75xx_Error DM75xx_DAC_Set_Mode (DM75xx_Board_Descriptor ∗ *handle,* dm75xx_dac_channel_t *dac,* dm75xx_dac_mode_t *mode*)

Set the DAC mode for a specified channel.

Parameters

handle	`Address of device's library board descriptor.`
dac	The specified DAC channel.
mode	The specified mode.

Return values

0	Success.
-1	Failure. `Please see the ioctl(2) man page for information on possible values errno` may have in this case.

Referenced by main().

4.19.2.30 DM75xx_Error DM75xx_DAC_Set_Range (DM75xx_Board_Descriptor ∗ *handle,* dm75xx_dac_channel_t *dac,* dm75xx_dac_range_t *range*)

Set the DAC output range for a specified channel.

Parameters

handle	`Address of device's library board descriptor.`
dac	The specified DAC channel.
range	The specified output range.

Return values

0	Success.
-1	Failure. `Please see the ioctl(2) man page for information on possible values errno` may have in this case.

Referenced by main().

4.19.2.31 DM75xx_Error DM75xx_DAC_Set_Rate (DM75xx_Board_Descriptor ∗ *handle,* dm75xx_dac_freq_t *freq,* uint32_t *rate,* float ∗ *actualRate*)

Set the DAC conversion rate.

Parameters

handle	`Address of device's library board descriptor.`
freq	The specified primary frequency.
rate	The chosen rate for conversion (in Hz).
actualRate	Address of the variable to store the precise rate the clock was set to.

Return values

0	Success.
-1	Failure. `Please see the ioctl(2) man page for information on possible values errno` may have in this case.

Note

This function calls DM75xx_DAC_Set_Clock()

Referenced by main().

4.19.2.32 DM75xx_Error DM75xx_DAC_Set_Update_Counter (DM75xx_Board_Descriptor ∗ *handle,* dm75xx_dac_channel_t *dac,* uint16_t *data*)

Set the DAC update counter for a specified channel.

Parameters

handle	`Address of device's library board descriptor.`
dac	The specific DAC channel.
data	The value to write to the DAC update counter

Return values

0	Success.
-1	Failure. `Please see the ioctl(2) man page for information on possible values errno` may have in this case.

Referenced by main().

4.19.2.33 DM75xx_Error DM75xx_DAC_Set_Update_Source (DM75xx_Board_Descriptor ∗ *handle,* dm75xx_dac_channel_t *dac,* dm75xx_dac_update_src_t *src*)

Set the DAC Update Source for the specified channel.

Parameters

handle	`Address of device's library board descriptor.`
dac	The specified DAC channel.
src	The specified update source.

Return values

0	Success.
-1	Failure. `Please see the ioctl(2) man page for information on possible values errno` may have in this case.

Referenced by main().

4.19.2.34 DM75xx_Error DM75xx_DAC_Setup (DM75xx_Board_Descriptor ∗ *handle,* dm75xx_dac_channel_t *dac,* dm75xx_dac_range_t *range,* dm75xx_dac_update_src_t *src,* dm75xx_dac_mode_t *mode*)

Setup a DAC channel.

Parameters

handle	`Address of device's library board descriptor.`

dac	The specified DAC channel.
range	The specified DAC range.
src	The specified DAC update source.
mode	The specified DAC mode.

Return values

0	Success.
-1	Failure. `Please see the ioctl(2) man page for information on possible values errno` may have in this case.

Referenced by main().

4.19.2.35 DM75xx_Error DM75xx_DAC_Soft_Update (DM75xx_Board_Descriptor * *handle*, uint8_t *dac*)

DM75xx_Library_DAC_Functions DM75xx user library digital to analog.

DM75xx_Library_ADC_Functions

```
Cause a DAC software update on the specified channel.
```

Parameters

handle	`Address of device's library board descriptor.`
dac	The specific DAC channel(s).

Return values

0	Success.
-1	Failure. `Please see the ioctl(2) man page for information on possible values errno` may have in this case.

4.19.2.36 DM75xx_Error DM75xx_DAC_Start (DM75xx_Board_Descriptor * *handle*)

Causes a DAC Software Start.

Parameters

handle	`Address of device's library board descriptor.`

Return values

0	Success.
-1	Failure. `Please see the ioctl(2) man page for information on possible values errno` may have in this case.

Note

This function calls DM75xx_DAC_Set_Clock()

4.19.2.37 DM75xx_Error DM75xx_DAC_Stop (DM75xx_Board_Descriptor ∗ *handle*)

Causes a DAC Software Stop.

Parameters

handle	`Address of device's library board descriptor.`

Return values

0	Success.
-1	Failure. `Please see the ioctl(2) man page for information on possible values errno` may have in this case.

Note

This function calls DM75xx_DAC_Set_Clock()

4.19.2.38 DM75xx_Error DM75xx_DCNT_Get_Count (DM75xx_Board_Descriptor ∗ *handle,* uint16_t ∗ *data*)

DM75xx_Library_DCNT_Functions DM75xx user library delay counter.

DM75xx_Library_ACNT_Functions

```
Get the Delay Counter value.
```

Parameters

handle	`Address of device's library board descriptor.`
data	Address of the variable to store the value.

Return values

0	Success.
-1	Failure. `Please see the ioctl(2) man page for information on possible values errno` may have in this case.

4.19.2.39 DM75xx_Error DM75xx_DCNT_Set_Count (DM75xx_Board_Descriptor ∗ *handle,* uint16_t *data*)

Set the Delay Counter value.

Parameters

handle	`Address of device’s library board descriptor.`
data	Value at which to set the Delay Counter.

Return values

0	Success.
-1	Failure. `Please see the ioctl(2) man page for information on possible values errno` may have in this case.

Referenced by main().

4.19.2.40 DM75xx_Error DM75xx_DIO_Clear_IRQ (DM75xx_Board_Descriptor ∗ *handle*)

Clear Digital I/O IRQ Status.

Parameters

handle	Address of device’s library board descriptor.

Return values

0	Success.
-1	Failure.

Please see the ioctl(2) man page for information on possible values errno may have in this case.

Referenced by ISR().

4.19.2.41 DM75xx_Error DM75xx_DIO_Clock (DM75xx_Board_Descriptor ∗ *handle,* dm75xx_dio_clk_t *clock*)

Set the Digital I/O Sample Clock.

Parameters

handle	Address of device’s library board descriptor.
clock	The clock

Return values

0	Success.
-1	Failure.

Please see the ioctl(2) man page for information on possible values errno may have in this case.

Referenced by main().

4.19.2.42 DM75xx_Error DM75xx_DIO_Enable_IRQ (DM75xx_Board_Descriptor ∗ *handle,* uint8_t *enable*)

Enable/Disable Digital I/O Interrupts.

Parameters

handle	Address of device’s library board descriptor.
enable	0 for Disable anything else for Enable

Return values

0	Success.
-1	Failure.

Please see the ioctl(2) man page for information on possible values errno may have in this case.

Referenced by main().

4.19.2.43 DM75xx_Error DM75xx_DIO_Get_Compare (DM75xx_Board_Descriptor ∗ *handle,* uint8_t ∗ *compare*)

Get the compare register for Digital I/O Port 0.

Parameters

handle	Address of device's library board descriptor.
compare	Address of the variable to store the value in the compare register.

Return values

0	Success.
-1	Failure.

Please see the ioctl(2) man page for information on possible values errno may have in this case.

Note

This register is used as a latch when in event mode. The value that caused the event will be latched to this register and can subsequently be read.

4.19.2.44 DM75xx_Error DM75xx_DIO_Get_Port (DM75xx_Board_Descriptor ∗ *handle,* dm75xx_dio_port_t *port,* uint8_t ∗ *data*)

Get the value from the specified Digital I/O Port.

Parameters

handle	Address of device's library board descriptor.
port	The specified Digital I/O Port.
data	The address of the variable to store the Digital I/O Port's value.

Return values

0	Success.
-1	Failure.

Please see the ioctl(2) man page for information on possible values errno may have in this case.

Referenced by main().

4.19.2.45 DM75xx_Error DM75xx_DIO_Get_Status (DM75xx_Board_Descriptor ∗ *handle,* uint8_t ∗ *data*)

Get the Digital I/O Status byte.

Parameters

handle	Address of device's library board descriptor.
data	Address of the variable to store the status byte.

Return values

0	Success.
-1	Failure.

Please see the ioctl(2) man page for information on possible values errno may have in this case.

4.19.2.46 DM75xx_Error DM75xx_DIO_IRQ_Mode (DM75xx_Board_Descriptor ∗ *handle,* dm75xx_dio_mode_t *mode*)

Set the IRQ Mode for Digital I/O.

Parameters

handle	Address of device's library board descriptor.
mode	Set event or match mode IRQ.

Return values

0	Success.
-1	Failure.

Please see the ioctl(2) man page for information on possible values errno may have in this case.

Referenced by main().

4.19.2.47 DM75xx_Error DM75xx_DIO_Reset (DM75xx_Board_Descriptor ∗ *handle*)

Clear Digital I/O Chip.

Parameters

handle	Address of device's library board descriptor.

Return values

0	Success.
-1	Failure.

Please see the ioctl(2) man page for information on possible values errno may have in this case.

4.19.2.48 DM75xx_Error DM75xx_DIO_Set_Compare (DM75xx_Board_Descriptor ∗ *handle,* uint8_t *compare*)

Set the compare register for Digital I/O Port 0.

Parameters

handle	Address of device's library board descriptor.
compare	The value to compare for Match Mode.

Return values

0	Success.
-1	Failure.

Please see the ioctl(2) man page for information on possible values errno may have in this case.

Note

A compare value can only be set for Digital I/O Port 0.

Referenced by ISR(), and main().

4.19.2.49 DM75xx_Error DM75xx_DIO_Set_Direction (DM75xx_Board_Descriptor ∗ *handle*, dm75xx_dio_port_t *port*, uint8_t *direction*)

Set the direction of the specified Digital I/O Port.

Parameters

handle	Address of device's library board descriptor.
port	The specified Digital I/O Port.
direction	The direction to set for the specified Digital I/O Port.

Return values

0	Success.
-1	Failure.

Please see the ioctl(2) man page for information on possible values errno may have in this case.

Note

Port 0 is bit directional and Port 1 is byte directional.

Referenced by main().

4.19.2.50 DM75xx_Error DM75xx_DIO_Set_Mask (DM75xx_Board_Descriptor ∗ *handle,* uint8_t *mask*)

Set Digital I/O Port 0 Mask.

Parameters

handle	Address of device's library board descriptor.
mask	The mask to set for Digital I/O Port 0.

Return values

0	Success.
-1	Failure.

Please see the ioctl(2) man page for information on possible values errno may have in this case.

Note

A mask value can only be set for Digital I/O Port 0.

Referenced by main().

4.19.2.51 DM75xx_Error DM75xx_DIO_Set_Port (DM75xx_Board_Descriptor ∗ *handle,* dm75xx_dio_port_t *port,* uint8_t *data*)

DM75xx_Library_DIO_Functions DM75xx user library digital input/output.

DM75xx_Library_DCNT_Functions

```
Set a specified Digital I/O Port to the given value.
```

Parameters

handle	Address of device's library board descriptor.
port	The specified Digital I/O Port.
data	The value to set on the Digital I/O Port.

Return values

0	Success.
-1	Failure.

Please see the ioctl(2) man page for information on possible values errno may have in this case.

Referenced by main().

4.19.2.52 DM75xx_Error DM75xx_DSP_CMD_Complete (DM75xx_Board_Descriptor ∗ *handle,* uint8_t ∗ *data*)

Checks if the last command given to the DSP is finished.

Parameters

handle	Address of the device's library board descriptor.
data	This value will be zero if the DSP has completed the instruction and greater than zero if an instruction is still being executed.

Return values

0	Success.
-1	Failure.

Referenced by main().

4.19.2.53 DM75xx_Error DM75xx_DSP_CMD_Send (DM75xx_Board_Descriptor ∗ *handle*, dm75xx_dsp_command_t *command*)

Issue a command to the 7540 onboard DSP.

Parameters

handle	Address of the device's library board descriptor.
command	The DSP Command to issue.

Return values

0	Success.
-1	Failure.

4.19.2.54 DM75xx_Error DM75xx_DSP_CMD_Status (DM75xx_Board_Descriptor ∗ *handle*, dm75xx_dsp_command_t *command*)

Checks whether or not a command successfully completed on the DSP.

Parameters

handle	Address of the device's library board descriptor.
command	The DSP command status that is being checked.

Return values

0	Success.
-1	Failure.

Referenced by main().

4.19.2.55 DM75xx_Error DM75xx_EINT_Polarity_Select (DM75xx_Board_Descriptor ∗ *handle*, dm75xx_ext_polarity_t *polarity*)

Set the External Interrupt polarity.

Parameters

handle	`Address of device's library board descriptor.`

polarity	Positive/Negative polarity select.

Return values

0	Success.
-1	Failure. `Please see the ioctl(2) man page for information on possible values errno` may have in this case.

Referenced by main().

4.19.2.56 DM75xx_Error DM75xx_ETRIG_Polarity_Select (DM75xx_Board_Descriptor ∗ *handle,* dm75xx_ext_polarity_t *polarity*)

DM75xx_Library_EXT_Functions DM75xx user library external trigger/interrupt.

DM75xx_Library_McBSP_Functions

```
Set the External Trigger polarity
```

Parameters

handle	`Address of device's library board descriptor.`
polarity	Positive/Negative polarity select.

Return values

0	Success.
-1	Failure. `Please see the ioctl(2) man page for information on possible values errno` may have in this case.

Referenced by main().

4.19.2.57 DM75xx_Error DM75xx_FIFO_Get_Status (DM75xx_Board_Descriptor ∗ *handle,* uint16_t ∗ *fifo_status*)

DM75xx_Library_STATUS_Functions DM75xx user library status.

DM75xx_Library_EXT_Functions

```
Get current FIFO Status
```

Parameters

handle	Address of device's library board descriptor.
fifo_status	Variable in which to store the current status.

Return values

0	Success.
-1	Failure.

Please see the ioctl(2) man page for information on possible values errno may have in this case.

Referenced by main().

4.19.2.58 DM75xx_Error DM75xx_Get_Temp (DM75xx_Board_Descriptor ∗ *handle,* uint8_t ∗ *temp*)

Get the temperature from the board.

Parameters

handle	Address of the device's library board descriptor.
temp	The temperature returned from the board.

Return values

0	Success
-1	Failure

Referenced by main().

4.19.2.59 DM75xx_Error DM75xx_HSDIN_Clear (DM75xx_Board_Descriptor ∗ *handle*)

Clear High Speed Digital FIFO.

Parameters

handle	Address of device's library board descriptor.

Return values

0	Success.
-1	Failure.

Please see the ioctl(2) man page for information on possible values errno may have in this case.

Referenced by main().

4.19.2.60 DM75xx_Error DM75xx_HSDIN_FIFO_Read (DM75xx_Board_Descriptor ∗ *handle,* uint16_t ∗ *data*)

Read value from High Speed Digital FIFO.

Parameters

handle	Address of device's library board descriptor.
data	Address of the variable to store the data.

Return values

0	Success.
-1	Failure.

Please see the ioctl(2) man page for information on possible values errno may have in this case.

Referenced by main().

4.19.2.61 DM75xx_Error DM75xx_HSDIN_Sample_Signal (DM75xx_Board_Descriptor ∗ *handle,* dm75xx_hsdin_signal_t *signal*)

Set HighSpeed digital sampling signal.

Parameters

handle	`Address of device's library board descriptor.`

signal	Sampling signal to select.

Return values

0	Success.
-1	Failure. `Please see the ioctl(2) man page for information on possible values errno` may have in this case.

Referenced by main().

4.19.2.62 DM75xx_Error DM75xx_HSDIN_Software_Sample (DM75xx_Board_Descriptor ∗ *handle*)

DM75xx_Library_HSDIN_Functions DM75xx user library high speed digital.

DM75xx_Library_DAC_Functions

```
Software high speed digital input sample command
```

Parameters

handle	`Address of device’s library board descriptor.`

Return values

0	Success.
-1	Failure. `Please see the ioctl(2) man page for information on possible values errno` may have in this case.

Note

This function calls DM75xx_DAC_Set_Clock()

4.19.2.63 DM75xx_Error DM75xx_McBSP_ADC_FIFO (DM75xx_Board_Descriptor ∗ *handle,* uint8_t *enable*)

DM75xx_Library_McBSP_Functions DM75xx user library mcbsp.

DM75xx_Library_UIO_Functions

```
Enable/Disable A/D FIFO to DSP
```

Parameters

handle	Address of device’s library board descriptor
enable	0x00 disables, 0xFF enables

Return values

0	Success
-1	Failure

Please see the ioctl(2) man page for information on possible values errno may have in this case.

4.19.2.64 DM75xx_Error DM75xx_McBSP_DAC_FIFO (DM75xx_Board_Descriptor ∗ *handle,* uint8_t *enable*)

Enable/Disable D/A FIFO to DSP.

Parameters

handle	Address of device's library board descriptor
enable	0x00 disables, 0xFF enables

Return values

0	Success
-1	Failure

Please see the ioctl(2) man page for information on possible values errno may have in this case.

4.19.2.65 DM75xx_Error DM75xx_SBUS_Enable (DM75xx_Board_Descriptor ∗ *handle,* dm75xx_sbus_t *sbus,* uint16_t *enable*)

Enable/Disable Syncbus.

Parameters

handle	`Address of device's library board descriptor.`
sbus	The specified SyncBus.
enable	Value determining whether to enable/disable the syncbus.

Return values

0	Success.
-1	Failure. `Please see the ioctl(2) man page for information on possible values errno` may have in this case.

4.19.2.66 DM75xx_Error DM75xx_SBUS_Set_Source (DM75xx_Board_Descriptor ∗ *handle,* dm75xx_sbus_t *sbus,* dm75xx_sbus_src_t *src*)

DM75xx_Library_SBUS_Functions DM75xx user library syncbus.

DM75xx_Library_HSDIN_Functions

```
    Set SyncBus Source
```

Parameters

handle	`Address of device's library board descriptor.`
sbus	The specified SyncBus.
src	Source to set for the specified SyncBus.

Return values

0	Success.
-1	Failure. `Please see the ioctl(2) man page for information on possible values errno` may have in this case.

Referenced by main().

4.19.2.67 DM75xx_Error DM75xx_UIO_Read (DM75xx_Board_Descriptor ∗ *handle*, uint32_t ∗ *data*)

Read the current status of the user I/O.

Parameters

handle	Address of device's library board descriptor
data	Address of the variable to store the read value.

Return values

0	Success
-1	Failure

Please see the ioctl(2) man page for information on possible values errno may have in this case.

4.19.2.68 DM75xx_Error DM75xx_UIO_Select (DM75xx_Board_Descriptor ∗ *handle*, dm75xx_uio_channel_t *channel*, dm75xx_uio_source_t *source*)

DM75xx_Library_UIO_Functions DM75xx user library user I/O.

DM75xx_Library_DIO_Functions

```
Selects the source of a user I/O signal
```

Parameters

handle	Address of device's library board descriptor
channel	The user output channel on which the signal will be sent
source	The source for the signal

Return values

0	Success
-1	Failure

Please see the ioctl(2) man page for information on possible values errno may have in this case.

Referenced by main().

4.19.2.69 DM75xx_Error DM75xx_UIO_Write (DM75xx_Board_Descriptor ∗ *handle*, uint32_t *data*)

Write the value of the user I/O.

Parameters

handle	Address of device's library board descriptor
data	Value to write to the user I/O

Return values

0	Success
-1	Failure

Please see the ioctl(2) man page for information on possible values errno may have in this case.

4.20 DM75xx user library board control functions

Functions

- DM75xx_Error DM75xx_Board_PCI_Master (DM75xx_Board_Descriptor ∗handle, uint8_t ∗pci_master)

  Determine whether or not a device is PCI master capable.
- DM75xx_Error DM75xx_Board_Reset (DM75xx_Board_Descriptor ∗handle)

  Reset a DM75xx device.
- DM75xx_Error DM75xx_Clear_ITMask (DM75xx_Board_Descriptor ∗handle, uint16_t mask)

  Clear Interrupts via Mask.
- DM75xx_Error DM75xx_Clear_IT_Overrun (DM75xx_Board_Descriptor ∗handle)

  Clear Interrupt Overrun Register.
- void DM75xx_Exit_On_Error (DM75xx_Board_Descriptor ∗handle, DM75xx_Error status, char ∗str)

  Tests the return status of a library function, and if it's an error we clean up the board and exit.
- DM75xx_Error DM75xx_Board_Init (DM75xx_Board_Descriptor ∗handle)

  Initialize a Board. This function performs the following to attempt to get the device into a known state:
- DM75xx_Error DM75xx_Interrupt_Enable (DM75xx_Board_Descriptor ∗handle, dm75xx_int_source_t int_↩ source)

  Enable one or more DM75xx interrupt source(s).
- DM75xx_Error DM75xx_Interrupt_Disable (DM75xx_Board_Descriptor ∗handle, dm75xx_int_source_t int_↩ source)

  Disable one or more DM75xx interrupt source(s).
- DM75xx_Error DM75xx_Interrupt_Check (DM75xx_Board_Descriptor ∗handle, dm75xx_int_source_t ∗int_↩ source)

  Returns the value of current active/enabled interrupts on the device.

4.20.1 Detailed Description

4.20.2 Function Documentation

4.20.2.1 DM75xx_Error DM75xx_Board_Init (DM75xx_Board_Descriptor ∗ *handle*)

Initialize a Board. This function performs the following to attempt to get the device into a known state:

Board Reset Clear A/D FIFO Clear D/A 1 FIFO Clear D/A 2 FIFO Clear High Speed Digital FIFO Clear Channel Gain Table Reset Digital I/O Chip Clear Digital Interrupts Clear Interrupts Clear Interrupt Overrun register

Parameters

handle	Address of device's library board descriptor.

Return values

0	Success.
-1	Failure.

Please see the ioctl(2) man page for information on possible values errno may have in this case.

Referenced by main().

4.20.2.2 DM75xx_Error DM75xx_Board_PCI_Master (DM75xx_Board_Descriptor ∗ *handle,* uint8_t ∗ *pci_master*)

Determine whether or not a device is PCI master capable.

Parameters

handle	Address of device's library board descriptor.
pci_master	Address where PCI master capable flag should be stored. Zero will be stored here if the device is not PCI master capable. A non-zero value will be stored here if the device is PCI master capable.

Return values

0	Success.
-1	Failure.

Please see the ioctl(2) man page for information on possible values errno may have in this case.

Referenced by main().

4.20.2.3 DM75xx_Error DM75xx_Board_Reset (DM75xx_Board_Descriptor ∗ *handle*)

Reset a DM75xx device.

Parameters

handle	Address of device's library board descriptor.

Return values

0	Success.
-1	Failure.

Please see the ioctl(2) man page for information on possible values errno may have in this case.

Note

This function does not reset the PLX chip or 8254 chips.

Referenced by main().

4.20.2.4 DM75xx_Error DM75xx_Clear_IT_Overrun (DM75xx_Board_Descriptor ∗ *handle*)

Clear Interrupt Overrun Register.

Parameters

handle	Address of device's library board descriptor.

Return values

0	Success.
-1	Failure.

Please see the ioctl(2) man page for information on possible values errno may have in this case.

4.20.2.5 DM75xx_Error DM75xx_Clear_ITMask (DM75xx_Board_Descriptor ∗ *handle*, uint16_t *mask*)

Clear Interrupts via Mask.

Parameters

handle	Address of device's library board descriptor.

mask	Mask of the interrupt bits to clear.

Return values

0	Success.
-1	Failure.

Please see the ioctl(2) man page for information on possible values errno may have in this case.

4.20.2.6 void DM75xx_Exit_On_Error (**DM75xx_Board_Descriptor** ∗ *handle,* **DM75xx_Error** *status,* char ∗ *str*)

Tests the return status of a library function, and if it's an error we clean up the board and exit.

Parameters

handle	Address of the device's library board descriptor.
status	The return status we are testing.
str	The string to print in the case of failure.

Referenced by ISR(), main(), and sigint_handler().

4.20.2.7 **DM75xx_Error** DM75xx_Interrupt_Check (**DM75xx_Board_Descriptor** ∗ *handle,* **dm75xx_int_source_t** ∗ *int_source*)

Returns the value of current active/enabled interrupts on the device.

Parameters

handle	Address of device's library board descriptor.
int_source	Address of variable to store the returned interrupt enable status

Return values

0	Success.
-1	Failure.

Please see the ioctl(2) man page for information on possible values errno may have in this case.

4.20.2.8 **DM75xx_Error** DM75xx_Interrupt_Disable (**DM75xx_Board_Descriptor** ∗ *handle,* **dm75xx_int_source_t** *int_source*)

Disable one or more DM75xx interrupt source(s).

Parameters

handle	Address of device's library board descriptor.
int_source	Interrupt source to disable.

Return values

0	Success.
-1	Failure.

Please see the ioctl(2) man page for information on possible values errno may have in this case.

Referenced by main().

4.20.2.9 **DM75xx_Error** DM75xx_Interrupt_Enable (**DM75xx_Board_Descriptor** ∗ *handle,* **dm75xx_int_source_t** *int_source*)

Enable one or more DM75xx interrupt source(s).

Parameters

handle	Address of device's library board descriptor.
int_source	Interrupt source to enable.

Return values

0	Success.
-1	Failure.

Please see the ioctl(2) man page for information on possible values errno may have in this case.

Referenced by main().

4.21 DM75xx user library DMA functions

Functions

- DM75xx_Error DM75xx_DMA_Buffer_Write (DM75xx_Board_Descriptor ∗handle, dm75xx_dma_channel_t channel, unsigned long num_ints)

  Copy the User Space buffers data incrementally into our Kernel Space buffer.
- DM75xx_Error DM75xx_DMA_Buffer_Read (DM75xx_Board_Descriptor ∗handle, dm75xx_dma_channel_t channel, unsigned long num_ints)

  Copy the Kernel Space buffers data incrementally into our User Space buffer.
- DM75xx_Error DM75xx_DMA_Buffer_Create (DM75xx_Board_Descriptor ∗handle, uint16_t ∗∗buffer, dm75xx_dma_channel_t channel, uint32_t samples)

  Create a buffer in which the user should place data from the device's DMA buffers.
- DM75xx_Error DM75xx_DMA_Buffer_Free (DM75xx_Board_Descriptor ∗handle, uint16_t ∗∗buffer, dm75xx_dma_channel_t channel)

  Free a buffer previously allocated with DM75xx_DMA_Buffer_Create().
- DM75xx_Error DM75xx_DMA_Init_Arb (DM75xx_Board_Descriptor ∗handle, dm75xx_dma_channel_t channel, dm75xx_dma_source_t source, dm75xx_dma_request_t request, uint32_t samples, uint32_t pci_address)

  Set up direct memory access (DMA) for the given DMA/FIFO channel to/from an arbitrary PCI address.
- DM75xx_Error DM75xx_DMA_Initialize (DM75xx_Board_Descriptor ∗handle, dm75xx_dma_channel_t channel, dm75xx_dma_source_t source, dm75xx_dma_request_t request, uint32_t samples, uint16_t ∗∗buf)

  Set up direct memory access (DMA) for the given DMA/FIFO channel.
- DM75xx_Error DM75xx_DMA_Abort (DM75xx_Board_Descriptor ∗handle, dm75xx_dma_channel_t channel)

  Abort any active transfer on the specified DMA channel.
- DM75xx_Error DM75xx_DMA_Enable (DM75xx_Board_Descriptor ∗handle, dm75xx_dma_channel_t channel, uint8_t enable)

  Set the enable bit for a particular DMA channel. To start DMA after this, call DM75xx_DMA_Start().
- DM75xx_Error DM75xx_DMA_Request_Source (DM75xx_Board_Descriptor ∗handle, dm75xx_dma_channel_t channel, dm75xx_dma_request_t request)

  Set the demand mode request source for a specified DMA channel.
- DM75xx_Error DM75xx_DMA_Start (DM75xx_Board_Descriptor ∗handle, dm75xx_dma_channel_t channel)

  Sets the start bit for a particular DMA Channel. DMA will start if the enable bit has been set by DM75xx_DMA_Enable().

4.21.1 Detailed Description

DM75xx_Library_BrdCtl_Functions

4.21.2 Function Documentation

4.21.2.1 DM75xx_Error DM75xx_DMA_Abort (DM75xx_Board_Descriptor ∗ *handle*, dm75xx_dma_channel_t *channel*)

Abort any active transfer on the specified DMA channel.

Parameters

handle	Address of device's library board descriptor.

channel	The channel on which to abort DMA transfer.

Return values

0	Success
-1	Failure

Referenced by main().

4.21.2.2 DM75xx_Error DM75xx_DMA_Buffer_Create (DM75xx_Board_Descriptor ∗ *handle*, uint16_t ∗∗ *buffer*, dm75xx_dma_channel_t *channel*, uint32_t *samples*)

Create a buffer in which the user should place data from the device's DMA buffers.

Parameters

handle	Address of device's library board descriptor.
buffer	Address of the pointer which create for the user.
channel	The DMA channel for which to create a user-space buffer.
samples	The size of the buffer required in samples.

Return values

0	Success.
-1	Failure.

Note

This function MUST be called if you are planning on use the DMA_Buffer_Read() or DMA_Buffer_Write() function calls to manage your DMA data.

Referenced by main().

4.21.2.3 DM75xx_Error DM75xx_DMA_Buffer_Free (DM75xx_Board_Descriptor ∗ *handle*, uint16_t ∗∗ *buffer*, dm75xx_dma_channel_t *channel*)

Free a buffer previously allocated with DM75xx_DMA_Buffer_Create().

Parameters

handle	Address of device's library board descriptor.
buffer	Address of the pointer which to free.
channel	The DMA Channel's buffer we want to free.

Return values

0	Success.
-1	Failure.

Note

This function MUST be called if you are planning on use the DMA_Buffer_Read() or DMA_Buffer_Write() function calls to manage your DMA data.

Referenced by main().

4.21.2.4 DM75xx_Error DM75xx_DMA_Buffer_Read (DM75xx_Board_Descriptor ∗ *handle*, dm75xx_dma_channel_t *channel*, unsigned long *num_ints*)

Copy the Kernel Space buffers data incrementally into our User Space buffer.

Parameters

handle	Address of device's library board descriptor.
channel	The DMA Channel on which to perform the operation.
num_ints	Number of DMA interrupt received, this helps us keep track of our place in the buffer.

Return values

0	Success
-1	Failure

Note

> Use this function if you would like the library to handle your DMA buffer reads. This function effectively copies from the buffer mapped to the kernel dma buffer by DMA_Initialize() into the buffer allocated by DMA_Buffer↩ _Create().

This function must be used in conjuction with DMA_Buffer_Create() and DMA_Buffer_Free().

The user is more than welcome to manage the buffers via memcpy().

Referenced by main().

4.21.2.5 DM75xx_Error DM75xx_DMA_Buffer_Write (DM75xx_Board_Descriptor ∗ *handle*, dm75xx_dma_channel_t *channel*, unsigned long *num_ints*)

Copy the User Space buffers data incrementally into our Kernel Space buffer.

Parameters

handle	Address of device's library board descriptor.
channel	The DMA Channel on which to perform the operation.
num_ints	Number of DMA interrupt received, this helps us keep track of our place in the buffer

Return values

0	Success
-1	

Note

> Use this function if you would like the library to handle your DMA buffer writes. This function effectively copies to the buffer mapped to the kernel dma buffer by DMA_Initialize() into the buffer allocated by DMA_Buffer_↩ Create().

This function must be used in conjuction with DMA_Buffer_Create() and DMA_Buffer_Free().

The user is more than welcome to manage the buffers via memcpy().

Referenced by main().

4.21.2.6 DM75xx_Error DM75xx_DMA_Enable (DM75xx_Board_Descriptor ∗ *handle*, dm75xx_dma_channel_t *channel*, uint8_t *enable*)

Set the enable bit for a particular DMA channel. To start DMA after this, call DM75xx_DMA_Start().

Parameters

handle	Address of device's library board descriptor.
channel	The DMA channel to enable on the specified device.
enable	0 for disable, >0 for enable

Returns

0

Success.

Returns

-1

Failure.

Referenced by main().

4.21.2.7 DM75xx_Error DM75xx_DMA_Init_Arb (DM75xx_Board_Descriptor ∗ *handle,* dm75xx_dma_channel_t *channel,* dm75xx_dma_source_t *source,* dm75xx_dma_request_t *request,* uint32_t *samples,* uint32_t *pci_address*)

Set up direct memory access (DMA) for the given DMA/FIFO channel to/from an arbitrary PCI address.

Parameters

handle	Address of device's library board descriptor.
channel	The DMA channel to use.
source	The FIFO to/from which DMA will be used.
request	The DREQ line source.
samples	The number of samples desired from the FIFO via DMA. For HD DMA, this value is required to be a multiple of half the FIFO size of the board. This is a limitation of the UTC1 DREQ source used for HD DMA.
pci_address	The PCI Address to transfer to/from.

Return values

0	Success.
-1	Failure. errno may be set as follows: • `EAGAIN` DMA has already been initialized for fifo. • `EINVAL` fifo is not valid. • `ENOMEM` Kernel memory allocation failed. • `EOPNOTSUPP` The device is not PCI master capable. Please see the ioctl(2) man page for information on other possible values errno may have in this case.

Note

When using DMA to/from arbitrary PCI addresses, no kernel buffer is allocated for DMA.
DMA to arbitrary PCI addresses can cause unknown behavior if you are not careful about what you are doing.
As the application designer, you have some flexibility to configure DMA as as your purpose suits. However, if this function fails with errno ENOMEM, you need to allocate smaller buffers.

Referenced by main().

4.21.2.8 DM75xx_Error DM75xx_DMA_Initialize (DM75xx_Board_Descriptor ∗ *handle,* dm75xx_dma_channel_t *channel,* dm75xx_dma_source_t *source,* dm75xx_dma_request_t *request,* uint32_t *samples,* uint16_t ∗∗ *buf*)

Set up direct memory access (DMA) for the given DMA/FIFO channel.

Parameters

handle	Address of device's library board descriptor.
channel	The DMA channel to use.
source	The FIFO to/from which DMA will be used.
request	The DREQ line source.
samples	The number of samples desired from the FIFO via DMA. For HD DMA, this value is required to be a multiple of half the FIFO size of the board. This is a limitation of the UTC1 DREQ source used for HD DMA.
buf	The user space buffer which will be mapped to the kernel space DMA buffer

Return values

0	Success.
-1	Failure. errno may be set as follows: • `EAGAIN` DMA has already been initialized for fifo. • `EINVAL` fifo is not valid. • `ENOMEM` Kernel memory allocation failed. • `EOPNOTSUPP` The device is not PCI master capable. Please see the ioctl(2) man page for information on other possible values errno may have in this case.

Note

Since a single DMA buffer must exist in physically contiguous memory, the probability of DMA buffer allocation failure increases as both the number of buffers to allocate and the size of each buffer increase.
Factors beyond the number and size of DMA buffers affect the probability of DMA buffer allocation failure. These factors include the number of processes on the system, how much system memory is already in use, and the presence of processes (such as the X server) which use a lot of memory.
System memory can be a scarce resource. Every system entity needs some amount of memory. Memory is being allocated and released all the time.
The default value for DM75xx_MAX_DMA_BUFFER_SIZE is 131,072 bytes (128 kilobytes or 65k samples). If you need to change this, edit include/dm75xx_driver.h, save the changes, recompile the driver, and reload the driver.
As the application designer, you have some flexibility to configure DMA as as your purpose suits. However, if this function fails with errno ENOMEM, you need to allocate smaller buffers.
This function also maps a user space buffer to the kernel memory buffer allocated for DMA. This was done to prevent successive calls to copy_to_user() or copy_from_user() as both of these functions will sleep if the user space buffer was paged out.

Referenced by main().

4.21.2.9 DM75xx_Error DM75xx_DMA_Request_Source (DM75xx_Board_Descriptor ∗ *handle,* dm75xx_dma_channel_t *channel,* dm75xx_dma_request_t *request*)

Set the demand mode request source for a specified DMA channel.

Parameters

handle	Address of the device's library board descriptor.
channel	The specified channel for which to set the request source.
request	The demand mode request source to set.

Returns

0

Success

Returns

-1

Failure

4.21.2.10 DM75xx_Error DM75xx_DMA_Start (DM75xx_Board_Descriptor ∗ *handle*, dm75xx_dma_channel_t *channel*)

Sets the start bit for a particular DMA Channel. DMA will start if the enable bit has been set by DM75xx_DMA_↩ Enable().

Parameters

handle	Address of device's library board descriptor.
channel	The DMA channel to start on the specified device.

Returns

0

Success.

Returns

-1

Failure.

Referenced by main().

4.22 DM75xx user library general functions

Functions

- DM75xx_Error DM75xx_Board_Close (DM75xx_Board_Descriptor ∗handle)

  Close a DM75xx device file.
- DM75xx_Error DM75xx_Board_Open (uint8_t dev_num, DM75xx_Board_Descriptor ∗∗handle)

  Open a DM75xx device file.
- DM75xx_Error DM75xx_FIFO_Size (DM75xx_Board_Descriptor ∗handle, unsigned int ∗data)

  Retrieve the FIFO size of the board from the kernel space device descriptor.
- DM75xx_Error DM75xx_Board_Type (DM75xx_Board_Descriptor ∗handle, dm75xx_board_t ∗data)

  Determine the family of the board (DM7520 or SDM7540/8540).
- DM75xx_Error DM75xx_InstallISR (DM75xx_Board_Descriptor ∗handle, void(∗isr_fnct)(unsigned int status))

  Install userspace ISR.
- DM75xx_Error DM75xx_RemoveISR (DM75xx_Board_Descriptor ∗handle)

  Uninstall userspace ISR.
- void ∗ DM75xx_WaitForInterrupt (void ∗ptr)

  Function that will have its own thread and wait for interrupts to occur. Once an interrupt is received this function will call our callback ISR and pass it the interrupt status.

4.22.1 Detailed Description

DM75xx_Library_DMA_Functions

4.22.2 Function Documentation

4.22.2.1 DM75xx_Error DM75xx_Board_Close (DM75xx_Board_Descriptor ∗ *handle*)

Close a DM75xx device file.

Parameters

handle	Address of device's library board descriptor.

Return values

0	Success.
-1	Failure. errno may be set as follows: • `ENODATA` handle is NULL. Please see the close(2) man page for information on other possible values errno may have in this case.

Note

This function frees the memory allocated for the library board descriptor.
When processing the close request, the driver disables PLX PCI interrupts, disables PLX local interrupt input, and disables PLX DMA channel 0/1 interrupts.

Warning

Whether or not this function succeeds, the library board descriptor must not be referenced in any way after the function returns.

Referenced by main().

4.22.2.2 DM75xx_Error DM75xx_Board_Open (uint8_t *dev_num,* DM75xx_Board_Descriptor ∗∗ *handle*)

Open a DM75xx device file.

Parameters

dev_num	Minor number of DM75xx device file.
handle	Address where address of memory allocated for library device descriptor should be stored. If the first open of a device file fails, then NULL will be stored here.

Return values

0	Success.
-1	Failure. errno may be set as follows: • `EBUSY` The DM75xx device file with minor number dev_num is already open. • `ENODEV` dev_num is not a valid DM75xx minor number; 2.4 kernel only. • `ENOMEM` Library device descriptor memory allocation failed. • `ENXIO` dev_num is not a valid DM75xx minor number; 2.6 kernel only. Please see the open(2) man page for information on other possible values errno may have in this case.

Note

Once a device file is open, it cannot be opened again until it is closed.
When processing the open request, the driver disables & clears all device interrupts, enables PLX PC↵ I interrupts, enables PLX local interrupt input, and enables PLX DMA channel 0/1 interrupts.

Referenced by main().

4.22.2.3 DM75xx_Error DM75xx_Board_Type (DM75xx_Board_Descriptor ∗ *handle,* dm75xx_board_t ∗ *data*)

Determine the family of the board (DM7520 or SDM7540/8540).

Parameters

handle	Address of the device's library board descriptor.
data	This value will be returned as 0 if the board is a DM7520 or a positive value if the board is an SDM7540/8540.

Return values

0	Success
-1	Failure

4.22.2.4 DM75xx_Error DM75xx_FIFO_Size (DM75xx_Board_Descriptor ∗ *handle,* unsigned int ∗ *data*)

Retrieve the FIFO size of the board from the kernel space device descriptor.

Parameters

handle	Address of the device's library board descriptor.
data	Address of the variable in which to store the fifo size.

Return values

0	Success
-1	Failure

Note

```
This function does not calculate the value of the FIFO upon each call.  The
FIFO size is determined at 'insmod' time and is stored in the drivers device
descriptor.
```

Referenced by main().

4.22.2.5 DM75xx_Error DM75xx_InstallISR (DM75xx_Board_Descriptor ∗ *handle*, void(∗)(unsigned int status) *isr_fnct*)

Install userspace ISR.

Parameters

handle	Address of the device's library board descriptor.
isr_fnct	Function pointer to the user ISR that will be called in the event of an interrupt

Return values

0	Success
-1	Failure

Note

Any previously installed ISR will be removed before installing a new ISR
This function creates another thread that runs DM75xx_WaitForInterrupt(). This thread is removed by a call to DM75xx_RemoveISR().

Referenced by main().

4.22.2.6 DM75xx_Error DM75xx_RemoveISR (DM75xx_Board_Descriptor ∗ *handle*)

Uninstall userspace ISR.

Parameters

handle	Address of the device's library board descriptor.

Return values

0	Success
-1	Failure

Note

```
This function makes an ioctl call into the kernel which makes a call to wake
the thread from the select() call.
```

Referenced by main().

4.22.2.7 void∗ DM75xx_WaitForInterrupt (void ∗ *ptr*)

Function that will have its own thread and wait for interrupts to occur. Once an interrupt is received this function will call our callback ISR and pass it the interrupt status.

Parameters

ptr	Pointer to be typecasted to the device handle.

Return values

0	Success
-1	Failure

Note

```
This function should not be called directly by the user.
```

4.23 DM75xx user library user timer/counter control

Functions

- DM75xx_Error DM75xx_UTC_Set_Clock_Source (DM75xx_Board_Descriptor ∗handle, dm75xx_utc_timer↩_t utc, dm75xx_utc_clk_t source)

  Set a User Timer/Counter Clock Source.
- DM75xx_Error DM75xx_UTC_Set_Gate (DM75xx_Board_Descriptor ∗handle, dm75xx_utc_timer_t utc, dm75xx_utc_gate gate)

  Set a User Timer/Counter Gate.
- DM75xx_Error DM75xx_UTC_Set_Mode (DM75xx_Board_Descriptor ∗handle, dm75xx_utc_timer_t utc, dm75xx_utc_mode mode)

  Set a User Timer/Counter Mode.
- DM75xx_Error DM75xx_UTC_Get_Mode (DM75xx_Board_Descriptor ∗handle, dm75xx_utc_timer_t utc, uint16_t ∗mode)

  Set a User Timer/Counter Mode.
- DM75xx_Error DM75xx_UTC_Set_Divisor (DM75xx_Board_Descriptor ∗handle, dm75xx_utc_timer_t utc, uint16_t rate)

  Set a User Timer/Counter Divisor.
- DM75xx_Error DM75xx_UTC_Get_Count (DM75xx_Board_Descriptor ∗handle, dm75xx_utc_timer_t utc, uint16_t ∗count)

  Return current value of a User Timer/Counter.
- DM75xx_Error DM75xx_UTC_Get_Status (DM75xx_Board_Descriptor ∗handle, dm75xx_utc_timer_t utc_↩select, uint8_t ∗utc_status)

  Return current status of a User Timer/Counter.
- DM75xx_Error DM75xx_UTC_Setup (DM75xx_Board_Descriptor ∗handle, dm75xx_utc_timer_t utc, dm75xx_utc_clk_t source, dm75xx_utc_gate gate, dm75xx_utc_mode mode, uint16_t divisor)

  Setup a User Timer/Counter.

4.23.1 Detailed Description

DM75xx_Library_General_Functions

4.23.2 Function Documentation

4.23.2.1 DM75xx_Error DM75xx_UTC_Get_Count (DM75xx_Board_Descriptor ∗ *handle,* dm75xx_utc_timer_t *utc,* uint16_t ∗ *count*)

Return current value of a User Timer/Counter.

Parameters

handle	Address of device's library board descriptor.
utc	User Timer/Counter we are configuring.
count	Variable to store the current count in.

Return values

0	Success.
-1	Failure.

Please see the ioctl(2) man page for information on possible values errno may have in this case.

Referenced by main().

4.23.2.2 DM75xx_Error DM75xx_UTC_Get_Mode (DM75xx_Board_Descriptor ∗ *handle,* dm75xx_utc_timer_t *utc,* uint16_t ∗ *mode*)

Set a User Timer/Counter Mode.

Parameters

handle	Address of device's library board descriptor.
utc	Which user timer/counter's mode to read.
mode	Variable to store the retrieved mode value.

Return values

0	Success.
-1	Failure.

Please see the ioctl(2) man page for information on possible values errno may have in this case.

Referenced by main().

4.23.2.3 DM75xx_Error DM75xx_UTC_Get_Status (DM75xx_Board_Descriptor ∗ *handle,* dm75xx_utc_timer_t *utc_select,* uint8_t ∗ *utc_status*)

Return current status of a User Timer/Counter.

Parameters

handle	Address of device's library board descriptor.
utc_select	User Timer/Counter we are configuring.
utc_status	Variable to store the current status in.

Return values

0	Success.
-1	Failure.

Please see the ioctl(2) man page for information on possible values errno may have in this case.

Referenced by main().

4.23.2.4 DM75xx_Error DM75xx_UTC_Set_Clock_Source (DM75xx_Board_Descriptor ∗ *handle,* dm75xx_utc_timer_t *utc,* dm75xx_utc_clk_t *source*)

Set a User Timer/Counter Clock Source.

Parameters

handle	Address of device's library board descriptor.
utc	User Timer/Counter we are configuring.
source	The User Timer/Counter source to be set.

Return values

0	Success.
-1	Failure.

Please see the ioctl(2) man page for information on possible values errno may have in this case.

Referenced by main().

4.23.2.5 DM75xx_Error DM75xx_UTC_Set_Divisor (DM75xx_Board_Descriptor ∗ *handle,* dm75xx_utc_timer_t *utc,* uint16_t *rate*)

Set a User Timer/Counter Divisor.

Parameters

handle	Address of device's library board descriptor.
utc	User Timer/Counter we are configuring.
rate	The rate to set for this User Timer/Counter

Return values

0	Success.
-1	Failure.

Please see the ioctl(2) man page for information on possible values errno may have in this case.

Referenced by main().

4.23.2.6 DM75xx_Error DM75xx_UTC_Set_Gate (DM75xx_Board_Descriptor ∗ *handle*, dm75xx_utc_timer_t *utc*, dm75xx_utc_gate *gate*)

Set a User Timer/Counter Gate.

Parameters

handle	Address of device's library board descriptor.
utc	User Timer/Counter we are configuring.
gate	The User Timer/Counter gate option.

Return values

0	Success.
-1	Failure.

Please see the ioctl(2) man page for information on possible values errno may have in this case.

Referenced by main().

4.23.2.7 DM75xx_Error DM75xx_UTC_Set_Mode (DM75xx_Board_Descriptor ∗ *handle*, dm75xx_utc_timer_t *utc*, dm75xx_utc_mode *mode*)

Set a User Timer/Counter Mode.

Parameters

handle	Address of device's library board descriptor.
utc	User Timer/Counter we are configuring.
mode	The User Timer/Counter mode to be set.

Return values

0	Success.
-1	Failure.

Please see the ioctl(2) man page for information on possible values errno may have in this case.

Referenced by main().

4.23.2.8 DM75xx_Error DM75xx_UTC_Setup (DM75xx_Board_Descriptor ∗ *handle*, dm75xx_utc_timer_t *utc*, dm75xx_utc_clk_t *source*, dm75xx_utc_gate *gate*, dm75xx_utc_mode *mode*, uint16_t *divisor*)

Setup a User Timer/Counter.

Parameters

handle	Address of device's library board descriptor.
utc	User Timer/Counter we are configuring.
source	The User Timer/Counter source to be set.
gate	The User Timer/Counter gate option to set.
mode	The User Timer/Counter mode option to set.
divisor	The divisor to set the User Timer/Counter with.

Return values

0	Success.
-1	Failure.

Please see the ioctl(2) man page for information on possible values errno may have in this case.

Referenced by main().

4.24 DM75xx user library burst clock control

Functions

- DM75xx_Error DM75xx_BCLK_Get_Count (DM75xx_Board_Descriptor ∗handle, uint16_t ∗data)

  Get the current Burst Clock count.
- DM75xx_Error DM75xx_BCLK_Set_Count (DM75xx_Board_Descriptor ∗handle, uint16_t data)

  Set the current Burst Clock count.
- DM75xx_Error DM75xx_BCLK_Set_Rate (DM75xx_Board_Descriptor ∗handle, dm75xx_bclk_freq_t freq, float rate, float ∗actualRate)

  Set the Burst Clock rate.
- DM75xx_Error DM75xx_BCLK_Set_Start (DM75xx_Board_Descriptor ∗handle, dm75xx_bclk_start_t start)

  Set Burst Clock start trigger.
- DM75xx_Error DM75xx_BCLK_Set_Frequency (DM75xx_Board_Descriptor ∗handle, dm75xx_bclk_freq_↩ t freq)

  Set the Burst Clock primary frequency.
- DM75xx_Error DM75xx_BCLK_Setup (DM75xx_Board_Descriptor ∗handle, dm75xx_bclk_start_t start, dm75xx_bclk_freq_t freq, float rate, float ∗actualRate)

  Setup Burst Clock.

4.24.1 Detailed Description

DM75xx_Library_UTC_Funtions

4.24.2 Function Documentation

4.24.2.1 DM75xx_Error DM75xx_BCLK_Get_Count (DM75xx_Board_Descriptor ∗ *handle,* uint16_t ∗ *data*)

Get the current Burst Clock count.

Parameters

handle	Address of device's library board descriptor.
data	Address of the variable to store the value.

Return values

0	Success.
-1	Failure.

Please see the ioctl(2) man page for information on possible values errno may have in this case.

4.24.2.2 DM75xx_Error DM75xx_BCLK_Set_Count (DM75xx_Board_Descriptor ∗ *handle,* uint16_t *data*)

Set the current Burst Clock count.

Parameters

handle	Address of device's library board descriptor.
data	Value to write to the Burst Clock count.

Return values

0	Success.
-1	Failure.

Please see the ioctl(2) man page for information on possible values errno may have in this case.

4.24.2.3 DM75xx_Error DM75xx_BCLK_Set_Frequency (DM75xx_Board_Descriptor ∗ *handle,* dm75xx_bclk_freq_t *freq*)

Set the Burst Clock primary frequency.

Parameters

handle	Address of device's library board descriptor.
freq	Frequency to select.

Return values

0	Success.
-1	Failure.

Please see the ioctl(2) man page for information on possible values errno may have in this case.

4.24.2.4 DM75xx_Error DM75xx_BCLK_Set_Rate (DM75xx_Board_Descriptor ∗ *handle,* dm75xx_bclk_freq_t *freq,* float *rate,* float ∗ *actualRate*)

Set the Burst Clock rate.

Parameters

handle	Address of device's library board descriptor.
freq	Set the Burst Clock primary frequency.
rate	The desired Burst Clock rate.
actualRate	The actual rate set.

Return values

0	Success.
-1	Failure.

Please see the ioctl(2) man page for information on possible values errno may have in this case.

Referenced by main().

4.24.2.5 DM75xx_Error DM75xx_BCLK_Set_Start (DM75xx_Board_Descriptor ∗ *handle,* dm75xx_bclk_start_t *start*)

Set Burst Clock start trigger.

Parameters

handle	Address of device's library board descriptor.
start	Start trigger to set.

Return values

0	Success.
-1	Failure.

Please see the ioctl(2) man page for information on possible values errno may have in this case.

Referenced by main().

4.24.2.6 DM75xx_Error DM75xx_BCLK_Setup (DM75xx_Board_Descriptor ∗ *handle*, dm75xx_bclk_start_t *start*, dm75xx_bclk_freq_t *freq*, float *rate*, float ∗ *actualRate*)

Setup Burst Clock.

Parameters

handle	Address of device's library board descriptor.
start	Burst Clock start trigger.
freq	Burst Clock primary frequency.
rate	Rate at which to set the clock.
actualRate	Rate at which the clock is actually set.

Return values

0	Success.
-1	Failure.

Please see the ioctl(2) man page for information on possible values errno may have in this case.

Referenced by main().

4.25 DM75xx user library pacer clock control

Functions

- DM75xx_Error DM75xx_PCLK_Set_Frequency (DM75xx_Board_Descriptor ∗handle, dm75xx_pclk_freq_↩ t pclk_freq)

  Set the Pacer Clock frequency.
- DM75xx_Error DM75xx_PCLK_Set_Source (DM75xx_Board_Descriptor ∗handle, dm75xx_pclk_select_↩ t pclk_select)

  Set the Pacer Clock source.
- DM75xx_Error DM75xx_PCLK_Set_Start (DM75xx_Board_Descriptor ∗handle, dm75xx_pclk_start_t pclk↩ _start)

  Set the Pacer Clock start trigger.
- DM75xx_Error DM75xx_PCLK_Set_Stop (DM75xx_Board_Descriptor ∗handle, dm75xx_pclk_stop_t pclk_↩ stop)

  Set the Pacer Clock stop trigger.
- DM75xx_Error DM75xx_PCLK_Read (DM75xx_Board_Descriptor ∗handle, uint32_t ∗pacer_value)

  Read the current pacer clock value.
- DM75xx_Error DM75xx_PCLK_Set_Trigger_Mode (DM75xx_Board_Descriptor ∗handle, dm75xx_pclk_↩ mode_t pclk_mode)

  Set the Pacer Clock trigger mode.
- DM75xx_Error DM75xx_PCLK_Set_Count (DM75xx_Board_Descriptor ∗handle, uint32_t count)

  Set the Pacer Clock Count.
- DM75xx_Error DM75xx_PCLK_Set_Rate (DM75xx_Board_Descriptor ∗handle, dm75xx_pclk_freq_t freq, float rate, float ∗actualRate)

  Set the Pacer Clock Rate.
- DM75xx_Error DM75xx_PCLK_Setup (DM75xx_Board_Descriptor ∗handle, dm75xx_pclk_select_t pclk_↩ select, dm75xx_pclk_freq_t pclk_freq, dm75xx_pclk_mode_t pclk_mode, dm75xx_pclk_start_t pclk_start, dm75xx_pclk_stop_t pclk_stop, float rate, float ∗actualRate)

  Setup the Pacer Clock.
- DM75xx_Error DM75xx_PCLK_Start (DM75xx_Board_Descriptor ∗handle)

  Software Pacer Clock Start.
- DM75xx_Error DM75xx_PCLK_Stop (DM75xx_Board_Descriptor ∗handle)

  Software Pacer Clock Stop.

4.25.1 Detailed Description

DM75xx_Library_BCLK_Funtions

4.25.2 Function Documentation

4.25.2.1 DM75xx_Error DM75xx_PCLK_Read (DM75xx_Board_Descriptor ∗ *handle,* uint32_t ∗ *pacer_value*)

Read the current pacer clock value.

Parameters

handle	Address of device's library board descriptor.
pacer_value	Address of the variable to store the value of the pacer

Return values

0	Success.
-1	Failure.

Please see the ioctl(2) man page for information on possible values errno may have in this case.

4.25.2.2 DM75xx_Error DM75xx_PCLK_Set_Count (DM75xx_Board_Descriptor ∗ *handle,* uint32_t *count*)

Set the Pacer Clock Count.

Parameters

handle	Address of device's library board descriptor.
count	Pacer Clock count

Return values

0	Success.
-1	Failure.

Please see the ioctl(2) man page for information on possible values errno may have in this case.

4.25.2.3 DM75xx_Error DM75xx_PCLK_Set_Frequency (DM75xx_Board_Descriptor ∗ *handle,* dm75xx_pclk_freq_t *pclk_freq*)

Set the Pacer Clock frequency.

Parameters

handle	Address of device's library board descriptor.
pclk_freq	Frequency to set for the pacer clock.

Return values

0	Success.
-1	Failure.

Please see the ioctl(2) man page for information on possible values errno may have in this case.

Referenced by main().

4.25.2.4 DM75xx_Error DM75xx_PCLK_Set_Rate (DM75xx_Board_Descriptor ∗ *handle,* dm75xx_pclk_freq_t *freq,* float *rate,* float ∗ *actualRate*)

Set the Pacer Clock Rate.

Parameters

handle	Address of device's library board descriptor.
freq	Pacer Clock primary frequency.
rate	Rate desired or the Pacer Clock.
actualRate	Address to store the actual rate value.

Return values

0	Success.
-1	Failure.

Please see the ioctl(2) man page for information on possible values errno may have in this case.

Referenced by main().

4.25.2.5 DM75xx_Error DM75xx_PCLK_Set_Source (DM75xx_Board_Descriptor ∗ *handle,* dm75xx_pclk_select_t *pclk_select*)

Set the Pacer Clock source.

Parameters

handle	Address of device's library board descriptor.
pclk_select	Source for the Pacer Clock

Return values

0	Success.
-1	Failure.

Please see the ioctl(2) man page for information on possible values errno may have in this case.

Referenced by main().

4.25.2.6 DM75xx_Error DM75xx_PCLK_Set_Start (DM75xx_Board_Descriptor ∗ *handle,* dm75xx_pclk_start_t *pclk_start*)

Set the Pacer Clock start trigger.

Parameters

handle	Address of device's library board descriptor.
pclk_start	Start trigger for the Pacer Clock

Return values

0	Success.
-1	Failure.

Please see the ioctl(2) man page for information on possible values errno may have in this case.

Referenced by main().

4.25.2.7 DM75xx_Error DM75xx_PCLK_Set_Stop (DM75xx_Board_Descriptor ∗ *handle,* dm75xx_pclk_stop_t *pclk_stop*)

Set the Pacer Clock stop trigger.

Parameters

handle	Address of device's library board descriptor.
pclk_stop	Stop trigger for the Pacer Clock

Return values

0	Success.
-1	Failure.

Please see the ioctl(2) man page for information on possible values errno may have in this case.

Referenced by main().

4.25.2.8 DM75xx_Error DM75xx_PCLK_Set_Trigger_Mode (DM75xx_Board_Descriptor ∗ *handle,* dm75xx_pclk_mode_t *pclk_mode*)

Set the Pacer Clock trigger mode.

Parameters

handle	Address of device's library board descriptor.

pclk_mode	Mode in which to set the pacer clock.

Return values

0	Success.
-1	Failure.

Please see the ioctl(2) man page for information on possible values errno may have in this case.

Referenced by main().

4.25.2.9 DM75xx_Error DM75xx_PCLK_Setup (DM75xx_Board_Descriptor ∗ *handle,* dm75xx_pclk_select_t *pclk_select,* dm75xx_pclk_freq_t *pclk_freq,* dm75xx_pclk_mode_t *pclk_mode,* dm75xx_pclk_start_t *pclk_start,* dm75xx_pclk_stop_t *pclk_stop,* float *rate,* float ∗ *actualRate*)

Setup the Pacer Clock.

Parameters

handle	Address of device's library board descriptor.
pclk_select	Select Internal/External Pacer Clock.
pclk_freq	Select the primary clock frequency.
pclk_mode	Select the trigger mode.
pclk_start	Select the start trigger.
pclk_stop	Select the stop trigger.
rate	Desired rate for the Pacer Clock.
actualRate	Rate actually set.

Return values

0	Success.
-1	Failure.

Please see the ioctl(2) man page for information on possible values errno may have in this case.

Referenced by main().

4.25.2.10 DM75xx_Error DM75xx_PCLK_Start (DM75xx_Board_Descriptor ∗ *handle*)

Software Pacer Clock Start.

Parameters

handle	Address of device's library board descriptor.

Return values

0	Success.
-1	Failure.

Please see the ioctl(2) man page for information on possible values errno may have in this case.

Referenced by main().

4.25.2.11 DM75xx_Error DM75xx_PCLK_Stop (DM75xx_Board_Descriptor ∗ *handle*)

Software Pacer Clock Stop.

Parameters

handle	Address of device's library board descriptor.

Return values

0	Success.
-1	Failure.

Please see the ioctl(2) man page for information on possible values errno may have in this case.

Referenced by main().

4.26 DM75xx user library channel gain table

Functions

- DM75xx_Error DM75xx_CGT_Reset (DM75xx_Board_Descriptor ∗handle)

  Reset Channel Gain Table.
- DM75xx_Error DM75xx_CGT_Clear (DM75xx_Board_Descriptor ∗handle)

  Clear Channel Gain Table.
- DM75xx_Error DM75xx_CGT_Create_Entry (dm75xx_cgt_entry_t ∗cgt, uint16_t ∗cgt_entry)

  Create a channel gain table entry.
- DM75xx_Error DM75xx_CGT_Write (DM75xx_Board_Descriptor ∗handle, dm75xx_cgt_entry_t cgt)

  Write a channel gain table entry. This function utilizes DM75xx_CGT_Create_Entry() to create the 16 bit entry.
- DM75xx_Error DM75xx_CGT_Latch (DM75xx_Board_Descriptor ∗handle, dm75xx_cgt_entry_t cgt)

  Write ADC channel gain table latch for single channel sampling.
- DM75xx_Error DM75xx_CGT_Enable (DM75xx_Board_Descriptor ∗handle, uint16_t enable)

  Enable/disable A/D channel gain table.
- DM75xx_Error DM75xx_DT_Enable (DM75xx_Board_Descriptor ∗handle, uint16_t enable)

  Enable/disable Digital Table.
- DM75xx_Error DM75xx_DT_Write_Entry (DM75xx_Board_Descriptor ∗handle, uint8_t data)

  Write Digital Table entry.
- DM75xx_Error DM75xx_CGT_Pause (DM75xx_Board_Descriptor ∗handle, uint16_t pause)

  Pause the Channel Gain Table.

4.26.1 Detailed Description

DM75xx_Library_PCLK_Functions

4.26.2 Function Documentation

4.26.2.1 DM75xx_Error DM75xx_CGT_Clear (DM75xx_Board_Descriptor ∗ *handle*)

Clear Channel Gain Table.

Parameters

handle	Address of device's library board descriptor.

Return values

0	Success.
-1	Failure.

Please see the ioctl(2) man page for information on possible values errno may have in this case.

4.26.2.2 DM75xx_Error DM75xx_CGT_Create_Entry (dm75xx_cgt_entry_t ∗ *cgt*, uint16_t ∗ *cgt_entry*)

Create a channel gain table entry.

Parameters

cgt	Struct that holds the values for the channel gain table.

cgt_entry	The channel gain table converted to a uint16_t for register entry.

Return values

0	Success.
-1	Failure.

Note

This function should not be called by the user.

4.26.2.3 DM75xx_Error DM75xx_CGT_Enable (DM75xx_Board_Descriptor ∗ *handle,* uint16_t *enable*)

Enable/disable A/D channel gain table.

Parameters

handle	`Address of device's library board descriptor.`
enable	A 0 denotes CGT Disable and CG Latch Enable, a 1 denotes CGT Enable and CG Latch Disable.

Return values

0	Success.
-1	Failure. `Please see the ioctl(2) man page for information on possible values errno` may have in this case.

Referenced by main().

4.26.2.4 DM75xx_Error DM75xx_CGT_Latch (DM75xx_Board_Descriptor ∗ *handle,* dm75xx_cgt_entry_t *cgt*)

Write ADC channel gain table latch for single channel sampling.

Parameters

handle	`Address of device's library board descriptor.`
cgt	Channel gain table entry to write to the latch.

Return values

0	Success.
-1	Failure. `Please see the ioctl(2) man page for information on possible values errno` may have in this case.

Referenced by main().

4.26.2.5 DM75xx_Error DM75xx_CGT_Pause (DM75xx_Board_Descriptor ∗ *handle,* uint16_t *pause*)

Pause the Channel Gain Table.

Parameters

handle	`Address of device's library board descriptor.`
pause	Enable/Disable CGT Pause.

Return values

0	Success.
-1	Failure. `Please see the ioctl(2) man page for information on possible values errno` may have in this case.

Note

```
Pause is ignored in burst mode.
```

4.26.2.6 DM75xx_Error DM75xx_CGT_Reset (DM75xx_Board_Descriptor ∗ *handle*)

Reset Channel Gain Table.

Parameters

handle	Address of device's library board descriptor.

Return values

0	Success.
-1	Failure.

Please see the ioctl(2) man page for information on possible values errno may have in this case.

4.26.2.7 DM75xx_Error DM75xx_CGT_Write (DM75xx_Board_Descriptor ∗ *handle,* dm75xx_cgt_entry_t *cgt*)

Write a channel gain table entry. This function utilizes DM75xx_CGT_Create_Entry() to create the 16 bit entry.

Parameters

handle	`Address of device's library board descriptor.`
cgt	The channel gain table entry to write.

Return values

0	Success.
-1	Failure. `Please see the ioctl(2) man page for information on possible values errno` may have in this case.

Referenced by main().

4.26.2.8 DM75xx_Error DM75xx_DT_Enable (DM75xx_Board_Descriptor ∗ *handle,* uint16_t *enable*)

Enable/disable Digital Table.

Parameters

handle	`Address of device's library board descriptor.`
enable	Enable/Disable the Digital Table and Digital I/O Port 1.

Return values

0	Success.
-1	Failure. `Please see the ioctl(2) man page for information on possible values errno` may have in this case.

4.26.2.9 DM75xx_Error DM75xx_DT_Write_Entry (DM75xx_Board_Descriptor ∗ *handle,* uint8_t *data*)

Write Digital Table entry.

Parameters

handle	`Address of device's library board descriptor.`
data	Entry to add to the digital table.

Return values

0	Success.
-1	Failure. `Please see the ioctl(2) man page for information on possible values errno` may have in this case.

4.27 DM75xx type definition header file

Modules

- DM75xx type enumerations
- DM75xx type definition structures

4.27.1 Detailed Description

4.28 DM75xx type enumerations

Modules

- DM75xx type PCI enumerations
- DM75xx type DSP enumerations
- DM75xx type DMA
- DM75xx type Interrupt
- DM75xx type Analog DIO
- DM75xx type User Output
- DM75xx type Digital Input/Output
- DM75xx type External Trigger/Interrupt
- DM75xx type SyncBus enumerations
- DM75xx type HighSpeed Digital enumerations
- DM75xx type Digital to Analog enumerations
- DM75xx type Analog to Digital enumerations
- DM75xx type Burst Clock enumerations
- DM75xx type Pacer Clock enumerations
- DM75xx type timer/counter enumerations

4.28.1 Detailed Description

4.29 DM75xx type PCI enumerations

Typedefs

- typedef enum dm75xx_pci_region_num dm75xx_pci_region_num_t

  Standard PCI region number type.
- typedef enum dm75xx_pci_region_access_size dm75xx_pci_region_access_size_t

  Standard PCI region access size type.
- typedef enum _dm75xx_board dm75xx_board_t

  DM75xx Board Type.

Enumerations

- enum dm75xx_pci_region_num { DM75xx_PLX_MEM = 0, DM75xx_PLX_IO, DM75xx_LAS0, DM75xx_LAS1 }

  Standard PCI region number.
- enum dm75xx_pci_region_access_size { DM75xx_PCI_REGION_ACCESS_8 = 0, DM75xx_PCI_REGION↩_ACCESS_16, DM75xx_PCI_REGION_ACCESS_32 }

  Desired size in bits of access to standard PCI region.
- enum _dm75xx_board { DM75xx_BOARD_DM7520 = 0, DM75xx_BOARD_SDM7540 }

  DM75xx Board.

4.29.1 Detailed Description

4.29.2 Enumeration Type Documentation

4.29.2.1 enum _dm75xx_board

DM75xx Board.

Enumerator

DM75xx_BOARD_DM7520	DM7520
DM75xx_BOARD_SDM7540	SDM7540

Definition at line 125 of file dm75xx_types.h.

4.29.2.2 enum dm75xx_pci_region_access_size

Desired size in bits of access to standard PCI region.

Enumerator

DM75xx_PCI_REGION_ACCESS_8	8-bit access
DM75xx_PCI_REGION_ACCESS_16	16-bit access
DM75xx_PCI_REGION_ACCESS_32	32-bit access

Definition at line 92 of file dm75xx_types.h.

4.29.2.3 enum dm75xx_pci_region_num

Standard PCI region number.

Enumerator

DM75xx_PLX_MEM	Memory-mapped PLX registers
DM75xx_PLX_IO	I/O-mapped PLX registers
DM75xx_LAS0	Memory-mapped LAS 0 registers
DM75xx_LAS1	Memory-mapped LAS 1 registers

Definition at line 53 of file dm75xx_types.h.

4.30 DM75xx type DSP enumerations

Typedefs

- typedef enum _dm75xx_dsp_command dm75xx_dsp_command_t

  DSP Command Type.

Enumerations

- enum _dm75xx_dsp_command {
  DM75xx_DSP_CAL_AUTO = 1, DM75xx_DSP_FLASH_DOWNLOAD = 2, DM75xx_DSP_USER_RUN = 3,
  DM75xx_DSP_USER_UPGRADE = 4,
  DM75xx_DSP_INT_FLASH_ERASE = 5, DM75xx_DSP_EXT_FLASH_ERASE = 6, DM75xx_DSP_ATTE↩
  NTION = 7, DM75xx_DSP_CAL_DEFAULT = 8,
  DM75xx_DSP_CAL_VERSION = 10, DM75xx_DSP_BOOT_VERSION = 11 }

  DSP Command.

4.30.1 Detailed Description

DM75xx_Types_PCI_Enumerations

4.30.2 Enumeration Type Documentation

4.30.2.1 enum _dm75xx_dsp_command

DSP Command.

Enumerator

DM75xx_DSP_CAL_AUTO	Auto Calibration of SDM7540
DM75xx_DSP_FLASH_DOWNLOAD	Internal Flash Download
DM75xx_DSP_USER_RUN	Run User Program
DM75xx_DSP_USER_UPGRADE	Upgrade User Program
DM75xx_DSP_INT_FLASH_ERASE	Erase Internal Flash
DM75xx_DSP_EXT_FLASH_ERASE	Erase External Flash
DM75xx_DSP_ATTENTION	Check if DSP still alive
DM75xx_DSP_CAL_DEFAULT	Load Default Calibration
DM75xx_DSP_CAL_VERSION	Get Calibration Algorithm Version
DM75xx_DSP_BOOT_VERSION	Get Boot Loader Version

Definition at line 151 of file dm75xx_types.h.

4.31 DM75xx type DMA

Typedefs

- typedef enum _dm75xx_dma_flag dm75xx_dma_flag_t

    DMA Control Flag Type.
- typedef enum _dm75xx_dma_reset dm75x_dma_reset_t

    DMA Status Reset Flag Type.
- typedef enum _dm75xx_dma_request dm75xx_dma_request_t

    DMA Demand Mode Source Type.
- typedef enum _dm75xx_dma_source dm75xx_dma_source_t

    DMA Source Type.
- typedef enum _dm75xx_dma_channel dm75xx_dma_channel_t

    DMA Channel Type.

Enumerations

- enum _dm75xx_dma_flag {
  DM75xx_DMA_FLAG_INIT = 0x01, DM75xx_DMA_FLAG_MMAP = 0x02, DM75xx_DMA_FLAG_RESET = 0x04, DM75xx_DMA_FLAG_NONDEMAND = 0x08,
  DM75xx_DMA_FLAG_STATUS = 0x10, DM75xx_DMA_FLAG_ARB = 0x20 }

    DMA Control Flag.
- enum _dm75xx_dma_reset { DM75xx_DMA_RESET_SEL = 0x01, DM75xx_DMA_RESET_VAL = 0x10 }

    DMA Status Reset Flag.
- enum _dm75xx_dma_request {
  DM75xx_DMA_DEMAND_DISABLE = 0, DM75xx_DMA_DEMAND_SCNT_ADC = 1, DM75xx_DMA_DEM←AND_SCNT_DAC1 = 2, DM75xx_DMA_DEMAND_SCNT_DAC2 = 3,
  DM75xx_DMA_DEMAND_UTC1 = 4, DM75xx_DMA_DEMAND_FIFO_ADC = 8, DM75xx_DMA_DEMAN←D_FIFO_DAC1 = 9, DM75xx_DMA_DEMAND_FIFO_DAC2 = 10 }

    DMA Demand Mode Source.
- enum _dm75xx_dma_source { DM75xx_DMA_FIFO_ADC = 0, DM75xx_DMA_FIFO_DAC1, DM75xx_DM←A_FIFO_DAC2, DM75xx_DMA_FIFO_HSDIN }

    DMA Local Source.
- enum _dm75xx_dma_channel { DM75xx_DMA_CHANNEL_0 = 0, DM75xx_DMA_CHANNEL_1 }

    DMA Channel.

4.31.1 Detailed Description

DM75xx_Types_DSP_Enumerations

4.31.2 Enumeration Type Documentation

4.31.2.1 enum _dm75xx_dma_channel

DMA Channel.

Enumerator

DM75xx_DMA_CHANNEL_0	DMA Channel 0
DM75xx_DMA_CHANNEL_1	DMA Channel 1

Definition at line 337 of file dm75xx_types.h.

4.31.2.2 enum _dm75xx_dma_flag

DMA Control Flag.

Enumerator

DM75xx_DMA_FLAG_INIT DMA Initialized
DM75xx_DMA_FLAG_MMAP DMA Memory Map
DM75xx_DMA_FLAG_RESET DMA Reset DREQ
DM75xx_DMA_FLAG_NONDEMAND DMA Non Demand Mode
DM75xx_DMA_FLAG_STATUS DMA Channel Status
DM75xx_DMA_FLAG_ARB DMA Arbitrary

Definition at line 213 of file dm75xx_types.h.

4.31.2.3 enum _dm75xx_dma_request

DMA Demand Mode Source.

Enumerator

DM75xx_DMA_DEMAND_DISABLE Request Disable
DM75xx_DMA_DEMAND_SCNT_ADC A/D Sample Counter
DM75xx_DMA_DEMAND_SCNT_DAC1 D/A 1 Sample Counter
DM75xx_DMA_DEMAND_SCNT_DAC2 D/A 2 Sample Counter
DM75xx_DMA_DEMAND_UTC1 User Timer/Counter 1
DM75xx_DMA_DEMAND_FIFO_ADC A/D FIFO Half Full
DM75xx_DMA_DEMAND_FIFO_DAC1 D/A 1 FIFO Half Empty
DM75xx_DMA_DEMAND_FIFO_DAC2 D/A 2 FIFO Half Empty

Definition at line 267 of file dm75xx_types.h.

4.31.2.4 enum _dm75xx_dma_reset

DMA Status Reset Flag.

Enumerator

DM75xx_DMA_RESET_SEL DMA 0
DM75xx_DMA_RESET_VAL DMA Channel Reset Value

Definition at line 248 of file dm75xx_types.h.

4.31.2.5 enum _dm75xx_dma_source

DMA Local Source.

Enumerator

DM75xx_DMA_FIFO_ADC DMA A/D FIFO
DM75xx_DMA_FIFO_DAC1 DMA D/A 1 FIFO
DM75xx_DMA_FIFO_DAC2 DMA D/A 2 FIFO
DM75xx_DMA_FIFO_HSDIN DMA HSDIN FIFO

Definition at line 310 of file dm75xx_types.h.

4.32 DM75xx type Interrupt

Typedefs

- typedef enum _dm75xx_int_source dm75xx_int_source_t

  Interrupt Source Type.

Enumerations

- enum _dm75xx_int_source {
  DM75xx_INT_FIFO_WRITE = 0x0001, DM75xx_INT_CGT_RESET = 0x0002, DM75xx_INT_RESERVED = 0x0004, DM75xx_INT_CGT_PAUSE = 0x0008,
  DM75xx_INT_ABOUT = 0x0010, DM75xx_INT_DELAY = 0x0020, DM75xx_INT_SCNT_ADC = 0x0040, D↩ M75xx_INT_SCNT_DAC1 = 0x0080,
  DM75xx_INT_SCNT_DAC2 = 0x0100, DM75xx_INT_UTC1 = 0x0200, DM75xx_INT_UTC1_INV = 0x0400, DM75xx_INT_UTC2 = 0x0800,
  DM75xx_INT_DIO = 0x1000, DM75xx_INT_EXTERNAL = 0x2000, DM75xx_INT_ETRIG_RISING = 0x4000, DM75xx_INT_ETRIG_FALLING = 0x8000,
  DM75xx_INT_DMA_0 = 0x00200000, DM75xx_INT_DMA_1 = 0x00400000, DM75xx_INT_ALGDIO_POS↩ _PIN1 = 0x04000000, DM75xx_INT_ALGDIO_POS_PIN2 = 0x08000000,
  DM75xx_INT_ALGDI0_NEG_PIN1 = 0x10000000, DM75xx_INT_ALGDIO_NEG_PIN2 = 0x20000000 }

  Interrupt Source.

4.32.1 Detailed Description

DM75xx_Types_DMA_Enumerations

4.32.2 Enumeration Type Documentation

4.32.2.1 enum _dm75xx_int_source

Interrupt Source.

Enumerator

DM75xx_INT_FIFO_WRITE	FIFO Write
DM75xx_INT_CGT_RESET	Reset CGT
DM75xx_INT_RESERVED	Reserved
DM75xx_INT_CGT_PAUSE	Pause CGT
DM75xx_INT_ABOUT	About Counter Out
DM75xx_INT_DELAY	Delay Counter Out
DM75xx_INT_SCNT_ADC	A/D Sample Counter
DM75xx_INT_SCNT_DAC1	D/A 1 Update Counter
DM75xx_INT_SCNT_DAC2	D/A 2 Update Counter
DM75xx_INT_UTC1	User Timer/Counter 1 Out
DM75xx_INT_UTC1_INV	Inverted User Timer/Counter 1 Out
DM75xx_INT_UTC2	User Timer/Counter 2 Out
DM75xx_INT_DIO	Digital Interrupt
DM75xx_INT_EXTERNAL	External Interrupt
DM75xx_INT_ETRIG_RISING	External Trigger Rising Edge

DM75xx_INT_ETRIG_FALLING	External Trigger Falling Edge
DM75xx_INT_DMA_0	DMA Channel 0 – ENABLED BY DEFAULT
DM75xx_INT_DMA_1	DMA Channel 1 – ENABLED BY DEFAULT
DM75xx_INT_ALGDIO_POS_PIN1	Analog DIO Pin 1 Pos Edge
DM75xx_INT_ALGDIO_POS_PIN2	Analog DIO Pin 2 Pos Edge
DM75xx_INT_ALGDI0_NEG_PIN1	Analog DIO Pin 1 Neg Edge
DM75xx_INT_ALGDIO_NEG_PIN2	Analog DIO Pin 2 Neg Edge

Definition at line 366 of file dm75xx_types.h.

4.33 DM75xx type Analog DIO

Typedefs

- typedef enum _dm75xx_algdio_mask dm75xx_algdio_mask_t

  Analog DIO Mask Type.
- typedef enum _dm75xx_algdio_pin dm75xx_algdio_pin_t

  Analog DIO Pin Type.
- typedef enum
  _dm75xx_algdio_direction dm75xx_algdio_direction_t

  Analog DIO Direction Type.

Enumerations

- enum _dm75xx_algdio_mask { DM75xx_ALGDIO_MASKED = 0, DM75xx_ALGDIO_UNMASKED }

  Analog DIO Mask.
- enum _dm75xx_algdio_pin { DM75xx_ALGDIO_PIN1 = 0, DM75xx_ALGDIO_PIN2 }

  Analog DIO Pins.
- enum _dm75xx_algdio_direction { DM75xx_ALGDIO_INPUT = 0, DM75xx_ALGDIO_OUTPUT }

  Analog DIO Direction.

4.33.1 Detailed Description

DM75xx_Types_INT_Enumerations

4.33.2 Enumeration Type Documentation

4.33.2.1 enum _dm75xx_algdio_direction

Analog DIO Direction.

Enumerator

DM75xx_ALGDIO_INPUT	Input
DM75xx_ALGDIO_OUTPUT	Output

Definition at line 513 of file dm75xx_types.h.

4.33.2.2 enum _dm75xx_algdio_mask

Analog DIO Mask.

Enumerator

DM75xx_ALGDIO_MASKED	Masked
DM75xx_ALGDIO_UNMASKED	Unmasked

Definition at line 475 of file dm75xx_types.h.

4.33.2.3 enum _dm75xx_algdio_pin

Analog DIO Pins.

Enumerator

DM75xx_ALGDIO_PIN1	Pin 1
DM75xx_ALGDIO_PIN2	Pin 2

Definition at line 494 of file dm75xx_types.h.

4.34 DM75xx type User Output

Typedefs

- typedef enum _dm75xx_uio_channel dm75xx_uio_channel_t

  User I/O Channel Type.
- typedef enum _dm75xx_uio_source dm75xx_uio_source_t

  User I/O Source Type.

Enumerations

- enum _dm75xx_uio_channel { DM75xx_UIO0 = 0, DM75xx_UIO1 }

  User I/O Channel.
- enum _dm75xx_uio_source { DM75xx_UIO_ADC = 0, DM75xx_UIO_DAC1, DM75xx_UIO_DAC2, DM75xx↩ _UIO_PRG }

  User I/O Source.

4.34.1 Detailed Description

DM75xx_Types_ALGDIO_Enumerations

4.34.2 Enumeration Type Documentation

4.34.2.1 enum _dm75xx_uio_channel

User I/O Channel.

Enumerator

DM75xx_UIO0	User I/O Channel 0
DM75xx_UIO1	User I/O Channel 1

Definition at line 542 of file dm75xx_types.h.

4.34.2.2 enum _dm75xx_uio_source

User I/O Source.

Enumerator

DM75xx_UIO_ADC	A/D Conversion Signal
DM75xx_UIO_DAC1	D/A 1 Update
DM75xx_UIO_DAC2	D/A 2 Update
DM75xx_UIO_PRG	Software Programmable

Definition at line 561 of file dm75xx_types.h.

4.35 DM75xx type Digital Input/Output

Typedefs

- typedef enum _dm75xx_dio_clk dm75xx_dio_clk_t

  Digital I/O Clock Type.
- typedef enum _dm75xx_dio_mode dm75xx_dio_mode_t

  Digital I/O IRQ Mode Type.
- typedef enum _dm75xx_dio_port dm75xx_dio_port_t

  Digital I/O Port Type.

Enumerations

- enum _dm75xx_dio_clk { DM75xx_DIO_CLK_8MHZ = 0, DM75xx_DIO_CLK_UTC1 }

  Digital I/O Clock.
- enum _dm75xx_dio_mode { DM75xx_DIO_MODE_EVENT = 0, DM75xx_DIO_MODE_MATCH }

  Digital I/O IRQ Mode.
- enum _dm75xx_dio_port { DM75xx_DIO_PORT0 = 0, DM75xx_DIO_PORT1 }

  Digital I/O Port.

4.35.1 Detailed Description

DM75xx_Types_UIO_Enumerations

4.35.2 Enumeration Type Documentation

4.35.2.1 enum _dm75xx_dio_clk

Digital I/O Clock.

Enumerator

DM75xx_DIO_CLK_8MHZ	8MHZ Clock
DM75xx_DIO_CLK_UTC1	Programmable Clock

Definition at line 598 of file dm75xx_types.h.

4.35.2.2 enum _dm75xx_dio_mode

Digital I/O IRQ Mode.

Enumerator

DM75xx_DIO_MODE_EVENT	Event Mode
DM75xx_DIO_MODE_MATCH	Match Mode

Definition at line 617 of file dm75xx_types.h.

4.35.2.3 enum _dm75xx_dio_port

Digital I/O Port.

Enumerator

DM75xx_DIO_PORT0	Port 0
DM75xx_DIO_PORT1	Port 1

Definition at line 636 of file dm75xx_types.h.

4.36 DM75xx type External Trigger/Interrupt

Typedefs

- typedef enum _dm75xx_ext_polarity dm75xx_ext_polarity_t

  Polarity Type.

Enumerations

- enum _dm75xx_ext_polarity { DM75xx_EXT_POLARITY_POS = 0, DM75xx_EXT_POLARITY_NEG }

  Polarity.

4.36.1 Detailed Description

DM75xx_Types_DIO_Enumerations

4.36.2 Enumeration Type Documentation

4.36.2.1 enum _dm75xx_ext_polarity

Polarity.

Enumerator

DM75xx_EXT_POLARITY_POS	Positive Edge
DM75xx_EXT_POLARITY_NEG	Negative Edge

Definition at line 665 of file dm75xx_types.h.

4.37 DM75xx type SyncBus enumerations

Typedefs

- typedef enum _dm75xx_sbus dm75xx_sbus_t

  SyncBus Enumeration Type.
- typedef enum _dm75xx_sbus_src dm75xx_sbus_src_t

  SyncBus Source Select Type.

Enumerations

- enum _dm75xx_sbus { DM75xx_SBUS0 = 0, DM75xx_SBUS1, DM75xx_SBUS2 }

  SyncBus Enumerations.
- enum _dm75xx_sbus_src {
  DM75xx_SBUS_SRC_SOFT_ADC = 0, DM75xx_SBUS_SRC_PCLK = 1, DM75xx_SBUS_SRC_PCLK_S↩
  TART = 1, DM75xx_SBUS_SRC_BCLK = 2,
  DM75xx_SBUS_SRC_PCLK_STOP = 2, DM75xx_SBUS_SRC_DIG_IT = 3, DM75xx_SBUS_SRC_DAC1 =
  3, DM75xx_SBUS_SRC_ETRIG = 4,
  DM75xx_SBUS_SRC_DAC2 = 4, DM75xx_SBUS_SRC_DAC_UPDATE = 5, DM75xx_SBUS_SRC_EPCLK
  = 5, DM75xx_SBUS_SRC_DAC_CLK = 6,
  DM75xx_SBUS_SRC_ETRIG2 = 6, DM75xx_SBUS_SRC_UTC2 = 7 }

  SyncBus Source Select.

4.37.1 Detailed Description

DM75xx_Types_EXT_Enumerations

4.37.2 Enumeration Type Documentation

4.37.2.1 enum _dm75xx_sbus

SyncBus Enumerations.

Enumerator

DM75xx_SBUS0	SyncBus 0
DM75xx_SBUS1	SyncBus 1
DM75xx_SBUS2	SyncBus 2

Definition at line 694 of file dm75xx_types.h.

4.37.2.2 enum _dm75xx_sbus_src

SyncBus Source Select.

Enumerator

DM75xx_SBUS_SRC_SOFT_ADC	Software A/D Start
DM75xx_SBUS_SRC_PCLK	Pacer Clock
DM75xx_SBUS_SRC_PCLK_START	Software Pacer Start
DM75xx_SBUS_SRC_BCLK	Burst Clock
DM75xx_SBUS_SRC_PCLK_STOP	Software Pacer Stop

DM75xx_SBUS_SRC_DIG_IT	Digital Interrupt
DM75xx_SBUS_SRC_DAC1	Software D/A1 Update
DM75xx_SBUS_SRC_ETRIG	External Trigger
DM75xx_SBUS_SRC_DAC2	Software D/A2 Update
DM75xx_SBUS_SRC_DAC_UPDATE	Simultaneous D/A Update
DM75xx_SBUS_SRC_EPCLK	External Pacer Clock
DM75xx_SBUS_SRC_DAC_CLK	D/A Clock
DM75xx_SBUS_SRC_ETRIG2	External Trigger
DM75xx_SBUS_SRC_UTC2	User Timer/Counter 2 Out

Definition at line 717 of file dm75xx_types.h.

4.38 DM75xx type HighSpeed Digital enumerations

Typedefs

- typedef enum _dm75xx_hsdin_signal dm75xx_hsdin_signal_t

    HSDIN Sampling Signal Type.

Enumerations

- enum _dm75xx_hsdin_signal {
DM75xx_HSDIN_SIGNAL_SOFTWARE = 0, DM75xx_HSDIN_SIGNAL_ADC, DM75xx_HSDIN_SIGNAL_↩
UTC0, DM75xx_HSDIN_SIGNAL_UTC1,
DM75xx_HSDIN_SIGNAL_UTC2, DM75xx_HSDIN_SIGNAL_EPCLK, DM75xx_HSDIN_SIGNAL_ETRIG }

    HSDIN Sampling Signal.

4.38.1 Detailed Description

DM75xx_Types_SBUS_Enumerations

4.38.2 Enumeration Type Documentation

4.38.2.1 enum _dm75xx_hsdin_signal

HSDIN Sampling Signal.

Enumerator

DM75xx_HSDIN_SIGNAL_SOFTWARE	Software
DM75xx_HSDIN_SIGNAL_ADC	A/D Conversion Signal
DM75xx_HSDIN_SIGNAL_UTC0	User Timer/Counter 0 Out
DM75xx_HSDIN_SIGNAL_UTC1	User Timer/Counter 1 Out
DM75xx_HSDIN_SIGNAL_UTC2	User Timer/Counter 2 Out
DM75xx_HSDIN_SIGNAL_EPCLK	External Pacer Clock
DM75xx_HSDIN_SIGNAL_ETRIG	External Trigger

Definition at line 794 of file dm75xx_types.h.

4.39 DM75xx type Digital to Analog enumerations

Typedefs

- typedef enum _dm75xx_dac_freq dm75xx_dac_freq_t

  DAC primary clock source type.
- typedef enum _dm75xx_dac_clk_stop dm75xx_dac_clk_stop_t

  DAC clock stop source type.
- typedef enum _dm75xx_dac_clk_start dm75xx_dac_clk_start_t

  DAC clock start source type.
- typedef enum _dm75xx_dac_mode dm75xx_dac_mode_t

  DAC Cycle Mode Type.
- typedef enum _dm75xx_dac_update_src dm75xx_dac_update_src_t

  DAC Update Source Type.
- typedef enum _dm75xx_dac_range dm75xx_dac_range_t

  DAC Output Range Type.
- typedef enum _dm75xx_dac_channel dm75xx_dac_channel_t
- typedef enum _dm75xx_dac_clk_mode dm75xx_dac_clk_mode_t

  DAC Clock Mode Type.

Enumerations

- enum _dm75xx_dac_freq { DM75xx_DAC_FREQ_8_MHZ = 0, DM75xx_DAC_FREQ_20_MHZ }

  DAC primary clock source.
- enum _dm75xx_dac_clk_stop {
  DM75xx_DAC_CLK_STOP_SOFTWARE_PACER = 0, DM75xx_DAC_CLK_STOP_ETRIG, DM75xx_DA←C_CLK_STOP_DIG_IT, DM75xx_DAC_CLK_STOP_UTC2,
  DM75xx_DAC_CLK_STOP_SBUS0, DM75xx_DAC_CLK_STOP_SBUS1, DM75xx_DAC_CLK_STOP_SB←US2, DM75xx_DAC_CLK_STOP_SOFTWARE,
  DM75xx_DAC_CLK_STOP_DAC1_UCNT, DM75xx_DAC_CLK_STOP_DAC2_UCNT }

  DAC clock stop source.
- enum _dm75xx_dac_clk_start {
  DM75xx_DAC_CLK_START_SOFTWARE_PACER = 0, DM75xx_DAC_CLK_START_ETRIG, DM75xx_D←AC_CLK_START_DIG_IT, DM75xx_DAC_CLK_START_UTC2,
  DM75xx_DAC_CLK_START_SBUS0, DM75xx_DAC_CLK_START_SBUS1, DM75xx_DAC_CLK_START←_SBUS2, DM75xx_DAC_CLK_START_SOFTWARE }

  DAC clock start source.
- enum _dm75xx_dac_mode { DM75xx_DAC_MODE_NOT_CYCLE = 0, DM75xx_DAC_MODE_CYCLE }

  DAC Cycle Mode.
- enum _dm75xx_dac_update_src {
  DM75xx_DAC_UPDATE_SOFTWARE = 0, DM75xx_DAC_UPDATE_CGT, DM75xx_DAC_UPDATE_CL←OCK, DM75xx_DAC_UPDATE_EPCLK,
  DM75xx_DAC_UPDATE_SBUS0, DM75xx_DAC_UPDATE_SBUS1, DM75xx_DAC_UPDATE_SBUS2 }

  DAC Update Source.
- enum _dm75xx_dac_range { DM75xx_DAC_RANGE_UNIPOLAR_5 = 0, DM75xx_DAC_RANGE_UNIPO←LAR_10, DM75xx_DAC_RANGE_BIPOLAR_5, DM75xx_DAC_RANGE_BIPOLAR_10 }

  DAC Output Range.
- enum _dm75xx_dac_channel { DM75xx_DAC1 = 1, DM75xx_DAC2 }

  DAC channels.
- enum _dm75xx_dac_clk_mode { DM75xx_DAC_CLK_FREE_RUN = 0, DM75xx_DAC_CLK_START_STOP }

  DAC Clock Mode.

4.39.1 Detailed Description

DM75xx_Types_HSDIN_Enumerations

4.39.2 Typedef Documentation

4.39.2.1 typedef enum _dm75xx_dac_channel dm75xx_dac_channel_t

DAC channel type

Definition at line 1057 of file dm75xx_types.h.

4.39.3 Enumeration Type Documentation

4.39.3.1 enum _dm75xx_dac_channel

DAC channels.

Note: These are given these values specifically so they can be bitwise combined and compared.

Enumerator

DM75xx_DAC1	Digital to Analog channel 1
DM75xx_DAC2	Digital to Analog channel 2

Definition at line 1044 of file dm75xx_types.h.

4.39.3.2 enum _dm75xx_dac_clk_mode

DAC Clock Mode.

Enumerator

DM75xx_DAC_CLK_FREE_RUN	Free Run Mode
DM75xx_DAC_CLK_START_STOP	Start/Stop Mode

Definition at line 1062 of file dm75xx_types.h.

4.39.3.3 enum _dm75xx_dac_clk_start

DAC clock start source.

Enumerator

DM75xx_DAC_CLK_START_SOFTWARE_PACER	Software pacer start
DM75xx_DAC_CLK_START_ETRIG	External trigger
DM75xx_DAC_CLK_START_DIG_IT	Digital Interrupt
DM75xx_DAC_CLK_START_UTC2	User Timer/Counter 2 out
DM75xx_DAC_CLK_START_SBUS0	SyncBus 0
DM75xx_DAC_CLK_START_SBUS1	SyncBus 1
DM75xx_DAC_CLK_START_SBUS2	SyncBus 2
DM75xx_DAC_CLK_START_SOFTWARE	Software DAC clock start

Definition at line 913 of file dm75xx_types.h.

4.39.3.4 enum _dm75xx_dac_clk_stop

DAC clock stop source.

Enumerator

DM75xx_DAC_CLK_STOP_SOFTWARE_PACER	Software Pacer Stop
DM75xx_DAC_CLK_STOP_ETRIG	External trigger
DM75xx_DAC_CLK_STOP_DIG_IT	Digital Interrupt
DM75xx_DAC_CLK_STOP_UTC2	User Timer/Counter 2 out
DM75xx_DAC_CLK_STOP_SBUS0	Syncbus 0
DM75xx_DAC_CLK_STOP_SBUS1	Syncbus 1
DM75xx_DAC_CLK_STOP_SBUS2	Syncbus 2
DM75xx_DAC_CLK_STOP_SOFTWARE	Software DAC clock stop
DM75xx_DAC_CLK_STOP_DAC1_UCNT	DAC1 Update Counter
DM75xx_DAC_CLK_STOP_DAC2_UCNT	DAC2 Update Counter

Definition at line 862 of file dm75xx_types.h.

4.39.3.5 enum _dm75xx_dac_freq

DAC primary clock source.

Enumerator

DM75xx_DAC_FREQ_8_MHZ	8 MHz Clock
DM75xx_DAC_FREQ_20_MHZ	20 MHz Clock

Definition at line 843 of file dm75xx_types.h.

4.39.3.6 enum _dm75xx_dac_mode

DAC Cycle Mode.

Enumerator

DM75xx_DAC_MODE_NOT_CYCLE	Not cycle
DM75xx_DAC_MODE_CYCLE	Cycle

Definition at line 956 of file dm75xx_types.h.

4.39.3.7 enum _dm75xx_dac_range

DAC Output Range.

Enumerator

DM75xx_DAC_RANGE_UNIPOLAR_5	Unipolar 0V to 5V
DM75xx_DAC_RANGE_UNIPOLAR_10	Unipolar 0V to 10V
DM75xx_DAC_RANGE_BIPOLAR_5	Bipolar -5V to 5V
DM75xx_DAC_RANGE_BIPOLAR_10	Bipolar -10V to 10V

Definition at line 1014 of file dm75xx_types.h.

4.39.3.8 enum _dm75xx_dac_update_src

DAC Update Source.

Enumerator

DM75xx_DAC_UPDATE_SOFTWARE	Software DAC Update
DM75xx_DAC_UPDATE_CGT	CGT Controlled Update
DM75xx_DAC_UPDATE_CLOCK	DAC Clock
DM75xx_DAC_UPDATE_EPCLK	External pacer clock
DM75xx_DAC_UPDATE_SBUS0	Syncbus 0
DM75xx_DAC_UPDATE_SBUS1	Syncbus 1
DM75xx_DAC_UPDATE_SBUS2	Syncbus 2

Definition at line 975 of file dm75xx_types.h.

4.40 DM75xx type Analog to Digital enumerations

Typedefs

- typedef enum _dm75xx_adc_scnt_src dm75xx_adc_scnt_src_t

  ADC Sample Counter Source Type.
- typedef enum
  _dm75xx_adc_conv_signal dm75xx_adc_conv_signal_t

  ADC Conversion Signal Select type.

Enumerations

- enum _dm75xx_adc_scnt_src { DM75xx_ADC_SCNT_SRC_CGT = 0, DM75xx_ADC_SCNT_SRC_FIFO }

  ADC Sample Counter Source.
- enum _dm75xx_adc_conv_signal {
  DM75xx_ADC_CONV_SIGNAL_SOFTWARE = 0, DM75xx_ADC_CONV_SIGNAL_PCLK, DM75xx_ADC↩
  _CONV_SIGNAL_BCLK, DM75xx_ADC_CONV_SIGNAL_DIG_IT,
  DM75xx_ADC_CONV_SIGNAL_DAC1_MRKR1, DM75xx_ADC_CONV_SIGNAL_DAC2_MRKR1, D↩
  M75xx_ADC_CONV_SIGNAL_SBUS0, DM75xx_ADC_CONV_SIGNAL_SBUS1,
  DM75xx_ADC_CONV_SIGNAL_SBUS2 }

  ADC Conversion Signal Select.

4.40.1 Detailed Description

DM75xx_Types_DAC_Enumerations

4.40.2 Enumeration Type Documentation

4.40.2.1 enum _dm75xx_adc_conv_signal

ADC Conversion Signal Select.

Enumerator

DM75xx_ADC_CONV_SIGNAL_SOFTWARE	Software
DM75xx_ADC_CONV_SIGNAL_PCLK	Pacer Clock
DM75xx_ADC_CONV_SIGNAL_BCLK	Burst Clock
DM75xx_ADC_CONV_SIGNAL_DIG_IT	Digital Interrupt
DM75xx_ADC_CONV_SIGNAL_DAC1_MRKR1	DAC1 Marker Bit 1
DM75xx_ADC_CONV_SIGNAL_DAC2_MRKR1	DAC2 Marker Bit 2
DM75xx_ADC_CONV_SIGNAL_SBUS0	SyncBus 0
DM75xx_ADC_CONV_SIGNAL_SBUS1	SyncBus 1
DM75xx_ADC_CONV_SIGNAL_SBUS2	SyncBus 2

Definition at line 1110 of file dm75xx_types.h.

4.40.2.2 enum _dm75xx_adc_scnt_src

ADC Sample Counter Source.

Enumerator

DM75xx_ADC_SCNT_SRC_CGT	Reset Channel Gain Table
DM75xx_ADC_SCNT_SRC_FIFO	A/D FIFO Write

Definition at line 1091 of file dm75xx_types.h.

4.41 DM75xx type Burst Clock enumerations

Typedefs

- typedef enum _dm75xx_bclk_freq dm75xx_bclk_freq_t

  Burst Clock primary frequency type.
- typedef enum _dm75xx_bclk_start dm75xx_bclk_start_t

  Burst Clock Start Trigger Type.

Enumerations

- enum _dm75xx_bclk_freq { DM75xx_BCLK_FREQ_8_MHZ = 0, DM75xx_BCLK_FREQ_20_MHZ }

  Burst Clock primary frequency.
- enum _dm75xx_bclk_start {
  DM75xx_BCLK_START_SOFTWARE = 0, DM75xx_BCLK_START_PACER, DM75xx_BCLK_START_ET↩
  RIG, DM75xx_BCLK_START_DIG_IT,
  DM75xx_BCLK_START_SBUS0, DM75xx_BCLK_START_SBUS1, DM75xx_BCLK_START_SBUS2 }

  Burst Clock Start Trigger.

4.41.1 Detailed Description

DM75xx_Types_ADC_Enumerations

4.41.2 Enumeration Type Documentation

4.41.2.1 enum _dm75xx_bclk_freq

Burst Clock primary frequency.

Enumerator

DM75xx_BCLK_FREQ_8_MHZ	8 MHz Clock
DM75xx_BCLK_FREQ_20_MHZ	20 MHz Clock

Definition at line 1168 of file dm75xx_types.h.

4.41.2.2 enum _dm75xx_bclk_start

Burst Clock Start Trigger.

Enumerator

DM75xx_BCLK_START_SOFTWARE	Software A/D
DM75xx_BCLK_START_PACER	Pacer Clock
DM75xx_BCLK_START_ETRIG	External Trigger
DM75xx_BCLK_START_DIG_IT	Digital Interrupt
DM75xx_BCLK_START_SBUS0	SyncBus 0
DM75xx_BCLK_START_SBUS1	SyncBus 1
DM75xx_BCLK_START_SBUS2	SyncBus 2

Definition at line 1188 of file dm75xx_types.h.

4.42 DM75xx type Pacer Clock enumerations

Typedefs

- typedef enum _dm75xx_pclk_mode dm75xx_pclk_mode_t

  Pacer Clock Trigger Mode Type.
- typedef enum _dm75xx_pclk_stop dm75xx_pclk_stop_t

  Pacer Clock Stop Type.
- typedef enum _dm75xx_pclk_start dm75xx_pclk_start_t

  Pacer Clock Start Type.
- typedef enum _dm75xx_pclk_select dm75xx_pclk_select_t

  Pacer Clock Select Type.
- typedef enum _dm75xx_pclk_freq dm75xx_pclk_freq_t

  Pacer Clock Frequency type.

Enumerations

- enum _dm75xx_pclk_mode { DM75xx_PCLK_NO_REPEAT = 0, DM75xx_PCLK_REPEAT }

  Pacer Clock Trigger Mode.
- enum _dm75xx_pclk_stop {
  DM75xx_PCLK_STOP_SOFTWARE = 0, DM75xx_PCLK_STOP_ETRIG, DM75xx_PCLK_STOP_DIGITA←L_IT, DM75xx_PCLK_STOP_ACNT,
  DM75xx_PCLK_STOP_UTC2, DM75xx_PCLK_STOP_SBUS0, DM75xx_PCLK_STOP_SBUS1, DM75xx←_PCLK_STOP_SBUS2,
  DM75xx_PCLK_STOP_ASOFTWARE, DM75xx_PCLK_STOP_AETRIG, DM75xx_PCLK_STOP_ADIGIT←AL_IT, DM75xx_PCLK_STOP_RES,
  DM75xx_PCLK_STOP_AUTC2, DM75xx_PCLK_STOP_ASBUS0, DM75xx_PCLK_STOP_ASBUS1, D←M75xx_PCLK_STOP_ASBUS2 }

  Pacer Clock Stop.
- enum _dm75xx_pclk_start {
  DM75xx_PCLK_START_SOFTWARE = 0, DM75xx_PCLK_START_ETRIG, DM75xx_PCLK_START_DIG←ITAL_IT, DM75xx_PCLK_START_UTC2,
  DM75xx_PCLK_START_SBUS0, DM75xx_PCLK_START_SBUS1, DM75xx_PCLK_START_SBUS2, D←M75xx_PCLK_START_RES,
  DM75xx_PCLK_START_DSOFTWARE, DM75xx_PCLK_START_DETRIG, DM75xx_PCLK_START_DDI←GITAL_IT, DM75xx_PCLK_START_DUTC2,
  DM75xx_PCLK_START_DSBUS0, DM75xx_PCLK_START_DSBUS1, DM75xx_PCLK_START_DSBUS2,
  DM75xx_PCLK_START_ETRIG_GATE }

  Pacer Clock Start.
- enum _dm75xx_pclk_select { DM75xx_PCLK_EXTERNAL = 0, DM75xx_PCLK_INTERNAL }

  Pacer Clock Select.
- enum _dm75xx_pclk_freq { DM75xx_PCLK_FREQ_8_MHZ = 0, DM75xx_PCLK_FREQ_20_MHZ }

  Pacer Clock Frequency Select.

4.42.1 Detailed Description

DM75xx_Types_BCLK_Enumerations

4.42.2 Enumeration Type Documentation

4.42.2.1 enum _dm75xx_pclk_freq

Pacer Clock Frequency Select.

Enumerator

DM75xx_PCLK_FREQ_8_MHZ 8Mhz Frequency

DM75xx_PCLK_FREQ_20_MHZ 20Mhz Frequency

Definition at line 1436 of file dm75xx_types.h.

4.42.2.2 enum _dm75xx_pclk_mode

Pacer Clock Trigger Mode.

Enumerator

DM75xx_PCLK_NO_REPEAT Single Cycle Mode

DM75xx_PCLK_REPEAT Repeat Mode

Definition at line 1239 of file dm75xx_types.h.

4.42.2.3 enum _dm75xx_pclk_select

Pacer Clock Select.

Enumerator

DM75xx_PCLK_EXTERNAL External Pacer Clock

DM75xx_PCLK_INTERNAL Internal Pacer Clock

Definition at line 1414 of file dm75xx_types.h.

4.42.2.4 enum _dm75xx_pclk_start

Pacer Clock Start.

Enumerator

DM75xx_PCLK_START_SOFTWARE Software

DM75xx_PCLK_START_ETRIG External Trigger

DM75xx_PCLK_START_DIGITAL_IT Digital Interrupt

DM75xx_PCLK_START_UTC2 User Timer/Counter 2 Out

DM75xx_PCLK_START_SBUS0 SyncBus 0

DM75xx_PCLK_START_SBUS1 SyncBus 1

DM75xx_PCLK_START_SBUS2 SyncBus 2

DM75xx_PCLK_START_RES Reserved

DM75xx_PCLK_START_DSOFTWARE Delayed Software

DM75xx_PCLK_START_DETRIG Delayed External Trigger

DM75xx_PCLK_START_DDIGITAL_IT Delayed Digital Interrupt

DM75xx_PCLK_START_DUTC2 Delayed User Timer/Counter 2 Out

DM75xx_PCLK_START_DSBUS0 Delayed SyncBus 0

DM75xx_PCLK_START_DSBUS1 Delayed SyncBus 1

DM75xx_PCLK_START_DSBUS2 Delayed SyncBus 2

DM75xx_PCLK_START_ETRIG_GATE External Trigger Gated

Definition at line 1339 of file dm75xx_types.h.

4.42.2.5 enum _dm75xx_pclk_stop

Pacer Clock Stop.

Enumerator

DM75xx_PCLK_STOP_SOFTWARE	Software
DM75xx_PCLK_STOP_ETRIG	External Trigger
DM75xx_PCLK_STOP_DIGITAL_IT	Digital Interrupt
DM75xx_PCLK_STOP_ACNT	About Counter
DM75xx_PCLK_STOP_UTC2	User Timer/Counter 2 Out
DM75xx_PCLK_STOP_SBUS0	SyncBus 0
DM75xx_PCLK_STOP_SBUS1	SyncBus 1
DM75xx_PCLK_STOP_SBUS2	SyncBus 2
DM75xx_PCLK_STOP_ASOFTWARE	About Software
DM75xx_PCLK_STOP_AETRIG	About External Trigger
DM75xx_PCLK_STOP_ADIGITAL_IT	About Digital Interrupt
DM75xx_PCLK_STOP_RES	Reserved
DM75xx_PCLK_STOP_AUTC2	About User Timer/Counter 2 Out
DM75xx_PCLK_STOP_ASBUS0	About SyncBus 0
DM75xx_PCLK_STOP_ASBUS1	About SyncBus 1
DM75xx_PCLK_STOP_ASBUS2	About SyncBus 2

Definition at line 1263 of file dm75xx_types.h.

4.43 DM75xx type timer/counter enumerations

Typedefs

- typedef enum _dm75xx_utc_timer dm75xx_utc_timer_t

  8254 timer/counter type
- typedef enum _dm75xx_utc_clk dm75xx_utc_clk_t

  8254 timer/counter clock selector type
- typedef enum _dm75xx_utc_gate dm75xx_utc_gate

  8254 timer/counter gate selector type
- typedef enum _dm75xx_utc_mode dm75xx_utc_mode

  8254 timer/counter waverform mode selector type

Enumerations

- enum _dm75xx_utc_timer { DM75xx_UTC_0 = 0, DM75xx_UTC_1, DM75xx_UTC_2 }

  8254 timers/counters
- enum _dm75xx_utc_clk {
  DM75xx_CUTC_8_MHZ = 0, DM75xx_CUTC_EXT_TC_CLOCK_1 = 1, DM75xx_CUTC_EXT_TC_CLOC↩
  K_2 = 2, DM75xx_CUTC_EXT_PCLK = 3,
  DM75xx_CUTC_UTC_0_OUT = 4, DM75xx_CUTC_UTC_1_OUT = 4, DM75xx_CUTC_HSDIN_SIGNAL = 5
  }

  8254 timer/counter clock selectors
- enum _dm75xx_utc_gate {
  DM75xx_GUTC_NOT_GATED = 0, DM75xx_GUTC_GATED = 1, DM75xx_GUTC_EXT_TC_CLK_1 = 2, D↩
  M75xx_GUTC_EXT_TC_CLK_2 = 3,
  DM75xx_GUTC_UTC_0_OUT = 4, DM75xx_GUTC_UTC_1_OUT = 4 }

  8254 timer/counter gate selectors
- enum _dm75xx_utc_mode {
  DM75xx_UTC_MODE_EVENT_COUNTER = 0, DM75xx_UTC_MODE_PROG_ONE_SHOT, DM75xx_UT↩
  C_MODE_RATE_GENERATOR, DM75xx_UTC_MODE_SQUARE_WAVE,
  DM75xx_UTC_MODE_SOFTWARE_STROBE, DM75xx_UTC_MODE_HARDWARE_STROBE }

  8254 timer/counter waveform mode selectors

4.43.1 Detailed Description

DM75xx_Types_PCLK_Enumerations

4.43.2 Enumeration Type Documentation

4.43.2.1 enum _dm75xx_utc_clk

8254 timer/counter clock selectors

Enumerator

DM75xx_CUTC_8_MHZ	8 MHz clock
DM75xx_CUTC_EXT_TC_CLOCK_1	External Timer Counter Clock 1
DM75xx_CUTC_EXT_TC_CLOCK_2	External Timer Counter Clock 2
DM75xx_CUTC_EXT_PCLK	External Pacer Clock
DM75xx_CUTC_UTC_0_OUT	User Timer/Counter 0 Out
DM75xx_CUTC_UTC_1_OUT	User Timer/Counter 1 Out

DM75xx_CUTC_HSDIN_SIGNAL High Speed Digital Input Sample Signal

Definition at line 1504 of file dm75xx_types.h.

4.43.2.2 enum _dm75xx_utc_gate

8254 timer/counter gate selectors

Enumerator

DM75xx_GUTC_NOT_GATED Logic 0

DM75xx_GUTC_GATED Logic 1

DM75xx_GUTC_EXT_TC_CLK_1 8254 timer/counter

DM75xx_GUTC_EXT_TC_CLK_2 8254 timer/counter

DM75xx_GUTC_UTC_0_OUT 8254 timer/counter

DM75xx_GUTC_UTC_1_OUT 8254 timer/counter

Definition at line 1561 of file dm75xx_types.h.

4.43.2.3 enum _dm75xx_utc_mode

8254 timer/counter waveform mode selectors

Enumerator

DM75xx_UTC_MODE_EVENT_COUNTER Event counter

DM75xx_UTC_MODE_PROG_ONE_SHOT Programmable one shot

DM75xx_UTC_MODE_RATE_GENERATOR Rate generator

DM75xx_UTC_MODE_SQUARE_WAVE Square wave generator

DM75xx_UTC_MODE_SOFTWARE_STROBE Software triggered strobe

DM75xx_UTC_MODE_HARDWARE_STROBE Hardware triggered strobe

Definition at line 1613 of file dm75xx_types.h.

4.43.2.4 enum _dm75xx_utc_timer

8254 timers/counters

Enumerator

DM75xx_UTC_0 Timer 0 on 8254 chip

DM75xx_UTC_1 Timer 1 on 8254 chip

DM75xx_UTC_2 Timer 2 on 8254 chip

Definition at line 1471 of file dm75xx_types.h.

4.44 DM75xx type definition structures

Data Structures

- struct _dm75xx_int_status

  Interrupts status.
- struct _dm75xx_cgt_entry

  Channel gain table entry.
- struct dm75xx_pci_access_request

  PCI region access request descriptor. This structure holds information about a request to read data from or write data to one of a device's PCI regions.

Typedefs

- typedef enum _dm75xx_fifo_status dm75xx_fifo_status_t

  FIFO Status Type.
- typedef struct _dm75xx_int_status dm75xx_int_status_t

  Interrupt status type.
- typedef struct _dm75xx_cgt_entry dm75xx_cgt_entry_t

  Channel gain table entry type.
- typedef struct
  dm75xx_pci_access_request dm75xx_pci_access_request_t

  PCI region access request descriptor type.

Enumerations

- enum _dm75xx_fifo_status {
  DM75xx_FIFO_DAC1_NOT_EMPTY = 0x0001, DM75xx_FIFO_DAC1_HALF_EMPTY = 0x0002, DM75xx↩
  _FIFO_DAC1_NOT_FULL = 0x0004, DM75xx_FIFO_DAC2_NOT_EMPTY = 0x0010,
  DM75xx_FIFO_DAC2_HALF_EMPTY = 0x0020, DM75xx_FIFO_DAC2_NOT_FULL = 0x0040, DM75xx_F↩
  IFO_ADC_NOT_EMPTY = 0x0100, DM75xx_FIFO_ADC_HALF_EMPTY = 0x0200,
  DM75xx_FIFO_ADC_NOT_FULL = 0x0400, DM75xx_FIFO_HSDIN_NOT_EMPTY = 0x1000, DM75xx_FI↩
  FO_HSDIN_HALF_EMPTY = 0x2000, DM75xx_FIFO_HSDIN_NOT_FULL = 0x4000 }

  FIFO status.

4.44.1 Detailed Description

DM75xx_Types_Enumerations

4.44.2 Enumeration Type Documentation

4.44.2.1 enum _dm75xx_fifo_status

FIFO status.

Enumerator

DM75xx_FIFO_DAC1_NOT_EMPTY	DAC1 FIFO Not Empty
DM75xx_FIFO_DAC1_HALF_EMPTY	DAC1 FIFO Half Empty
DM75xx_FIFO_DAC1_NOT_FULL	DAC1 FIFO Not Full
DM75xx_FIFO_DAC2_NOT_EMPTY	DAC2 FIFO Not Empty

DM75xx_FIFO_DAC2_HALF_EMPTY DAC2 FIFO Half Empty

DM75xx_FIFO_DAC2_NOT_FULL DAC2 FIFO Not Full

DM75xx_FIFO_ADC_NOT_EMPTY ADC FIFO Not Empty

DM75xx_FIFO_ADC_HALF_EMPTY ADC FIFO Half Empty

DM75xx_FIFO_ADC_NOT_FULL ADC FIFO Not Full

DM75xx_FIFO_HSDIN_NOT_EMPTY HSDIN FIFO Not Empty

DM75xx_FIFO_HSDIN_HALF_EMPTY HSDIN FIFO Half Empty

DM75xx_FIFO_HSDIN_NOT_FULL HSDIN FIFO Not Full

Definition at line 1679 of file dm75xx_types.h.

Chapter 5

Data Structure Documentation

5.1 _dm75xx_cgt_entry Struct Reference

Channel gain table entry.

```
#include <dm75xx_types.h>
```

Data Fields

- uint8_t channel:4
- uint8_t gain:3
- uint8_t nrse:1
- uint8_t range:2
- uint8_t ground:1
- uint8_t pause:1
- uint8_t dac1:1
- uint8_t dac2:1
- uint8_t skip:1
- uint8_t reserved:1

5.1.1 Detailed Description

Channel gain table entry.

Definition at line 1768 of file dm75xx_types.h.

5.1.2 Field Documentation

5.1.2.1 uint8_t channel

Analog input channel

Definition at line 1772 of file dm75xx_types.h.

Referenced by main().

5.1.2.2 uint8_t dac1

DAC1 Update

Definition at line 1796 of file dm75xx_types.h.

Referenced by main().

5.1.2.3 uint8_t dac2

DAC2 Update

Definition at line 1800 of file dm75xx_types.h.

Referenced by main().

5.1.2.4 uint8_t gain

Gain

Definition at line 1776 of file dm75xx_types.h.

Referenced by main().

5.1.2.5 uint8_t ground

Single/Differential

Definition at line 1788 of file dm75xx_types.h.

Referenced by main().

5.1.2.6 uint8_t nrse

AGND/AINSENSE

Definition at line 1780 of file dm75xx_types.h.

Referenced by main().

5.1.2.7 uint8_t pause

Pause

Definition at line 1792 of file dm75xx_types.h.

Referenced by main().

5.1.2.8 uint8_t range

Output Range

Definition at line 1784 of file dm75xx_types.h.

Referenced by main().

5.1.2.9 uint8_t reserved

Reserved

Definition at line 1808 of file dm75xx_types.h.

5.1.2.10 uint8_t skip

Skip

Definition at line 1804 of file dm75xx_types.h.

Referenced by main().

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

- include/dm75xx_types.h

5.2 _dm75xx_int_status Struct Reference

Interrupts status.

```
#include <dm75xx_types.h>
```

Data Fields

- int int_remaining
- unsigned int int_missed
- uint32_t status

5.2.1 Detailed Description

Interrupts status.

Definition at line 1738 of file dm75xx_types.h.

5.2.2 Field Documentation

5.2.2.1 unsigned int int_missed

Number of interrupts missed

Definition at line 1750 of file dm75xx_types.h.

5.2.2.2 int int_remaining

Number of interrupts remaining in the interrupt status queue

Definition at line 1744 of file dm75xx_types.h.

5.2.2.3 uint32_t status

Interrupt Status

Definition at line 1756 of file dm75xx_types.h.

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

- include/dm75xx_types.h

5.3 DM75xx_Board_Descriptor Struct Reference

DM75xx board descriptor. This structure holds information about a device needed by the library.

```
#include <dm75xx_library.h>
```

Data Fields

- int file_descriptor
- void(∗ isr)(unsigned int status)
- pthread_t pid
- int thread_status
- uint16_t ∗ k_buf [2]
- uint16_t ∗ u_buf [2]
- unsigned long k_buf_siz [2]
- unsigned long u_buf_siz [2]

5.3.1 Detailed Description

DM75xx board descriptor. This structure holds information about a device needed by the library.

Definition at line 183 of file dm75xx_library.h.

5.3.2 Field Documentation

5.3.2.1 int file_descriptor

File descriptor for device returned from open()

Definition at line 189 of file dm75xx_library.h.

5.3.2.2 void(∗ isr)(unsigned int status)

The currently installed user-space ISR for this device.

Definition at line 193 of file dm75xx_library.h.

5.3.2.3 uint16_t∗ k_buf[2]

Pointer to Kernel Space Buffers

Definition at line 205 of file dm75xx_library.h.

5.3.2.4 unsigned long k_buf_siz[2]

Kernel Buffer Sizes

Definition at line 213 of file dm75xx_library.h.

5.3.2.5 pthread_t pid

The parent ID of the thread watching for interrupts.

Definition at line 197 of file dm75xx_library.h.

5.3.2.6 int thread_status

Status of thread after execution

Definition at line 201 of file dm75xx_library.h.

5.3.2.7 uint16_t∗ u_buf[2]

Pointer to User Space DMA Buffers

Definition at line 209 of file dm75xx_library.h.

5.3.2.8 unsigned long u_buf_siz[2]

User Space Buffer Sizes

Definition at line 217 of file dm75xx_library.h.

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

- include/dm75xx_library.h

5.4 dm75xx_device_descriptor Struct Reference

DM75xx device descriptor. This structure holds information about a device needed by the kernel.

```
#include <dm75xx_driver.h>
```

Data Fields

- char name [DM75xx_DEVICE_NAME_LENGTH]
- dm75xx_board_t board_type
- dm75xx_pci_region_t pci [PCI_ROM_RESOURCE]
- dm75xx_int_source_t int_control
- uint32_t int_status [DM75xx_INT_QUEUE_SIZE]
- unsigned int int_queue_in
- unsigned int int_queue_out
- unsigned int int_queue_missed
- unsigned int int_count
- unsigned int fifo_size
- spinlock_t lock
- uint8_t reference_count
- unsigned int irq_number
- uint8_t remove_isr_flag
- wait_queue_head_t int_wait_queue
- uint32_t dma_size [DM75xx_DMA_CHANNELS]
- dm75xx_dma_descriptor_t dma_buffers [DM75xx_DMA_CHANNELS]
- dm75xx_dma_chain_descriptor_t ∗ dma_chain [DM75xx_DMA_CHANNELS]
- dm75xx_dma_flag_t dma_flag [DM75xx_DMA_CHANNELS]

5.4.1 Detailed Description

DM75xx device descriptor. This structure holds information about a device needed by the kernel.

Definition at line 289 of file dm75xx_driver.h.

5.4.2 Field Documentation

5.4.2.1 dm75xx_board_t board_type

Flag which indicates if the board has SDM7540/8540 functionality

Definition at line 302 of file dm75xx_driver.h.

5.4.2.2 dm75xx_dma_descriptor_t dma_buffers[DM75xx_DMA_CHANNELS]

Per DMA channel buffer information

Definition at line 393 of file dm75xx_driver.h.

5.4.2.3 dm75xx_dma_chain_descriptor_t* dma_chain[DM75xx_DMA_CHANNELS]

Per DMA channel chaining descriptors

Definition at line 399 of file dm75xx_driver.h.

5.4.2.4 dm75xx_dma_flag_t dma_flag[DM75xx_DMA_CHANNELS]

Flag used for DMA control

Definition at line 405 of file dm75xx_driver.h.

5.4.2.5 uint32_t dma_size[DM75xx_DMA_CHANNELS]

Per-FIFO channel DMA transfer size in bytes

Definition at line 387 of file dm75xx_driver.h.

5.4.2.6 unsigned int fifo_size

The board's FIFO capacity.

Definition at line 350 of file dm75xx_driver.h.

5.4.2.7 dm75xx_int_source_t int_control

Interrupt Control

Definition at line 314 of file dm75xx_driver.h.

5.4.2.8 unsigned int int_count

Number of interrupts in the queue

Definition at line 344 of file dm75xx_driver.h.

5.4.2.9 unsigned int int_queue_in

Number of entries in the interrupt status queue

Definition at line 326 of file dm75xx_driver.h.

5.4.2.10 unsigned int int_queue_missed

Number of interrupts missed because of a full queue

Definition at line 338 of file dm75xx_driver.h.

5.4.2.11 unsigned int int_queue_out

Number of entries read from the interrupt status queue

Definition at line 332 of file dm75xx_driver.h.

5.4.2.12 uint32_t int_status[DM75xx_INT_QUEUE_SIZE]

Interrupt status queue

Definition at line 320 of file dm75xx_driver.h.

5.4.2.13 wait_queue_head_t int_wait_queue

Queue of processes waiting to be woken up when an interrupt occurs

Definition at line 381 of file dm75xx_driver.h.

5.4.2.14 unsigned int irq_number

IRQ line number

Definition at line 369 of file dm75xx_driver.h.

5.4.2.15 spinlock_t lock

Concurrency control

Definition at line 356 of file dm75xx_driver.h.

5.4.2.16 char name[DM75xx_DEVICE_NAME_LENGTH]

Device name used when requesting resources; a NUL terminated string of the form rtd-dm75xx-x where x is the device minor number.

Definition at line 296 of file dm75xx_driver.h.

5.4.2.17 dm75xx_pci_region_t pci[PCI_ROM_RESOURCE]

Information about each of the standard PCI regions

Definition at line 308 of file dm75xx_driver.h.

5.4.2.18 uint8_t reference_count

Number of entities which have the device file open. Used to enforce single open semantics.

Definition at line 363 of file dm75xx_driver.h.

5.4.2.19 uint8_t remove_isr_flag

Used to assist in shutting down the thread waiting for interrupts

Definition at line 375 of file dm75xx_driver.h.

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

- include/dm75xx_driver.h

5.5 dm75xx_dma_chain_descriptor Struct Reference

Dm75xx DMA chaining descriptor.

```
#include <dm75xx_driver.h>
```

Data Fields

- uint32_t pci_address
- uint32_t local_address
- uint32_t transfer_size
- uint32_t descriptor_pointer

5.5.1 Detailed Description

Dm75xx DMA chaining descriptor.

Definition at line 225 of file dm75xx_driver.h.

5.5.2 Field Documentation

5.5.2.1 uint32_t descriptor_pointer

Descriptor Pointer

Definition at line 241 of file dm75xx_driver.h.

5.5.2.2 uint32_t local_address

Local Address

Definition at line 233 of file dm75xx_driver.h.

5.5.2.3 uint32_t pci_address

PCI Address

Definition at line 229 of file dm75xx_driver.h.

5.5.2.4 uint32_t transfer_size

Transfer Size

Definition at line 237 of file dm75xx_driver.h.

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

- include/dm75xx_driver.h

5.6 dm75xx_dma_descriptor Struct Reference

DM75xx DMA buffer descriptor. This structure holds allocation information for a single DMA buffer.

```
#include <dm75xx_driver.h>
```

Data Fields

- dma_addr_t bus_address
- void ∗ virtual_address
- unsigned long size

5.6.1 Detailed Description

DM75xx DMA buffer descriptor. This structure holds allocation information for a single DMA buffer.

Definition at line 257 of file dm75xx_driver.h.

5.6.2 Field Documentation

5.6.2.1 dma_addr_t bus_address

Bus/physical address

Definition at line 263 of file dm75xx_driver.h.

5.6.2.2 unsigned long size

Buffer size

Definition at line 275 of file dm75xx_driver.h.

5.6.2.3 void∗ virtual_address

Virtual address

Definition at line 269 of file dm75xx_driver.h.

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

- include/dm75xx_driver.h

5.7 dm75xx_ioctl_argument Union Reference

ioctl() request structure encapsulating all possible requests. This is what gets passed into the kernel from user space on the ioctl() call.

```
#include <dm75xx_ioctl.h>
```

Data Fields

- dm75xx_ioctl_region_readwrite_t readwrite
- dm75xx_ioctl_region_modify_t modify
- dm75xx_ioctl_dma_function_t dma_function
- dm75xx_ioctl_int_control_t int_control

5.7.1 Detailed Description

ioctl() request structure encapsulating all possible requests. This is what gets passed into the kernel from user space on the ioctl() call.

Definition at line 254 of file dm75xx_ioctl.h.

5.7.2 Field Documentation

5.7.2.1 dm75xx_ioctl_dma_function_t dma_function

DMA management function

Definition at line 272 of file dm75xx_ioctl.h.

5.7.2.2 dm75xx_ioctl_int_control_t int_control

Interrupt control function

Definition at line 278 of file dm75xx_ioctl.h.

5.7.2.3 dm75xx_ioctl_region_modify_t modify

PCI region read/modify/write

Definition at line 266 of file dm75xx_ioctl.h.

5.7.2.4 dm75xx_ioctl_region_readwrite_t readwrite

PCI region read and write

Definition at line 260 of file dm75xx_ioctl.h.

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

- include/dm75xx_ioctl.h

5.8 dm75xx_ioctl_dma_function Struct Reference

ioctl() request structure for performing a DMA function

```
#include <dm75xx_ioctl.h>
```

Data Fields

- dm75xx_dma_source_t source
- dm75xx_dma_channel_t channel
- dm75xx_dma_request_t request
- uint32_t size
- uint32_t pci_address
- uint8_t arb
- dm75xx_dma_manage_function_t function

5.8.1 Detailed Description

ioctl() request structure for performing a DMA function

Definition at line 169 of file dm75xx_ioctl.h.

5.8.2 Field Documentation

5.8.2.1 uint8_t arb

Flag indicating whether or not this transfer is to an arbitrary address

Definition at line 204 of file dm75xx_ioctl.h.

5.8.2.2 dm75xx_dma_channel_t channel

The DMA Channel on which to perform the specified operation

Definition at line 181 of file dm75xx_ioctl.h.

5.8.2.3 dm75xx_dma_manage_function_t function

DMA function to perform

Definition at line 210 of file dm75xx_ioctl.h.

5.8.2.4 uint32_t pci_address

PCI Address

Definition at line 199 of file dm75xx_ioctl.h.

5.8.2.5 dm75xx_dma_request_t request

Demand mode request source (DREQ)

Definition at line 187 of file dm75xx_ioctl.h.

5.8.2.6 uint32_t size

Contains the transfer size for the DMA channel

Definition at line 193 of file dm75xx_ioctl.h.

5.8.2.7 dm75xx_dma_source_t source

DMA Local FIFO Source

Definition at line 175 of file dm75xx_ioctl.h.

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

- include/dm75xx_ioctl.h

5.9 dm75xx_ioctl_int_control Struct Reference

ioctl() request structure for interrupt control.

```
#include <dm75xx_ioctl.h>
```

Data Fields

- dm75xx_int_source_t source
- dm75xx_int_control_function_t function

5.9.1 Detailed Description

ioctl() request structure for interrupt control.

Definition at line 226 of file dm75xx_ioctl.h.

5.9.2 Field Documentation

5.9.2.1 dm75xx_int_control_function_t function

Interrupt function to perofmr

Definition at line 238 of file dm75xx_ioctl.h.

5.9.2.2 dm75xx_int_source_t source

Interrupt Sources

Definition at line 232 of file dm75xx_ioctl.h.

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

- include/dm75xx_ioctl.h

5.10 dm75xx_ioctl_region_modify Struct Reference

ioctl() request structure for PCI region read/modify/write

```
#include <dm75xx_ioctl.h>
```

Data Fields

- dm75xx_pci_access_request_t access
- union {
    uint8_t mask8
    uint16_t mask16
    uint32_t mask32
  } mask

5.10.1 Detailed Description

ioctl() request structure for PCI region read/modify/write

Definition at line 115 of file dm75xx_ioctl.h.

5.10.2 Field Documentation

5.10.2.1 dm75xx_pci_access_request_t access

PCI region access request

Definition at line 121 of file dm75xx_ioctl.h.

5.10.2.2 union { ... } mask

Bit mask that controls which bits can be modified. A zero in a bit position means that the corresponding register bit should not be modified. A one in a bit position means that the corresponding register bit should be modified.

Note that it's possible to set bits outside of the mask depending upon the register value before modification. When processing the associated request code, the driver will silently prevent this from happening but will not return an indication that the mask or new value was incorrect.

5.10.2.3 uint16_t mask16

Mask for 16-bit operations

Definition at line 147 of file dm75xx_ioctl.h.

5.10.2.4 uint32_t mask32

Mask for 32-bit operations

Definition at line 153 of file dm75xx_ioctl.h.

5.10.2.5 uint8_t mask8

Mask for 8-bit operations

Definition at line 141 of file dm75xx_ioctl.h.

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

- include/dm75xx_ioctl.h

5.11 dm75xx_ioctl_region_readwrite Struct Reference

ioctl() request structure for read from or write to PCI region

`#include <dm75xx_ioctl.h>`

Data Fields

- dm75xx_pci_access_request_t access

5.11.1 Detailed Description

ioctl() request structure for read from or write to PCI region

Definition at line 96 of file dm75xx_ioctl.h.

5.11.2 Field Documentation

5.11.2.1 dm75xx_pci_access_request_t access

PCI region access request

Definition at line 102 of file dm75xx_ioctl.h.

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

- include/dm75xx_ioctl.h

5.12 dm75xx_pci_access_request Struct Reference

PCI region access request descriptor. This structure holds information about a request to read data from or write data to one of a device's PCI regions.

```
#include <dm75xx_types.h>
```

Data Fields

- dm75xx_pci_region_access_size_t size
- dm75xx_pci_region_num_t region
- uint16_t offset
- union {
    uint8_t data8
    uint16_t data16
    uint32_t data32
  } data

5.12.1 Detailed Description

PCI region access request descriptor. This structure holds information about a request to read data from or write data to one of a device's PCI regions.

Definition at line 1824 of file dm75xx_types.h.

5.12.2 Field Documentation

5.12.2.1 union { ... } data

Data to write or the data read

5.12.2.2 uint16_t data16

16-bit value

Definition at line 1860 of file dm75xx_types.h.

5.12.2.3 uint32_t data32

32-bit value

Definition at line 1866 of file dm75xx_types.h.

5.12.2.4 uint8_t data8

8-bit value

Definition at line 1854 of file dm75xx_types.h.

5.12.2.5 uint16_t offset

Offset within region to access

Definition at line 1842 of file dm75xx_types.h.

5.12.2.6 dm75xx_pci_region_num_t region

The PCI region to access

Definition at line 1836 of file dm75xx_types.h.

5.12.2.7 dm75xx_pci_region_access_size_t size

Size of access in bits

Definition at line 1830 of file dm75xx_types.h.

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

- include/dm75xx_types.h

5.13 dm75xx_pci_region Struct Reference

DM75xx PCI region descriptor. This structure holds information about one of a device's PCI memory regions.

`#include <dm75xx_driver.h>`

Data Fields

- unsigned long io_addr
- unsigned long length
- unsigned int phys_addr
- void ∗ virt_addr
- uint8_t allocated

5.13.1 Detailed Description

DM75xx PCI region descriptor. This structure holds information about one of a device's PCI memory regions.

Definition at line 177 of file dm75xx_driver.h.

5.13.2 Field Documentation

5.13.2.1 uint8_t allocated

Flag indicating whether or not the I/O-mapped memory ranged was allocated. A value of zero means the memory range was not allocated. Any other value means the memory range was allocated.

Definition at line 211 of file dm75xx_driver.h.

5.13.2.2 unsigned long io_addr

I/O port number if I/O mapped

Definition at line 183 of file dm75xx_driver.h.

5.13.2.3 unsigned long length

Length of region in bytes

Definition at line 189 of file dm75xx_driver.h.

5.13.2.4 unsigned int phys_addr

Region's physical address if memory mapped or I/O port number if I/O mapped

Definition at line 196 of file dm75xx_driver.h.

5.13.2.5 void∗ virt_addr

Address at which region is mapped in kernel virtual address space if memory mapped

Definition at line 203 of file dm75xx_driver.h.

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

- include/dm75xx_driver.h

Chapter 6

File Documentation

6.1 examples/about_intrpt.c File Reference

This example program demonstrates the use of the About Counter Interrupt.

```
#include <errno.h>
#include <error.h>
#include <fcntl.h>
#include <getopt.h>
#include <limits.h>
#include <signal.h>
#include <stdint.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include "dm75xx_library.h"
```

Macros

- #define UTC2_RATE 400
- #define ADC_RATE 1000

Functions

- static void usage (void)

  Print information on stderr about how the program is to be used. After doing so, the program is exited.
- void ISR (uint32_t status)

  User-Space ISR.
- int main (int argument_count, char ∗∗arguments)

  Main program code.

Variables

- static char ∗ program_name
- static int interrupts

6.1.1 Detailed Description

This example program demonstrates the use of the About Counter Interrupt.

```
Samples are gathered via the Pacer Clock which is triggered by User Timer/
Counter 2.  The About Counter is loaded to trigger an interrupt every 100
samples.


-------------------------------------------------------------------------
This file and its contents are copyright (C) RTD Embedded Technologies,
Inc.  All Rights Reserved.

This software is licensed as described in the RTD End-User Software License
Agreement.  For a copy of this agreement, refer to the file LICENSE.TXT
(which should be included with this software) or contact RTD Embedded
Technologies, Inc.
-------------------------------------------------------------------------


    $Id: about_intrpt.c 99065 2016-04-26 18:03:23Z rgroner $
```

Definition in file about_intrpt.c.

6.1.2 Macro Definition Documentation

6.1.2.1 #define ADC_RATE 1000

Sampling rate

Definition at line 58 of file about_intrpt.c.

Referenced by main().

6.1.2.2 #define UTC2_RATE 400

Rate for User Timer/Counter 2

Definition at line 54 of file about_intrpt.c.

Referenced by main().

6.1.3 Variable Documentation

6.1.3.1 int interrupts `[static]`

Variable used to count how many interrupts occurred

Definition at line 50 of file about_intrpt.c.

Referenced by ISR(), and main().

6.1.3.2 char∗ program_name `[static]`

Name of the program as invoked on the command line.

Definition at line 46 of file about_intrpt.c.

Referenced by main(), and usage().

6.2 examples/adc_abrst.c File Reference

This example program demonstrates the use of the About Counter.

```
#include <errno.h>
#include <error.h>
#include <fcntl.h>
#include <getopt.h>
#include <limits.h>
#include <signal.h>
#include <stdint.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include "dm75xx_library.h"
```

Macros

- #define NUM_CHANNELS 8
- #define PACER_RATE 125000
- #define BURST_RATE 1250000
- #define UTC2_RATE 400

Functions

- static void usage (void)

    Print information on stderr about how the program is to be used. After doing so, the program is exited.
- int main (int argument_count, char ∗∗arguments)

    Main program code.

Variables

- static char ∗ program_name

6.2.1 Detailed Description

This example program demonstrates the use of the About Counter.

```
This example program utilizes the About Counter to demonstrate 'Multi-Burst'
sampling.

The board is configured to sample 8 channels 3 times for each 8254 User
Timer/Counter 2 Out trigger received.

Samples are gathered in this way until the FIFO is filled at which point the
program will print the samples to the screen.
```

```
$Id: adc_abrst.c 99065 2016-04-26 18:03:23Z rgroner $
```

Definition in file adc_abrst.c.

6.2.2 Macro Definition Documentation

6.2.2.1 #define BURST_RATE 1250000

Burst Clock Rate

Definition at line 62 of file adc_abrst.c.

Referenced by main().

6.2.2.2 #define NUM_CHANNELS 8

The number of channels to sample

Definition at line 54 of file adc_abrst.c.

Referenced by main().

6.2.2.3 #define PACER_RATE 125000

Pacer Clock Rate

Definition at line 58 of file adc_abrst.c.

Referenced by main().

6.2.2.4 #define UTC2_RATE 400

User Timer/Counter 2 Rate

Definition at line 66 of file adc_abrst.c.

Referenced by main().

6.2.3 Variable Documentation

6.2.3.1 char* program_name [static]

Name of the program as invoked on the command line.

Definition at line 50 of file adc_abrst.c.

Referenced by main(), and usage().

6.3 examples/adc_dac_simul.c File Reference

Demonstrates simultaneous Analog and Digital sampling.

```
#include <errno.h>
#include <error.h>
#include <fcntl.h>
#include <getopt.h>
#include <limits.h>
#include <math.h>
#include <signal.h>
#include <stdint.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include "dm75xx_library.h"
```

Macros

- #define DAC_FIFO 0x10000
- #define ADC_FIFO 0x8000
- #define DAC_RATE 40000
- #define ADC_RATE 25000
- #define DAT_FILE "./test.dat"

Functions

- static void usage (void)

  Print information on stderr about how the program is to be used. After doing so, the program is exited.
- static void sigint_handler (int sig_num)

  Exit gracefully if the user enters CTRL-C.
- void ISR (uint32_t status)

  User-Space ISR.
- int main (int argument_count, char **arguments)

  Main program code.

Variables

- static char * program_name
- static volatile int adc_ints
- static volatile int dac_ints
- volatile uint8_t exit_program

6.3.1 Detailed Description

Demonstrates simultaneous Analog and Digital sampling.

```
Program samples out the DAC and in the ADC simultaneously via DMA.
Different driver FIFO sizes are emulated to show the diversity of the
DMA engine.  Also, the DAC and ADC are sampled at different rates to show
that each interrupt source can be handled at various times.  The data
captured during this example program is saved to a file named
'test.txt'.
```

```
$Id: adc_dac_simul.c 99065 2016-04-26 18:03:23Z rgroner $
```

Definition in file adc_dac_simul.c.

6.3.2 Macro Definition Documentation

6.3.2.1 #define ADC_FIFO 0x8000

Size of the ADC FIFO to emulate in the driver

Definition at line 66 of file adc_dac_simul.c.

Referenced by main().

6.3.2.2 #define ADC_RATE 25000

ADC Sample rate

Definition at line 74 of file adc_dac_simul.c.

Referenced by main().

6.3.2.3 #define DAC_FIFO 0x10000

Size of the DAC FIFO to emulate in the driver

Definition at line 62 of file adc_dac_simul.c.

Referenced by main().

6.3.2.4 #define DAC_RATE 40000

DAC Sample rate

Definition at line 70 of file adc_dac_simul.c.

Referenced by main().

6.3.2.5 #define DAT_FILE "./test.dat"

Filname to dump the data

Definition at line 82 of file adc_dac_simul.c.

Referenced by main().

6.3.3 Variable Documentation

6.3.3.1 volatile int adc_ints [static]

Variable used to count how many interrupts occurred

Definition at line 54 of file adc_dac_simul.c.

Referenced by ISR(), and main().

6.3.3.2 volatile int dac_ints [static]

Variable used to count how many interrupts occurred

Definition at line 58 of file adc_dac_simul.c.

Referenced by ISR(), and main().

6.3.3.3 volatile uint8_t exit_program

Variable to allow graceful exit from Ctrl-C

Definition at line 78 of file adc_dac_simul.c.

Referenced by main(), and sigint_handler().

6.3.3.4 char∗ program_name [static]

Program name as invoked on the command line

Definition at line 50 of file adc_dac_simul.c.

Referenced by main(), and usage().

6.4 examples/adc_dma.c File Reference

Demonstrates the use of Analog to Digital sampling with DMA.

```
#include <errno.h>
#include <error.h>
#include <fcntl.h>
#include <getopt.h>
#include <limits.h>
#include <signal.h>
#include <stdint.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include "dm75xx_library.h"
```

Macros

- #define FIFO 0x10000
- #define NUM_DATA (FIFO ∗ 10)
- #define NUM_INTS (NUM_DATA/(FIFO/2))
- #define ADC_RATE 1250000

Functions

- static void usage (void)

    Print information on stderr about how the program is to be used. After doing so, the program is exited.
- static void sigint_handler (int sig_num)

    Exit gracefully if the user enters CTRL-C.
- void ISR (uint32_t status)

    User-Space ISR.
- void setup_ctrlc_handler ()

    Handler to detect when user hits Ctrl-C.
- int main (int argument_count, char ∗∗arguments)

    Main program code.

Variables

- static char ∗ program_name
- static volatile int interrupts
- DM75xx_Board_Descriptor ∗ board

6.4.1 Detailed Description

Demonstrates the use of Analog to Digital sampling with DMA.

```
This program captures about 1 Million samples at maximum speed (1.25MHz) and
displays the samples to the screen.

This example is meant to show that full speed acquisition is now available
in the driver.

-------------------------------------------------------------------------
This file and its contents are copyright (C) RTD Embedded Technologies,
Inc.  All Rights Reserved.

This software is licensed as described in the RTD End-User Software License
Agreement.  For a copy of this agreement, refer to the file LICENSE.TXT
(which should be included with this software) or contact RTD Embedded
Technologies, Inc.
-------------------------------------------------------------------------

    $Id: adc_dma.c 99108 2016-04-27 21:12:59Z rgroner $
```

Definition in file adc_dma.c.

6.4.2 Macro Definition Documentation

6.4.2.1 #define ADC_RATE 1250000

Sampling rate

Definition at line 68 of file adc_dma.c.

Referenced by main().

6.4.2.2 #define FIFO 0x10000

Size of the FIFO, in samples, to emulate in the driver

Definition at line 56 of file adc_dma.c.

Referenced by main().

6.4.2.3 #define NUM_DATA (FIFO ∗ 10)

Amount of data, in samples, we want from the board

Definition at line 60 of file adc_dma.c.

Referenced by main().

6.4.2.4 #define NUM_INTS (NUM_DATA/(FIFO/2))

Number of user ISR interrupts until we have the amount of data we want.

Definition at line 64 of file adc_dma.c.

Referenced by main().

6.4.3 Function Documentation

6.4.3.1 void setup_ctrlc_handler ( )

Handler to detect when user hits Ctrl-C.

Return values

None.	

Definition at line 141 of file adc_dma.c.

References sigint_handler().

Referenced by main().

6.4.4 Variable Documentation

6.4.4.1 DM75xx_Board_Descriptor∗ board

Board descriptor

Definition at line 72 of file adc_dma.c.

Referenced by main().

6.4.4.2 volatile int interrupts [static]

Variable used to count how many interrupts occurred

Definition at line 52 of file adc_dma.c.

Referenced by ISR(), and main().

6.4.4.3 char∗ program_name [static]

Name of the program as invoked on the command line.

Definition at line 48 of file adc_dma.c.

Referenced by main(), and usage().

6.5 examples/adc_dma_continuous.c File Reference

Demonstrates the use of Digital to Analog sampling via DMA.

```
#include <errno.h>
#include <error.h>
#include <fcntl.h>
#include <getopt.h>
#include <limits.h>
#include <signal.h>
#include <stdint.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include "dm75xx_library.h"
```

Macros

- #define NUM_DATA 0x80000
- #define FIFO 0x10000
- #define NUM_INTS (NUM_DATA/(FIFO/2))
- #define DAT_FILE "./test.dat"
- #define ADC_RATE 50000

Functions

- static void usage (void)

  Print information on stderr about how the program is to be used. After doing so, the program is exited.
- void ISR (uint32_t status)

  User-Space ISR.
- void sigint_handler (int sig_num, siginfo_t ∗info, void ∗ptr)

  Exit gracefully if the user enters CTRL-C.
- int main (int argument_count, char ∗∗arguments)

  Main program code.

Variables

- static char ∗ program_name
- static volatile int interrupts
- volatile uint8_t exit_program

6.5.1 Detailed Description

Demonstrates the use of Digital to Analog sampling via DMA.

```
This example program is similar to adc_dma except data is sampled at a
slower rate and instead of logging to a buffer, the data is dumped to disk.
This program will continually gather A/D samples until you ask it to quit.

This program will run until the user presses Ctrl+C to quit.
```

```
------------------------------------------------------------------------
This file and its contents are copyright (C) RTD Embedded Technologies,
Inc.  All Rights Reserved.

This software is licensed as described in the RTD End-User Software License
Agreement.  For a copy of this agreement, refer to the file LICENSE.TXT
(which should be included with this software) or contact RTD Embedded
Technologies, Inc.
------------------------------------------------------------------------

$Id: adc_dma_continuous.c 99065 2016-04-26 18:03:23Z rgroner $
```

Definition in file adc_dma_continuous.c.

6.5.2 Macro Definition Documentation

6.5.2.1 #define ADC_RATE 50000

Sampling rate

Definition at line 76 of file adc_dma_continuous.c.

Referenced by main().

6.5.2.2 #define DAT_FILE "./test.dat"

Filname to dump the data

Definition at line 68 of file adc_dma_continuous.c.

Referenced by main().

6.5.2.3 #define FIFO 0x10000

Size of the FIFO to emulate in the driver

Definition at line 60 of file adc_dma_continuous.c.

Referenced by main().

6.5.2.4 #define NUM_DATA 0x80000

Amount of data we want from the board

Definition at line 56 of file adc_dma_continuous.c.

6.5.2.5 #define NUM_INTS (NUM_DATA/(FIFO/2))

Number of user ISR interrupts until we have that much data

Definition at line 64 of file adc_dma_continuous.c.

6.5.3 Variable Documentation

6.5.3.1 volatile uint8_t exit_program

Variable to allow graceful exit from Ctrl-C

Definition at line 72 of file adc_dma_continuous.c.

Referenced by main(), and sigint_handler().

6.5.3.2 volatile int interrupts `[static]`

Variable used to count how many interrupts occurred

Definition at line 52 of file adc_dma_continuous.c.

Referenced by ISR(), and main().

6.5.3.3 char∗ program_name `[static]`

Name of the program as invoked on the command line.

Definition at line 48 of file adc_dma_continuous.c.

Referenced by main(), and usage().

6.6 examples/adc_hd_simul.c File Reference

Demonstrates Analog to Digital and High Speed Digital simultaneously.

```
#include <errno.h>
#include <error.h>
#include <fcntl.h>
#include <getopt.h>
#include <limits.h>
#include <signal.h>
#include <stdint.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include "dm75xx_library.h"
```

Macros

- #define ADC_NUM_DATA 0x80000
- #define ADC_FIFO 0x10000
- #define HD_NUM_DATA 0x80000
- #define HD_FIFO 0x8000
- #define ADC_NUM_INTS (ADC_NUM_DATA/(ADC_FIFO/2))
- #define HD_NUM_INTS (HD_NUM_DATA/(HD_FIFO/2))
- #define ADC_RATE 50000
- #define HSDIN_RATE 30000

Functions

- static void usage (void)

  Print information on stderr about how the program is to be used. After doing so, the program is exited.
- static void sigint_handler (int sig_num)

  Exit gracefully if the user enters CTRL-C.
- void ISR (uint32_t status)

  User-Space ISR.
- int main (int argument_count, char ∗∗arguments)

  Main program code.

Variables

- static char * program_name
- static volatile int dma_0_ints
- static volatile int dma_1_ints
- DM75xx_Board_Descriptor * board

6.6.1 Detailed Description

Demonstrates Analog to Digital and High Speed Digital simultaneously.

This program simultaneously samples Analog and High Speed Digital data acquisition via DMA.

About 500,000 samples are gathered on each source at various speeds and driver FIFO sizes. This is done to show the versatility of the driver's DMA engine.

The samples are printed to the screen at the end of the program.

```
--------------------------------------------------------------------------
This file and its contents are copyright (C) RTD Embedded Technologies,
Inc.  All Rights Reserved.

This software is licensed as described in the RTD End-User Software License
Agreement.  For a copy of this agreement, refer to the file LICENSE.TXT
(which should be included with this software) or contact RTD Embedded
Technologies, Inc.
--------------------------------------------------------------------------
```

```
$Id: adc_hd_simul.c 99065 2016-04-26 18:03:23Z rgroner $
```

Definition in file adc_hd_simul.c.

6.6.2 Macro Definition Documentation

6.6.2.1 #define ADC_FIFO 0x10000

Size of the FIFO to emulate in the driver

Definition at line 64 of file adc_hd_simul.c.

Referenced by main().

6.6.2.2 #define ADC_NUM_DATA 0x80000

Amount of data we want from the board

Definition at line 60 of file adc_hd_simul.c.

Referenced by main().

6.6.2.3 #define ADC_NUM_INTS (ADC_NUM_DATA/(ADC_FIFO/2))

Number of user ISR interrupts until we have the amount of data we want.

Definition at line 76 of file adc_hd_simul.c.

Referenced by main().

6.6.2.4 #define ADC_RATE 50000

A/D Sampling rate

Definition at line 84 of file adc_hd_simul.c.

Referenced by main().

6.6.2.5 #define HD_FIFO 0x8000

Size of the FIFO to emulate in the driver

Definition at line 72 of file adc_hd_simul.c.

Referenced by main().

6.6.2.6 #define HD_NUM_DATA 0x80000

Amount of data we want from the board

Definition at line 68 of file adc_hd_simul.c.

Referenced by main().

6.6.2.7 #define HD_NUM_INTS (HD_NUM_DATA/(HD_FIFO/2))

Number of user ISR interrupts until we have the amount of data we want.

Definition at line 80 of file adc_hd_simul.c.

Referenced by main().

6.6.2.8 #define HSDIN_RATE 30000

HD Sampling rate

Definition at line 88 of file adc_hd_simul.c.

Referenced by main().

6.6.3 Variable Documentation

6.6.3.1 DM75xx_Board_Descriptor* board

Board descriptor

Definition at line 92 of file adc_hd_simul.c.

6.6.3.2 volatile int dma_0_ints `[static]`

Variable used to count how many interrupts occurred

Definition at line 52 of file adc_hd_simul.c.

Referenced by ISR(), and main().

6.6.3.3 volatile int dma_1_ints `[static]`

Variable used to count how many interrupts occurred

Definition at line 56 of file adc_hd_simul.c.

Referenced by ISR(), and main().

6.6.3.4 char∗ program_name [static]

Name of the program as invoked on the command line.

Definition at line 48 of file adc_hd_simul.c.

Referenced by main(), and usage().

6.7 examples/adc_multi.c File Reference

Demonstrates the use of Analog to Digital Burst sampling.

```
#include <errno.h>
#include <error.h>
#include <fcntl.h>
#include <getopt.h>
#include <limits.h>
#include <signal.h>
#include <stdint.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include "dm75xx_library.h"
```

Macros

- #define NUM_CHANNELS 16
- #define BURST_RATE 1000000
- #define PACER_RATE 100000

Functions

- static void usage (void)

  Print information on stderr about how the program is to be used. After doing so, the program is exited.
- int main (int argument_count, char ∗∗arguments)

  Main program code.

Variables

- static char ∗ program_name

6.7.1 Detailed Description

Demonstrates the use of Analog to Digital Burst sampling.

```
This example program using the Pacer Clock and Burst Clock to perform
Burst sampling.  Burst sampling is near simultaneous sampling of a given
number of channels as configured per the channel gain table.  In this
```

```
example we sample from all 16 channels on each Pacer Clock conversion
signal.  Samples are acquired until the FIFO is filled then they are
printed to the screen.


--------------------------------------------------------------------------
This file and its contents are copyright (C) RTD Embedded Technologies,
Inc.  All Rights Reserved.

This software is licensed as described in the RTD End-User Software License
Agreement.  For a copy of this agreement, refer to the file LICENSE.TXT
(which should be included with this software) or contact RTD Embedded
Technologies, Inc.
--------------------------------------------------------------------------


    $Id: adc_multi.c 99065 2016-04-26 18:03:23Z rgroner $
```

Definition in file adc_multi.c.

6.7.2 Macro Definition Documentation

6.7.2.1 #define BURST_RATE 1000000

Burst Clock Rate

Definition at line 57 of file adc_multi.c.

Referenced by main().

6.7.2.2 #define NUM_CHANNELS 16

The number of channels to sample

Definition at line 53 of file adc_multi.c.

Referenced by main().

6.7.2.3 #define PACER_RATE 100000

Pacer Clock Rate

Definition at line 61 of file adc_multi.c.

Referenced by main().

6.7.3 Variable Documentation

6.7.3.1 char∗ program_name [static]

Name of the program as invoked on the command line.

Definition at line 49 of file adc_multi.c.

Referenced by main(), and usage().

6.8 examples/adc_single.c File Reference

Demonstrates the use of Analog to Digital Conversion.

```
#include <errno.h>
#include <error.h>
#include <fcntl.h>
#include <getopt.h>
#include <limits.h>
#include <signal.h>
#include <stdint.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include "dm75xx_library.h"
```

Macros

- #define ADC_RATE 10000

Functions

- static void usage (void)

  Print information on stderr about how the program is to be used. After doing so, the program is exited.
- int main (int argument_count, char **arguments)

  Main program code.

Variables

- static char * program_name

6.8.1 Detailed Description

Demonstrates the use of Analog to Digital Conversion.

```
This example program demonstrates simple Analog sampling on a single
channel.  When sampling on a signal channel the channel gain table latch
must be set as shown in this example program.  Samples are gathered until
the FIFO is filled, then they are printed to the screen.
```

```
    $Id: adc_single.c 99107 2016-04-27 21:09:01Z rgroner $
```

Definition in file adc_single.c.

6.8.2 Macro Definition Documentation

6.8.2.1 #define ADC_RATE 10000

Sample rate

Definition at line 51 of file adc_single.c.

Referenced by main().

6.8.3 Variable Documentation

6.8.3.1 char∗ program_name `[static]`

Name of the program as invoked on the command line.

Definition at line 47 of file adc_single.c.

Referenced by main(), and usage().

6.9 examples/adc_soft_trig.c File Reference

Demonstrates the use of the Software Trigger.

```
#include <errno.h>
#include <error.h>
#include <fcntl.h>
#include <getopt.h>
#include <limits.h>
#include <signal.h>
#include <stdint.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include "dm75xx_library.h"
```

Functions

- static void usage (void)

  Print information on stderr about how the program is to be used. After doing so, the program is exited.
- static void sigint_handler (int sig_num)

  Exit gracefully if the user enters CTRL-C.
- int main (int argument_count, char ∗∗arguments)

  Main program code.

Variables

- static char ∗ program_name
- int exit_program = 0

6.9.1 Detailed Description

Demonstrates the use of the Software Trigger.

```
This example program demonstrates simple Analog sampling on a single
channel using software triggering.


--------------------------------------------------------------------------
This file and its contents are copyright (C) RTD Embedded Technologies,
Inc.  All Rights Reserved.

This software is licensed as described in the RTD End-User Software License
Agreement.  For a copy of this agreement, refer to the file LICENSE.TXT
(which should be included with this software) or contact RTD Embedded
Technologies, Inc.
--------------------------------------------------------------------------


    $Id: adc_soft_trig.c 99065 2016-04-26 18:03:23Z rgroner $
```

Definition in file adc_soft_trig.c.

6.9.2 Variable Documentation

6.9.2.1 int exit_program = 0

Flag indicating the user wants to exit.

Definition at line 50 of file adc_soft_trig.c.

Referenced by main(), and sigint_handler().

6.9.2.2 char∗ program_name [static]

Name of the program as invoked on the command line.

Definition at line 45 of file adc_soft_trig.c.

Referenced by main(), and usage().

6.10 examples/adc_trig_ext.c File Reference

Demonstrates the use of Analog to Digital sampling via external trigger.

```
#include <errno.h>
#include <error.h>
#include <fcntl.h>
#include <getopt.h>
#include <limits.h>
#include <signal.h>
#include <stdint.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/select.h>
#include <unistd.h>
#include "dm75xx_library.h"
```

Macros

- #define FIFO 0x1000
- #define NUM_DATA (FIFO ∗ 4)

- #define NUM_INTS (NUM_DATA/(FIFO/2))
- #define ADC_RATE 1000

Functions

- static void usage (void)

  Print information on stderr about how the program is to be used. After doing so, the program is exited.
- static void sigint_handler (int sig_num)

  Exit gracefully if the user enters CTRL-C.
- void ISR (uint32_t status)

  User-Space ISR.
- int kbhit (void)

  Implementation of kbhit()
- int main (int argument_count, char ∗∗arguments)

  Main program code.

Variables

- static char ∗ program_name
- static volatile int interrupts
- DM75xx_Board_Descriptor ∗ board

6.10.1 Detailed Description

Demonstrates the use of Analog to Digital sampling via external trigger.

```
This example program uses the external trigger to toggle the Pacer Clock.
While the External Trigger is high the Pacer Clock will run and while it is
low the Pacer Clock will stop.  The status of the Pacer Clock will be
printed to the screen as External Trigger Edge interrupts are received by
the user-space ISR.

Digital I/O Port 1 is used as an input to the external trigger.  The value
on Port 1 is toggled with the strike of a key on the keyboard.  This
effectively enables/disables acquisition.

Note: This program uses DMA
```

```
--------------------------------------------------------------------------
This file and its contents are copyright (C) RTD Embedded Technologies,
Inc.  All Rights Reserved.

This software is licensed as described in the RTD End-User Software License
Agreement.  For a copy of this agreement, refer to the file LICENSE.TXT
(which should be included with this software) or contact RTD Embedded
Technologies, Inc.
--------------------------------------------------------------------------
```

```
    $Id: adc_trig_ext.c 99065 2016-04-26 18:03:23Z rgroner $
```

Definition in file adc_trig_ext.c.

6.10.2 Macro Definition Documentation

6.10.2.1 #define ADC_RATE 1000

Sampling rate

Definition at line 75 of file adc_trig_ext.c.

Referenced by main().

6.10.2.2 #define FIFO 0x1000

Size of the FIFO to emulate in the driver

Definition at line 63 of file adc_trig_ext.c.

Referenced by main().

6.10.2.3 #define NUM_DATA (FIFO ∗ 4)

Amount of data we want from the board

Definition at line 67 of file adc_trig_ext.c.

Referenced by main().

6.10.2.4 #define NUM_INTS (NUM_DATA/(FIFO/2))

Number of user ISR interrupts until we have the amount of data we want.

Definition at line 71 of file adc_trig_ext.c.

Referenced by main().

6.10.3 Variable Documentation

6.10.3.1 DM75xx_Board_Descriptor∗ board

Board descriptor

Definition at line 79 of file adc_trig_ext.c.

6.10.3.2 volatile int interrupts `[static]`

Variable used to count how many interrupts occurred

Definition at line 59 of file adc_trig_ext.c.

Referenced by ISR(), and main().

6.10.3.3 char∗ program_name `[static]`

Name of the program as invoked on the command line.

Definition at line 55 of file adc_trig_ext.c.

Referenced by main(), and usage().

6.11 examples/adio_event.c File Reference

Demonstrates the use of the Digital I/O Event Mode to generate interrupts.

```
#include <errno.h>
#include <error.h>
#include <fcntl.h>
#include <getopt.h>
#include <limits.h>
#include <signal.h>
#include <stdint.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include "dm75xx_library.h"
```

Functions

- static void usage (void)

    Print information on stderr about how the program is to be used. After doing so, the program is exited.
- void ISR (uint32_t status)

    User-Space ISR.
- int main (int argument_count, char ∗∗arguments)

    Main program code.

Variables

- static char ∗ program_name
- DM75xx_Board_Descriptor ∗ board
- int interrupts

6.11.1 Detailed Description

Demonstrates the use of the Digital I/O Event Mode to generate interrupts.

```
Unlike Match mode where an interrupt only occurs when
Port 0 contains a certain value, event mode interrupts are triggered
whenever the value at Port 0 changes.  The same value array used in
the match mode example is iterated through here, and you will see an
interrupt triggered each time.

Port0 must be connected to Port1 bit-per-bit (Port0 bit 0 connected to
Port1 bit 0 and so on...).

--------------------------------------------------------------------------
This file and its contents are copyright (C) RTD Embedded Technologies,
Inc.  All Rights Reserved.

This software is licensed as described in the RTD End-User Software License
Agreement.  For a copy of this agreement, refer to the file LICENSE.TXT
(which should be included with this software) or contact RTD Embedded
Technologies, Inc.
--------------------------------------------------------------------------


    $Id: adio_event.c 99065 2016-04-26 18:03:23Z rgroner $
```

Definition in file adio_event.c.

6.11.2 Variable Documentation

6.11.2.1 DM75xx_Board_Descriptor∗ board

Board's device descriptor

Definition at line 56 of file adio_event.c.

6.11.2.2 int interrupts

Variable to count interrupts

Definition at line 60 of file adio_event.c.

Referenced by ISR(), and main().

6.11.2.3 char∗ program_name `[static]`

Program name as invoked on the command line.

Definition at line 52 of file adio_event.c.

Referenced by main(), and usage().

6.12 examples/adio_match.c File Reference

Demonstrates the use of the Digital I/O Match Mode to generate interrupts.

```
#include <errno.h>
#include <error.h>
#include <fcntl.h>
#include <getopt.h>
#include <limits.h>
#include <signal.h>
#include <stdint.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include "dm75xx_library.h"
```

Functions

- static void usage (void)

  Print information on stderr about how the program is to be used. After doing so, the program is exited.
- void ISR (uint32_t status)

  User-Space ISR.
- int main (int argument_count, char ∗∗arguments)

  Main program code.

Variables

- static char ∗ program_name
- DM75xx_Board_Descriptor ∗ board
- int interrupts

6.12.1 Detailed Description

Demonstrates the use of the Digital I/O Match Mode to generate interrupts.

```
The compare register is first set to 0xAB.  When this
value is written out Port1 and received by Port0 an interrupt is
received.  In the user-space ISR, the DIO interrupt is cleared and the
compare register is changed to 0x3C.  When 0x3C is written out Port1
and received by Port0 another interrupt is received.

Port0 must be connected to Port1 bit-per-bit (Port0 bit 0 connected to
Port1 bit 0 and so on ...).


--------------------------------------------------------------------------
This file and its contents are copyright (C) RTD Embedded Technologies,
Inc.  All Rights Reserved.

This software is licensed as described in the RTD End-User Software License
Agreement.  For a copy of this agreement, refer to the file LICENSE.TXT
(which should be included with this software) or contact RTD Embedded
Technologies, Inc.
--------------------------------------------------------------------------


    $Id: adio_match.c 99065 2016-04-26 18:03:23Z rgroner $
```

Definition in file adio_match.c.

6.12.2 Variable Documentation

6.12.2.1 DM75xx_Board_Descriptor* board

Board's device descriptor

Definition at line 56 of file adio_match.c.

6.12.2.2 int interrupts

Variable to count the number of interrupts

Definition at line 60 of file adio_match.c.

Referenced by ISR(), and main().

6.12.2.3 char* program_name [static]

Program name as invoked on the command line.

Definition at line 52 of file adio_match.c.

Referenced by main(), and usage().

6.13 examples/analog_dio.c File Reference

Demonstrates the use of Analog DIO Connector on the SDM7540.

```
#include <errno.h>
#include <error.h>
#include <fcntl.h>
#include <getopt.h>
#include <limits.h>
#include <signal.h>
#include <stdint.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include "dm75xx_library.h"
```

Functions

- static void usage (void)
  Print information on stderr about how the program is to be used. After doing so, the program is exited.
- void ISR (uint32_t status)
  User-Space ISR.
- int main (int argument_count, char **arguments)
  Main program code.

Variables

- static char * program_name
- unsigned int interrupts

6.13.1 Detailed Description

Demonstrates the use of Analog DIO Connector on the SDM7540.

```
This example shows use of the Analog Connector DIO.  This consists of pins 1
and 2 on CN9 of the SDM7540.  In this particular example, pin 1 is set as
an output and pin 2 is set as an input.  Pin 2 rising edge interrupt is
enabled.  Pulses are sent out pin 1 until pin 2 receives 2 interrupts.  Pin
1 should be connect to pin 2.

--------------------------------------------------------------------------
This file and its contents are copyright (C) RTD Embedded Technologies,
Inc.  All Rights Reserved.

This software is licensed as described in the RTD End-User Software License
Agreement.  For a copy of this agreement, refer to the file LICENSE.TXT
(which should be included with this software) or contact RTD Embedded
Technologies, Inc.
--------------------------------------------------------------------------

    $Id: analog_dio.c 99065 2016-04-26 18:03:23Z rgroner $
```

Definition in file analog_dio.c.

6.13.2 Variable Documentation

6.13.2.1 unsigned int interrupts

Global used to keep track of the nubmer of interrupts received

Definition at line 52 of file analog_dio.c.

Referenced by ISR().

6.13.2.2 char∗ program_name [static]

Name of the program as invoked on the command line.

Definition at line 48 of file analog_dio.c.

Referenced by main(), and usage().

6.14 examples/calibrate.c File Reference

Demonstrates auto-calibration of SDM7540/8540.

```
#include <errno.h>
#include <error.h>
#include <fcntl.h>
#include <getopt.h>
#include <limits.h>
#include <signal.h>
#include <stdint.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
#include <unistd.h>
#include "dm75xx_library.h"
```

Functions

- static void usage (void)
  Print information on stderr about how the program is to be used. After doing so, the program is exited.
- int main (int argument_count, char ∗∗arguments)
  Main program code.

Variables

- static char ∗ program_name

6.14.1 Detailed Description

Demonstrates auto-calibration of SDM7540/8540.

```
    This program utilizes the on-board DSP to auto calibrate the A/D and D/A
    converters.

---------------------------------------------------------------------------
This file and its contents are copyright (C) RTD Embedded Technologies,
Inc.  All Rights Reserved.

This software is licensed as described in the RTD End-User Software License
Agreement.  For a copy of this agreement, refer to the file LICENSE.TXT
(which should be included with this software) or contact RTD Embedded
Technologies, Inc.
---------------------------------------------------------------------------
```

```
$Id: calibrate.c 99065 2016-04-26 18:03:23Z rgroner $
```

Definition in file calibrate.c.

6.14.2 Variable Documentation

6.14.2.1 char∗ program_name [static]

Name of the program as invoked on the command line.

Definition at line 46 of file calibrate.c.

Referenced by main(), and usage().

6.15 examples/cgt_reset_intrpt.c File Reference

Demonstrates the use of the Channgel Gain Reset Interrupt.

```
#include <errno.h>
#include <error.h>
#include <fcntl.h>
#include <getopt.h>
#include <limits.h>
#include <signal.h>
#include <stdint.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include "dm75xx_library.h"
```

Macros

- #define NUM_CHANNELS 16
- #define ADC_RATE 200

Functions

- static void usage (void)

  Print information on stderr about how the program is to be used. After doing so, the program is exited.
- void ISR (uint32_t status)

  User-Space ISR.
- int main (int argument_count, char ∗∗arguments)

  Main program code.

Variables

- static char ∗ program_name
- int interrupts

6.15.1 Detailed Description

Demonstrates the use of the Channgel Gain Reset Interrupt.

```
This example program will gather samples on each of the 16 channels
triggered by the Pacer Clock.  Every 16 samples the channel gain table will
reset back to the beginning.  When this reset occurs a channel gain table
reset will be received.

The program will receive 64 CGT reset interrupts before quitting.


--------------------------------------------------------------------------
This file and its contents are copyright (C) RTD Embedded Technologies,
Inc.  All Rights Reserved.

This software is licensed as described in the RTD End-User Software License
Agreement.  For a copy of this agreement, refer to the file LICENSE.TXT
(which should be included with this software) or contact RTD Embedded
Technologies, Inc.
--------------------------------------------------------------------------


$Id: cgt_reset_intrpt.c 99065 2016-04-26 18:03:23Z rgroner $
```

Definition in file cgt_reset_intrpt.c.

6.15.2 Macro Definition Documentation

6.15.2.1 #define ADC_RATE 200

Sample rate

Definition at line 61 of file cgt_reset_intrpt.c.

Referenced by main().

6.15.2.2 #define NUM_CHANNELS 16

The number of channels to sample

Definition at line 53 of file cgt_reset_intrpt.c.

Referenced by main().

6.15.3 Variable Documentation

6.15.3.1 int interrupts

Global used to keep track of the number of interrupts received.

Definition at line 57 of file cgt_reset_intrpt.c.

Referenced by ISR(), and main().

6.15.3.2 char* program_name [static]

Name of the program as invoked on the command line.

Definition at line 49 of file cgt_reset_intrpt.c.

Referenced by main(), and usage().

6.16 examples/dac.c File Reference

Demonstrates the use of Digital to Analog conversion.

```
#include <errno.h>
#include <error.h>
#include <fcntl.h>
#include <getopt.h>
#include <limits.h>
#include <math.h>
#include <signal.h>
#include <stdint.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include "dm75xx_library.h"
```

Macros

- #define DAC_RATE 10000

Functions

- static void usage (void)

  Print information on stderr about how the program is to be used. After doing so, the program is exited.
- int main (int argument_count, char ∗∗arguments)

  Main program code.

Variables

- static char ∗ program_name

6.16.1 Detailed Description

Demonstrates the use of Digital to Analog conversion.

```
This example program produces a Sine Wave on DAC1 and a Saw-Toothed wave on
DAC2.


---------------------------------------------------------------------------
This file and its contents are copyright (C) RTD Embedded Technologies,
Inc.  All Rights Reserved.

This software is licensed as described in the RTD End-User Software License
Agreement.  For a copy of this agreement, refer to the file LICENSE.TXT
(which should be included with this software) or contact RTD Embedded
Technologies, Inc.
---------------------------------------------------------------------------


    $Id: dac.c 99065 2016-04-26 18:03:23Z rgroner $
```

Definition in file dac.c.

6.16.2 Macro Definition Documentation

6.16.2.1 #define DAC_RATE 10000

Sample rate

Definition at line 50 of file dac.c.

Referenced by main().

6.16.3 Variable Documentation

6.16.3.1 char* program_name [static]

Program name as invoked on the command line

Definition at line 46 of file dac.c.

Referenced by main(), and usage().

6.17 examples/dac_dma.c File Reference

Demonstrates the use of Digital to Analog conversion with DMA.

```
#include <errno.h>
#include <error.h>
#include <fcntl.h>
#include <getopt.h>
#include <limits.h>
#include <math.h>
#include <signal.h>
#include <stdint.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include "dm75xx_library.h"
```

Macros

- #define FIFO 0x10000
- #define DAC_RATE 100000

Functions

- static void usage (void)

  Print information on stderr about how the program is to be used. After doing so, the program is exited.
- static void sigint_handler (int sig_num)

  Exit gracefully if the user enters CTRL-C.
- void ISR (uint32_t status)

  User-Space ISR.
- int main (int argument_count, char **arguments)

  Main program code.

Variables

- static char ∗ program_name
- static volatile int interrupts
- volatile uint8_t exit_program

6.17.1 Detailed Description

Demonstrates the use of Digital to Analog conversion with DMA.

```
This program displays a Sine Wave out DAC2.  This sign wave is repeated via
DMA and continued to be displayed until the user presses CTRL+C to end the
program.


--------------------------------------------------------------------------
This file and its contents are copyright (C) RTD Embedded Technologies,
Inc.  All Rights Reserved.

This software is licensed as described in the RTD End-User Software License
Agreement.  For a copy of this agreement, refer to the file LICENSE.TXT
(which should be included with this software) or contact RTD Embedded
Technologies, Inc.
--------------------------------------------------------------------------


$Id: dac_dma.c 99065 2016-04-26 18:03:23Z rgroner $
```

Definition in file dac_dma.c.

6.17.2 Macro Definition Documentation

6.17.2.1 #define DAC_RATE 100000

Sample rate

Definition at line 59 of file dac_dma.c.

Referenced by main().

6.17.2.2 #define FIFO 0x10000

Size of the FIFO to emulate in the driver

Definition at line 55 of file dac_dma.c.

Referenced by main().

6.17.3 Variable Documentation

6.17.3.1 volatile uint8_t exit_program

Variable to allow graceful exit from Ctrl-C

Definition at line 63 of file dac_dma.c.

Referenced by main(), and sigint_handler().

6.17.3.2 volatile int interrupts [static]

Variable used to count how many interrupts occurred

Definition at line 51 of file dac_dma.c.

Referenced by ISR(), and main().

6.17.3.3 char∗ program_name [static]

Program name as invoked on the command line

Definition at line 47 of file dac_dma.c.

Referenced by main(), and usage().

6.18 examples/delay_intrpt.c File Reference

This example program demonstrates the use of the Delay Counter Interrupt.

```
#include <errno.h>
#include <error.h>
#include <fcntl.h>
#include <getopt.h>
#include <limits.h>
#include <signal.h>
#include <stdint.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include "dm75xx_library.h"
```

Macros

- #define ADC_RATE 10

Functions

- static void usage (void)

  Print information on stderr about how the program is to be used. After doing so, the program is exited.
- void ISR (uint32_t status)

  User-Space ISR.
- int main (int argument_count, char ∗∗arguments)

  Main program code.

Variables

- static char ∗ program_name
- static int interrupts

6.18.1 Detailed Description

This example program demonstrates the use of the Delay Counter Interrupt.

```
Samples are gathered via the Pacer Clock which is triggered by Delay
Software Start.  It will wait to collect samples until 10 samples have been
read then it will collect 10 samples.  This process will be repeated via use
of the delay and about counters.  An interrupt will occur each time the
delay counter hits 0 delay counter.


--------------------------------------------------------------------------
This file and its contents are copyright (C) RTD Embedded Technologies,
Inc.  All Rights Reserved.

This software is licensed as described in the RTD End-User Software License
Agreement.  For a copy of this agreement, refer to the file LICENSE.TXT
(which should be included with this software) or contact RTD Embedded
Technologies, Inc.
--------------------------------------------------------------------------


    $Id: delay_intrpt.c 99065 2016-04-26 18:03:23Z rgroner $
```

Definition in file delay_intrpt.c.

6.18.2 Macro Definition Documentation

6.18.2.1 #define ADC_RATE 10

Sampling rate

Definition at line 56 of file delay_intrpt.c.

Referenced by main().

6.18.3 Variable Documentation

6.18.3.1 int interrupts [static]

Variable used to count how many interrupts occurred

Definition at line 52 of file delay_intrpt.c.

Referenced by ISR(), and main().

6.18.3.2 char* program_name [static]

Name of the program as invoked on the command line.

Definition at line 48 of file delay_intrpt.c.

Referenced by main(), and usage().

6.19 examples/dma_pci_arb.c File Reference

Demonstrates the ability to DMA to an arbitrary PCI address.

```
#include <errno.h>
#include <error.h>
#include <fcntl.h>
#include <getopt.h>
#include <limits.h>
#include <signal.h>
#include <stdint.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include "dm75xx_library.h"
```

Macros

- #define FIFO 0x10000
- #define NUM_DATA (FIFO ∗ 2)
- #define NUM_INTS (NUM_DATA/(FIFO/2))
- #define ADC_RATE 100000

Functions

- static void usage (void)

  Print information on stderr about how the program is to be used. After doing so, the program is exited.
- static void sigint_handler (int sig_num)

  Exit gracefully if the user enters CTRL-C.
- void ISR (uint32_t status)

  User-Space ISR.
- int main (int argument_count, char ∗∗arguments)

  Main program code.

Variables

- static char ∗ program_name
- static int interrupts
- DM75xx_Board_Descriptor ∗ board

6.19.1 Detailed Description

Demonstrates the ability to DMA to an arbitrary PCI address.

```
This program will perform one 8k DMA transfer to video buffer.  This example
program is used as a proof of concept that DMA is possible to an arbitrary
PCI address.  This is useful if you want to DMA data directly to another
device, such as a DSP.


---------------------------------------------------------------------------
This file and its contents are copyright (C) RTD Embedded Technologies,
Inc.  All Rights Reserved.

This software is licensed as described in the RTD End-User Software License
Agreement.  For a copy of this agreement, refer to the file LICENSE.TXT
(which should be included with this software) or contact RTD Embedded
Technologies, Inc.
---------------------------------------------------------------------------
```

```
$Id: dma_pci_arb.c 99065 2016-04-26 18:03:23Z rgroner $
```

Definition in file dma_pci_arb.c.

6.19.2 Macro Definition Documentation

6.19.2.1 #define ADC_RATE 100000

Sampling rate

Definition at line 67 of file dma_pci_arb.c.

Referenced by main().

6.19.2.2 #define FIFO 0x10000

Size of the FIFO, in samples, to emulate in the driver

Definition at line 55 of file dma_pci_arb.c.

Referenced by main().

6.19.2.3 #define NUM_DATA (FIFO ∗ 2)

Amount of data, in samples, we want from the board

Definition at line 59 of file dma_pci_arb.c.

6.19.2.4 #define NUM_INTS (NUM_DATA/(FIFO/2))

Number of user ISR interrupts until we have the amount of data we want.

Definition at line 63 of file dma_pci_arb.c.

Referenced by ISR(), and main().

6.19.3 Variable Documentation

6.19.3.1 DM75xx_Board_Descriptor∗ board

Board descriptor

Definition at line 71 of file dma_pci_arb.c.

6.19.3.2 int interrupts `[static]`

Variable used to count how many interrupts occurred

Definition at line 51 of file dma_pci_arb.c.

Referenced by ISR(), and main().

6.19.3.3 char∗ program_name `[static]`

Name of the program as invoked on the command line.

Definition at line 47 of file dma_pci_arb.c.

Referenced by main(), and usage().

6.20 examples/etrig_intrpt.c File Reference

Demonstrates the use of the External Trigger rising/falling interrupts.

```
#include <errno.h>
#include <error.h>
#include <fcntl.h>
#include <getopt.h>
#include <limits.h>
#include <signal.h>
#include <stdint.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include "dm75xx_library.h"
```

Macros

- #define RISING_EDGE 0
- #define FALLING_EDGE 1

Functions

- static void usage (void)

  Print information on stderr about how the program is to be used. After doing so, the program is exited.
- void ISR (uint32_t status)

  User-Space ISR.
- int main (int argument_count, char ∗∗arguments)

  Main program code.

Variables

- static char ∗ program_name
- DM75xx_Board_Descriptor ∗ board
- uint8_t interrupts
- uint16_t edge_val

6.20.1 Detailed Description

Demonstrates the use of the External Trigger rising/falling interrupts.

```
This program uses UTC1 out to set off the interrupts.  User Timer/Counter 1
Out Pin must be routed to External Trigger Pin.  This should cause an
interrupt to be received every second.


--------------------------------------------------------------------------
This file and its contents are copyright (C) RTD Embedded Technologies,
Inc.  All Rights Reserved.

This software is licensed as described in the RTD End-User Software License
Agreement.  For a copy of this agreement, refer to the file LICENSE.TXT
(which should be included with this software) or contact RTD Embedded
Technologies, Inc.
--------------------------------------------------------------------------
```

```
$Id: etrig_intrpt.c 99065 2016-04-26 18:03:23Z rgroner $
```

Definition in file etrig_intrpt.c.

6.20.2 Macro Definition Documentation

6.20.2.1 #define FALLING_EDGE 1

Value to denote falling edge interrupts

Definition at line 58 of file etrig_intrpt.c.

Referenced by main().

6.20.2.2 #define RISING_EDGE 0

Value to denote rising edge interrupts

Definition at line 54 of file etrig_intrpt.c.

Referenced by main().

6.20.3 Variable Documentation

6.20.3.1 DM75xx_Board_Descriptor* board

Board's device descriptor

Definition at line 50 of file etrig_intrpt.c.

6.20.3.2 uint16_t edge_val

Global used to indicate which edge value we will interrupt on.

Definition at line 66 of file etrig_intrpt.c.

Referenced by ISR(), and main().

6.20.3.3 uint8_t interrupts

Global used to keep track of the number of interrupts received.

Definition at line 62 of file etrig_intrpt.c.

Referenced by ISR(), and main().

6.20.3.4 char* program_name `[static]`

Program name as invoked on the command line.

Definition at line 46 of file etrig_intrpt.c.

Referenced by main(), and usage().

6.21 examples/hd.c File Reference

Demonstrates the use of high speed digital data acquisition.

```
#include <errno.h>
#include <error.h>
#include <fcntl.h>
#include <getopt.h>
#include <limits.h>
#include <signal.h>
#include <stdint.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include "dm75xx_library.h"
```

Macros

- #define TIMER_RATE 50000

Functions

- static void usage (void)

    Print information on stderr about how the program is to be used. After doing so, the program is exited.
- int main (int argument_count, char ∗∗arguments)

    Main program code.

Variables

- static char ∗ program_name

6.21.1 Detailed Description

Demonstrates the use of high speed digital data acquisition.

```
This example program simply gathers high speed digital data and displays
it to the screen when the FIFO is filled.


--------------------------------------------------------------------------
This file and its contents are copyright (C) RTD Embedded Technologies,
Inc.  All Rights Reserved.

This software is licensed as described in the RTD End-User Software License
Agreement.  For a copy of this agreement, refer to the file LICENSE.TXT
(which should be included with this software) or contact RTD Embedded
Technologies, Inc.
--------------------------------------------------------------------------


$Id: hd.c 99065 2016-04-26 18:03:23Z rgroner $
```

Definition in file hd.c.

6.21.2 Macro Definition Documentation

6.21.2.1 #define TIMER_RATE 50000

Rate of high speed acquisition

Definition at line 49 of file hd.c.

Referenced by main().

6.21.3 Variable Documentation

6.21.3.1 char∗ program_name [static]

Name of the program as invoked on the command line.

Definition at line 45 of file hd.c.

Referenced by main(), and usage().

6.22 examples/hd_dma.c File Reference

Demonstrates the use of High Speed Digital acquisition via DMA.

```
#include <errno.h>
#include <error.h>
#include <fcntl.h>
#include <getopt.h>
#include <limits.h>
#include <signal.h>
#include <stdint.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include "dm75xx_library.h"
```

Macros

- #define NUM_DATA 0x10000
- #define FIFO 0x4000
- #define NUM_INTS (NUM_DATA/(FIFO/2))
- #define RATE 50000

Functions

- static void usage (void)

  Print information on stderr about how the program is to be used. After doing so, the program is exited.
- static void sigint_handler (int sig_num)

  Exit gracefully if the user enters CTRL-C.
- void ISR (uint32_t status)

  User-Space ISR.
- int main (int argument_count, char ∗∗arguments)

  Main program code.

Variables

- static char ∗ program_name
- static int interrupts
- DM75xx_Board_Descriptor ∗ board

6.22.1 Detailed Description

Demonstrates the use of High Speed Digital acquisition via DMA.

```
This example program uses UTC1 as a demand mode source for HSDIN DMA
operations.  UTC1 will act as a sample counter and each time FIFO Half is
counted it will trigger a demand mode DMA.

---------------------------------------------------------------------------
This file and its contents are copyright (C) RTD Embedded Technologies,
Inc.  All Rights Reserved.

This software is licensed as described in the RTD End-User Software License
Agreement.  For a copy of this agreement, refer to the file LICENSE.TXT
(which should be included with this software) or contact RTD Embedded
Technologies, Inc.
---------------------------------------------------------------------------


$Id: hd_dma.c 99065 2016-04-26 18:03:23Z rgroner $
```

Definition in file hd_dma.c.

6.22.2 Macro Definition Documentation

6.22.2.1 #define FIFO 0x4000

Size of the FIFO to emulate in the driver

Definition at line 58 of file hd_dma.c.

Referenced by main().

6.22.2.2 #define NUM_DATA 0x10000

Amount of data we want from the board

Definition at line 54 of file hd_dma.c.

Referenced by main().

6.22.2.3 #define NUM_INTS (NUM_DATA/(FIFO/2))

Number of user ISR interrupts until we have the amount of data we want.

Definition at line 62 of file hd_dma.c.

Referenced by main().

6.22.2.4 #define RATE 50000

Rate at which samples are collected

Definition at line 66 of file hd_dma.c.

Referenced by main().

6.22.3 Variable Documentation

6.22.3.1 DM75xx_Board_Descriptor* board

Board descriptor

Definition at line 70 of file hd_dma.c.

6.22.3.2 int interrupts `[static]`

Variable used to count how many interrupts occurred

Definition at line 50 of file hd_dma.c.

Referenced by ISR(), and main().

6.22.3.3 char∗ program_name `[static]`

Name of the program as invoked on the command line.

Definition at line 46 of file hd_dma.c.

Referenced by main(), and usage().

6.23 examples/library_test.c File Reference

Program which tests the basic functionality of the library.

```
#include <errno.h>
#include <error.h>
#include <fcntl.h>
#include <getopt.h>
#include <limits.h>
#include <signal.h>
#include <stdint.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include "dm75xx_library.h"
```

Functions

- static void usage (void)

    Print information on stderr about how the program is to be used. After doing so, the program is exited.
- static void expect_failure_and_check (int status, int expected)

    Checks to make sure the correct error code was returned.
- static void expect_success (int status)

    Validates a successful return code.
- int main (int argument_count, char ∗∗arguments)

    Main program code.

Variables

- static char ∗ program_name

6.23.1 Detailed Description

Program which tests the basic functionality of the library.

```
This program is used to test the library API.  It passes various valid and
invalid parameters to each library function to ensure that only acceptable
values are considered valid.


--------------------------------------------------------------------------
This file and its contents are copyright (C) RTD Embedded Technologies,
Inc.  All Rights Reserved.

This software is licensed as described in the RTD End-User Software License
Agreement.  For a copy of this agreement, refer to the file LICENSE.TXT
(which should be included with this software) or contact RTD Embedded
Technologies, Inc.
--------------------------------------------------------------------------
```

Warning

This program ABSOLUTELY IS NOT INTENDED to be an example of how to program a board. Some of the techniques appearing herein can lead to erratic program or system behavior and are used only to cause specific error conditions.

`$Id: library_test.c 65468 2012-12-04 19:42:58Z rgroner $`

Definition in file library_test.c.

6.23.2 Variable Documentation

6.23.2.1 char∗ program_name [static]

Name of the program as invoked on the command line.

Definition at line 52 of file library_test.c.

Referenced by main(), and usage().

6.24 examples/temperature.c File Reference

Demonstrates the use of the SDM7540's on board temp sensor.

```
#include <errno.h>
#include <error.h>
#include <fcntl.h>
#include <getopt.h>
#include <limits.h>
#include <signal.h>
#include <stdint.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include "dm75xx_library.h"
```

Functions

- static void usage (void)

Print information on stderr about how the program is to be used. After doing so, the program is exited.

- int main (int argument_count, char ∗∗arguments)

  Main program code.

Variables

- static char ∗ program_name

6.24.1 Detailed Description

Demonstrates the use of the SDM7540's on board temp sensor.

```
This example will show how to use the SDM7540 family on board temperature
sensor which is on the I2C bus.


--------------------------------------------------------------------------
This file and its contents are copyright (C) RTD Embedded Technologies,
Inc.  All Rights Reserved.

This software is licensed as described in the RTD End-User Software License
Agreement.  For a copy of this agreement, refer to the file LICENSE.TXT
(which should be included with this software) or contact RTD Embedded
Technologies, Inc.
--------------------------------------------------------------------------


    $Id: temperature.c 99065 2016-04-26 18:03:23Z rgroner $
```

Definition in file temperature.c.

6.24.2 Variable Documentation

6.24.2.1 char∗ program_name [static]

Name of the program as invoked on the command line.

Definition at line 45 of file temperature.c.

Referenced by main(), and usage().

6.25 examples/timer.c File Reference

Demonstrates the use of the User Timer/Counters for keeping time.

```
#include <errno.h>
#include <error.h>
#include <fcntl.h>
#include <getopt.h>
#include <limits.h>
#include <signal.h>
#include <stdint.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include "dm75xx_library.h"
```

Macros

- #define TIMER_RATE 200

Functions

- static void usage (void)

  Print information on stderr about how the program is to be used. After doing so, the program is exited.
- static void sigint_handler (int sig_num)

  Exit gracefully if the user enters CTRL-C.
- int main (int argument_count, char ∗∗arguments)

  Main program code.

Variables

- static char ∗ program_name
- volatile uint8_t exit_program

6.25.1 Detailed Description

Demonstrates the use of the User Timer/Counters for keeping time.

```
This example program demonstrates using the 8254 User Timer/Counters as a
simple time keeper.

UTC 1 is being fed by UTC 0


--------------------------------------------------------------------------
This file and its contents are copyright (C) RTD Embedded Technologies,
Inc.  All Rights Reserved.

This software is licensed as described in the RTD End-User Software License
Agreement.  For a copy of this agreement, refer to the file LICENSE.TXT
(which should be included with this software) or contact RTD Embedded
Technologies, Inc.
--------------------------------------------------------------------------


$Id: timer.c 99065 2016-04-26 18:03:23Z rgroner $
```

Definition in file timer.c.

6.25.2 Macro Definition Documentation

6.25.2.1 #define TIMER_RATE 200

Rate of the Timer.

Definition at line 47 of file timer.c.

Referenced by main().

6.25.3 Variable Documentation

6.25.3.1 volatile uint8_t exit_program

Variable to allow graceful exit from Ctrl-C

Definition at line 55 of file timer.c.

Referenced by main(), and sigint_handler().

6.25.3.2 char∗ program_name `[static]`

Program name as invoked on the command line.

Definition at line 51 of file timer.c.

Referenced by main(), and usage().

6.26 examples/timer_intrpt.c File Reference

Demonstrates the use of the User Timer/Counter Out Interrupts.

```
#include <errno.h>
#include <error.h>
#include <fcntl.h>
#include <getopt.h>
#include <limits.h>
#include <signal.h>
#include <stdint.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include "dm75xx_library.h"
```

Macros

- #define UTC0 8000
- #define UTC1 2000

Functions

- static void usage (void)

  Print information on stderr about how the program is to be used. After doing so, the program is exited.
- void ISR (uint32_t status)

  User-Space ISR.
- int main (int argument_count, char ∗∗arguments)

  Main program code.

Variables

- static char ∗ program_name
- DM75xx_Board_Descriptor ∗ board
- uint8_t utc1_int
- uint8_t utc1_int_inverted

6.26.1 Detailed Description

Demonstrates the use of the User Timer/Counter Out Interrupts.

```
UTC0 is set to 1kHz and UTC1 is set to 1Hz.  An interrupt will occur every
UTC1 out and UTC1 Inverted Out.  This should end up causing an interrupt.
Every second from each source but the interrupts will be shifted half a
second apart. The program counts to ten interrupts, on each channel.
```

```
    $Id: timer_intrpt.c 99065 2016-04-26 18:03:23Z rgroner $
```

Definition in file timer_intrpt.c.

6.26.2 Macro Definition Documentation

6.26.2.1 #define UTC0 8000

Rate of user timer/counter 0

Definition at line 47 of file timer_intrpt.c.

Referenced by main().

6.26.2.2 #define UTC1 2000

Rate of user timer/counter 1

Definition at line 51 of file timer_intrpt.c.

Referenced by main().

6.26.3 Variable Documentation

6.26.3.1 DM75xx_Board_Descriptor∗ board

Board's device descriptor

Definition at line 59 of file timer_intrpt.c.

6.26.3.2 char∗ program_name [static]

Program name as invoked on the command line.

Definition at line 55 of file timer_intrpt.c.

Referenced by main(), and usage().

6.26.3.3 uint8_t utc1_int

Global used to keep track of the number of interrupts received.

Definition at line 63 of file timer_intrpt.c.

Referenced by ISR(), and main().

6.26.3.4 uint8_t utc1_int_inverted

Global used to keep track of the number of interrupts received.

Definition at line 67 of file timer_intrpt.c.

Referenced by ISR(), and main().

6.27 include/dm75xx_driver.h File Reference

Definitions for the DM75xx driver.

```
#include <linux/fs.h>
#include <linux/list.h>
#include <linux/pci.h>
#include <linux/spinlock.h>
#include <linux/types.h>
#include "dm75xx_ioctl.h"
#include "dm75xx_types.h"
#include "dm75xx_kernel.h"
```

Data Structures

- struct dm75xx_pci_region

  DM75xx PCI region descriptor. This structure holds information about one of a device's PCI memory regions.
- struct dm75xx_dma_chain_descriptor

  Dm75xx DMA chaining descriptor.
- struct dm75xx_dma_descriptor

  DM75xx DMA buffer descriptor. This structure holds allocation information for a single DMA buffer.
- struct dm75xx_device_descriptor

  DM75xx device descriptor. This structure holds information about a device needed by the kernel.

Macros

- #define DM75xx_DEVICE_NAME_LENGTH 22

  Maximum number of characters in device's name.
- #define DM7520_PCI_DEVICE_ID 0x7520

  DM7520 PCI device ID.
- #define DM7540_PCI_DEVICE_ID 0x7540

  DM7540 PCI device ID.
- #define RTD_PCI_VENDOR_ID 0x1435

  RTD Embedded Technologies PCI vendor ID.
- #define DM75xx_PCI_REGIONS PCI_ROM_RESOURCE

  Number of standard PCI regions.
- #define DM75xx_DMA_CHANNELS 2

Number of FIFO channels per device.
- #define DM75xx_MAX_DMA_BUFFER_SIZE 0x20000

  Maximum size in bytes of any DMA buffer.
- #define DM75xx_INT_QUEUE_SIZE 0x10

  Maximum size in entries of the interrupt status queue.

Typedefs

- typedef enum
  dm75xx_pci_region_access_dir dm75xx_pci_region_access_dir_t

  Standard PCI region access direction type.
- typedef struct dm75xx_pci_region dm75xx_pci_region_t

  DM75xx PCI region descriptor type.
- typedef struct
  dm75xx_dma_chain_descriptor dm75xx_dma_chain_descriptor_t

  DM75xx DMA Chaining descriptor type.
- typedef struct
  dm75xx_dma_descriptor dm75xx_dma_descriptor_t

  DM75xx DMA buffer descriptor type.
- typedef struct
  dm75xx_device_descriptor dm75xx_device_descriptor_t

  DM75xx device descriptor type.

Enumerations

- enum dm75xx_pci_region_access_dir { DM75xx_PCI_REGION_ACCESS_READ = 0, DM75xx_PCI_REG↩
  ION_ACCESS_WRITE }

  Direction of access to standard PCI region.

Functions

- static void dm75xx_access_pci_region (const dm75xx_device_descriptor_t ∗dm75xx, dm75xx_pci_access↩
  _request_t ∗pci_request, dm75xx_pci_region_access_dir_t direction)

  Read from or write to one of the standard PCI regions.
- static int dm75xx_allocate_irq (dm75xx_device_descriptor_t ∗dm75xx, const struct pci_dev ∗pci_device)

  Allocate an interrupt line for a DM75xx device.
- static void dm75xx_enable_plx_interrupts (const dm75xx_device_descriptor_t ∗dm75xx, uint8_t enable)

  Enable PLX interrupts for the specified DM75xx Device.
- static void dm75xx_enable_plx_dma (const dm75xx_device_descriptor_t ∗dm75xx, dm75xx_dma_channel↩
  _t channel)

  Configure PLX Mode register for the specified DMA Channel.
- static void dm75xx_get_pci_master_status (dm75xx_device_descriptor_t ∗dm75xx, uint8_t ∗pci_master)

  Determine whether or not a device is PCI master capable.
- static void dm75xx_initialize_device_descriptor (dm75xx_device_descriptor_t ∗dm75xx)

  Initialize the device descriptor for the specified DM75xx device.
- static void dm75xx_initialize_hardware (const dm75xx_device_descriptor_t ∗dm75xx)

  Initialize the specified DM75xx device.
- INTERRUPT_HANDLER_TYPE dm75xx_interrupt_handler (int irq_number, void ∗device_id)

  DM75xx device interrupt handler.
- static long dm75xx_ioctl (struct file ∗file, unsigned int request_code, unsigned long ioctl_param)

  Process ioctl(2) system calls directed toward a DM75xx device file.

- static void dm75xx_board_reset (dm75xx_device_descriptor_t ∗dm75xx)

  Performs a reset of the board and device descriptor.
- static void dm75xx_interrupt_enable (dm75xx_device_descriptor_t ∗dm75xx, dm75xx_int_source_t source, uint8_t enable)

  Performs the actual enable/disable of the interrupt sources.
- static int dm75xx_interrupt_control (dm75xx_device_descriptor_t ∗dm75xx, unsigned long ioctl_param)

  Control the interrupts on the boards. This includes enabling, disabling and checking the enable/disable status of the interrupts.
- static int dm75xx_get_interrupt (dm75xx_device_descriptor_t ∗dm75xx, unsigned long ioctl_param)

  Returns the top entry from the interrupt status queue.
- static void dm75xx_put_interrupt (dm75xx_device_descriptor_t ∗dm75xx, uint32_t interrupt)

  Adds an interrupt to the interrupt status queue.
- static int dm75xx_service_dma_function (dm75xx_device_descriptor_t ∗dm75xx, unsigned long ioctl_param)

  Process user space DMA function requests.
- static int dm75xx_dma_abort (dm75xx_device_descriptor_t ∗dm75xx, dm75xx_dma_channel_t channel)

  Aborts any DMA transfers on the given channel.
- static int dm75xx_dma_alloc_buffer (dm75xx_device_descriptor_t ∗dm75xx, dm75xx_dma_channel_t channel)

  Allocates a coherent and consistent buffer for our DMA operations.
- static int dm75xx_dreq_init (dm75xx_device_descriptor_t ∗dm75xx, dm75xx_dma_channel_t channel, dm75xx_dma_request_t dreq)

  Performs some DMA initialization work based on the DREQ source.
- static int dm75xx_dma_initialize (dm75xx_device_descriptor_t ∗dm75xx, dm75xx_ioctl_argument_t ∗ioctl_↩ argument)

  Initialize DMA for the specified channel and source for the DM75xx device.
- static void dm75xx_free_dma_mappings (dm75xx_device_descriptor_t ∗dm75xx, dm75xx_dma_channel_↩ t channel)

  Free all coherent/consistent DMA mappings for the given DMA channel on the specified DM75xx device.
- int dm75xx_load (void)

  Perform all actions necessary to initialize the DM75xx driver and devices.
- static int dm75xx_modify_pci_region (dm75xx_device_descriptor_t ∗dm75xx, unsigned long ioctl_param)

  Read an unsigned value from one of a device's PCI regions, modify certain bits in the value, and then write it back to the region.
- static int dm75xx_open (struct inode ∗inode, struct file ∗file)

  Prepare a DM75xx device file to be opened and used.
- static unsigned int dm75xx_poll (struct file ∗file, struct poll_table_struct ∗poll_table)

  Determine whether or not a DM75xx device is readable. This function supports the poll(2) and select(2) system calls.
- static int dm75xx_probe_devices (uint32_t ∗device_count, dm75xx_device_descriptor_t ∗∗device_↩ descriptors)

  Probe and set up all DM75xx devices.
- static int dm75xx_process_pci_regions (dm75xx_device_descriptor_t ∗dm75xx, const struct pci_dev ∗pci_↩ device)

  For each of the standard PCI regions, get the region's base address and length from kernel PCI resource information set up at boot. Also, remap any memory-mapped region into the kernel's virtual address space.
- static int dm75xx_register_char_device (int ∗major)

  Register the DM75xx character device and request dynamic allocation of a character device major number.
- static int dm75xx_release (struct inode ∗inode, struct file ∗file)

  Do all processing necessary after the last reference to a DM75xx device file is released elsewhere in the kernel.
- static void dm75xx_release_resources (void)

  Release any resources allocated by the driver.
- void dm75xx_unload (void)

  Perform all actions necessary to deinitialize the DM75xx driver and devices.

- static int dm75xx_unregister_char_device (void)

  Unregister the DM75xx character device and free the character device major number.
- static int dm75xx_validate_device (const dm75xx_device_descriptor_t ∗dm75xx)

  Given what is assumed to be the address of a DM75xx device descriptor, make sure it corresponds to a valid DM75xx device descriptor.
- static int dm75xx_validate_pci_access (const dm75xx_device_descriptor_t ∗dm75xx, const dm75xx_pci_↩ access_request_t ∗pci_request)

  Validate a user-space access to one of the device's PCI regions.
- static int dm75xx_get_fifo_size (dm75xx_device_descriptor_t ∗dm75xx, unsigned int ∗size)

  Measure the size of the fifo by filling it until it is half-full than doubling that value to get the size of the fifo.

Variables

- static struct file_operations dm75xx_file_ops

  File operations supported by driver.

6.27.1 Detailed Description

Definitions for the DM75xx driver.

```
--------------------------------------------------------------------------
This file and its contents are copyright (C) RTD Embedded Technologies,
Inc.  All Rights Reserved.

This software is licensed as described in the RTD End-User Software License
Agreement.  For a copy of this agreement, refer to the file LICENSE.TXT
(which should be included with this software) or contact RTD Embedded
Technologies, Inc.
--------------------------------------------------------------------------
```

Id

dm75xx_driver.h 99068 2016-04-26 18:25:17Z rgroner

Definition in file dm75xx_driver.h.

6.28 include/dm75xx_ioctl.h File Reference

Low level ioctl() request descriptor structure and request code definitions.

```
#include <linux/ioctl.h>
#include <linux/types.h>
#include "dm75xx_types.h"
```

Data Structures

- struct dm75xx_ioctl_region_readwrite

  ioctl() request structure for read from or write to PCI region
- struct dm75xx_ioctl_region_modify

  ioctl() request structure for PCI region read/modify/write
- struct dm75xx_ioctl_dma_function

  ioctl() request structure for performing a DMA function
- struct dm75xx_ioctl_int_control

ioctl() request structure for interrupt control.

- union dm75xx_ioctl_argument

    ioctl() request structure encapsulating all possible requests. This is what gets passed into the kernel from user space on the ioctl() call.

Macros

- #define DM75xx_IOCTL_MAGIC 'D'

    Unique 8-bit value used to generate unique ioctl() request codes.
- #define DM75xx_IOCTL_REQUEST_BASE 0x00

    First ioctl() request number.
- #define DM75xx_IOCTL_REGION_READ

    ioctl() request code for reading from a PCI region
- #define DM75xx_IOCTL_REGION_WRITE

    ioctl() request code for writing to a PCI region
- #define DM75xx_IOCTL_REGION_MODIFY

    ioctl() request code for PCI region read/modify/write
- #define DM75xx_IOCTL_DMA_FUNCTION

    ioctl() request code for DMA function
- #define DM75xx_IOCTL_WAKEUP

    ioctl() request code to wake up a sleeping driver function
- #define DM75xx_IOCTL_INT_STATUS

    ioctl() request code to get the interrupt status queue
- #define DM75xx_IOCTL_GET_FIFO_SIZE

    ioctl() request code to get the fifo size
- #define DM75xx_IOCTL_GET_BOARD_TYPE

    ioctl() request code to get the board type
- #define DM75xx_IOCTL_INT_CONTROL

    ioctl() request code to control interrupts
- #define DM75xx_IOCTL_RESET

    ioctl() request code to reset the board
- #define DM75xx_IOCTL_RESET_DMA_STATUS

    ioctl() request code to control DMA buffer status

Typedefs

- typedef enum
dm75xx_dma_manage_function dm75xx_dma_manage_function_t

    Functions supported by driver DMA management system.
- typedef enum
dm75xx_int_control_function dm75xx_int_control_function_t

    Functions supported by driver interrupt control system.
- typedef struct
dm75xx_ioctl_region_readwrite dm75xx_ioctl_region_readwrite_t
- typedef struct
dm75xx_ioctl_region_modify dm75xx_ioctl_region_modify_t

    ioctl() PCI region read/modify/write request descriptor type
- typedef struct
dm75xx_ioctl_dma_function dm75xx_ioctl_dma_function_t

    ioctl() request structure for performing a DMA function type.

- typedef struct
  dm75xx_ioctl_int_control dm75xx_ioctl_int_control_t

  ioctl() request structure for interrupt control.
- typedef union dm75xx_ioctl_argument dm75xx_ioctl_argument_t

  ioctl() request descriptor type

Enumerations

- enum dm75xx_dma_manage_function { DM75xx_DMA_FUNCTION_INITIALIZE = 0, DM75xx_DMA_FUN↩
  CTION_ABORT }

  Functions supported by driver DMA management system.
- enum dm75xx_int_control_function { DM75xx_INT_CONTROL_ENABLE = 0, DM75xx_INT_CONTROL_D↩
  ISABLE, DM75xx_INT_CONTROL_CHECK }

  Functions supported by driver interrupt control system.

6.28.1 Detailed Description

Low level ioctl() request descriptor structure and request code definitions.

```
---------------------------------------------------------------------------
This file and its contents are copyright (C) RTD Embedded Technologies,
Inc.  All Rights Reserved.

This software is licensed as described in the RTD End-User Software License
Agreement.  For a copy of this agreement, refer to the file LICENSE.TXT
(which should be included with this software) or contact RTD Embedded
Technologies, Inc.
---------------------------------------------------------------------------


$Id: dm75xx_ioctl.h 65466 2012-12-04 19:25:44Z rgroner $
```

Definition in file dm75xx_ioctl.h.

6.29 include/dm75xx_kernel.h File Reference

Kernel compatibility issues between 2.6.0 and 3.x kernels.

```
#include <asm/ptrace.h>
#include <linux/version.h>
#include <linux/interrupt.h>
#include <linux/cdev.h>
#include <linux/dma-mapping.h>
```

Macros

- #define INTERRUPT_HANDLER_TYPE static irqreturn_t

  Type returned by interrupt handler.
- #define DM75XX_IOCTL .unlocked_ioctl

  In Kernel 2.6.35, .ioctl was replaced with .unlocked_ioctl.
- #define IO_MEMORY_READ8 ioread8

  Entity which reads an 8-bit value from device I/O memory.
- #define IO_MEMORY_READ16 ioread16

  Entity which reads a 16-bit value from device I/O memory.

- #define IO_MEMORY_READ32 ioread32

    Entity which reads a 32-bit value from device I/O memory.
- #define IO_MEMORY_WRITE8 iowrite8

    Entity which writes an 8-bit value to device I/O memory.
- #define IO_MEMORY_WRITE16 iowrite16

    Entity which writes a 16-bit value to device I/O memory.
- #define IO_MEMORY_WRITE32 iowrite32

    Entity which writes a 32-bit value to device I/O memory.

Typedefs

- typedef irqreturn_t(∗ dm75xx_handler_t)(int, void ∗)

    Type definition for interrupt handling function.

6.29.1 Detailed Description

Kernel compatibility issues between 2.6.0 and 3.x kernels.

```
--------------------------------------------------------------------------
This file and its contents are copyright (C) RTD Embedded Technologies,
Inc.  All Rights Reserved.

This software is licensed as described in the RTD End-User Software License
Agreement.  For a copy of this agreement, refer to the file LICENSE.TXT
(which should be included with this software) or contact RTD Embedded
Technologies, Inc.
--------------------------------------------------------------------------
```

Id

dm75xx_kernel.h 65703 2012-12-13 17:04:04Z rgroner

Definition in file dm75xx_kernel.h.

6.30 include/dm75xx_library.h File Reference

DM75xx user library definitions.

```
#include <stdint.h>
#include <stdlib.h>
#include <pthread.h>
#include <sys/wait.h>
#include "dm75xx_types.h"
```

Data Structures

- struct DM75xx_Board_Descriptor

    DM75xx board descriptor. This structure holds information about a device needed by the library.

Macros

- #define DM75xx_INTERRUPT_ACTIVE(status, source) (((status) & (source)) ? 0xFF : 0x00)

  Determine whether or not the specified interrupt source has occurred in your the user space ISR.
- #define DM75xx_ADC_ANALOG_DATA(data) (((int16_t) (data)) >> 3)

  This macro will return the sample portion of raw analog data.
- #define DM75xx_ADC_MARKERS(data) ((data) & 0x07)

  This macro will turn the data marker portion of raw analog data.
- #define DM75xx_DAC_PACK_DATA(data, mcbsp_bit, data_markers)

  This macro will assemble a package to be sent to the Digital to Analog FIFO.

Typedefs

- typedef int DM75xx_Error

  DM75xx user library error code type.
- typedef struct
  DM75xx_Board_Descriptor DM75xx_Board_Descriptor

Functions

- DM75xx_Error DM75xx_Board_PCI_Master (DM75xx_Board_Descriptor ∗handle, uint8_t ∗pci_master)

  Determine whether or not a device is PCI master capable.
- DM75xx_Error DM75xx_Board_Reset (DM75xx_Board_Descriptor ∗handle)

  Reset a DM75xx device.
- DM75xx_Error DM75xx_Clear_ITMask (DM75xx_Board_Descriptor ∗handle, uint16_t mask)

  Clear Interrupts via Mask.
- DM75xx_Error DM75xx_Clear_IT_Overrun (DM75xx_Board_Descriptor ∗handle)

  Clear Interrupt Overrun Register.
- void DM75xx_Exit_On_Error (DM75xx_Board_Descriptor ∗handle, DM75xx_Error status, char ∗str)

  Tests the return status of a library function, and if it's an error we clean up the board and exit.
- DM75xx_Error DM75xx_Board_Init (DM75xx_Board_Descriptor ∗handle)

  Initialize a Board. This function performs the following to attempt to get the device into a known state:
- DM75xx_Error DM75xx_Interrupt_Enable (DM75xx_Board_Descriptor ∗handle, dm75xx_int_source_t int_↩
  source)

  Enable one or more DM75xx interrupt source(s).
- DM75xx_Error DM75xx_Interrupt_Disable (DM75xx_Board_Descriptor ∗handle, dm75xx_int_source_t int_↩
  source)

  Disable one or more DM75xx interrupt source(s).
- DM75xx_Error DM75xx_Interrupt_Check (DM75xx_Board_Descriptor ∗handle, dm75xx_int_source_t ∗int_↩
  source)

  Returns the value of current active/enabled interrupts on the device.
- DM75xx_Error DM75xx_DMA_Buffer_Write (DM75xx_Board_Descriptor ∗handle, dm75xx_dma_channel_t channel, unsigned long num_ints)

  Copy the User Space buffers data incrementally into our Kernel Space buffer.
- DM75xx_Error DM75xx_DMA_Buffer_Read (DM75xx_Board_Descriptor ∗handle, dm75xx_dma_channel_t channel, unsigned long num_ints)

  Copy the Kernel Space buffers data incrementally into our User Space buffer.
- DM75xx_Error DM75xx_DMA_Buffer_Create (DM75xx_Board_Descriptor ∗handle, uint16_t ∗∗buffer, dm75xx_dma_channel_t channel, uint32_t samples)

  Create a buffer in which the user should place data from the device's DMA buffers.
- DM75xx_Error DM75xx_DMA_Buffer_Free (DM75xx_Board_Descriptor ∗handle, uint16_t ∗∗buffer, dm75xx_dma_channel_t channel)

Free a buffer previously allocated with DM75xx_DMA_Buffer_Create().

- DM75xx_Error DM75xx_DMA_Init_Arb (DM75xx_Board_Descriptor ∗handle, dm75xx_dma_channel_t channel, dm75xx_dma_source_t source, dm75xx_dma_request_t request, uint32_t samples, uint32_t pci_↩ address)

  Set up direct memory access (DMA) for the given DMA/FIFO channel to/from an arbitrary PCI address.
- DM75xx_Error DM75xx_DMA_Initialize (DM75xx_Board_Descriptor ∗handle, dm75xx_dma_channel_t channel, dm75xx_dma_source_t source, dm75xx_dma_request_t request, uint32_t samples, uint16_t ∗∗buf)

  Set up direct memory access (DMA) for the given DMA/FIFO channel.
- DM75xx_Error DM75xx_DMA_Abort (DM75xx_Board_Descriptor ∗handle, dm75xx_dma_channel_t channel)

  Abort any active transfer on the specified DMA channel.
- DM75xx_Error DM75xx_DMA_Enable (DM75xx_Board_Descriptor ∗handle, dm75xx_dma_channel_t channel, uint8_t enable)

  Set the enable bit for a particular DMA channel. To start DMA after this, call DM75xx_DMA_Start().
- DM75xx_Error DM75xx_DMA_Request_Source (DM75xx_Board_Descriptor ∗handle, dm75xx_dma_↩ channel_t channel, dm75xx_dma_request_t request)

  Set the demand mode request source for a specified DMA channel.
- DM75xx_Error DM75xx_DMA_Start (DM75xx_Board_Descriptor ∗handle, dm75xx_dma_channel_t channel)

  Sets the start bit for a particular DMA Channel. DMA will start if the enable bit has been set by DM75xx_DMA_↩ Enable().
- DM75xx_Error DM75xx_Board_Close (DM75xx_Board_Descriptor ∗handle)

  Close a DM75xx device file.
- DM75xx_Error DM75xx_Board_Open (uint8_t dev_num, DM75xx_Board_Descriptor ∗∗handle)

  Open a DM75xx device file.
- DM75xx_Error DM75xx_FIFO_Size (DM75xx_Board_Descriptor ∗handle, unsigned int ∗data)

  Retrieve the FIFO size of the board from the kernel space device descriptor.
- DM75xx_Error DM75xx_Board_Type (DM75xx_Board_Descriptor ∗handle, dm75xx_board_t ∗data)

  Determine the family of the board (DM7520 or SDM7540/8540).
- DM75xx_Error DM75xx_InstallISR (DM75xx_Board_Descriptor ∗handle, void(∗isr_fnct)(unsigned int status))

  Install userspace ISR.
- DM75xx_Error DM75xx_RemoveISR (DM75xx_Board_Descriptor ∗handle)

  Uninstall userspace ISR.
- void ∗ DM75xx_WaitForInterrupt (void ∗ptr)

  Function that will have its own thread and wait for interrupts to occur. Once an interrupt is received this function will call our callback ISR and pass it the interrupt status.
- DM75xx_Error DM75xx_UTC_Set_Clock_Source (DM75xx_Board_Descriptor ∗handle, dm75xx_utc_timer↩ _t utc, dm75xx_utc_clk_t source)

  Set a User Timer/Counter Clock Source.
- DM75xx_Error DM75xx_UTC_Set_Gate (DM75xx_Board_Descriptor ∗handle, dm75xx_utc_timer_t utc, dm75xx_utc_gate gate)

  Set a User Timer/Counter Gate.
- DM75xx_Error DM75xx_UTC_Set_Mode (DM75xx_Board_Descriptor ∗handle, dm75xx_utc_timer_t utc, dm75xx_utc_mode mode)

  Set a User Timer/Counter Mode.
- DM75xx_Error DM75xx_UTC_Get_Mode (DM75xx_Board_Descriptor ∗handle, dm75xx_utc_timer_t utc, uint16_t ∗mode)

  Set a User Timer/Counter Mode.
- DM75xx_Error DM75xx_UTC_Set_Divisor (DM75xx_Board_Descriptor ∗handle, dm75xx_utc_timer_t utc, uint16_t rate)

  Set a User Timer/Counter Divisor.
- DM75xx_Error DM75xx_UTC_Get_Count (DM75xx_Board_Descriptor ∗handle, dm75xx_utc_timer_t utc, uint16_t ∗count)

  Return current value of a User Timer/Counter.

- DM75xx_Error DM75xx_UTC_Get_Status (DM75xx_Board_Descriptor ∗handle, dm75xx_utc_timer_t utc_↩ select, uint8_t ∗utc_status)

  Return current status of a User Timer/Counter.
- DM75xx_Error DM75xx_UTC_Setup (DM75xx_Board_Descriptor ∗handle, dm75xx_utc_timer_t utc, dm75xx_utc_clk_t source, dm75xx_utc_gate gate, dm75xx_utc_mode mode, uint16_t divisor)

  Setup a User Timer/Counter.
- DM75xx_Error DM75xx_BCLK_Get_Count (DM75xx_Board_Descriptor ∗handle, uint16_t ∗data)

  Get the current Burst Clock count.
- DM75xx_Error DM75xx_BCLK_Set_Count (DM75xx_Board_Descriptor ∗handle, uint16_t data)

  Set the current Burst Clock count.
- DM75xx_Error DM75xx_BCLK_Set_Rate (DM75xx_Board_Descriptor ∗handle, dm75xx_bclk_freq_t freq, float rate, float ∗actualRate)

  Set the Burst Clock rate.
- DM75xx_Error DM75xx_BCLK_Set_Start (DM75xx_Board_Descriptor ∗handle, dm75xx_bclk_start_t start)

  Set Burst Clock start trigger.
- DM75xx_Error DM75xx_BCLK_Set_Frequency (DM75xx_Board_Descriptor ∗handle, dm75xx_bclk_freq_↩ t freq)

  Set the Burst Clock primary frequency.
- DM75xx_Error DM75xx_BCLK_Setup (DM75xx_Board_Descriptor ∗handle, dm75xx_bclk_start_t start, dm75xx_bclk_freq_t freq, float rate, float ∗actualRate)

  Setup Burst Clock.
- DM75xx_Error DM75xx_PCLK_Set_Frequency (DM75xx_Board_Descriptor ∗handle, dm75xx_pclk_freq_↩ t pclk_freq)

  Set the Pacer Clock frequency.
- DM75xx_Error DM75xx_PCLK_Set_Source (DM75xx_Board_Descriptor ∗handle, dm75xx_pclk_select_↩ t pclk_select)

  Set the Pacer Clock source.
- DM75xx_Error DM75xx_PCLK_Set_Start (DM75xx_Board_Descriptor ∗handle, dm75xx_pclk_start_t pclk↩ _start)

  Set the Pacer Clock start trigger.
- DM75xx_Error DM75xx_PCLK_Set_Stop (DM75xx_Board_Descriptor ∗handle, dm75xx_pclk_stop_t pclk_↩ stop)

  Set the Pacer Clock stop trigger.
- DM75xx_Error DM75xx_PCLK_Read (DM75xx_Board_Descriptor ∗handle, uint32_t ∗pacer_value)

  Read the current pacer clock value.
- DM75xx_Error DM75xx_PCLK_Set_Trigger_Mode (DM75xx_Board_Descriptor ∗handle, dm75xx_pclk_↩ mode_t pclk_mode)

  Set the Pacer Clock trigger mode.
- DM75xx_Error DM75xx_PCLK_Set_Count (DM75xx_Board_Descriptor ∗handle, uint32_t count)

  Set the Pacer Clock Count.
- DM75xx_Error DM75xx_PCLK_Set_Rate (DM75xx_Board_Descriptor ∗handle, dm75xx_pclk_freq_t freq, float rate, float ∗actualRate)

  Set the Pacer Clock Rate.
- DM75xx_Error DM75xx_PCLK_Setup (DM75xx_Board_Descriptor ∗handle, dm75xx_pclk_select_t pclk_↩ select, dm75xx_pclk_freq_t pclk_freq, dm75xx_pclk_mode_t pclk_mode, dm75xx_pclk_start_t pclk_start, dm75xx_pclk_stop_t pclk_stop, float rate, float ∗actualRate)

  Setup the Pacer Clock.
- DM75xx_Error DM75xx_PCLK_Start (DM75xx_Board_Descriptor ∗handle)

  Software Pacer Clock Start.
- DM75xx_Error DM75xx_PCLK_Stop (DM75xx_Board_Descriptor ∗handle)

  Software Pacer Clock Stop.
- DM75xx_Error DM75xx_CGT_Reset (DM75xx_Board_Descriptor ∗handle)

Reset Channel Gain Table.

- DM75xx_Error DM75xx_CGT_Clear (DM75xx_Board_Descriptor ∗handle)

  Clear Channel Gain Table.
- DM75xx_Error DM75xx_CGT_Create_Entry (dm75xx_cgt_entry_t ∗cgt, uint16_t ∗cgt_entry)

  Create a channel gain table entry.
- DM75xx_Error DM75xx_CGT_Write (DM75xx_Board_Descriptor ∗handle, dm75xx_cgt_entry_t cgt)

  Write a channel gain table entry. This function utilizes DM75xx_CGT_Create_Entry() to create the 16 bit entry.
- DM75xx_Error DM75xx_CGT_Latch (DM75xx_Board_Descriptor ∗handle, dm75xx_cgt_entry_t cgt)

  Write ADC channel gain table latch for single channel sampling.
- DM75xx_Error DM75xx_CGT_Enable (DM75xx_Board_Descriptor ∗handle, uint16_t enable)

  Enable/disable A/D channel gain table.
- DM75xx_Error DM75xx_DT_Enable (DM75xx_Board_Descriptor ∗handle, uint16_t enable)

  Enable/disable Digital Table.
- DM75xx_Error DM75xx_DT_Write_Entry (DM75xx_Board_Descriptor ∗handle, uint8_t data)

  Write Digital Table entry.
- DM75xx_Error DM75xx_CGT_Pause (DM75xx_Board_Descriptor ∗handle, uint16_t pause)

  Pause the Channel Gain Table.

- DM75xx_Error DM75xx_ADC_FIFO_Read (DM75xx_Board_Descriptor ∗handle, uint16_t ∗value)

  DM75xx_Library_ADC_Functions DM75xx user library analog to digital.
- DM75xx_Error DM75xx_ADC_Software_Sample (DM75xx_Board_Descriptor ∗handle)

  Analog to Digital Software Sample.
- DM75xx_Error DM75xx_ADC_Conv_Signal (DM75xx_Board_Descriptor ∗handle, dm75xx_adc_conv_↩ signal_t adc_conv_signal)

  Select the A/D Conversion Signal.
- DM75xx_Error DM75xx_ADC_SCNT_Source (DM75xx_Board_Descriptor ∗handle, dm75xx_adc_scnt_src↩ _t src)

  Select the A/D Sample Counter Source.
- DM75xx_Error DM75xx_ADC_About_Enable (DM75xx_Board_Descriptor ∗handle, uint16_t enable)

  Enable/Disable About Counter stop.
- DM75xx_Error DM75xx_ADC_Clear (DM75xx_Board_Descriptor ∗handle)

  Clear Analag to Digital FIFO.
- DM75xx_Error DM75xx_ADC_SCNT_Read (DM75xx_Board_Descriptor ∗handle, uint16_t ∗data)

  Read the value in the A/D Sample Counter.
- DM75xx_Error DM75xx_ADC_SCNT_Load (DM75xx_Board_Descriptor ∗handle, uint16_t data)

  Load a value into the A/D Sample Counter.

- DM75xx_Error DM75xx_DAC_Soft_Update (DM75xx_Board_Descriptor ∗handle, uint8_t dac)

  DM75xx_Library_DAC_Functions DM75xx user library digital to analog.
- DM75xx_Error DM75xx_DAC_Get_Update_Counter (DM75xx_Board_Descriptor ∗handle, dm75xx_dac_↩ channel_t dac, uint16_t ∗data)

  Get DAC update counter for a specified channel.
- DM75xx_Error DM75xx_DAC_Set_Update_Counter (DM75xx_Board_Descriptor ∗handle, dm75xx_dac_↩ channel_t dac, uint16_t data)

  Set the DAC update counter for a specified channel.
- DM75xx_Error DM75xx_DAC_Set_Range (DM75xx_Board_Descriptor ∗handle, dm75xx_dac_channel_↩ t dac, dm75xx_dac_range_t range)

  Set the DAC output range for a specified channel.
- DM75xx_Error DM75xx_DAC_Set_Update_Source (DM75xx_Board_Descriptor ∗handle, dm75xx_dac_↩ channel_t dac, dm75xx_dac_update_src_t src)

  Set the DAC Update Source for the specified channel.

- DM75xx_Error DM75xx_DAC_Set_Mode (DM75xx_Board_Descriptor ∗handle, dm75xx_dac_channel_t dac, dm75xx_dac_mode_t mode)

  Set the DAC mode for a specified channel.
- DM75xx_Error DM75xx_DAC_FIFO_Write (DM75xx_Board_Descriptor ∗handle, dm75xx_dac_channel_↩ t dac, uint16_t data)

  Write a value to the DAC FIFO of a specified channel.
- DM75xx_Error DM75xx_DAC_Set_Frequency (DM75xx_Board_Descriptor ∗handle, dm75xx_dac_freq_↩ t freq)

  Set the primary slock frequency for DAC conversion.
- DM75xx_Error DM75xx_DAC_Set_Count (DM75xx_Board_Descriptor ∗handle, uint32_t count)

  Set the DAC Clock Count.
- DM75xx_Error DM75xx_DAC_Set_Rate (DM75xx_Board_Descriptor ∗handle, dm75xx_dac_freq_t freq, uint32_t rate, float ∗actualRate)

  Set the DAC conversion rate.
- DM75xx_Error DM75xx_DAC_Set_Clock_Stop (DM75xx_Board_Descriptor ∗handle, dm75xx_dac_clk_↩ stop_t stop)

  Set the DAC Clock Stop Value.
- DM75xx_Error DM75xx_DAC_Set_Clock_Start (DM75xx_Board_Descriptor ∗handle, dm75xx_dac_clk_↩ start_t start)

  Set the DAC Clock Start Value.
- DM75xx_Error DM75xx_DAC_Start (DM75xx_Board_Descriptor ∗handle)

  Causes a DAC Software Start.
- DM75xx_Error DM75xx_DAC_Stop (DM75xx_Board_Descriptor ∗handle)

  Causes a DAC Software Stop.
- DM75xx_Error DM75xx_DAC_Setup (DM75xx_Board_Descriptor ∗handle, dm75xx_dac_channel_t dac, dm75xx_dac_range_t range, dm75xx_dac_update_src_t src, dm75xx_dac_mode_t mode)

  Setup a DAC channel.
- DM75xx_Error DM75xx_DAC_Reset (DM75xx_Board_Descriptor ∗handle, dm75xx_dac_channel_t dac)

  Reset a DAC Fifo.
- DM75xx_Error DM75xx_DAC_Clear (DM75xx_Board_Descriptor ∗handle, dm75xx_dac_channel_t dac)

  Clear a DAC Fifo.
- DM75xx_Error DM75xx_DAC_Set_CLK_Mode (DM75xx_Board_Descriptor ∗handle, dm75xx_dac_clk_↩ mode_t clk_mode)

  Set DAC Clock Mode.

- DM75xx_Error DM75xx_HSDIN_Software_Sample (DM75xx_Board_Descriptor ∗handle)

  DM75xx_Library_HSDIN_Functions DM75xx user library high speed digital.
- DM75xx_Error DM75xx_HSDIN_Sample_Signal (DM75xx_Board_Descriptor ∗handle, dm75xx_hsdin_↩ signal_t signal)

  Set HighSpeed digital sampling signal.
- DM75xx_Error DM75xx_HSDIN_Clear (DM75xx_Board_Descriptor ∗handle)

  Clear High Speed Digital FIFO.
- DM75xx_Error DM75xx_HSDIN_FIFO_Read (DM75xx_Board_Descriptor ∗handle, uint16_t ∗data)

  Read value from High Speed Digital FIFO.

- DM75xx_Error DM75xx_SBUS_Set_Source (DM75xx_Board_Descriptor ∗handle, dm75xx_sbus_t sbus, dm75xx_sbus_src_t src)

  DM75xx_Library_SBUS_Functions DM75xx user library syncbus.
- DM75xx_Error DM75xx_SBUS_Enable (DM75xx_Board_Descriptor ∗handle, dm75xx_sbus_t sbus, uint16↩ _t enable)

  Enable/Disable Syncbus.

- DM75xx_Error DM75xx_ACNT_Get_Count (DM75xx_Board_Descriptor ∗handle, uint16_t ∗data)

    DM75xx_Library_ACNT_Functions DM75xx user library about counter.
- DM75xx_Error DM75xx_ACNT_Set_Count (DM75xx_Board_Descriptor ∗handle, uint16_t data)

    Set the About Counter value.

- DM75xx_Error DM75xx_DCNT_Get_Count (DM75xx_Board_Descriptor ∗handle, uint16_t ∗data)

    DM75xx_Library_DCNT_Functions DM75xx user library delay counter.
- DM75xx_Error DM75xx_DCNT_Set_Count (DM75xx_Board_Descriptor ∗handle, uint16_t data)

    Set the Delay Counter value.

- DM75xx_Error DM75xx_DIO_Set_Port (DM75xx_Board_Descriptor ∗handle, dm75xx_dio_port_t port, uint8_t data)

    DM75xx_Library_DIO_Functions DM75xx user library digital input/output.
- DM75xx_Error DM75xx_DIO_Get_Port (DM75xx_Board_Descriptor ∗handle, dm75xx_dio_port_t port, uint8_t ∗data)

    Get the value from the specified Digital I/O Port.
- DM75xx_Error DM75xx_DIO_Get_Status (DM75xx_Board_Descriptor ∗handle, uint8_t ∗data)

    Get the Digital I/O Status byte.
- DM75xx_Error DM75xx_DIO_Clear_IRQ (DM75xx_Board_Descriptor ∗handle)

    Clear Digital I/O IRQ Status.
- DM75xx_Error DM75xx_DIO_Reset (DM75xx_Board_Descriptor ∗handle)

    Clear Digital I/O Chip.
- DM75xx_Error DM75xx_DIO_Set_Direction (DM75xx_Board_Descriptor ∗handle, dm75xx_dio_port_t port, uint8_t direction)

    Set the direction of the specified Digital I/O Port.
- DM75xx_Error DM75xx_DIO_Set_Mask (DM75xx_Board_Descriptor ∗handle, uint8_t mask)

    Set Digital I/O Port 0 Mask.
- DM75xx_Error DM75xx_DIO_Set_Compare (DM75xx_Board_Descriptor ∗handle, uint8_t compare)

    Set the compare register for Digital I/O Port 0.
- DM75xx_Error DM75xx_DIO_Get_Compare (DM75xx_Board_Descriptor ∗handle, uint8_t ∗compare)

    Get the compare register for Digital I/O Port 0.
- DM75xx_Error DM75xx_DIO_IRQ_Mode (DM75xx_Board_Descriptor ∗handle, dm75xx_dio_mode_t mode)

    Set the IRQ Mode for Digital I/O.
- DM75xx_Error DM75xx_DIO_Clock (DM75xx_Board_Descriptor ∗handle, dm75xx_dio_clk_t clock)

    Set the Digital I/O Sample Clock.
- DM75xx_Error DM75xx_DIO_Enable_IRQ (DM75xx_Board_Descriptor ∗handle, uint8_t enable)

    Enable/Disable Digital I/O Interrupts.

- DM75xx_Error DM75xx_UIO_Select (DM75xx_Board_Descriptor ∗handle, dm75xx_uio_channel_t channel, dm75xx_uio_source_t source)

    DM75xx_Library_UIO_Functions DM75xx user library user I/O.
- DM75xx_Error DM75xx_UIO_Read (DM75xx_Board_Descriptor ∗handle, uint32_t ∗data)

    Read the current status of the user I/O.
- DM75xx_Error DM75xx_UIO_Write (DM75xx_Board_Descriptor ∗handle, uint32_t data)

    Write the value of the user I/O.

- DM75xx_Error DM75xx_McBSP_ADC_FIFO (DM75xx_Board_Descriptor ∗handle, uint8_t enable)

    DM75xx_Library_McBSP_Functions DM75xx user library mcbsp.
- DM75xx_Error DM75xx_McBSP_DAC_FIFO (DM75xx_Board_Descriptor ∗handle, uint8_t enable)

    Enable/Disable D/A FIFO to DSP.

- DM75xx_Error DM75xx_ETRIG_Polarity_Select (DM75xx_Board_Descriptor ∗handle, dm75xx_ext_↩ polarity_t polarity)

  DM75xx_Library_EXT_Functions DM75xx user library external trigger/interrupt.
- DM75xx_Error DM75xx_EINT_Polarity_Select (DM75xx_Board_Descriptor ∗handle, dm75xx_ext_polarity_t polarity)

  Set the External Interrupt polarity.
- DM75xx_Error DM75xx_FIFO_Get_Status (DM75xx_Board_Descriptor ∗handle, uint16_t ∗fifo_status)

  DM75xx_Library_STATUS_Functions DM75xx user library status.
- DM75xx_Error DM75xx_CLK_Get_Status (DM75xx_Board_Descriptor ∗handle, uint16_t ∗status)

  Get status of pacer/burst clocks.
- DM75xx_Error DM75xx_Calibrate (DM75xx_Board_Descriptor ∗handle, uint16_t dac1_value, uint16_↩ t dac2_value, dm75xx_dac_range_t dac1_range, dm75xx_dac_range_t dac2_range)

  DM75xx_Library_SDM7540_Functions DM75xx user library SDM7540 functions.
- DM75xx_Error DM75xx_DSP_CMD_Send (DM75xx_Board_Descriptor ∗handle, dm75xx_dsp_command_t command)

  Issue a command to the 7540 onboard DSP.
- DM75xx_Error DM75xx_DSP_CMD_Complete (DM75xx_Board_Descriptor ∗handle, uint8_t ∗data)

  Checks if the last command given to the DSP is finished.
- DM75xx_Error DM75xx_DSP_CMD_Status (DM75xx_Board_Descriptor ∗handle, dm75xx_dsp_command↩ _t command)

  Checks whether or not a command successfully completed on the DSP.
- DM75xx_Error DM75xx_ALGDIO_Get_Mask (DM75xx_Board_Descriptor ∗handle, dm75xx_algdio_mask_t ∗pin1, dm75xx_algdio_mask_t ∗pin2)

  Get the the mask of the Analog DIO.
- DM75xx_Error DM75xx_ALGDIO_Set_Mask (DM75xx_Board_Descriptor ∗handle, dm75xx_algdio_mask_t pin1, dm75xx_algdio_mask_t pin2)

  Set the Analog DIO Mask.
- DM75xx_Error DM75xx_ALGDIO_Get_Direction (DM75xx_Board_Descriptor ∗handle, dm75xx_algdio_↩ direction_t ∗pin1, dm75xx_algdio_direction_t ∗pin2)

  Get the Analog DIO Direction.
- DM75xx_Error DM75xx_ALGDIO_Set_Direction (DM75xx_Board_Descriptor ∗handle, dm75xx_algdio_↩ direction_t pin1, dm75xx_algdio_direction_t pin2)

  Set the Analog DIO Direction.
- DM75xx_Error DM75xx_ALGDIO_Set_Data (DM75xx_Board_Descriptor ∗handle, uint8_t pin1, uint8_t pin2)

  Set the Analog DIO pin values.
- DM75xx_Error DM75xx_ALGDIO_Get_Data (DM75xx_Board_Descriptor ∗handle, uint8_t ∗pin1, uint8_↩ t ∗pin2)

  Get the Analog DIO pin values.
- DM75xx_Error DM75xx_ALGDIO_Get_IRQ_Status (DM75xx_Board_Descriptor ∗handle, uint8_t ∗status)

  Get Analog DIO IRQ Status.
- DM75xx_Error DM75xx_Get_Temp (DM75xx_Board_Descriptor ∗handle, uint8_t ∗temp)

  Get the temperature from the board.

6.30.1 Detailed Description

DM75xx user library definitions.

```
$Id: dm75xx_library.h 99065 2016-04-26 18:03:23Z rgroner $
```

Definition in file dm75xx_library.h.

6.31 include/dm75xx_registers.h File Reference

Register definitions for DM75xx devices.

Macros

- #define DM75xx_PLX_ITCSR 0x68

  Interupt Control/Status.
- #define DM75xx_PLX_DMA_MODE0 0x80

  DMA Channel 0 Mode.
- #define DM75xx_PLX_DMA_PADR0 0x84

  DMA Channel 0 PCI Address.
- #define DM75xx_PLX_DMA_LADR0 0x88

  DMA Channel 0 Local Address.
- #define DM75xx_PLX_DMA_SIZE0 0x8C

  DMA Channel 0 Transfer Size (Bytes)
- #define DM75xx_PLX_DMA_DPR0 0x90

  DMA Channel 0 Descriptor Pointer.
- #define DM75xx_PLX_DMA_MODE1 0x94

  DMA Channel 1 Mode.
- #define DM75xx_PLX_DMA_PADR1 0x98

  DMA Channel 1 PCI Address.
- #define DM75xx_PLX_DMA_LADR1 0x9C

  DMA Channel 1 Local Address.
- #define DM75xx_PLX_DMA_SIZE1 0xA0

  DMA Channel 1 Transfer Size (Bytes)
- #define DM75xx_PLX_DMA_DPR1 0xA4

  DMA Channel 1 Descriptor Pointer.
- #define DM75xx_PLX_DMA_CSR0 0xA8

  DMA Command/Status channel 0.
- #define DM75xx_PLX_DMA_CSR1 0xA9

  DMA Command/Status channel 1.
- #define DM75xx_PLX_DMA_ARB 0xAC

  DMA Arbitration.
- #define DM75xx_PLX_DMA_THR 0xB0

  DMA Threshold.
- #define DM75xx_PLX_SRAM 0xF4

  PLX LAS1 SRAM access space.

- #define DM75xx_LAS0_MT_MODE 0x00

  Read Master/Target Only mode and Firmware version.
- #define DM75xx_LAS0_SPARE_04 0x04

  Spare 0x04.
- #define DM75xx_LAS0_USER_IO 0x08

  Read/Write user inputs.
- #define DM75xx_LAS0_DAC_CLK_ST 0x0C

  Start/Stop software clock.
- #define DM75xx_LAS0_FIFO_STATUS 0x10

  Read FIFO Status.
- #define DM75xx_LAS0_DAC1 0x14

  Software DAC1 Update.
- #define DM75xx_LAS0_DAC2 0x18

  Software DAC2 Update.
- #define DM75xx_LAS0_SPARE_1C 0x1C

  Spare 0x1C.
- #define DM75xx_LAS0_SPARE_20 0x20

  Spare 0x20.
- #define DM75xx_LAS0_DAC 0x24

  Software simultaneous DAC1 and DAC2 Update.
- #define DM75xx_LAS0_PACER 0x28

  Start/Stop software Pacer.
- #define DM75xx_LAS0_TIMER 0x2C

  Read: Read Timer Counter Status Write: Software HSDIN Sample Command.
- #define DM75xx_LAS0_IT 0x30

  Read: Read interrupt status Write: Write IT enable mask.
- #define DM75xx_LAS0_CLEAR_IT 0x34

  Read: Clear ITs via mask Write: Set IT clear mask.
- #define DM75xx_LAS0_IT_OVERRUN 0x38

  Read: Read IT overrrun Write: Clear IT overrun.
- #define DM75xx_LAS0_SPARE_3C 0x3C

  Spare 0x3C.
- #define DM75xx_LAS0_PCLK 0x40

  Read/Write Pacer Clock.
- #define DM75xx_LAS0_BCLK 0x44

  Read/Write Burst Clock.
- #define DM75xx_LAS0_ADC_SCNT 0x48

  Read/Write ADC sample counter.
- #define DM75xx_LAS0_DAC1_UCNT 0x4C

  Read/Write DAC1 update counter.
- #define DM75xx_LAS0_DAC2_UCNT 0x50

  Read/Write DAC2 update counter.
- #define DM75xx_LAS0_DCNT 0x54

  Read/Write delay counter.
- #define DM75xx_LAS0_ACNT 0x58

  Read/Write about counter.
- #define DM75xx_LAS0_DAC_CLK 0x5C

  Read/Write DAC clock.
- #define DM75xx_LAS0_UTC0 0x60

  Read/Write UTC0 value.
- #define DM75xx_LAS0_UTC1 0x64

Read/Write UTC1 value.
- #define DM75xx_LAS0_UTC2 0x68

    Read/Write UTC2 value.
- #define DM75xx_LAS0_UTC_CTRL 0x6C

    UTC Control.
- #define DM75xx_LAS0_DIO0 0x70

    Read/Program digital input port 0.
- #define DM75xx_LAS0_DIO1 0x74

    Read/Program digital input port 1.
- #define DM75xx_LAS0_DIO_CTRL 0x78

    Clear digital IRQ status, read/program port 0 direction, mask, or compare register.
- #define DM75xx_LAS0_DIO_STATUS 0x7C

    Read DIO Status or Program digital control/interrupts.
- #define DM75xx_LAS0_DSP 0xB0

    Read: DSP Command register to be written from the Host side and read from the DSP Write: DSP status written to by the DSP and read from the Host side.
- #define DM75xx_LAS0_ALGDIO_MASK 0xE0

    Read/Write analog connection DIO mask.
- #define DM75xx_LAS0_ALGDIO_DATA 0xE4

    Read/Write analog connection DIO data.
- #define DM75xx_LAS0_ALGDIO_DIR 0xE8

    read/Write analog connection DIO direction
- #define DM75xx_LAS0_ALGDIO_IRQ 0xEC

    Read analog connection DIO IRQ status.
- #define DM75xx_LAS0_I2C_ADDR 0xC0

    I2C Bus Address.
- #define DM75xx_LAS0_I2C_PTR 0xC4

    I2C Bus Pointer.
- #define DM75xx_LAS0_I2C_DATA 0xC8

    I2C Bus Data.
- #define DM75xx_LAS0_I2C_GO 0xCC

    I2C Bus Go.
- #define DM75xx_LAS0_I2C_READ 0xD0

    I2C Bus Read.
- #define DM75xx_LAS0_BOARD_RESET 0x100

    Software board reset.
- #define DM75xx_LAS0_DMA0_SRC 0x104

    DMA Channel 0 Source.
- #define DM75xx_LAS0_DMA1_SRC 0x108

    DMA Channel 1 Source.
- #define DM75xx_LAS0_DMA_RSTRQST0 0x1CC

    Reset DMA Channel 0 Request Machine.
- #define DM75xx_LAS0_DMA_RSTRQST1 0x1D0

    Reset DMA Channel 1 Request Machine.
- #define DM75xx_LAS0_ADC_CONV 0x10C

    Select ADC Conversion Signal.
- #define DM75xx_LAS0_BURST_START 0x110

    Select Burst Clock Start Trigger.
- #define DM75xx_LAS0_PACER_START 0x114

    Select Pacer Clock Start Trigger.
- #define DM75xx_LAS0_PACER_STOP 0x118

Select Pacer Clock Stop Trigger.
- #define DM75xx_LAS0_ACNT_ENABLE 0x11C

  About Counter Stop Enable.
- #define DM75xx_LAS0_PACER_MODE 0x120

  Pacer Clock Start Trigger Mode.
- #define DM75xx_LAS0_HSDIN_START 0x124

  Select HighSpeed Digital Sampling Signal.
- #define DM75xx_LAS0_HSDIN_FIFO_CLR 0x128

  Clear HighSpeed Digital FIFO.
- #define DM75xx_LAS0_ADC_FIFO_CLR 0x12C

  Clear ADC FIFO.
- #define DM75xx_LAS0_CGT_WRITE 0x130

  Write CGT Multi-Channel.
- #define DM75xx_LAS0_CGT_LATCH 0x134

  Write CGT Latch Single-Channel.
- #define DM75xx_LAS0_DT_WRITE 0x138

  Write Digital Table.
- #define DM75xx_LAS0_CGT_ENABLE 0x13C

  Enable CGT.
- #define DM75xx_LAS0_DT_ENABLE 0x140

  Enable Digital Table.
- #define DM75xx_LAS0_PAUSE_TABLE 0x144

  Table Pause Enable.
- #define DM75xx_LAS0_CGT_RESET 0x148

  Reset CGT.
- #define DM75xx_LAS0_CGT_CLEAR 0x14C

  Clear CGT.
- #define DM75xx_LAS0_DAC1_RANGE 0x150

  Select DAC1 Output Range.
- #define DM75xx_LAS0_DAC1_SRC 0x154

  Select DAC1 Update Source.
- #define DM75xx_LAS0_DAC1_CYCLE 0x158

  Select DAC1 Cycle Mode.
- #define DM75xx_LAS0_DAC1_RESET 0x15C

  Reset DAC1 FIFO.
- #define DM75xx_LAS0_DAC1_CLEAR 0x160

  Clear DAC1 FIFO.
- #define DM75xx_LAS0_DAC2_RANGE 0x164

  Select DAC2 Output Range.
- #define DM75xx_LAS0_DAC2_SRC 0x168

  Select DAC2 Update Source.
- #define DM75xx_LAS0_DAC2_CYCLE 0x16C

  Select DAC2 Cycle Mode.
- #define DM75xx_LAS0_DAC2_RESET 0x170

  Reset DAC2 FIFO.
- #define DM75xx_LAS0_DAC2_CLEAR 0x174

  Clear DAC2 FIFO.
- #define DM75xx_LAS0_SBUS0_SOURCE 0x184

  Select SyncBus 0 Source.
- #define DM75xx_LAS0_SBUS0_ENABLE 0x188

  Syncbus 0 Enable.

- #define DM75xx_LAS0_SBUS1_SOURCE 0x18C

  Select SyncBus 1 Source.
- #define DM75xx_LAS0_SBUS1_ENABLE 0x190

  SyncBus 1 Enable.
- #define DM75xx_LAS0_SBUS2_SOURCE 0x198

  Select SyncBus 2 Source.
- #define DM75xx_LAS0_SBUS2_ENABLE 0x19C

  SyncBus 2 Enable.
- #define DM75xx_LAS0_ETRG_POLARITY 0x1A4

  Select External Trigger Polarity.
- #define DM75xx_LAS0_EINT_POLARITY 0x1A8

  Select External Interrupt Polarity.
- #define DM75xx_LAS0_UTC0_CLOCK 0x1AC

  Select UTC0 Clock.
- #define DM75xx_LAS0_UTC0_GATE 0x1B0

  Select UTC0 Gate.
- #define DM75xx_LAS0_UTC1_CLOCK 0x1B4

  Select UTC1 Clock.
- #define DM75xx_LAS0_UTC1_GATE 0x1B8

  Select UTC1 Gate.
- #define DM75xx_LAS0_UTC2_CLOCK 0x1BC

  Select UTC2 Clock.
- #define DM75xx_LAS0_UTC2_GATE 0x1C0

  Select UTC2 Gate.
- #define DM75xx_LAS0_UIO0_SELECT 0x1C4

  Select User Output Signal 0.
- #define DM75xx_LAS0_UIO1_SELECT 0x1C8

  Select User Output Signal 1.
- #define DM75xx_LAS0_ADC_SCNT_SRC 0x178

  Select ADC Sample Counter Source.
- #define DM75xx_LAS0_PACER_SELECT 0x180

  Select Pacer Clock.
- #define DM75xx_LAS0_DAC_CLK_START 0x1D4

  Select DAC Clock Start.
- #define DM75xx_LAS0_DAC_CLK_STOP 0x1D8

  Select DAC Clock Stop.
- #define DM75xx_LAS0_PCLK_FREQ 0x1DC

  Select Pacer Clock Frequency.
- #define DM75xx_LAS0_BCLK_FREQ 0x1E0

  Select Burst Clock Frequency.
- #define DM75xx_LAS0_DAC_CLK_SOURCE 0x1E4

  Select DAC Clock Source.
- #define DM75xx_LAS0_DAC_CLK_MODE 0x1E8

  Select DAC Clock Mode.
- #define DM75xx_LAS0_MCBSP_AD_CTRL 0x1EC

  ADC FIFO Data to DSP Enable.
- #define DM75xx_LAS0_MCBSP_DA_CTRL 0x1F0

  DAC FIFO Data from DSP Enable.
- #define DM75xx_LAS0_FIFO_ADR_MODE 0x1F4

  Select FIFO Addressing Mode.
- #define DM75xx_LAS1_ADC_FIFO 0x00

Read ADC FIFO.
- #define DM75xx_LAS1_HSDIN_FIFO 0x04

  Read HighSpeed Digital FIFO.
- #define DM75xx_LAS1_DAC1_FIFO 0x08

  Write DAC1 FIFO.
- #define DM75xx_LAS1_DAC2_FIFO 0x0C

  Write DAC2 FIFO.
- #define DMALADDR_ADC 0x40000000

  Local Address for ADC FIFO.
- #define DMALADDR_HSDIN 0x40000004

  Local Address for HSDIN FIFO.
- #define DMALADDR_DAC1 0x40000008

  Local Address for DAC1 FIFO.
- #define DMALADDR_DAC2 0x4000000C

  Local Address for DAC2 FIFO.
- #define DISABLED 0x00

  Generic Disable Logic 0x00.
- #define ENABLED 0x01

  Generic Enable Logic 0x01.
- #define NO_ARG 0x00

  Dummy Value.
- #define BIT_00 0x00000001

  Bit 0.
- #define BIT_01 0x00000002

  Bit 1.
- #define BIT_02 0x00000004

  Bit 2.
- #define BIT_03 0x00000008

  Bit 3.
- #define BIT_04 0x00000010

  Bit 4.
- #define BIT_05 0x00000020

  Bit 5.
- #define BIT_06 0x00000040

  Bit 6.
- #define BIT_07 0x00000080

  Bit 7.
- #define BIT_08 0x00000100

  Bit 8.
- #define BIT_09 0x00000200

  Bit 9.
- #define BIT_10 0x00000400

  Bit 10.
- #define BIT_11 0x00000800

  Bit 11.
- #define BIT_12 0x00001000

  Bit 12.
- #define BIT_13 0x00002000

  Bit 13.
- #define BIT_14 0x00004000

  Bit 14.

- #define BIT_15 0x00008000

  Bit 15.
- #define BIT_16 0x00010000

  Bit 16.
- #define BIT_17 0x00020000

  Bit 17.
- #define BIT_18 0x00040000

  Bit 18.
- #define BIT_19 0x00080000

  Bit 19.
- #define BIT_20 0x00100000

  Bit 20.
- #define BIT_21 0x00200000

  Bit 21.
- #define BIT_22 0x00400000

  Bit 22.
- #define BIT_23 0x00800000

  Bit 23.
- #define BIT_24 0x01000000

  Bit 24.
- #define BIT_25 0x02000000

  Bit 25.
- #define BIT_26 0x04000000

  Bit 26.
- #define BIT_27 0x08000000

  Bit 27.
- #define BIT_28 0x10000000

  Bit 28.
- #define BIT_29 0x20000000

  Bit 29.
- #define BIT_30 0x40000000

  Bit 30.
- #define BIT_31 0x80000000

  Bit 31.
- #define PLX_DMA_WIDTH_16 BIT_00

  PLX DMA Local Bus Width.
- #define PLX_DMA_READY BIT_06

  PLX DMA Ready.
- #define PLX_DMA_LOCAL_BURST BIT_08

  PLX DMA Local Burst.
- #define PLX_DMA_CHAINING BIT_09

  PLX DMA Chaining.
- #define PLX_DMA_DONE_IT BIT_10

  PLX DMA Done Interrupt.
- #define PLX_DMA_LA_MODE BIT_11

  PLX DMA Local Addressing Mode.
- #define PLX_DMA_DEMAND_MODE BIT_12

  PLX DMA Demand Mode.
- #define PLX_DMA_PCI_IT BIT_17

  PLX DMA PCI Interrupt Enable.
- #define PLX_DMA_CONFIG PLX_DMA_WIDTH_16|PLX_DMA_READY|PLX_DMA_LOCAL_BURST|PLX↩
_DMA_DONE_IT|PLX_DMA_LA_MODE|PLX_DMA_PCI_IT

  Demand Mode DMA Configuration.

6.31.1 Detailed Description

Register definitions for DM75xx devices.

```
----------------------------------------------------------------------------
This file and its contents are copyright (C) RTD Embedded Technologies,
Inc.  All Rights Reserved.

This software is licensed as described in the RTD End-User Software License
Agreement.  For a copy of this agreement, refer to the file LICENSE.TXT
(which should be included with this software) or contact RTD Embedded
Technologies, Inc.
----------------------------------------------------------------------------


$Id: dm75xx_registers.h 56268 2011-10-25 18:55:35Z rgroner $
```

Definition in file dm75xx_registers.h.

6.32 include/dm75xx_types.h File Reference

Type definitions used both in kernel and user space.

Data Structures

- struct _dm75xx_int_status
  Interrupts status.
- struct _dm75xx_cgt_entry
  Channel gain table entry.
- struct dm75xx_pci_access_request
  PCI region access request descriptor. This structure holds information about a request to read data from or write data to one of a device's PCI regions.

Typedefs

- typedef enum dm75xx_pci_region_num dm75xx_pci_region_num_t
  Standard PCI region number type.
- typedef enum
  dm75xx_pci_region_access_size dm75xx_pci_region_access_size_t
  Standard PCI region access size type.
- typedef enum _dm75xx_board dm75xx_board_t
  DM75xx Board Type.
- typedef enum _dm75xx_dsp_command dm75xx_dsp_command_t
  DSP Command Type.
- typedef enum _dm75xx_dma_flag dm75xx_dma_flag_t
  DMA Control Flag Type.
- typedef enum _dm75xx_dma_reset dm75x_dma_reset_t
  DMA Status Reset Flag Type.
- typedef enum _dm75xx_dma_request dm75xx_dma_request_t
  DMA Demand Mode Source Type.
- typedef enum _dm75xx_dma_source dm75xx_dma_source_t
  DMA Source Type.
- typedef enum _dm75xx_dma_channel dm75xx_dma_channel_t
  DMA Channel Type.

- typedef enum _dm75xx_int_source dm75xx_int_source_t

  Interrupt Source Type.
- typedef enum _dm75xx_algdio_mask dm75xx_algdio_mask_t

  Analog DIO Mask Type.
- typedef enum _dm75xx_algdio_pin dm75xx_algdio_pin_t

  Analog DIO Pin Type.
- typedef enum _dm75xx_algdio_direction dm75xx_algdio_direction_t

  Analog DIO Direction Type.
- typedef enum _dm75xx_uio_channel dm75xx_uio_channel_t

  User I/O Channel Type.
- typedef enum _dm75xx_uio_source dm75xx_uio_source_t

  User I/O Source Type.
- typedef enum _dm75xx_dio_clk dm75xx_dio_clk_t

  Digital I/O Clock Type.
- typedef enum _dm75xx_dio_mode dm75xx_dio_mode_t

  Digital I/O IRQ Mode Type.
- typedef enum _dm75xx_dio_port dm75xx_dio_port_t

  Digital I/O Port Type.
- typedef enum _dm75xx_ext_polarity dm75xx_ext_polarity_t

  Polarity Type.
- typedef enum _dm75xx_sbus dm75xx_sbus_t

  SyncBus Enumeration Type.
- typedef enum _dm75xx_sbus_src dm75xx_sbus_src_t

  SyncBus Source Select Type.
- typedef enum _dm75xx_hsdin_signal dm75xx_hsdin_signal_t

  HSDIN Sampling Signal Type.
- typedef enum _dm75xx_dac_freq dm75xx_dac_freq_t

  DAC primary clock source type.
- typedef enum _dm75xx_dac_clk_stop dm75xx_dac_clk_stop_t

  DAC clock stop source type.
- typedef enum _dm75xx_dac_clk_start dm75xx_dac_clk_start_t

  DAC clock start source type.
- typedef enum _dm75xx_dac_mode dm75xx_dac_mode_t

  DAC Cycle Mode Type.
- typedef enum _dm75xx_dac_update_src dm75xx_dac_update_src_t

  DAC Update Source Type.
- typedef enum _dm75xx_dac_range dm75xx_dac_range_t

  DAC Output Range Type.
- typedef enum _dm75xx_dac_channel dm75xx_dac_channel_t
- typedef enum _dm75xx_dac_clk_mode dm75xx_dac_clk_mode_t

  DAC Clock Mode Type.
- typedef enum _dm75xx_adc_scnt_src dm75xx_adc_scnt_src_t

  ADC Sample Counter Source Type.
- typedef enum _dm75xx_adc_conv_signal dm75xx_adc_conv_signal_t

  ADC Conversion Signal Select type.
- typedef enum _dm75xx_bclk_freq dm75xx_bclk_freq_t

  Burst Clock primary frequency type.
- typedef enum _dm75xx_bclk_start dm75xx_bclk_start_t

  Burst Clock Start Trigger Type.

- typedef enum _dm75xx_pclk_mode dm75xx_pclk_mode_t

  Pacer Clock Trigger Mode Type.
- typedef enum _dm75xx_pclk_stop dm75xx_pclk_stop_t

  Pacer Clock Stop Type.
- typedef enum _dm75xx_pclk_start dm75xx_pclk_start_t

  Pacer Clock Start Type.
- typedef enum _dm75xx_pclk_select dm75xx_pclk_select_t

  Pacer Clock Select Type.
- typedef enum _dm75xx_pclk_freq dm75xx_pclk_freq_t

  Pacer Clock Frequency type.
- typedef enum _dm75xx_utc_timer dm75xx_utc_timer_t

  8254 timer/counter type
- typedef enum _dm75xx_utc_clk dm75xx_utc_clk_t

  8254 timer/counter clock selector type
- typedef enum _dm75xx_utc_gate dm75xx_utc_gate

  8254 timer/counter gate selector type
- typedef enum _dm75xx_utc_mode dm75xx_utc_mode

  8254 timer/counter waverform mode selector type
- typedef enum _dm75xx_fifo_status dm75xx_fifo_status_t

  FIFO Status Type.
- typedef struct _dm75xx_int_status dm75xx_int_status_t

  Interrupt status type.
- typedef struct _dm75xx_cgt_entry dm75xx_cgt_entry_t

  Channel gain table entry type.
- typedef struct
  dm75xx_pci_access_request dm75xx_pci_access_request_t

  PCI region access request descriptor type.

Enumerations

- enum dm75xx_pci_region_num { DM75xx_PLX_MEM = 0, DM75xx_PLX_IO, DM75xx_LAS0, DM75xx_LAS1 }

  Standard PCI region number.
- enum dm75xx_pci_region_access_size { DM75xx_PCI_REGION_ACCESS_8 = 0, DM75xx_PCI_REGION↩_ACCESS_16, DM75xx_PCI_REGION_ACCESS_32 }

  Desired size in bits of access to standard PCI region.
- enum _dm75xx_board { DM75xx_BOARD_DM7520 = 0, DM75xx_BOARD_SDM7540 }

  DM75xx Board.
- enum _dm75xx_dsp_command {
  DM75xx_DSP_CAL_AUTO = 1, DM75xx_DSP_FLASH_DOWNLOAD = 2, DM75xx_DSP_USER_RUN = 3,
  DM75xx_DSP_USER_UPGRADE = 4,
  DM75xx_DSP_INT_FLASH_ERASE = 5, DM75xx_DSP_EXT_FLASH_ERASE = 6, DM75xx_DSP_ATTE↩NTION = 7, DM75xx_DSP_CAL_DEFAULT = 8,
  DM75xx_DSP_CAL_VERSION = 10, DM75xx_DSP_BOOT_VERSION = 11 }

  DSP Command.
- enum _dm75xx_dma_flag {
  DM75xx_DMA_FLAG_INIT = 0x01, DM75xx_DMA_FLAG_MMAP = 0x02, DM75xx_DMA_FLAG_RESET = 0x04, DM75xx_DMA_FLAG_NONDEMAND = 0x08,
  DM75xx_DMA_FLAG_STATUS = 0x10, DM75xx_DMA_FLAG_ARB = 0x20 }

  DMA Control Flag.
- enum _dm75xx_dma_reset { DM75xx_DMA_RESET_SEL = 0x01, DM75xx_DMA_RESET_VAL = 0x10 }

  DMA Status Reset Flag.

- enum _dm75xx_dma_request {
  DM75xx_DMA_DEMAND_DISABLE = 0, DM75xx_DMA_DEMAND_SCNT_ADC = 1, DM75xx_DMA_DEM↩
  AND_SCNT_DAC1 = 2, DM75xx_DMA_DEMAND_SCNT_DAC2 = 3,
  DM75xx_DMA_DEMAND_UTC1 = 4, DM75xx_DMA_DEMAND_FIFO_ADC = 8, DM75xx_DMA_DEMAN↩
  D_FIFO_DAC1 = 9, DM75xx_DMA_DEMAND_FIFO_DAC2 = 10 }

  DMA Demand Mode Source.
- enum _dm75xx_dma_source { DM75xx_DMA_FIFO_ADC = 0, DM75xx_DMA_FIFO_DAC1, DM75xx_DM↩
  A_FIFO_DAC2, DM75xx_DMA_FIFO_HSDIN }

  DMA Local Source.
- enum _dm75xx_dma_channel { DM75xx_DMA_CHANNEL_0 = 0, DM75xx_DMA_CHANNEL_1 }

  DMA Channel.
- enum _dm75xx_int_source {
  DM75xx_INT_FIFO_WRITE = 0x0001, DM75xx_INT_CGT_RESET = 0x0002, DM75xx_INT_RESERVED =
  0x0004, DM75xx_INT_CGT_PAUSE = 0x0008,
  DM75xx_INT_ABOUT = 0x0010, DM75xx_INT_DELAY = 0x0020, DM75xx_INT_SCNT_ADC = 0x0040, D↩
  M75xx_INT_SCNT_DAC1 = 0x0080,
  DM75xx_INT_SCNT_DAC2 = 0x0100, DM75xx_INT_UTC1 = 0x0200, DM75xx_INT_UTC1_INV = 0x0400,
  DM75xx_INT_UTC2 = 0x0800,
  DM75xx_INT_DIO = 0x1000, DM75xx_INT_EXTERNAL = 0x2000, DM75xx_INT_ETRIG_RISING = 0x4000,
  DM75xx_INT_ETRIG_FALLING = 0x8000,
  DM75xx_INT_DMA_0 = 0x00200000, DM75xx_INT_DMA_1 = 0x00400000, DM75xx_INT_ALGDIO_POS↩
  _PIN1 = 0x04000000, DM75xx_INT_ALGDIO_POS_PIN2 = 0x08000000,
  DM75xx_INT_ALGDI0_NEG_PIN1 = 0x10000000, DM75xx_INT_ALGDIO_NEG_PIN2 = 0x20000000 }

  Interrupt Source.
- enum _dm75xx_algdio_mask { DM75xx_ALGDIO_MASKED = 0, DM75xx_ALGDIO_UNMASKED }

  Analog DIO Mask.
- enum _dm75xx_algdio_pin { DM75xx_ALGDIO_PIN1 = 0, DM75xx_ALGDIO_PIN2 }

  Analog DIO Pins.
- enum _dm75xx_algdio_direction { DM75xx_ALGDIO_INPUT = 0, DM75xx_ALGDIO_OUTPUT }

  Analog DIO Direction.
- enum _dm75xx_uio_channel { DM75xx_UIO0 = 0, DM75xx_UIO1 }

  User I/O Channel.
- enum _dm75xx_uio_source { DM75xx_UIO_ADC = 0, DM75xx_UIO_DAC1, DM75xx_UIO_DAC2, DM75xx↩
  _UIO_PRG }

  User I/O Source.
- enum _dm75xx_dio_clk { DM75xx_DIO_CLK_8MHZ = 0, DM75xx_DIO_CLK_UTC1 }

  Digital I/O Clock.
- enum _dm75xx_dio_mode { DM75xx_DIO_MODE_EVENT = 0, DM75xx_DIO_MODE_MATCH }

  Digital I/O IRQ Mode.
- enum _dm75xx_dio_port { DM75xx_DIO_PORT0 = 0, DM75xx_DIO_PORT1 }

  Digital I/O Port.
- enum _dm75xx_ext_polarity { DM75xx_EXT_POLARITY_POS = 0, DM75xx_EXT_POLARITY_NEG }

  Polarity.
- enum _dm75xx_sbus { DM75xx_SBUS0 = 0, DM75xx_SBUS1, DM75xx_SBUS2 }

  SyncBus Enumerations.
- enum _dm75xx_sbus_src {
  DM75xx_SBUS_SRC_SOFT_ADC = 0, DM75xx_SBUS_SRC_PCLK = 1, DM75xx_SBUS_SRC_PCLK_S↩
  TART = 1, DM75xx_SBUS_SRC_BCLK = 2,
  DM75xx_SBUS_SRC_PCLK_STOP = 2, DM75xx_SBUS_SRC_DIG_IT = 3, DM75xx_SBUS_SRC_DAC1 =
  3, DM75xx_SBUS_SRC_ETRIG = 4,
  DM75xx_SBUS_SRC_DAC2 = 4, DM75xx_SBUS_SRC_DAC_UPDATE = 5, DM75xx_SBUS_SRC_EPCLK
  = 5, DM75xx_SBUS_SRC_DAC_CLK = 6,
  DM75xx_SBUS_SRC_ETRIG2 = 6, DM75xx_SBUS_SRC_UTC2 = 7 }

  SyncBus Source Select.

- enum _dm75xx_hsdin_signal {
  DM75xx_HSDIN_SIGNAL_SOFTWARE = 0, DM75xx_HSDIN_SIGNAL_ADC, DM75xx_HSDIN_SIGNAL_↩ UTC0, DM75xx_HSDIN_SIGNAL_UTC1,
  DM75xx_HSDIN_SIGNAL_UTC2, DM75xx_HSDIN_SIGNAL_EPCLK, DM75xx_HSDIN_SIGNAL_ETRIG }

  HSDIN Sampling Signal.
- enum _dm75xx_dac_freq { DM75xx_DAC_FREQ_8_MHZ = 0, DM75xx_DAC_FREQ_20_MHZ }

  DAC primary clock source.
- enum _dm75xx_dac_clk_stop {
  DM75xx_DAC_CLK_STOP_SOFTWARE_PACER = 0, DM75xx_DAC_CLK_STOP_ETRIG, DM75xx_DA↩ C_CLK_STOP_DIG_IT, DM75xx_DAC_CLK_STOP_UTC2,
  DM75xx_DAC_CLK_STOP_SBUS0, DM75xx_DAC_CLK_STOP_SBUS1, DM75xx_DAC_CLK_STOP_SB↩ US2, DM75xx_DAC_CLK_STOP_SOFTWARE,
  DM75xx_DAC_CLK_STOP_DAC1_UCNT, DM75xx_DAC_CLK_STOP_DAC2_UCNT }

  DAC clock stop source.
- enum _dm75xx_dac_clk_start {
  DM75xx_DAC_CLK_START_SOFTWARE_PACER = 0, DM75xx_DAC_CLK_START_ETRIG, DM75xx_D↩ AC_CLK_START_DIG_IT, DM75xx_DAC_CLK_START_UTC2,
  DM75xx_DAC_CLK_START_SBUS0, DM75xx_DAC_CLK_START_SBUS1, DM75xx_DAC_CLK_START↩ _SBUS2, DM75xx_DAC_CLK_START_SOFTWARE }

  DAC clock start source.
- enum _dm75xx_dac_mode { DM75xx_DAC_MODE_NOT_CYCLE = 0, DM75xx_DAC_MODE_CYCLE }

  DAC Cycle Mode.
- enum _dm75xx_dac_update_src {
  DM75xx_DAC_UPDATE_SOFTWARE = 0, DM75xx_DAC_UPDATE_CGT, DM75xx_DAC_UPDATE_CL↩ OCK, DM75xx_DAC_UPDATE_EPCLK,
  DM75xx_DAC_UPDATE_SBUS0, DM75xx_DAC_UPDATE_SBUS1, DM75xx_DAC_UPDATE_SBUS2 }

  DAC Update Source.
- enum _dm75xx_dac_range { DM75xx_DAC_RANGE_UNIPOLAR_5 = 0, DM75xx_DAC_RANGE_UNIPO↩ LAR_10, DM75xx_DAC_RANGE_BIPOLAR_5, DM75xx_DAC_RANGE_BIPOLAR_10 }

  DAC Output Range.
- enum _dm75xx_dac_channel { DM75xx_DAC1 = 1, DM75xx_DAC2 }

  DAC channels.
- enum _dm75xx_dac_clk_mode { DM75xx_DAC_CLK_FREE_RUN = 0, DM75xx_DAC_CLK_START_STOP }

  DAC Clock Mode.
- enum _dm75xx_adc_scnt_src { DM75xx_ADC_SCNT_SRC_CGT = 0, DM75xx_ADC_SCNT_SRC_FIFO }

  ADC Sample Counter Source.
- enum _dm75xx_adc_conv_signal {
  DM75xx_ADC_CONV_SIGNAL_SOFTWARE = 0, DM75xx_ADC_CONV_SIGNAL_PCLK, DM75xx_ADC↩ _CONV_SIGNAL_BCLK, DM75xx_ADC_CONV_SIGNAL_DIG_IT,
  DM75xx_ADC_CONV_SIGNAL_DAC1_MRKR1, DM75xx_ADC_CONV_SIGNAL_DAC2_MRKR1, D↩ M75xx_ADC_CONV_SIGNAL_SBUS0, DM75xx_ADC_CONV_SIGNAL_SBUS1,
  DM75xx_ADC_CONV_SIGNAL_SBUS2 }

  ADC Conversion Signal Select.
- enum _dm75xx_bclk_freq { DM75xx_BCLK_FREQ_8_MHZ = 0, DM75xx_BCLK_FREQ_20_MHZ }

  Burst Clock primary frequency.
- enum _dm75xx_bclk_start {
  DM75xx_BCLK_START_SOFTWARE = 0, DM75xx_BCLK_START_PACER, DM75xx_BCLK_START_ET↩ RIG, DM75xx_BCLK_START_DIG_IT,
  DM75xx_BCLK_START_SBUS0, DM75xx_BCLK_START_SBUS1, DM75xx_BCLK_START_SBUS2 }

  Burst Clock Start Trigger.
- enum _dm75xx_pclk_mode { DM75xx_PCLK_NO_REPEAT = 0, DM75xx_PCLK_REPEAT }

  Pacer Clock Trigger Mode.

- enum _dm75xx_pclk_stop {
  DM75xx_PCLK_STOP_SOFTWARE = 0, DM75xx_PCLK_STOP_ETRIG, DM75xx_PCLK_STOP_DIGITA↩
  L_IT, DM75xx_PCLK_STOP_ACNT,
  DM75xx_PCLK_STOP_UTC2, DM75xx_PCLK_STOP_SBUS0, DM75xx_PCLK_STOP_SBUS1, DM75xx↩
  _PCLK_STOP_SBUS2,
  DM75xx_PCLK_STOP_ASOFTWARE, DM75xx_PCLK_STOP_AETRIG, DM75xx_PCLK_STOP_ADIGIT↩
  AL_IT, DM75xx_PCLK_STOP_RES,
  DM75xx_PCLK_STOP_AUTC2, DM75xx_PCLK_STOP_ASBUS0, DM75xx_PCLK_STOP_ASBUS1, D↩
  M75xx_PCLK_STOP_ASBUS2 }

  Pacer Clock Stop.
- enum _dm75xx_pclk_start {
  DM75xx_PCLK_START_SOFTWARE = 0, DM75xx_PCLK_START_ETRIG, DM75xx_PCLK_START_DIG↩
  ITAL_IT, DM75xx_PCLK_START_UTC2,
  DM75xx_PCLK_START_SBUS0, DM75xx_PCLK_START_SBUS1, DM75xx_PCLK_START_SBUS2, D↩
  M75xx_PCLK_START_RES,
  DM75xx_PCLK_START_DSOFTWARE, DM75xx_PCLK_START_DETRIG, DM75xx_PCLK_START_DDI↩
  GITAL_IT, DM75xx_PCLK_START_DUTC2,
  DM75xx_PCLK_START_DSBUS0, DM75xx_PCLK_START_DSBUS1, DM75xx_PCLK_START_DSBUS2,
  DM75xx_PCLK_START_ETRIG_GATE }

  Pacer Clock Start.
- enum _dm75xx_pclk_select { DM75xx_PCLK_EXTERNAL = 0, DM75xx_PCLK_INTERNAL }

  Pacer Clock Select.
- enum _dm75xx_pclk_freq { DM75xx_PCLK_FREQ_8_MHZ = 0, DM75xx_PCLK_FREQ_20_MHZ }

  Pacer Clock Frequency Select.
- enum _dm75xx_utc_timer { DM75xx_UTC_0 = 0, DM75xx_UTC_1, DM75xx_UTC_2 }

  8254 timers/counters
- enum _dm75xx_utc_clk {
  DM75xx_CUTC_8_MHZ = 0, DM75xx_CUTC_EXT_TC_CLOCK_1 = 1, DM75xx_CUTC_EXT_TC_CLOC↩
  K_2 = 2, DM75xx_CUTC_EXT_PCLK = 3,
  DM75xx_CUTC_UTC_0_OUT = 4, DM75xx_CUTC_UTC_1_OUT = 4, DM75xx_CUTC_HSDIN_SIGNAL = 5
  }

  8254 timer/counter clock selectors
- enum _dm75xx_utc_gate {
  DM75xx_GUTC_NOT_GATED = 0, DM75xx_GUTC_GATED = 1, DM75xx_GUTC_EXT_TC_CLK_1 = 2, D↩
  M75xx_GUTC_EXT_TC_CLK_2 = 3,
  DM75xx_GUTC_UTC_0_OUT = 4, DM75xx_GUTC_UTC_1_OUT = 4 }

  8254 timer/counter gate selectors
- enum _dm75xx_utc_mode {
  DM75xx_UTC_MODE_EVENT_COUNTER = 0, DM75xx_UTC_MODE_PROG_ONE_SHOT, DM75xx_UT↩
  C_MODE_RATE_GENERATOR, DM75xx_UTC_MODE_SQUARE_WAVE,
  DM75xx_UTC_MODE_SOFTWARE_STROBE, DM75xx_UTC_MODE_HARDWARE_STROBE }

  8254 timer/counter waveform mode selectors
- enum _dm75xx_fifo_status {
  DM75xx_FIFO_DAC1_NOT_EMPTY = 0x0001, DM75xx_FIFO_DAC1_HALF_EMPTY = 0x0002, DM75xx↩
  _FIFO_DAC1_NOT_FULL = 0x0004, DM75xx_FIFO_DAC2_NOT_EMPTY = 0x0010,
  DM75xx_FIFO_DAC2_HALF_EMPTY = 0x0020, DM75xx_FIFO_DAC2_NOT_FULL = 0x0040, DM75xx_F↩
  IFO_ADC_NOT_EMPTY = 0x0100, DM75xx_FIFO_ADC_HALF_EMPTY = 0x0200,
  DM75xx_FIFO_ADC_NOT_FULL = 0x0400, DM75xx_FIFO_HSDIN_NOT_EMPTY = 0x1000, DM75xx_FI↩
  FO_HSDIN_HALF_EMPTY = 0x2000, DM75xx_FIFO_HSDIN_NOT_FULL = 0x4000 }

  FIFO status.

6.32.1 Detailed Description

Type definitions used both in kernel and user space.

Id

dm75xx_types.h 79763 2014-06-13 21:01:02Z rgroner

Definition in file dm75xx_types.h.

Index