Linux Software (DM7820/DM9820)

Generated by Doxygen 1.8.1

Wed Jul 8 2015 09:17:58

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 DM7820 driver header file

Modules

- DM7820 driver enumerations
- DM7820 driver macros
- DM7820 driver constants
- DM7820 driver structures
- DM7820 driver forward declarations
- DM7820 driver functions

4.1.1 Detailed Description

4.2 DM7820 driver enumerations

Typedefs

- typedef enum dm7820_pci_region_access_dir dm7820_pci_region_access_dir_t

  Standard PCI region access direction type.

Enumerations

- enum dm7820_pci_region_access_dir { DM7820_PCI_REGION_ACCESS_READ = 0, DM7820_PCI_REGION_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 dm7820_pci_region_access_dir

Direction of access to standard PCI region.

Enumerator:

DM7820_PCI_REGION_ACCESS_READ Read from the region

DM7820_PCI_REGION_ACCESS_WRITE Write to the region

Definition at line 60 of file dm7820_driver.h.

4.3 DM7820 driver macros

Macros

- #define DM7820_DEVICE_NAME_LENGTH 22

  Maximum number of characters in device's name.
- #define DM7820_PCI_DEVICE_ID 0x7820

  DM7820 PCI device ID.
- #define RTD_PCI_VENDOR_ID 0x1435

  RTD Embedded Technologies PCI vendor ID.
- #define DM7820_PCI_REGIONS PCI_ROM_RESOURCE

  Number of standard PCI regions.
- #define DM7820_FIFO_CHANNELS 2

  Number of FIFO channels per device.
- #define DM7820_MAX_DMA_BUFFER_SIZE 0x40000

  Maximum size in bytes of any DMA buffer.
- #define DM7820_MAX_DMA_BUFFER_COUNT 16

  Maximum number of DMA buffers per DMA/FIFO channel.
- #define DM7820_INT_QUEUE_SIZE 0x10

  Maximum number of entries in the interrupt status queue;.

4.3.1 Detailed Description

DM7820_Driver_Enumerations

4.3.2 Macro Definition Documentation

4.3.2.1 #define DM7820_MAX_DMA_BUFFER_COUNT 16

Maximum number of DMA buffers per DMA/FIFO channel.

Note

Be aware that the probability of DMA buffer allocation failure increases as the number of buffers per DMA/FIFO channel increases.
If this default value does not suit your needs, you can change it and then recompile the driver.

Definition at line 159 of file dm7820_driver.h.

4.3.2.2 #define DM7820_MAX_DMA_BUFFER_SIZE 0x40000

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.

Definition at line 144 of file dm7820_driver.h.

4.4 DM7820 driver constants

4.4.1 Detailed Description

DM7820_Driver_Macros

4.5 DM7820 driver structures

Data Structures

- struct dm7820_pci_region

  DM7820 PCI region descriptor. This structure holds information about one of a device's PCI memory regions.
- struct dm7820_dma_descriptor_t

  DM7820 DMA buffer descriptor. This structure holds allocation information for a single DMA buffer.
- struct dm7820_dma_list_item_t

  DM7820 DMA buffer list item.
- struct dm7820_device_descriptor

  DM7820 device descriptor. This structure holds information about a device needed by the kernel.
- struct dm7820_interrupt_status_source

  Interrupt source information for a single Interrupt Status Register bit.

Typedefs

- typedef struct dm7820_pci_region dm7820_pci_region_t

  DM7820 PCI region descriptor type.
- typedef struct dm7820_device_descriptor dm7820_device_descriptor_t

  DM7820 device descriptor type.
- typedef struct dm7820_interrupt_status_source dm7820_interrupt_status_source_t

  Interrupt Status Register bit interrupt source information type.

4.5.1 Detailed Description

DM7820_Driver_Constants

4.6 DM7820 driver forward declarations

Variables

- static struct file_operations dm7820_file_ops

  File operations supported by driver.

4.6.1 Detailed Description

DM7820_Driver_Structures

4.7 DM7820 driver functions

Functions

- static void dm7820_access_pci_region (const dm7820_device_descriptor_t *dm7820_device, dm7820_pci_access_request_t *pci_request, dm7820_pci_region_access_dir_t direction)

  Read from or write to one of the standard PCI regions.
- static int dm7820_allocate_irq (dm7820_device_descriptor_t *dm7820_device, const struct pci_dev *pci_device)

  Allocate an interrupt line for a DM7820 device.
- static void dm7820_disable_all_interrupts (const dm7820_device_descriptor_t *dm7820_device)

  Disable all non-PLX interrupts for the specified DM7820 device.
- static void dm7820_enable_plx_interrupts (const dm7820_device_descriptor_t *dm7820_device, uint8_t enable)

  Disable or enable PLX interrupts for the specified DM7820 device.
- static void dm7820_free_dma_mappings (dm7820_device_descriptor_t *dm7820_device, dm7820_fifo_queue fifo)

  Free all coherent/consistent DMA mappings for the given DMA/FIFO channel on the specified DM7820 device.
- static int dm7820_get_interrupt_status (dm7820_device_descriptor_t *dm7820_device, unsigned long ioctl_param)

  Get interrupt status for the specified DM7820 device, optionally waiting for an interrupt to occur before returning the status.
- static void dm7820_get_pci_master_status (dm7820_device_descriptor_t *dm7820_device, uint8_t *pci_master)

  Determine whether or not a device is PCI master capable.
- static void dm7820_initialize_device_descriptor (dm7820_device_descriptor_t *dm7820_device)

  Initialize the device descriptor for the specified DM7820 device.
- static int dm7820_initialize_dma (dm7820_device_descriptor_t *dm7820_device, dm7820_ioctl_argument_t *ioctl_argument)

  Initialize DMA for the specified DM7820 device.
- dma_addr_t dm7820_get_buffer_phy_addr (dm7820_device_descriptor_t *dm7820_device, dm7820_fifo_queue fifo)

  Returns the physical address of the next available DMA buffer.
- static int dm7820_dma_read (dm7820_device_descriptor_t *dm7820_device, dm7820_ioctl_argument_t *ioctl_argument)

  Read from DMA buffer to copy to user.
- static int dm7820_dma_write (dm7820_device_descriptor_t *dm7820_device, dm7820_ioctl_argument_t *ioctl_argument)

  Write to DMA buffer.
- static int dm7820_dma_stop (dm7820_device_descriptor_t *dm7820_device, dm7820_fifo_queue fifo)

  Stops a DMA transfer on the specified channel if one is currently running.
- static int dm7820_dma_pause (dm7820_device_descriptor_t *dm7820_device, dm7820_fifo_queue fifo)

  Pause DMA – Used by the STOP_DMA function.
- static int dm7820_dma_check_xfer (dm7820_device_descriptor_t *dm7820_device, dm7820_fifo_queue fifo)

  Checks if there is a transfer currently underway for the specified DMA/FIFO channel.
- static void dm7820_initialize_hardware (const dm7820_device_descriptor_t *dm7820_device)

  Initialize the specified DM7820 device.
- static void dm7820_int_queue_add (dm7820_device_descriptor_t *dm7820_device, dm7820_interrupt_source source)

  Add an interrupt source to the queue.
- static dm7820_interrupt_info dm7820_dequeue_interrupt (dm7820_device_descriptor_t *dm7820_device)

Remove an interrupt from the front of the queue.

- static irqreturn_t dm7820_interrupt_handler (int irq_number, void *device_id)

  DM7820 device interrupt handler.
- static long dm7820_ioctl (struct file *file, unsigned int request_code, unsigned long ioctl_param)

  Process ioctl(2) system calls directed toward a DM7820 device file.
- int dm7820_load (void)

  Perform all actions necessary to initialize the DM7820 driver and devices.
- static int dm7820_modify_pci_region (dm7820_device_descriptor_t *dm7820_device, 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 dm7820_open (struct inode *inode, struct file *file)

  Prepare a DM7820 device file to be opened and used.
- static unsigned int dm7820_poll (struct file *file, struct poll_table_struct *poll_table)

  Determine whether or not a DM7820 device is readable. This function supports the poll(2) and select(2) system calls.
- static int dm7820_probe_device_blocks (dm7820_device_descriptor_t *dm7820_device)

  Probe and set up all functional blocks on a device.
- static int dm7820_probe_devices (uint32_t *device_count, dm7820_device_descriptor_t **device_descriptors)

  Probe and set up all DM7820 devices.
- static int dm7820_process_pci_regions (dm7820_device_descriptor_t *dm7820_device, 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 dm7820_read_pci_region (dm7820_device_descriptor_t *dm7820_device, unsigned long ioctl_param)

  Read an unsigned value from one of a device's PCI regions.
- static int dm7820_register_char_device (int *major)

  Register the DM7820 character device and request dynamic allocation of a character device major number.
- static int dm7820_release (struct inode *inode, struct file *file)

  Do all processing necessary after the last reference to a DM7820 device file is released elsewhere in the kernel.
- static void dm7820_release_resources (void)

  Release any resources allocated by the driver.
- static int dm7820_service_dma_function (dm7820_device_descriptor_t *dm7820_device, unsigned long ioctl_param)

  Process user space DMA function request.
- void dm7820_unload (void)

  Perform all actions necessary to deinitialize the DM7820 driver and devices.
- static int dm7820_unregister_char_device (void)

  Unregister the DM7820 character device and free the character device major number.
- static int dm7820_validate_device (const dm7820_device_descriptor_t *dm7820_device)

  Given what is assumed to be the address of a DM7820 device descriptor, make sure it corresponds to a valid DM7820 device descriptor.
- static int dm7820_validate_pci_access (const dm7820_device_descriptor_t *dm7820_device, const dm7820_pci_access_request_t *pci_request)

  Validate a user-space access to one of a device's PCI regions.
- static int dm7820_write_pci_region (dm7820_device_descriptor_t *dm7820_device, unsigned long ioctl_param)

  Write an unsigned value to one of a device's PCI regions.

4.7.1 Detailed Description

DM7820_Driver_Forward_Declarations

4.7.2 Function Documentation

4.7.2.1 static void dm7820_access_pci_region (const dm7820_device_descriptor_t * *dm7820_device,* dm7820_pci_access_request_t * *pci_request,* dm7820_pci_region_access_dir_t *direction*) [static]

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

Parameters

dm7820_device	Address of device's DM7820 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.

4.7.2.2 static int dm7820_allocate_irq (dm7820_device_descriptor_t * *dm7820_device,* const struct pci_dev * *pci_device*) [static]

Allocate an interrupt line for a DM7820 device.

Parameters

dm7820_device	Address of device's DM7820 device descriptor.
pci_device	Address of kernel's PCI device structure for the current DM7820 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.

4.7.2.3 static dm7820_interrupt_info dm7820_dequeue_interrupt (dm7820_device_descriptor_t * *dm7820_device*) [static]

Remove an interrupt from the front of the queue.

Parameters

dm7820_device	Address of device's DM7820 device descriptor.

Return values

dm7820_interrupt_info	Information about the interrupt and the queue

Note

None

4.7.2.4 static void dm7820_disable_all_interrupts (const dm7820_device_descriptor_t * *dm7820_device*) [static]

Disable all non-PLX interrupts for the specified DM7820 device.

Parameters

dm7820_device	Address of device's DM7820 device descriptor.

4.7.2.5 static int dm7820_dma_check_xfer (dm7820_device_descriptor_t * *dm7820_device,* dm7820_fifo_queue *fifo*) [static]

Checks if there is a transfer currently underway for the specified DMA/FIFO channel.

Parameters

dm7820_device	Address of device's DM7820 device descriptor.
fifo	The specified DMA/FIFO channel.

Return values

0	No transfer currently underway. This could also denote that a transfer has finished.
1	There is a transfer currently underway..

4.7.2.6 static int dm7820_dma_pause (dm7820_device_descriptor_t * *dm7820_device,* dm7820_fifo_queue *fifo*) [static]

Pause DMA – Used by the STOP_DMA function.

Parameters

dm7820_device	Address of device's DM7820 device descriptor.
fifo	The specified DMA/FIFO channel

Return values

0	Success.

4.7.2.7 static int dm7820_dma_read (dm7820_device_descriptor_t * *dm7820_device,* dm7820_ioctl_argument_t * *ioctl_argument*) [static]

Read from DMA buffer to copy to user.

Parameters

dm7820_device	Address of device's DM7820 device descriptor.
ioctl_argument	Address of kernel's ioctl() request structure.

Return values

0	Success.

Note

```
This function uses a busy wait while waiting for the expected number of DMA
transactions to take place.
```

4.7.2.8 static int dm7820_dma_stop (dm7820_device_descriptor_t * *dm7820_device*, dm7820_fifo_queue *fifo*) [static]

Stops a DMA transfer on the specified channel if one is currently running.

Parameters

dm7820_device	Address of device's DM7820 device descriptor.
fifo	The specific FIFO/DMA channel on which to abort the transfer.

Return values

0	Success.

4.7.2.9 static int dm7820_dma_write (dm7820_device_descriptor_t * *dm7820_device*, dm7820_ioctl_argument_t * *ioctl_argument*) [static]

Write to DMA buffer.

Parameters

dm7820_device	Address of device's DM7820 device descriptor.
ioctl_argument	Address of kernel's ioctl() request structure.

Return values

0	Success.

4.7.2.10 static void dm7820_enable_plx_interrupts (const dm7820_device_descriptor_t * *dm7820_device*, uint8_t *enable*) [static]

Disable or enable PLX interrupts for the specified DM7820 device.

Parameters

dm7820_device	Address of device's DM7820 device descriptor.
enable	Flag indicating whether or not PLX interrupts should be enabled. A value of zero means disable PLX interrupts. Any other value means enable PLX interrupts.

4.7.2.11 static void dm7820_free_dma_mappings (dm7820_device_descriptor_t * *dm7820_device*, dm7820_fifo_queue *fifo*) [static]

Free all coherent/consistent DMA mappings for the given DMA/FIFO channel on the specified DM7820 device.

Parameters

dm7820_device	Address of device's DM7820 device descriptor.
fifo	The DMA/FIFO channel to free mappings for.

Note

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

4.7.2.12 dma_addr_t dm7820_get_buffer_phy_addr (dm7820_device_descriptor_t * *dm7820_device*, dm7820_fifo_queue *fifo*)

Returns the physical address of the next available DMA buffer.

Parameters

dm7820_device	Address of the device's DM7820 device descriptor.
fifo	FIFO to get physical address of.

Return values

0	Success.

4.7.2.13 static int dm7820_get_interrupt_status (dm7820_device_descriptor_t * *dm7820_device*, unsigned long *ioctl_param*) [static]

Get interrupt status for the specified DM7820 device, optionally waiting for an interrupt to occur before returning the status.

Parameters

dm7820_device	Address of device's DM7820 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.

4.7.2.14 static void dm7820_get_pci_master_status (dm7820_device_descriptor_t * *dm7820_device*, uint8_t * *pci_master*) [static]

Determine whether or not a device is PCI master capable.

Parameters

dm7820_device	Address of device's DM7820 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.

4.7.2.15 static void dm7820_initialize_device_descriptor (dm7820_device_descriptor_t * *dm7820_device*) [static]

Initialize the device descriptor for the specified DM7820 device.

Parameters

dm7820_device	Address of device's DM7820 device descriptor.

4.7.2.16 static int dm7820_initialize_dma (dm7820_device_descriptor_t * *dm7820_device*, dm7820_ioctl_argument_t * *ioctl_argument*) [static]

Initialize DMA for the specified DM7820 device.

Parameters

dm7820_device	Address of device's DM7820 device descriptor.
ioctl_argument	Address of kernel's ioctl() request structure.

Return values

0	Success.
<	0

Failure.

The following values may be returned:

- -EAGAIN DMA has already been initialized.
- -EINVAL The number of DMA buffers to allocate is zero.
- -EINVAL The number of DMA buffers to allocate exceeds the default value DM7820_MAX_DMA_BUFFER_COUNT.
- -EINVAL The DMA buffer size is zero.
- -EINVAL The DMA buffer size is not evenly divisible by two.
- -EINVAL The DMA buffer size exceeds the default value DM7820_MAX_DMA_BUFFER_SIZE.
- -ENOMEM Kernel memory allocation failed.
- -EOPNOTSUPP The device is not PCI master capable.

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.
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.

4.7.2.17 static void dm7820_initialize_hardware (const dm7820_device_descriptor_t * *dm7820_device*) [static]

Initialize the specified DM7820 device.

Parameters

dm7820_device	Address of device's DM7820 device descriptor.

Note

When initializing a device, the driver: 1) resets the board, 2) disables PLX PCI interrupts, 3) disables PLX local interrupt input, 4) disables PLX DMA channel 0/1 interrupts, 5) sets up PLX DMA Channel 0/1 Mode Registers, and 6) sets up PLX DMA Channel 0/1 Local Address Registers.

4.7.2.18 static void dm7820_int_queue_add (dm7820_device_descriptor_t * *dm7820_device,* dm7820_interrupt_source *source*) [static]

Add an interrupt source to the queue.

Parameters

dm7820_device	Address of device's DM7820 device descriptor.
source	The source of the interrupt

Note

None

4.7.2.19 static irqreturn_t dm7820_interrupt_handler (int *irq_number,* void * *device_id*) [static]

DM7820 device interrupt handler.

Parameters

irq_number	Interrupt line number.
device_id	Address of device's DM7820 device descriptor. This is set on request_irq() call.

Return values

IRQ_HANDLED	Interrupt successfully processed; 2.6 kernel only.
IRQ_NONE	Interrupt could not be processed; 2.6 kernel only.

Note

This function does not return a value on 2.4 kernels.

4.7.2.20 static long dm7820_ioctl (struct file * *file*, unsigned int *request_code*, unsigned long *ioctl_param*) [static]

Process ioctl(2) system calls directed toward a DM7820 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 dm7820_validate_device(), dm7820_read_pci_region(), dm7820_write_pci_region(), and dm7820_modify_pci_region() for information on other possible values returned in this case.

4.7.2.21 int dm7820_load (void)

Perform all actions necessary to initialize the DM7820 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 dm7820_probe_devices() and dm7820_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.

4.7.2.22 static int dm7820_modify_pci_region (dm7820_device_descriptor_t * *dm7820_device*, 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

dm7820_device	Address of device's DM7820 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 dm7820_validate_pci_access() for information on other possible values returned in this case.

4.7.2.23 static int dm7820_open (struct inode * *inode*, struct file * *file*) [static]

Prepare a DM7820 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 DM7820 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.

4.7.2.24 static unsigned int dm7820_poll (struct file * *file*, struct poll_table_struct * *poll_table*) [static]

Determine whether or not a DM7820 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 DM7820 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.

4.7.2.25 static int dm7820_probe_device_blocks (dm7820_device_descriptor_t * *dm7820_device*) [static]

Probe and set up all functional blocks on a device.

Parameters

dm7820_device	Address of device's DM7820 device descriptor.

Return values

0	Success.
$<$	0

Failure.

The following values may be returned:

- `-EIO` Unknown or unexpected block ID found.
- `-EBADSLT` Internal driver error.

Note

At the present time, this function does not do much. Without additional firmware support for determining what blocks exist at driver load time and what resources the blocks use, the best that can be done is to assume a fixed layout of functional blocks. This function serves as a placeholder for when functional block information can be probed when the driver is loaded.

4.7.2.26 static int dm7820_probe_devices (uint32_t * *device_count*, dm7820_device_descriptor_t ** *device_descriptors*) [static]

Probe and set up all DM7820 devices.

Parameters

device_count	Address where DM7820 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 DM7820 devices found.
- -ENOMEM Device descriptor memory allocation failed.

Please see the descriptions of dm7820_process_pci_regions(), dm7820_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 DM7820 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.

4.7.2.27 static int dm7820_process_pci_regions (dm7820_device_descriptor_t * *dm7820_device*, 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

dm7820_device	Address of device's DM7820 device descriptor.
pci_device	Address of kernel's PCI device structure for the current DM7820 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.

4.7.2.28 static int dm7820_read_pci_region (dm7820_device_descriptor_t * *dm7820_device*, unsigned long *ioctl_param*) [static]

Read an unsigned value from one of a device's PCI regions.

Parameters

dm7820_device	Address of device's DM7820 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 dm7820_validate_pci_access() for information on other possible values returned in this case.

4.7.2.29 static int dm7820_register_char_device (int * *major*) [static]

Register the DM7820 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.

4.7.2.30 static int dm7820_release (struct inode * *inode*, struct file * *file*) [static]

Do all processing necessary after the last reference to a DM7820 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 dm7820_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.

4.7.2.31 static void dm7820_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.

4.7.2.32 static int dm7820_service_dma_function (dm7820_device_descriptor_t * *dm7820_device*, unsigned long *ioctl_param*) [static]

Process user space DMA function request.

Parameters

dm7820_device	Address of device's DM7820 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.
- -EINVAL DMA/FIFO channel to operate upon is not valid.
- -ENOSYS DMA function request is not valid.

Please see the descriptions of dm7820_initialize_dma(), and ???????? for information on other possible values returned in this case.

4.7.2.33 static int dm7820_unregister_char_device (void) [static]

Unregister the DM7820 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.

4.7.2.34 static int dm7820_validate_device (const dm7820_device_descriptor_t * *dm7820_device*) [static]

Given what is assumed to be the address of a DM7820 device descriptor, make sure it corresponds to a valid DM7820 device descriptor.

Parameters

dm7820_device	Address of device descriptor to be verified.

Return values

0	Success.
<	0

Failure.

The following values may be returned:

- -EBADFD dm7820_device is not a valid DM7820 device descriptor address.

4.7.2.35 static int dm7820_validate_pci_access (const dm7820_device_descriptor_t * *dm7820_device,* const dm7820_pci_access_request_t * *pci_request*) [static]

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

Parameters

dm7820_device	Address of device's DM7820 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 valid.
- -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.

4.7.2.36 static int dm7820_write_pci_region (dm7820_device_descriptor_t * *dm7820_device,* unsigned long *ioctl_param*) [static]

Write an unsigned value to one of a device's PCI regions.

Parameters

dm7820_device	Address of device's DM7820 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 dm7820_validate_pci_access() for information on other possible values returned in this case.

4.8 DM7820 global variable header file

Variables

- static dm7820_interrupt_control_t dm7820_interrupt_control [ ]

  Table of information needed to acknowledge, disable, and enable all interrupt sources.
- dm7820_minor_int_reg_layout_t dm7820_minor_int_reg_layout [ ]

  Table of information providing layout of all minor interrupt registers.

4.8.1 Detailed Description

4.9 DM7820 ioctl header file

Modules

- DM7820 ioctl enumerations
- DM7820 ioctl structures
- DM7820 ioctl macros

4.9.1 Detailed Description

4.10 DM7820 ioctl enumerations

Typedefs

- typedef enum dm7820_dma_manage_function dm7820_dma_manage_function_t

  Functions supported by driver DMA management system.

Enumerations

- enum dm7820_dma_manage_function { DM7820_DMA_FUNCTION_INITIALIZE = 0, DM7820_DMA_FUNCTION_STOP, DM7820_DMA_FUNCTION_READ, DM7820_DMA_FUNCTION_WRITE, DM7820_DMA_GET_BUFFER_ADDR }

  Functions supported by driver DMA management system.

4.10.1 Detailed Description

4.10.2 Enumeration Type Documentation

4.10.2.1 enum dm7820_dma_manage_function

Functions supported by driver DMA management system.

Enumerator:

DM7820_DMA_FUNCTION_INITIALIZE DMA initialization

DM7820_DMA_FUNCTION_STOP DMA stop

DM7820_DMA_FUNCTION_READ DMA read

DM7820_DMA_FUNCTION_WRITE DMA write

DM7820_DMA_GET_BUFFER_ADDR Get Address of DMA Buffer

Definition at line 58 of file dm7820_ioctl.h.

4.11 DM7820 ioctl structures

Data Structures

- struct dm7820_ioctl_region_readwrite

  ioctl() request structure for read from or write to PCI region
- struct dm7820_ioctl_region_modify

  ioctl() request structure for PCI region read/modify/write
- struct dm7820_ioctl_interrupt_status

  ioctl() request structure for getting interrupt status and waiting for an interrupt to occur
- struct dm7820_dma_initialize_arguments

  Arguments for DMA initialization function.
- union dm7820_dma_function_arguments

  Structure encapsulating arguments to all possible DMA functions.
- struct dm7820_ioctl_dma_function

  ioctl() request structure for performing a DMA function
- union dm7820_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 dm7820_ioctl_region_readwrite dm7820_ioctl_region_readwrite_t
- typedef struct dm7820_ioctl_region_modify dm7820_ioctl_region_modify_t

  ioctl() PCI region read/modify/write request descriptor type
- typedef struct dm7820_ioctl_interrupt_status dm7820_ioctl_interrupt_status_t

  ioctl() interrupt status request descriptor type
- typedef struct dm7820_dma_initialize_arguments dm7820_dma_initialize_arguments_t

  Arguments for DMA initialization function.
- typedef union dm7820_dma_function_arguments dm7820_dma_function_arguments_t

  Structure encapsulating arguments to all possible DMA functions.
- typedef struct dm7820_ioctl_dma_function dm7820_ioctl_dma_function_t

  ioctl() request structure for performing a DMA function
- typedef union dm7820_ioctl_argument dm7820_ioctl_argument_t

  ioctl() request descriptor type

4.11.1 Detailed Description

DM7820_Ioctl_Enumerations

4.11.2 Typedef Documentation

4.11.2.1 typedef struct dm7820_ioctl_region_readwrite dm7820_ioctl_region_readwrite_t

typedef for the PCI region access request type

Definition at line 120 of file dm7820_ioctl.h.

4.12 DM7820 ioctl macros

Macros

- #define DM7820_IOCTL_MAGIC 'D'

  Unique 8-bit value used to generate unique ioctl() request codes.
- #define DM7820_IOCTL_REQUEST_BASE 0x00

  First ioctl() request number.
- #define DM7820_IOCTL_REGION_READ

  ioctl() request code for reading from a PCI region
- #define DM7820_IOCTL_REGION_WRITE

  ioctl() request code for writing to a PCI region
- #define DM7820_IOCTL_REGION_MODIFY

  ioctl() request code for PCI region read/modify/write
- #define DM7820_IOCTL_GET_INTERRUPT_STATUS

  ioctl() request code for getting interrupt status and waiting for an interrupt to occur
- #define DM7820_IOCTL_DMA_FUNCTION

  ioctl() request code for DMA function
- #define DM7820_IOCTL_WAKEUP

  ioctl() request code for User ISR thread wake up
- #define DM7820_IOCTL_INTERRUPT_INFO

  ioctl() request code to retrieve interrupt status information

4.12.1 Detailed Description

DM7820_Ioctl_Structures

4.13 DM7820 user library header file

Modules

- DM7820 user library macros
- DM7820 user library type definitions
- DM7820 user library structures
- DM7820 user library functions

4.13.1 Detailed Description

4.14 DM7820 user library macros

Macros

- #define DM7820_Return_Status(status, string) if(status != 0) { error(EXIT_FAILURE,errno, "ERROR: %s FAILED\n",string); }
- #define DM7820_DMA_DEMAND_OFF_PCI_TO_DM7820 0x00
- #define DM7820_DMA_DEMAND_OFF_DM7820_TO_PCI 0x01
- #define DM7820_DMA_DEMAND_ON_PCI_TO_DM7820 0x02
- #define DM7820_DMA_DEMAND_ON_DM7820_TO_PCI 0x03
- #define DM7820_INCENC_DISABLE_PHASE_FILTER_TRANSITION(filter, transition) ((filter) |= (1 << (transition)))

  Configure an incremental encoder phase filter so that counter value update is disabled for the given transition.
- #define DM7820_INCENC_ENABLE_PHASE_FILTER_TRANSITION(filter, transition) ((filter) &= ~(1 << (transition)))

  Configure an incremental encoder phase filter so that counter value update is enabled for the given transition.
- #define DM7820_INCENC_RESET_PHASE_FILTER(filter) ((filter) = 0x00)

  Reset an incremental encoder phase filter.
- #define DM7820_INTERRUPT_STATUS_IS_SOURCE_PENDING(status, source) (((status) & (0x1LL << (source))) ? 0xFF : 0x00)

  Determine whether or not the specified interrupt source is pending in the interrupt status obtained via DM7820_General_Get_Interrupt_Status().

4.14.1 Detailed Description

4.14.2 Macro Definition Documentation

4.14.2.1 #define DM7820_DMA_DEMAND_OFF_DM7820_TO_PCI 0x01

Demand Mode Off, DM7820 to PCI

Definition at line 69 of file dm7820_library.h.

4.14.2.2 #define DM7820_DMA_DEMAND_OFF_PCI_TO_DM7820 0x00

Demand Mode Off, PCI to DM7820

Definition at line 63 of file dm7820_library.h.

4.14.2.3 #define DM7820_DMA_DEMAND_ON_DM7820_TO_PCI 0x03

Demand Mode On (DREQ Signal Controls DMA), PCI to DM7820

Definition at line 81 of file dm7820_library.h.

Referenced by do_dma(), and main().

4.14.2.4 #define DM7820_DMA_DEMAND_ON_PCI_TO_DM7820 0x02

Demand Mode On (DREQ Signal Controls DMA), DM7820 to PCI

Definition at line 75 of file dm7820_library.h.

Referenced by main().

4.14.2.5 #define DM7820_INCENC_DISABLE_PHASE_FILTER_TRANSITION(*filter, transition*) ((filter) |= (1 << (transition)))

Configure an incremental encoder phase filter so that counter value update is disabled for the given transition.

Parameters

filter	The phase filter to disable transition counter update in.
transition	The transition to disable counter update for.

Definition at line 99 of file dm7820_library.h.

Referenced by main().

4.14.2.6 #define DM7820_INCENC_ENABLE_PHASE_FILTER_TRANSITION(*filter, transition*) ((filter) &= ~(1 << (transition)))

Configure an incremental encoder phase filter so that counter value update is enabled for the given transition.

Parameters

filter	The phase filter to enable transition counter update in.
transition	The transition to enable counter update for.

Definition at line 118 of file dm7820_library.h.

4.14.2.7 #define DM7820_INCENC_RESET_PHASE_FILTER(*filter*) ((filter) = 0x00)

Reset an incremental encoder phase filter.

Parameters

filter	The phase filter to reset.

Note

This macro sets the phase filter to the default phase filter after a device reset, i.e. enables counter update for all transitions in the filter.

Definition at line 136 of file dm7820_library.h.

Referenced by main().

4.14.2.8 #define DM7820_INTERRUPT_STATUS_IS_SOURCE_PENDING(*status, source*) (((status) & (0x1LL << (source))) ? 0xFF : 0x00)

Determine whether or not the specified interrupt source is pending in the interrupt status obtained via DM7820_General_Get_Interrupt_Status().

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 165 of file dm7820_library.h.

4.14.2.9 #define DM7820_Return_Status(*status, string*) if(status != 0) { error(EXIT_FAILURE,errno, "ERROR: %s FAILED\n",string); }

Check library function return status

Definition at line 56 of file dm7820_library.h.

Referenced by clean_up(), disable_timers(), do_8254(), do_digital_io(), do_dma(), do_fifo(), do_incenc(), do_pwm(), ISR(), and main().

4.15 DM7820 user library type definitions

Typedefs

- typedef int DM7820_Error

  DM7820 user library error code type.
- typedef uint8_t dm7820_incenc_phase_filter

  Incremental encoder phase filter type.

4.15.1 Detailed Description

DM7820_Library_Macros

4.16 DM7820 user library structures

Data Structures

- struct DM7820_Board_Descriptor

  DM7820 board descriptor. This structure holds information about a device needed by the library.

Typedefs

- typedef struct DM7820_Board_Descriptor DM7820_Board_Descriptor

4.16.1 Detailed Description

DM7820_Library_Types

4.16.2 Typedef Documentation

4.16.2.1 typedef struct DM7820_Board_Descriptor DM7820_Board_Descriptor

DM7820 board descriptor type

Definition at line 239 of file dm7820_library.h.

4.17 DM7820 user library functions

Modules

- DM7820 user library advanced interrupt functions
- DM7820 user library FIFO functions
- DM7820 user library general functions
- DM7820 user library incremental encoder functions
- DM7820 user library pulse width modulator functions
- DM7820 user library programmable clock functions
- DM7820 user library standard I/O functions
- DM7820 user library 8254 timer/counter functions

4.17.1 Detailed Description

DM7820_Library_Structures

4.18 DM7820 user library advanced interrupt functions

Functions

- DM7820_Error DM7820_AdvInt_Get_Status (DM7820_Board_Descriptor *handle, dm7820_advint_interrupt interrupt, uint8_t *occurred)

  Determine whether or not a status condition has occurred for the given advanced interrupt.
- DM7820_Error DM7820_AdvInt_Read_Capture (DM7820_Board_Descriptor *handle, dm7820_advint_interrupt interrupt, DM7820_StdIO_Port port, uint16_t *value)

  Read the capture register value for the given advanced interrupt and standard I/O port.
- DM7820_Error DM7820_AdvInt_Set_Compare (DM7820_Board_Descriptor *handle, dm7820_advint_interrupt interrupt, DM7820_StdIO_Port port, uint16_t value)

  Load the compare register for the given advanced interrupt and standard I/O port.
- DM7820_Error DM7820_AdvInt_Set_Mask (DM7820_Board_Descriptor *handle, dm7820_advint_interrupt interrupt, DM7820_StdIO_Port port, uint16_t value)

  Load the mask register for the given advanced interrupt and standard I/O port.
- DM7820_Error DM7820_AdvInt_Set_Master (DM7820_Board_Descriptor *handle, dm7820_advint_interrupt interrupt, dm7820_advint_master_clock master)

  Select the master clock for the given advanced interrupt.
- DM7820_Error DM7820_AdvInt_Set_Mode (DM7820_Board_Descriptor *handle, dm7820_advint_interrupt interrupt, dm7820_advint_mode mode)

  Set the mode for the given advanced interrupt.

4.18.1 Detailed Description

4.18.2 Function Documentation

4.18.2.1 DM7820_Error DM7820_AdvInt_Get_Status (DM7820_Board_Descriptor * *handle*, dm7820_advint_interrupt *interrupt*, uint8_t * *occurred*)

Determine whether or not a status condition has occurred for the given advanced interrupt.

Parameters

handle	Address of device's library board descriptor.
interrupt	The advanced interrupt to get status of.
occurred	Address where occurrence flag should be stored. Zero will be stored here if no status condition has occurred for the interrupt. A non-zero value will be stored here if a status condition has occurred for the interrupt.

Return values

0	Success.
-1	Failure. errno may be set as follows: • `EINVAL` interrupt is not valid. Please see the ioctl(2) man page for information on other possible values errno may have in this case.

Warning

If you are using interrupts, the information returned from this function is unreliable because the driver's interrupt handler clears all advanced interrupt status flags during interrupt acknowledgment.

Note

This function reads the advanced interrupt status and then clears the board's advanced interrupt status flag if it is set. The hardware will not reassert the flag until the next time the condition occurs for the advanced interrupt.

Referenced by main().

4.18.2.2 DM7820_Error DM7820_AdvInt_Read_Capture (DM7820_Board_Descriptor * *handle,* dm7820_advint_interrupt *interrupt,* DM7820_StdIO_Port *port,* uint16_t * *value*)

Read the capture register value for the given advanced interrupt and standard I/O port.

Parameters

handle	Address of device's library board descriptor.
interrupt	The advanced interrupt to read capture register for.
port	The standard I/O port to read capture register for.
value	The address where capture register value should be stored.

Return values

0	Success.
-1	Failure. errno may be set as follows: • EINVAL interrupt is not valid. • EINVAL port is not valid. Please see the ioctl(2) man page for information on other possible values errno may have in this case.

Referenced by main().

4.18.2.3 DM7820_Error DM7820_AdvInt_Set_Compare (DM7820_Board_Descriptor * *handle,* dm7820_advint_interrupt *interrupt,* DM7820_StdIO_Port *port,* uint16_t *value*)

Load the compare register for the given advanced interrupt and standard I/O port.

Parameters

handle	Address of device's library board descriptor.
interrupt	The advanced interrupt to load compare register for.
port	The standard I/O port to load compare register value for.
value	The value to load into the compare register.

Return values

0	Success.
-1	Failure. errno may be set as follows: • EINVAL interrupt is not valid. • EINVAL port is not valid. Please see the ioctl(2) man page for information on other possible values errno may have in this case.

Referenced by main().

4.18.2.4 DM7820_Error DM7820_AdvInt_Set_Mask (DM7820_Board_Descriptor * *handle,* dm7820_advint_interrupt *interrupt,* DM7820_StdIO_Port *port,* uint16_t *value*)

Load the mask register for the given advanced interrupt and standard I/O port.

Parameters

handle	Address of device's library board descriptor.
interrupt	The advanced interrupt to load mask register for.
port	The standard I/O port to load mask register value for.
value	The value to load into the mask register. A zero in a bit position means that the corresponding port bit can generate an event or match interrupt. A one in a bit position means that the corresponding port bit cannot generate an event or match interrupt.

Return values

0	Success.
-1	Failure. errno may be set as follows: • EINVAL interrupt is not valid. • EINVAL port is not valid. Please see the ioctl(2) man page for information on other possible values errno may have in this case.

Warning

For advanced interrupts to work properly, any port and/or bits not being used to generate an event or match interrupt must be programmed via the mask register to be ignored.

Referenced by main().

4.18.2.5 DM7820_Error DM7820_AdvInt_Set_Master (DM7820_Board_Descriptor * *handle,* dm7820_advint_interrupt *interrupt,* dm7820_advint_master_clock *master*)

Select the master clock for the given advanced interrupt.

Parameters

handle	Address of device's library board descriptor.
interrupt	The advanced interrupt to select master clock for.
master	The master clock to select.

Return values

0	Success.
-1	Failure. errno may be set as follows: • EINVAL interrupt is not valid. • EINVAL master is not valid. • EOPNOTSUPP master is equal to DM7820_ADVINT_MASTER_RESERVED. Please see the ioctl(2) man page for information on other possible values errno may have in this case.

Note

This function yields different results depending upon what interrupt mode you are using. When using event or match mode, this function selects the sampling clock source. When using strobe mode, this function selects the strobe signal source. This behavior is determined by the hardware.

Warning

When using strobe mode, do not select the 25 MHz clock as the strobe signal source because strobe interrupts will be generated every 40 nanoseconds.

Referenced by main().

4.18.2.6 DM7820_Error DM7820_AdvInt_Set_Mode (DM7820_Board_Descriptor * *handle,* dm7820_advint_interrupt *interrupt,* dm7820_advint_mode *mode*)

Set the mode for the given advanced interrupt.

Parameters

handle	Address of device's library board descriptor.
interrupt	The advanced interrupt to set mode for.
mode	The advanced interrupt mode to set.

Return values

0	Success.
-1	Failure. errno may be set as follows: • EINVAL interrupt is not valid. • EINVAL mode is not valid. Please see the ioctl(2) man page for information on other possible values errno may have in this case.

Referenced by main().

4.19 DM7820 user library FIFO functions

Functions

- DM7820_Error DM7820_FIFO_Enable (DM7820_Board_Descriptor *handle, dm7820_fifo_queue fifo, uint8_t enable)

  Enable or disable the given FIFO.
- DM7820_Error DM7820_FIFO_Get_Status (DM7820_Board_Descriptor *handle, dm7820_fifo_queue fifo, dm7820_fifo_status_condition condition, uint8_t *occurred)

  Determine whether or not the specified status condition has occurred for the given FIFO.
- DM7820_Error DM7820_FIFO_DMA_Initialize (DM7820_Board_Descriptor *handle, dm7820_fifo_queue fifo, uint32_t buffer_count, uint32_t buffer_size)

  Set up direct memory access (DMA) for the given DMA/FIFO channel.
- DM7820_Error DM7820_FIFO_DMA_Create_Buffer (uint16_t **buf, uint32_t size)

  Creates a user space DMA buffer.
- DM7820_Error DM7820_FIFO_DMA_Free_Buffer (uint16_t **buf, uint32_t size)

  Frees a previously created user space buffer.
- DM7820_Error DM7820_FIFO_DMA_Read (DM7820_Board_Descriptor *handle, dm7820_fifo_queue fifo, void *user_buffer, uint32_t num_bufs)

  Reads the DMA buffers in the driver.
- DM7820_Error DM7820_FIFO_DMA_Write (DM7820_Board_Descriptor *handle, dm7820_fifo_queue fifo, void *user_buffer, uint32_t num_bufs)

  Copies a user buffer to DMA buffers to be sent into a FIFO.
- DM7820_Error DM7820_Stop_DMA (DM7820_Board_Descriptor *handle, dm7820_fifo_queue fifo)

  Aborts a DMA transfer on a given channel.
- DM7820_Error DM7820_FIFO_DMA_Configure (DM7820_Board_Descriptor *handle, dm7820_fifo_queue fifo, uint8_t direction, uint32_t transfer_size)

  Configure the specified DMA channel.
- DM7820_Error DM7820_FIFO_DMA_Enable (DM7820_Board_Descriptor *handle, dm7820_fifo_queue fifo, uint8_t enable, uint8_t start)

  Enable and/or Start a DMA channel.
- DM7820_Error DM7820_FIFO_Read (DM7820_Board_Descriptor *handle, dm7820_fifo_queue fifo, uint16_t *data)

  Read a single value from the given FIFO.
- DM7820_Error DM7820_FIFO_Set_DMA_Request (DM7820_Board_Descriptor *handle, dm7820_fifo_queue fifo, dm7820_fifo_dma_request source)

  Set DMA request source for the given FIFO.
- DM7820_Error DM7820_FIFO_Set_Data_Input (DM7820_Board_Descriptor *handle, dm7820_fifo_queue fifo, dm7820_fifo_data_input input)

  Set data input for the given FIFO.
- DM7820_Error DM7820_FIFO_Set_Input_Clock (DM7820_Board_Descriptor *handle, dm7820_fifo_queue fifo, dm7820_fifo_input_clock clock)

  Set input clock for the given FIFO.
- DM7820_Error DM7820_FIFO_Set_Output_Clock (DM7820_Board_Descriptor *handle, dm7820_fifo_queue fifo, dm7820_fifo_output_clock clock)

  Set output clock for the given FIFO.
- DM7820_Error DM7820_FIFO_Write (DM7820_Board_Descriptor *handle, dm7820_fifo_queue fifo, uint16_t data)

  Write a single value to the given FIFO.

4.19.1 Detailed Description

DM7820_Library_AdvInt_Functions

4.19.2 Function Documentation

4.19.2.1 DM7820_Error DM7820_FIFO_DMA_Configure (DM7820_Board_Descriptor * *handle,* dm7820_fifo_queue *fifo,* uint8_t *direction,* uint32_t *transfer_size*)

Configure the specified DMA channel.

Parameters

handle	Address of device's library board descriptor.
fifo	The specified DMA/FIFO channel to configure.
direction	The direction of the DMA transfer to/from PCI
transfer_size	The size of the DMA transfer

Return values

0	Success.
-1	Failure. errno may be set as follows: • EINVAL fifo is not valid. • EINVAL direction is not valid. • ENOTTY ioctl call is invalid. • EFAULT could get ioctl_arguments from userspace. Please see the ioctl(2) man page for information on other possible values errno may have in this case.

Referenced by do_dma(), and main().

4.19.2.2 DM7820_Error DM7820_FIFO_DMA_Create_Buffer (uint16_t ** *buf,* uint32_t *size*)

Creates a user space DMA buffer.

Parameters

buf	Pointer to the buffer to be set by malloc.
size	The size of the buffer to allocate.

Return values

0	Success
-1	Failure. errno may be set as follows: • ENOMEM Memory allocation for the buffer failed, or memory was not mapped correctly for lock/unlock, or an attempt was made to lock more memory than is allowed. • ENOSYS The implementation does not support memory locking. • EAGAIN Some or all of the memory could not be locked. • EINVAL The memory size was not a multiple of (PAGESIZE). • EPERM The calling process did not have approprate priveleges to perform memory locking.

Referenced by main().

4.19.2.3 DM7820_Error DM7820_FIFO_DMA_Enable (DM7820_Board_Descriptor * *handle,* dm7820_fifo_queue *fifo,* uint8_t *enable,* uint8_t *start*)

Enable and/or Start a DMA channel.

Parameters

handle	Address of device's library board descriptor.
fifo	The DMA channel to enable.
enable	0x00 to disable, 0xFF to enable
start	0xFF to start

Return values

0	Success.
-1	Failure. errno may be set as follows: • EINVAL fifo is not valid. Please see the ioctl(2) man page for information on other possible values errno may have in this case.

Referenced by clean_up(), and main().

4.19.2.4 DM7820_Error DM7820_FIFO_DMA_Free_Buffer (uint16_t ** *buf,* uint32_t *size*)

Frees a previously created user space buffer.

Parameters

buf	A pointer to the buffer to be released.
size	Size of the buffer to be released.

Return values

0	Success
-1	Failure. errno may be set as follows: • ENOMEM Memory allocation for the buffer failed, or memory was not mapped correctly for lock/unlock. • ENOSYS The implementation does not support memory locking/unlocking. • EAGAIN Some or all of the memory could not be unlocked. • EINVAL The memory size was not a multiple of (PAGESIZE). • EPERM The calling process did not have approprate priveleges to perform memory unlocking.

Referenced by main().

4.19.2.5 DM7820_Error DM7820_FIFO_DMA_Initialize (DM7820_Board_Descriptor * *handle,* dm7820_fifo_queue *fifo,* uint32_t *buffer_count,* uint32_t *buffer_size*)

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

Parameters

handle	Address of device's library board descriptor.
fifo	The FIFO to set up DMA for.
buffer_count	The number of DMA buffers to allocate for the DMA/FIFO channel.
buffer_size	The size of the DMA buffer to allocate for the DMA/FIFO channel.

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. • EINVAL buffer_count is zero. • EINVAL buffer_count exceeds the default value DM7820_MAX_DMA_BUFFER_COUNT. • EINVAL buffer_size is zero. • EINVAL buffer_values exceeds the default value DM7820_MAX_DMA_BUFFER_SIZE. • 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 DM7820_MAX_DMA_BUFFER_COUNT is 16. If you need to change this, edit include/dm7820_driver.h, save the changes, recompile the driver, and reload the driver.
The default value for DM7820_MAX_DMA_BUFFER_SIZE is 262,144 bytes (256 kilobytes). If you need to change this, edit include/dm7820_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 decrease the number of buffers requested or ask for smaller buffers. In extreme cases, you may need to do both.
The values you specify here are used for all DMA transfers afterward.

Referenced by do_dma(), and main().

4.19.2.6 DM7820_Error DM7820_FIFO_DMA_Read (DM7820_Board_Descriptor * *handle*, dm7820_fifo_queue *fifo*, void * *user_buffer*, uint32_t *num_bufs*)

Reads the DMA buffers in the driver.

Parameters

handle	Address of device's library board descriptor.
fifo	The FIFO to determine status of.
user_buffer	Virtual address of the buffer used to copy the DMA buffers back to userspace
num_bufs	The number of buffers to copy from the devices linked list of DMA buffers.

Return values

0	Success
-1	Failure

Referenced by main().

4.19.2.7 DM7820_Error DM7820_FIFO_DMA_Write (DM7820_Board_Descriptor * *handle*, dm7820_fifo_queue *fifo*, void * *user_buffer*, uint32_t *num_bufs*)

Copies a user buffer to DMA buffers to be sent into a FIFO.

Parameters

handle	Address of device's library board descriptor.
fifo	The FIFO to determine status of.
user_buffer	Virtual address of the userspace buffer used to fill DMA buffers
num_bufs	The number of buffers to copy to the devices linked list of DMA buffers.

Return values

0	Success
-1	Failure

Referenced by main().

4.19.2.8 DM7820_Error DM7820_FIFO_Enable (DM7820_Board_Descriptor * *handle*, dm7820_fifo_queue *fifo*, uint8_t *enable*)

Enable or disable the given FIFO.

Parameters

handle	Address of device's library board descriptor.
fifo	The FIFO to enable or disable.
enable	Flag indicating whether or not the FIFO should be enabled. A value of zero means disable the FIFO. Any other value means enable the FIFO.

Return values

0	Success.
-1	Failure. errno may be set as follows: • EINVAL fifo is not valid. Please see the ioctl(2) man page for information on other possible values errno may have in this case.

Note

The hardware clears a FIFO when it is disabled.

Referenced by clean_up(), do_fifo(), and main().

4.19.2.9 DM7820_Error DM7820_FIFO_Get_Status (DM7820_Board_Descriptor * *handle,* dm7820_fifo_queue *fifo,* dm7820_fifo_status_condition *condition,* uint8_t * *occurred*)

Determine whether or not the specified status condition has occurred for the given FIFO.

Parameters

handle	Address of device's library board descriptor.
fifo	The FIFO to determine status of.
condition	The status condition to check for.
occurred	Address where occurrence flag should be stored. Zero will be stored here if the specified condition has not occurred. A non-zero value will be stored here if the specified condition occurred.

Return values

0	Success.
-1	Failure. errno may be set as follows: • EINVAL fifo is not valid. • EINVAL condition is not valid. Please see the ioctl(2) man page for information on other possible values errno may have in this case.

Warning

If you are using interrupts, the information returned from this function is unreliable because the driver's interrupt handler clears all FIFO status flags during interrupt acknowledgment.

Note

If condition is DM7820_FIFO_STATUS_FULL or DM7820_FIFO_STATUS_EMPTY, this function clears the board's corresponding FIFO status flag before reading the FIFO status to get accurate status. If the condition is still pending after the clear, the hardware will reassert the status flag.
If condition is DM7820_FIFO_STATUS_READ_REQUEST, DM7820_FIFO_STATUS_WRITE_REQUEST, DM7820_FIFO_STATUS_OVERFLOW, or DM7820_FIFO_STATUS_UNDERFLOW, this function reads the FIFO status and then clears the board's corresponding FIFO status flag if it is set. The hardware will not reassert the flag until the next time the condition occurs.

Referenced by get_fifo_status(), and main().

4.19.2.10 DM7820_Error DM7820_FIFO_Read (DM7820_Board_Descriptor * *handle,* dm7820_fifo_queue *fifo,* uint16_t * *data*)

Read a single value from the given FIFO.

Parameters

handle	Address of device's library board descriptor.
fifo	The FIFO to read.
data	Address where value read from FIFO should be stored.

Return values

0	Success.
-1	Failure. errno may be set as follows: • EINVAL fifo is not valid. Please see the ioctl(2) man page for information on other possible values errno may have in this case.

Referenced by main().

4.19.2.11 DM7820_Error DM7820_FIFO_Set_Data_Input (DM7820_Board_Descriptor * *handle,* dm7820_fifo_queue *fifo,* dm7820_fifo_data_input *input*)

Set data input for the given FIFO.

Parameters

handle	Address of device's library board descriptor.
fifo	The FIFO to set data input for.
input	The data input to set.

Return values

0	Success.
-1	Failure. errno may be set as follows: • EINVAL fifo is not valid. • EINVAL input is not valid. • EOPNOTSUPP input is not supported by fifo. Please see the ioctl(2) man page for information on other possible values errno may have in this case.

Referenced by do_fifo(), and main().

4.19.2.12 DM7820_Error DM7820_FIFO_Set_DMA_Request (DM7820_Board_Descriptor * *handle,* dm7820_fifo_queue *fifo,* dm7820_fifo_dma_request *source*)

Set DMA request source for the given FIFO.

Parameters

handle	Address of device's library board descriptor.
fifo	The FIFO to set DMA request source for.
source	The DMA request source to set.

Return values

0	Success.
-1	Failure. errno may be set as follows: • EINVAL fifo is not valid. • EINVAL source is not valid. Please see the ioctl(2) man page for information on other possible values errno may have in this case.

Referenced by do_fifo(), and main().

4.19.2.13 DM7820_Error DM7820_FIFO_Set_Input_Clock (DM7820_Board_Descriptor * *handle,* dm7820_fifo_queue *fifo,* dm7820_fifo_input_clock *clock*)

Set input clock for the given FIFO.

Parameters

handle	Address of device's library board descriptor.
fifo	The FIFO to set input clock for.
clock	The input clock to set.

Return values

0	Success.
-1	Failure. errno may be set as follows: • EINVAL fifo is not valid. • EINVAL clock is not valid. • EOPNOTSUPP clock is equal to DM7820_FIFO_INPUT_CLOCK_RESERVED_1, DM7820_FIFO_INPUT_CLOCK_RESERVED_2, DM7820_FIFO_INPUT_CLOCK_RESERVED_3, or DM7820_FIFO_INPUT_CLOCK_RESERVED_4. Please see the ioctl(2) man page for information on other possible values errno may have in this case.

Referenced by do_fifo(), and main().

4.19.2.14 DM7820_Error DM7820_FIFO_Set_Output_Clock (DM7820_Board_Descriptor * *handle,* dm7820_fifo_queue *fifo,* dm7820_fifo_output_clock *clock*)

Set output clock for the given FIFO.

Parameters

handle	Address of device's library board descriptor.
fifo	The FIFO to set output clock for.
clock	The output clock to set.

Return values

0	Success.
-1	Failure. errno may be set as follows: • EINVAL fifo is not valid. • EINVAL clock is not valid. • EOPNOTSUPP clock is equal to DM7820_FIFO_OUTPUT_CLOCK_RESERVED_1, DM7820_FIFO_OUTPUT_CLOCK_RESERVED_2, DM7820_FIFO_OUTPUT_CLOCK_RESERVED_3, or DM7820_FIFO_OUTPUT_CLOCK_RESERVED_4. Please see the ioctl(2) man page for information on other possible values errno may have in this case.

Referenced by do_fifo(), and main().

4.19.2.15 DM7820_Error DM7820_FIFO_Write (DM7820_Board_Descriptor * *handle,* dm7820_fifo_queue *fifo,* uint16_t *data*)

Write a single value to the given FIFO.

Parameters

handle	Address of device's library board descriptor.
fifo	The FIFO to write.
data	Data to write to FIFO.

Return values

0	Success.
-1	Failure. errno may be set as follows: • EINVAL fifo is not valid. Please see the ioctl(2) man page for information on other possible values errno may have in this case.

Referenced by main().

4.19.2.16 DM7820_Error DM7820_Stop_DMA (DM7820_Board_Descriptor * *handle,* dm7820_fifo_queue *fifo*)

Aborts a DMA transfer on a given channel.

Parameters

handle	Address of device's library board descriptor.
fifo	The specified DMA/FIFO channel to configure.

Return values

0	Success
-1	Failure

4.20 DM7820 user library general functions

Functions

- DM7820_Error DM7820_General_Close_Board (DM7820_Board_Descriptor *handle)

  Close a DM7820 device file.
- DM7820_Error DM7820_General_Enable_Interrupt (DM7820_Board_Descriptor *handle, dm7820_interrupt_source source, uint8_t enable)

  Enable or disable the given interrupt source.
- DM7820_Error DM7820_General_Get_Interrupt_Status (DM7820_Board_Descriptor *handle, dm7820_interrupt_info *interrupt_info, uint8_t wait_for_interrupt)

  Get a device's interrupt status, optionally waiting for an interrupt to occur.
- DM7820_Error DM7820_General_Open_Board (uint8_t dev_num, DM7820_Board_Descriptor **handle)

  Open a DM7820 device file.
- DM7820_Error DM7820_General_Get_Version_Info (DM7820_Board_Descriptor *handle, uint8_t *fpga_type_id, uint8_t *fpga_version, uint16_t *svn_version)

  Read a device's FPGA and source code revision control versions.
- DM7820_Error DM7820_General_Is_PCI_Master (DM7820_Board_Descriptor *handle, uint8_t *pci_master)

  Determine whether or not a device is PCI master capable.
- DM7820_Error DM7820_General_Reset (DM7820_Board_Descriptor *handle)

  Reset a DM7820 device.
- DM7820_Error DM7820_General_RemoveISR (DM7820_Board_Descriptor *handle)

  Uninstall userspace ISR.
- DM7820_Error DM7820_General_StartThread (int(*fnct)(void *), void *data)

  Creates thread to watch for interrupts and call userspace ISR.
- void * DM7820_General_WaitForInterrupt (void *ptr)

  Waits for DMA Done interrupts.
- DM7820_Error DM7820_General_SetISRPriority (DM7820_Board_Descriptor *handle, int priority)

  Changes the Priority for the ISR thread.

4.20.1 Detailed Description

DM7820_Library_FIFO_Functions

4.20.2 Function Documentation

4.20.2.1 DM7820_Error DM7820_General_Close_Board (DM7820_Board_Descriptor * *handle*)

Close a DM7820 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 clean_up(), and main().

4.20.2.2 DM7820_Error DM7820_General_Enable_Interrupt (DM7820_Board_Descriptor * *handle,* dm7820_interrupt_source *source,* uint8_t *enable*)

Enable or disable the given interrupt source.

Parameters

handle	Address of device's library board descriptor.
source	The interrupt source to change state of.
enable	Flag indicating whether or not the interrupt source should be enabled. A value of zero means disable the interrupt source. Any other value means enable the interrupt source.

Return values

0	Success.
-1	Failure. errno may be set as follows: • `EINVAL` source is not valid. • `EOPNOTSUPP` source is valid but cannot be modified from user space. Please see the ioctl(2) man page for information on other possible values errno may have in this case.

Note

Some interrupts are controlled by an enable register secondary to the Interrupt Enable Register. When disabling such an interrupt, if all interrupts in the second register are disabled then this function will also disable the interrupt in the Interrupt Enable Register. For example, suppose only the empty interrupt is enabled for FIFO 0. In this case, the FIFO 0 interrupt would be enabled in the Interrupt Enable Register and the empty interrupt would be enabled in the FIFO 0 Interrupt Register. When the FIFO 0 empty interrupt is disabled, there are no more FIFO 0 interrupts enabled so this function would also disable FIFO 0 interrupts.
When enabling an interrupt source, this function clears all applicable interrupt status flags.
The driver's interrupt handler disables each FIFO interrupt after it occurs to prevent possible flooding by these interrupts. If you are using a FIFO interrupt, it must be reenabled before it can be utilized again.

Referenced by main().

4.20.2.3 DM7820_Error DM7820_General_Get_Interrupt_Status (DM7820_Board_Descriptor * *handle,* dm7820_interrupt_info * *interrupt_info,* uint8_t *wait_for_interrupt*)

Get a device's interrupt status, optionally waiting for an interrupt to occur.

Parameters

handle	Address of device's library board descriptor.
interrupt_info	Address where interrupt information should be stored.
wait_for_interrupt	Flag indicating whether or not to wait for an interrupt to occur. A value of zero means do not wait for an interrupt and return whatever interrupt status is available. Any other value means wait for an interrupt to occur and then return its status.

Return values

0	Success.
-1	Failure. errno may be set as follows: • `EINTR` The process was waiting for an interrupt but received a signal before an interrupt occurred. This is not a fatal error but rather means the wait should be retried. Please see the ioctl(2) man page for information on other possible values errno may have in this case.

Note

If this function is being used to wait for interrupts, signals can wake up the process before an interrupt occurs. If a signal is delivered to the process during a wait, the application is responsible for dealing with the premature awakening in a reasonable manner.
When this function is being used to wait for interrupts, it can be woken up by a signal before an interrupt occurs and an interrupt may be missed if signals are delivered rapidly enough or at inopportune times. To decrease the chances of this, it is strongly suggested that you 1) do not use signals or 2) minimize their use in your application.
This function disables all interrupts for a very brief time to obtain accurate status information. If you call the function repeatedly in a loop (such as when busy-waiting for an interrupt to occur), this can interfere with system interrupts. It is strongly suggested that you do not busy-wait for interrupts.

Referenced by main().

4.20.2.4 DM7820_Error DM7820_General_Get_Version_Info (DM7820_Board_Descriptor * *handle,* uint8_t * *fpga_type_id,* uint8_t * *fpga_version,* uint16_t * *svn_version*)

Read a device's FPGA and source code revision control versions.

Parameters

handle	Address of device's library board descriptor.
fpga_type_id	Address where FPGA version information type identifier field should be stored.
fpga_version	Address where FPGA version information version identifier field should be stored.
svn_version	Address where source code revision control version identifier should be stored.

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().

4.20.2.5 DM7820_Error DM7820_General_Is_PCI_Master (DM7820_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().

4.20.2.6 DM7820_Error DM7820_General_Open_Board (uint8_t *dev_num,* DM7820_Board_Descriptor ** *handle*)

Open a DM7820 device file.

Parameters

dev_num	Minor number of DM7820 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 DM7820 device file with minor number dev_num is already open. • `ENODEV` dev_num is not a valid DM7820 minor number; 2.4 kernel only. • `ENOMEM` Library device descriptor memory allocation failed. • `ENXIO` dev_num is not a valid DM7820 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 PCI interrupts, enables PLX local interrupt input, and enables PLX DMA channel 0/1 interrupts.

Referenced by main().

4.20.2.7 DM7820_Error DM7820_General_RemoveISR (DM7820_Board_Descriptor * *handle*)

Uninstall userspace ISR.

Parameters

handle	Address of the device's library board descriptor.

Return values

0	Success
-1	Failure

Referenced by main().

4.20.2.8 DM7820_Error DM7820_General_Reset (DM7820_Board_Descriptor * *handle*)

Reset a DM7820 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.

Referenced by main().

4.20.2.9 DM7820_Error DM7820_General_SetISRPriority (DM7820_Board_Descriptor * *handle*, int *priority*)

Changes the Priority for the ISR thread.

Parameters

handle	Address of the device's library board descriptor.
priority	Value to change the thread's priority to. (99 highest priority, 1 lowest priority)

Return values

0	Success

Referenced by main().

4.21 DM7820 user library incremental encoder functions

Functions

- DM7820_Error DM7820_IncEnc_Configure (DM7820_Board_Descriptor *handle, dm7820_incenc_encoder encoder, dm7820_incenc_phase_filter phase_filter, dm7820_incenc_input_mode input_mode, uint8_t enable_input_filter, dm7820_incenc_channel_mode channel_mode, uint8_t enable_index)

  Configure the given incremental encoder.
- DM7820_Error DM7820_IncEnc_Enable (DM7820_Board_Descriptor *handle, dm7820_incenc_encoder encoder, uint8_t enable)

  Enable or disable the given incremental encoder.
- DM7820_Error DM7820_IncEnc_Enable_Hold (DM7820_Board_Descriptor *handle, dm7820_incenc_encoder encoder, uint8_t enable)

  Enable or disable value register hold for the given incremental encoder.
- DM7820_Error DM7820_IncEnc_Get_Independent_Value (DM7820_Board_Descriptor *handle, dm7820_incenc_encoder encoder, dm7820_incenc_channel channel, uint16_t *value)

  Get 16-bit counter value of the given independent incremental encoder channel.
- DM7820_Error DM7820_IncEnc_Get_Joined_Value (DM7820_Board_Descriptor *handle, dm7820_incenc_encoder encoder, uint32_t *value)

  Get 32-bit counter value of the given independent incremental encoder whose channels are joined.
- DM7820_Error DM7820_IncEnc_Get_Status (DM7820_Board_Descriptor *handle, dm7820_incenc_encoder encoder, dm7820_incenc_status_condition condition, uint8_t *occurred)

  Determine whether or not the specified status condition has occurred for the given incremental encoder.
- DM7820_Error DM7820_IncEnc_Set_Independent_Value (DM7820_Board_Descriptor *handle, dm7820_incenc_encoder encoder, dm7820_incenc_channel channel, uint16_t value)

  Set 16-bit counter value for the given independent incremental encoder channel.
- DM7820_Error DM7820_IncEnc_Set_Joined_Value (DM7820_Board_Descriptor *handle, dm7820_incenc_encoder encoder, uint32_t value)

  Set 32-bit counter value for the given incremental encoder whose channels are joined.
- DM7820_Error DM7820_IncEnc_Set_Master (DM7820_Board_Descriptor *handle, dm7820_incenc_encoder encoder, dm7820_incenc_master_clock master)

  Set the master clock for the given incremental encoder.

4.21.1 Detailed Description

DM7820_Library_General_Functions

4.21.2 Function Documentation

4.21.2.1 DM7820_Error DM7820_IncEnc_Configure (DM7820_Board_Descriptor * *handle,* dm7820_incenc_encoder *encoder,* dm7820_incenc_phase_filter *phase_filter,* dm7820_incenc_input_mode *input_mode,* uint8_t *enable_input_filter,* dm7820_incenc_channel_mode *channel_mode,* uint8_t *enable_index*)

Configure the given incremental encoder.

Parameters

handle	Address of device's library board descriptor.
encoder	The incremental encoder to configure.
phase_filter	Mask for allowing/disallowing input state transitions from changing the counter.

The DM7820_INCENC_DISABLE_PHASE_FILTER_TRANSITION, DM7820_INCENC_ENABLE_PHASE_FILTER_TRANSITION, and DM7820_INCENC_RESET_PHASE_FILTER macros should be used modify transitions in

the phase filter.

Parameters

input_mode	Incremental encoder input mode.
enable_input_filter	Flag indicating whether or not the input filter should be enabled. A value of zero means disable the input filter. Any other value means enable the input filter.
channel_mode	Incremental encoder channel mode.
enable_index	Flag indicating whether or not the index input should be enabled. A value of zero means disable the index input. Any other value means enable the index input.

Return values

0	Success.
-1	Failure. errno may be set as follows: • EINVAL encoder is not valid. • EINVAL input_mode is not valid. • EINVAL channel_mode is not valid. Please see the ioctl(2) man page for information on other possible values errno may have in this case.

Note

The incremental encoder should be disabled before calling this function.

Referenced by do_incenc(), and main().

4.21.2.2 DM7820_Error DM7820_IncEnc_Enable (DM7820_Board_Descriptor * *handle,* dm7820_incenc_encoder *encoder,* uint8_t *enable*)

Enable or disable the given incremental encoder.

Parameters

handle	Address of device's library board descriptor.
encoder	The incremental encoder to change state of.
enable	Flag indicating whether or not the incremental encoder should be enabled. A value of zero means disable the incremental encoder. Any other value means enable the incremental encoder.

Return values

0	Success.
-1	Failure. errno may be set as follows: • EINVAL encoder is not valid. Please see the ioctl(2) man page for information on other possible values errno may have in this case.

Referenced by do_incenc(), and main().

4.21.2.3 DM7820_Error DM7820_IncEnc_Enable_Hold (DM7820_Board_Descriptor * *handle,* dm7820_incenc_encoder *encoder,* uint8_t *enable*)

Enable or disable value register hold for the given incremental encoder.

Parameters

handle	Address of device's library board descriptor.
encoder	The incremental encoder to change value register hold state of.
enable	Flag indicating whether or not value register hold should be enabled. A value of zero means disable value register hold. Any other value means enable value register hold.

Return values

0	Success.
-1	Failure. errno may be set as follows: • EINVAL encoder is not valid. Please see the ioctl(2) man page for information on other possible values errno may have in this case.

Referenced by do_incenc(), and main().

4.21.2.4 DM7820_Error DM7820_IncEnc_Get_Independent_Value (DM7820_Board_Descriptor * *handle,* dm7820_incenc_encoder *encoder,* dm7820_incenc_channel *channel,* uint16_t * *value*)

Get 16-bit counter value of the given independent incremental encoder channel.

Parameters

handle	Address of device's library board descriptor.
encoder	The incremental encoder to get counter value of.
channel	The incremental encoder channel to get counter value of.
value	Address where counter value should be stored.

Return values

0	Success.
-1	Failure. errno may be set as follows: • EINVAL encoder is not valid. • EINVAL channel is not valid. • EOPNOTSUPP Incremental encoder channels A and B are joined. Please see the ioctl(2) man page for information on other possible values errno may have in this case.

Referenced by main().

4.21.2.5 DM7820_Error DM7820_IncEnc_Get_Joined_Value (DM7820_Board_Descriptor * *handle,* dm7820_incenc_encoder *encoder,* uint32_t * *value*)

Get 32-bit counter value of the given independent incremental encoder whose channels are joined.

Parameters

handle	Address of device's library board descriptor.
encoder	The incremental encoder to get counter value of.
value	Address where counter value should be stored.

Return values

0	Success.
-1	Failure. errno may be set as follows: • EINVAL encoder is not valid. • EINVAL channel is not valid. • EOPNOTSUPP Incremental encoder channels A and B are independent. Please see the ioctl(2) man page for information on other possible values errno may have in this case.

Referenced by main().

4.21.2.6 DM7820_Error DM7820_IncEnc_Get_Status (DM7820_Board_Descriptor * *handle*, dm7820_incenc_encoder *encoder*, dm7820_incenc_status_condition *condition*, uint8_t * *occurred*)

Determine whether or not the specified status condition has occurred for the given incremental encoder.

Parameters

handle	Address of device's library board descriptor.
encoder	The incremental encoder to get status of.
condition	The status condition to check for.
occurred	Address where occurrence flag should be stored. Zero will be stored here if the specified condition has not occurred. A non-zero value will be stored here if the specified condition occurred.

Return values

0	Success.
-1	Failure. errno may be set as follows: • EINVAL encoder is not valid. • EINVAL condition is not valid. Please see the ioctl(2) man page for information on other possible values errno may have in this case.

Warning

If you are using interrupts, the information returned from this function is unreliable because the driver's interrupt handler clears all incremental encoder interrupt status flags during interrupt acknowledgment.

Note

This function reads the incremental encoder status and then clears the board's specified encoder status flag if it is set. The hardware will not reassert the flag until the next time the specified condition occurs for the encoder.

Referenced by main().

4.21.2.7 DM7820_Error DM7820_IncEnc_Set_Independent_Value (DM7820_Board_Descriptor * *handle,* dm7820_incenc_encoder *encoder,* dm7820_incenc_channel *channel,* uint16_t *value*)

Set 16-bit counter value for the given independent incremental encoder channel.

Parameters

handle	Address of device's library board descriptor.
encoder	The incremental encoder to set counter value for.
channel	The incremental encoder channel to set counter value for.
value	The counter value to set.

Return values

0	Success.
-1	Failure. errno may be set as follows: • EINVAL encoder is not valid. • EINVAL channel is not valid. • EOPNOTSUPP Incremental encoder channels A and B are joined. Please see the ioctl(2) man page for information on other possible values errno may have in this case.

Referenced by do_incenc(), and main().

4.21.2.8 DM7820_Error DM7820_IncEnc_Set_Joined_Value (DM7820_Board_Descriptor * *handle,* dm7820_incenc_encoder *encoder,* uint32_t *value*)

Set 32-bit counter value for the given incremental encoder whose channels are joined.

Parameters

handle	Address of device's library board descriptor.
encoder	The incremental encoder to set counter value for.
value	The counter value to set.

Return values

0	Success.
-1	Failure. errno may be set as follows: • EINVAL encoder is not valid. • EINVAL channel is not valid. • EOPNOTSUPP Incremental encoder channels A and B are independent. Please see the ioctl(2) man page for information on other possible values errno may have in this case.

Referenced by main().

4.21.2.9 DM7820_Error DM7820_IncEnc_Set_Master (DM7820_Board_Descriptor * *handle,* dm7820_incenc_encoder *encoder,* dm7820_incenc_master_clock *master*)

Set the master clock for the given incremental encoder.

Parameters

handle	Address of device's library board descriptor.
encoder	The incremental encoder to set master clock for.
master	The master clock to select.

Return values

0	Success.
-1	Failure. errno may be set as follows: • `EINVAL` encoder is not valid. • `EINVAL` master is not valid. • `EOPNOTSUPP` master is equal to DM7820_INCENC_MASTER_RESERVED. Please see the ioctl(2) man page for information on other possible values errno may have in this case.

Note

The incremental encoder should be disabled before calling this function.

Referenced by do_incenc(), and main().

4.22 DM7820 user library pulse width modulator functions

Functions

- DM7820_Error DM7820_PWM_Enable (DM7820_Board_Descriptor *handle, dm7820_pwm_modulator pwm, uint8_t enable)

  Enable or disable the given pulse width modulator (PWM).
- DM7820_Error DM7820_PWM_Set_Period (DM7820_Board_Descriptor *handle, dm7820_pwm_modulator pwm, uint32_t period)

  Set the period for the given pulse width modulator (PWM).
- DM7820_Error DM7820_PWM_Set_Period_Master (DM7820_Board_Descriptor *handle, dm7820_pwm_modulator pwm, dm7820_pwm_period_master_clock master)

  Set the period master clock for the given pulse width modulator (PWM).
- DM7820_Error DM7820_PWM_Set_Width (DM7820_Board_Descriptor *handle, dm7820_pwm_modulator pwm, dm7820_pwm_output output, uint16_t width)

  Set the width for the specified output on the given pulse width modulator (PWM).
- DM7820_Error DM7820_PWM_Set_Width_Master (DM7820_Board_Descriptor *handle, dm7820_pwm_modulator pwm, dm7820_pwm_width_master_clock master)

  Set the width master clock for the given pulse width modulator (PWM).

4.22.1 Detailed Description

DM7820_Library_IncEnc_Functions

4.22.2 Function Documentation

4.22.2.1 DM7820_Error DM7820_PWM_Enable (DM7820_Board_Descriptor * *handle,* dm7820_pwm_modulator *pwm,* uint8_t *enable*)

Enable or disable the given pulse width modulator (PWM).

Parameters

handle	Address of device's library board descriptor.
pwm	The pulse width modulator to change state of.
enable	Flag indicating whether or not the pulse width modulator should be enabled. A value of zero means disable the PWM. Any other value means enable the PWM.

Return values

0	Success.
-1	Failure. errno may be set as follows: • EINVAL pwm is not valid. Please see the ioctl(2) man page for information on other possible values errno may have in this case.

Referenced by do_pwm(), and main().

4.22.2.2 DM7820_Error DM7820_PWM_Set_Period (DM7820_Board_Descriptor * *handle,* dm7820_pwm_modulator *pwm,* uint32_t *period*)

Set the period for the given pulse width modulator (PWM).

Parameters

handle	Address of device's library board descriptor.
pwm	The pulse width modulator to set period for.
period	Value to set as the period.

Return values

0	Success.
-1	Failure. errno may be set as follows: • EINVAL pwm is not valid. • ERANGE period is greater than 0x10000. • ERANGE period is equal to 0. Please see the ioctl(2) man page for information on other possible values errno may have in this case.

Note

The pulse width modulator should be disabled before calling this function.

Referenced by do_pwm(), and main().

4.22.2.3 DM7820_Error DM7820_PWM_Set_Period_Master (DM7820_Board_Descriptor * *handle,* dm7820_pwm_modulator *pwm,* dm7820_pwm_period_master_clock *master*)

Set the period master clock for the given pulse width modulator (PWM).

Parameters

handle	Address of device's library board descriptor.
pwm	The pulse width modulator to set period master clock for.
master	The period master clock to select.

Return values

0	Success.
-1	Failure. errno may be set as follows: • EINVAL pwm is not valid. • EINVAL master is not valid. • EOPNOTSUPP master is equal to DM7820_PWM_PERIOD_MASTER_RESERVED. Please see the ioctl(2) man page for information on other possible values errno may have in this case.

Note

The pulse width modulator should be disabled before calling this function.

Referenced by do_pwm(), and main().

4.22.2.4 DM7820_Error DM7820_PWM_Set_Width (DM7820_Board_Descriptor * *handle,* dm7820_pwm_modulator *pwm,* dm7820_pwm_output *output,* uint16_t *width*)

Set the width for the specified output on the given pulse width modulator (PWM).

Parameters

handle	Address of device's library board descriptor.
pwm	The pulse width modulator to set output width for.
output	The pulse width modulator to set width for.
width	Value to set as width.

Return values

0	Success.
-1	Failure. errno may be set as follows: • EINVAL pwm is not valid. • EINVAL output is not valid. Please see the ioctl(2) man page for information on other possible values errno may have in this case.

Referenced by do_pwm(), and main().

4.22.2.5 DM7820_Error DM7820_PWM_Set_Width_Master (DM7820_Board_Descriptor * *handle,* dm7820_pwm_modulator *pwm,* dm7820_pwm_width_master_clock *master*)

Set the width master clock for the given pulse width modulator (PWM).

Parameters

handle	Address of device's library board descriptor.
pwm	The pulse width modulator to set width master clock for.
master	The width master clock to select.

Return values

0	Success.
-1	Failure. errno may be set as follows: • EINVAL pwm is not valid. • EINVAL master is not valid. • EOPNOTSUPP master is equal to DM7820_PWM_WIDTH_MASTER_RESERVED. Please see the ioctl(2) man page for information on other possible values errno may have in this case.

Note

The pulse width modulator should be disabled before calling this function.

Referenced by do_pwm(), and main().

4.23 DM7820 user library programmable clock functions

Functions

- DM7820_Error DM7820_PrgClk_Set_Master (DM7820_Board_Descriptor *handle, dm7820_prgclk_clock clock, dm7820_prgclk_master_clock master)

  Select the master clock for the given programmable clock.
- DM7820_Error DM7820_PrgClk_Set_Mode (DM7820_Board_Descriptor *handle, dm7820_prgclk_clock clock, dm7820_prgclk_mode mode)

  Select the mode for the given programmable clock.
- DM7820_Error DM7820_PrgClk_Set_Period (DM7820_Board_Descriptor *handle, dm7820_prgclk_clock clock, uint32_t period)

  Set the period for the given programmable clock.
- DM7820_Error DM7820_PrgClk_Set_Start_Trigger (DM7820_Board_Descriptor *handle, dm7820_prgclk_clock clock, dm7820_prgclk_start_trigger start)

  Set the start trigger for the given programmable clock.
- DM7820_Error DM7820_PrgClk_Set_Stop_Trigger (DM7820_Board_Descriptor *handle, dm7820_prgclk_clock clock, dm7820_prgclk_stop_trigger stop)

  Set the stop trigger for the given programmable clock.

4.23.1 Detailed Description

DM7820_Library_PWM_Functions

4.23.2 Function Documentation

4.23.2.1 DM7820_Error DM7820_PrgClk_Set_Master (DM7820_Board_Descriptor * *handle,* dm7820_prgclk_clock *clock,* dm7820_prgclk_master_clock *master*)

Select the master clock for the given programmable clock.

Parameters

handle	Address of device's library board descriptor.
clock	The programmable clock to select master clock for.
master	The master clock to select.

Return values

0	Success.
-1	Failure. errno may be set as follows: • EINVAL clock is not valid. • EINVAL master is not valid. • EOPNOTSUPP master is equal to DM7820_PRGCLK_MASTER_RESERVED. Please see the ioctl(2) man page for information on other possible values errno may have in this case.

Note

The clock should be disabled before calling this function.

Referenced by main().

4.23.2.2 DM7820_Error DM7820_PrgClk_Set_Mode (DM7820_Board_Descriptor * *handle,* dm7820_prgclk_clock *clock,* dm7820_prgclk_mode *mode*)

Select the mode for the given programmable clock.

Parameters

handle	Address of device's library board descriptor.
clock	The programmable clock to select mode for.
mode	The mode to select.

Return values

0	Success.
-1	Failure. errno may be set as follows: • EINVAL clock is not valid. • EINVAL mode is not valid. • EOPNOTSUPP mode is equal to DM7820_PRGCLK_MODE_RESERVED. Please see the ioctl(2) man page for information on other possible values errno may have in this case.

Note

When changing from one-shot to continuous mode or from continuous to one-shot mode, the clock must be disabled as an intermediate step. For example, if a programmable clock is in continuous mode and you want to change it to one-shot mode, you first disable the clock and then set continuous mode.

Referenced by clean_up(), and main().

4.23.2.3 DM7820_Error DM7820_PrgClk_Set_Period (DM7820_Board_Descriptor * *handle,* dm7820_prgclk_clock *clock,* uint32_t *period*)

Set the period for the given programmable clock.

Parameters

handle	Address of device's library board descriptor.
clock	The programmable clock to set period for.
period	Value to set as the period.

Return values

0	Success.
-1	Failure. errno may be set as follows: • EINVAL clock is not valid. • ERANGE period is greater than 0x10000. • ERANGE period is equal to 0. Please see the ioctl(2) man page for information on other possible values errno may have in this case.

Referenced by main().

4.23.2.4 DM7820_Error DM7820_PrgClk_Set_Start_Trigger (DM7820_Board_Descriptor * *handle,* dm7820_prgclk_clock *clock,* dm7820_prgclk_start_trigger *start*)

Set the start trigger for the given programmable clock.

Parameters

handle	Address of device's library board descriptor.
clock	The programmable clock to set start trigger for.
start	The start trigger to set.

Return values

0	Success.
-1	Failure. errno may be set as follows: • EINVAL clock is not valid. • EINVAL start is not valid. Please see the ioctl(2) man page for information on other possible values errno may have in this case.

Note

The clock should be disabled before calling this function.

Referenced by main().

4.23.2.5 DM7820_Error DM7820_PrgClk_Set_Stop_Trigger (DM7820_Board_Descriptor * *handle,* dm7820_prgclk_clock *clock,* dm7820_prgclk_stop_trigger *stop*)

Set the stop trigger for the given programmable clock.

Parameters

handle	Address of device's library board descriptor.
clock	The programmable clock to set stop trigger for.
stop	The stop trigger to set.

Return values

0	Success.
-1	Failure. errno may be set as follows: • EINVAL clock is not valid. • EINVAL stop is not valid. Please see the ioctl(2) man page for information on other possible values errno may have in this case.

Note

The clock should be disabled before calling this function.

Referenced by main().

4.24 DM7820 user library standard I/O functions

Functions

- DM7820_Error DM7820_StdIO_Get_Input (DM7820_Board_Descriptor *handle, DM7820_StdIO_Port port, uint16_t *value)

  Read a value from the given standard I/O port.
- DM7820_Error DM7820_StdIO_Set_IO_Mode (DM7820_Board_Descriptor *handle, DM7820_StdIO_Port port, uint16_t bits, DM7820_StdIO_IO_Mode mode)

  Set the mode for specific bits in the given standard I/O port.
- DM7820_Error DM7820_StdIO_Set_Output (DM7820_Board_Descriptor *handle, DM7820_StdIO_Port port, uint16_t value)

  Write a value to the given standard I/O port.
- DM7820_Error DM7820_StdIO_Set_Periph_Mode (DM7820_Board_Descriptor *handle, DM7820_StdIO_Port port, uint16_t bits, DM7820_StdIO_Periph_Mode mode)

  Set the peripheral output mode for specific bits in the given standard I/O port.
- DM7820_Error DM7820_StdIO_Strobe_Input (DM7820_Board_Descriptor *handle, DM7820_StdIO_Strobe strobe, uint8_t *state)

  Determine state of given strobe signal.
- DM7820_Error DM7820_StdIO_Strobe_Mode (DM7820_Board_Descriptor *handle, DM7820_StdIO_Strobe strobe, uint8_t output)

  Set the direction (input or output) for the given strobe signal.
- DM7820_Error DM7820_StdIO_Strobe_Output (DM7820_Board_Descriptor *handle, DM7820_StdIO_Strobe strobe, uint8_t state)

  Set state of given strobe signal.

4.24.1 Detailed Description

DM7820_Library_PrgClk_Functions

4.24.2 Function Documentation

4.24.2.1 DM7820_Error DM7820_StdIO_Get_Input (DM7820_Board_Descriptor * *handle,* DM7820_StdIO_Port *port,* uint16_t * *value*)

Read a value from the given standard I/O port.

Parameters

handle	Address of device's library board descriptor.
port	The port to read.
value	Address where port value should be stored.

Note

Any port bit not set to input has whatever value currently being output on that bit.

Return values

0	Success.
-1	Failure. errno may be set as follows: • EINVAL port is not valid. Please see the ioctl(2) man page for information on other possible values errno may have in this case.

Referenced by main().

4.24.2.2 DM7820_Error DM7820_StdIO_Set_IO_Mode (DM7820_Board_Descriptor * *handle,* DM7820_StdIO_Port *port,* uint16_t *bits,* DM7820_StdIO_IO_Mode *mode*)

Set the mode for specific bits in the given standard I/O port.

Parameters

handle	Address of device's library board descriptor.
port	The port to set mode for.
bits	Bit mask indicating which port bits should have their mode set. A zero in a bit position means the corresponding port bit mode should not be set. A one in a bit position means the corresponding port bit mode should be set.
mode	The operating mode to set.

Return values

0	Success.
-1	Failure. errno may be set as follows: • EINVAL port is not valid. • EINVAL mode is not valid. Please see the ioctl(2) man page for information on other possible values errno may have in this case.

Referenced by do_digital_io(), and main().

4.24.2.3 DM7820_Error DM7820_StdIO_Set_Output (DM7820_Board_Descriptor * *handle,* DM7820_StdIO_Port *port,* uint16_t *value*)

Write a value to the given standard I/O port.

Parameters

handle	Address of device's library board descriptor.
port	The port to write.
value	Value to write.

Note

Any port bit not set to output is ignored.

Return values

0	Success.
-1	Failure. errno may be set as follows: • EINVAL port is not valid. Please see the ioctl(2) man page for information on other possible values errno may have in this case.

Referenced by main().

4.24.2.4 DM7820_Error DM7820_StdIO_Set_Periph_Mode (DM7820_Board_Descriptor * *handle,* DM7820_StdIO_Port *port,* uint16_t *bits,* DM7820_StdIO_Periph_Mode *mode*)

Set the peripheral output mode for specific bits in the given standard I/O port.

Parameters

handle	Address of device's library board descriptor.
port	The port to set peripheral output mode for.
bits	Bit mask indicating which port bits should have their peripheral output mode set. A zero in a bit position means the corresponding port bit peripheral output mode should not be set. A one in a bit position means the corresponding port bit peripheral output mode should be set.
mode	The peripheral output mode to set.

Return values

0	Success.
-1	Failure. errno may be set as follows: • `EINVAL` port is not valid. • `EINVAL` mode is not valid. • `EOPNOTSUPP` port does not support mode. Please see the ioctl(2) man page for information on other possible values errno may have in this case.

Referenced by do_digital_io(), and main().

4.24.2.5 DM7820_Error DM7820_StdIO_Strobe_Input (DM7820_Board_Descriptor * *handle,* DM7820_StdIO_Strobe *strobe,* uint8_t * *state*)

Determine state of given strobe signal.

Parameters

handle	Address of device's library board descriptor.
strobe	Strobe signal to check state of.
state	Address where strobe signal state should be stored. Zero will be stored here if the strobe signal is low. A non-zero value will be stored here if the strobe signal is high.

Return values

0	Success.
-1	Failure. errno may be set as follows: • `EINVAL` strobe is not valid. Please see the ioctl(2) man page for information on other possible values errno may have in this case.

Referenced by main().

4.24.2.6 DM7820_Error DM7820_StdIO_Strobe_Mode (DM7820_Board_Descriptor * *handle,* DM7820_StdIO_Strobe *strobe,* uint8_t *output*)

Set the direction (input or output) for the given strobe signal.

Parameters

handle	Address of device's library board descriptor.
strobe	Strobe signal to set direction of.
output	Flag indicating whether or not the strobe signal should be set to output. A value of zero means set the signal to input. Any other value means set the signal to output.

Return values

0	Success.
-1	Failure. errno may be set as follows: • EINVAL strobe is not valid. Please see the ioctl(2) man page for information on other possible values errno may have in this case.

Referenced by main().

4.24.2.7 DM7820_Error DM7820_StdIO_Strobe_Output (DM7820_Board_Descriptor * *handle*, DM7820_StdIO_Strobe *strobe*, uint8_t *state*)

Set state of given strobe signal.

Parameters

handle	Address of device's library board descriptor.
strobe	Strobe signal to set state of.
state	Flag indicating what state the signal should be set to. A value of zero set the signal low. Any other value sets the signal high.

Return values

0	Success.
-1	Failure. errno may be set as follows: • EINVAL strobe is not valid. Please see the ioctl(2) man page for information on other possible values errno may have in this case.

Referenced by main().

4.25 DM7820 user library 8254 timer/counter functions

Functions

- DM7820_Error DM7820_TmrCtr_Get_Status (DM7820_Board_Descriptor *handle, dm7820_tmrctr_timer timer, uint8_t *occurred)

  Determine whether or not a status condition has occurred for the given timer/counter.
- DM7820_Error DM7820_TmrCtr_Program (DM7820_Board_Descriptor *handle, dm7820_tmrctr_timer timer, dm7820_tmrctr_waveform waveform, dm7820_tmrctr_count_mode count_mode, uint16_t divisor)

  Program the given 8254 timer/counter. This will 1) set the waveform mode, 2) set the count mode, and 3) load a divisor into the timer.
- DM7820_Error DM7820_TmrCtr_Read (DM7820_Board_Descriptor *handle, dm7820_tmrctr_timer timer, uint16_t *value)

  Read the value of the given 8254 timer/counter.
- DM7820_Error DM7820_TmrCtr_Select_Clock (DM7820_Board_Descriptor *handle, dm7820_tmrctr_timer timer, dm7820_tmrctr_clock clock)

  Select the clock input for the given 8254 timer/counter.
- DM7820_Error DM7820_TmrCtr_Select_Gate (DM7820_Board_Descriptor *handle, dm7820_tmrctr_timer timer, dm7820_tmrctr_gate gate)

  Select the gate input for the given 8254 timer/counter.

4.25.1 Detailed Description

DM7820_Library_StdIO_Functions

4.25.2 Function Documentation

4.25.2.1 DM7820_Error DM7820_TmrCtr_Get_Status (DM7820_Board_Descriptor * *handle*, dm7820_tmrctr_timer *timer*, uint8_t * *occurred*)

Determine whether or not a status condition has occurred for the given timer/counter.

Parameters

handle	Address of device's library board descriptor.
timer	8254 timer/counter to get status of.
occurred	Address where occurrence flag should be stored. Zero will be stored here if no status condition has occurred for the timer. A non-zero value will be stored here if a status condition has occurred for the timer.

Return values

0	Success.
-1	Failure. errno may be set as follows: • EINVAL timer is not valid. Please see the ioctl(2) man page for information on other possible values errno may have in this case.

Warning

If you are using interrupts, the information returned from this function is unreliable because the driver's interrupt handler clears all timer/counter interrupt status flags during interrupt acknowledgment.

Note

This function reads the timer/counter status and then clears the board's timer status flag if it is set. The hardware will not reassert the flag until the next condition occurs for the timer.

Referenced by main().

4.25.2.2 DM7820_Error DM7820_TmrCtr_Program (DM7820_Board_Descriptor * *handle,* dm7820_tmrctr_timer *timer,* dm7820_tmrctr_waveform *waveform,* dm7820_tmrctr_count_mode *count_mode,* uint16_t *divisor*)

Program the given 8254 timer/counter. This will 1) set the waveform mode, 2) set the count mode, and 3) load a divisor into the timer.

Parameters

handle	Address of device's library board descriptor.
timer	8254 timer/counter to select clock input for.
waveform	The waveform mode to program.
count_mode	The count mode to program.
divisor	16-bit divisor to load into the timer.

Return values

0	Success.
-1	Failure. errno may be set as follows: • EINVAL timer is not valid. • EINVAL waveform is not valid. • EINVAL count_mode is not valid. Please see the ioctl(2) man page for information on other possible values errno may have in this case.

Referenced by do_8254(), and main().

4.25.2.3 DM7820_Error DM7820_TmrCtr_Read (DM7820_Board_Descriptor * *handle,* dm7820_tmrctr_timer *timer,* uint16_t * *value*)

Read the value of the given 8254 timer/counter.

Parameters

handle	Address of device's library board descriptor.
timer	8254 timer/counter to read value of.
value	Address where timer/counter value should be stored.

Return values

0	Success.
-1	Failure. errno may be set as follows: • EINVAL timer is not valid. Please see the ioctl(2) man page for information on other possible values errno may have in this case.

Referenced by main().

4.25.2.4 DM7820_Error DM7820_TmrCtr_Select_Clock (DM7820_Board_Descriptor * *handle,* dm7820_tmrctr_timer *timer,* dm7820_tmrctr_clock *clock*)

Select the clock input for the given 8254 timer/counter.

Parameters

handle	Address of device's library board descriptor.
timer	8254 timer/counter to select clock input for.
clock	Clock input to select.

Return values

0	Success.
-1	Failure. errno may be set as follows: • EINVAL timer is not valid. • EINVAL clock is not valid. • EOPNOTSUPP clock is equal to DM7820_TMRCTR_CLOCK_RESERVED. Please see the ioctl(2) man page for information on other possible values errno may have in this case.

Referenced by do_8254(), and main().

4.25.2.5 DM7820_Error DM7820_TmrCtr_Select_Gate (DM7820_Board_Descriptor * *handle,* dm7820_tmrctr_timer *timer,* dm7820_tmrctr_gate *gate*)

Select the gate input for the given 8254 timer/counter.

Parameters

handle	Address of device's library board descriptor.
timer	8254 timer/counter to select gate input for.
gate	Gate input to select.

Return values

0	Success.
-1	Failure. errno may be set as follows: • EINVAL timer is not valid. • EINVAL gate is not valid. Please see the ioctl(2) man page for information on other possible values errno may have in this case.

Referenced by disable_timers(), do_8254(), and main().

4.26 DM7820 register header file

Modules

- DM7820 register BAR0 (memory-mapped PLX registers) offsets
- DM7820 register BAR1 (I/O-mapped PLX registers) offsets
- DM7820 register BAR2 (memory-mapped FPGA registers) offsets
- DM7820 register PCI region lengths
- DM7820 register functional block identifiers
- DM7820 register BAR2 macros

4.26.1 Detailed Description

4.27 DM7820 register BAR0 (memory-mapped PLX registers) offsets

Macros

- #define DM7820_BAR0_INTCSR 0x68

  PLX interrupt control/status.
- #define DM7820_BAR0_DMAMODE0 0x80

  PLX DMA channel 0 mode.
- #define DM7820_BAR0_DMAPADR0 0x84

  PLX DMA channel 0 PCI address.
- #define DM7820_BAR0_DMALADR0 0x88

  PLX DMA channel 0 local address.
- #define DM7820_BAR0_DMASIZ0 0x8C

  PLX DMA channel 0 transfer size.
- #define DM7820_BAR0_DMADPR0 0x90

  PLX DMA channel 0 descriptor pointer.
- #define DM7820_BAR0_DMAMODE1 0x94

  PLX DMA channel 1 mode.
- #define DM7820_BAR0_DMAPADR1 0x98

  PLX DMA channel 1 PCI address.
- #define DM7820_BAR0_DMALADR1 0x9C

  PLX DMA channel 1 local address.
- #define DM7820_BAR0_DMASIZ1 0xA0

  PLX DMA channel 1 transfer size.
- #define DM7820_BAR0_DMADPR1 0xA4

  PLX DMA channel 1 descriptor pointer.
- #define DM7820_BAR0_DMACSR0 0xA8

  PLX DMA channel 0 command/status.
- #define DM7820_BAR0_DMACSR1 0xA9

  PLX DMA channel 1 command/status.
- #define DM7820_BAR0_DMAARB 0xAC

  PLX DMA arbitration.
- #define DM7820_BAR0_DMATHR 0xB0

  PLX DMA threshold.
- #define DM7820_BAR0_DMADA0 0xB4

  PLX DMA channel 0 PCI dual address cycles upper address.
- #define DM7820_BAR0_DMADA1 0xB8

  PLX DMA channel 1 PCI dual address cycles upper address.

4.27.1 Detailed Description

4.28 DM7820 register BAR1 (I/O-mapped PLX registers) offsets

Macros

- #define DM7820_BAR1_INTCSR 0x68

  PLX interrupt control/status.
- #define DM7820_BAR1_DMAMODE0 0x80

  PLX DMA channel 0 mode.
- #define DM7820_BAR1_DMAPADR0 0x84

  PLX DMA channel 0 PCI address.
- #define DM7820_BAR1_DMALADR0 0x88

  PLX DMA channel 0 local address.
- #define DM7820_BAR1_DMASIZ0 0x8C

  PLX DMA channel 0 transfer size.
- #define DM7820_BAR1_DMADPR0 0x90

  PLX DMA channel 0 descriptor pointer.
- #define DM7820_BAR1_DMAMODE1 0x94

  PLX DMA channel 1 mode.
- #define DM7820_BAR1_DMAPADR1 0x98

  PLX DMA channel 1 PCI address.
- #define DM7820_BAR1_DMALADR1 0x9C

  PLX DMA channel 1 local address.
- #define DM7820_BAR1_DMASIZ1 0xA0

  PLX DMA channel 1 transfer size.
- #define DM7820_BAR1_DMADPR1 0xA4

  PLX DMA channel 1 descriptor pointer.
- #define DM7820_BAR1_DMACSR0 0xA8

  PLX DMA channel 0 command/status.
- #define DM7820_BAR1_DMACSR1 0xA9

  PLX DMA channel 1 command/status.
- #define DM7820_BAR1_DMAARB 0xAC

  PLX DMA arbitration.
- #define DM7820_BAR1_DMATHR 0xB0

  PLX DMA threshold.
- #define DM7820_BAR1_DMADA0 0xB4

  PLX DMA channel 0 PCI dual address cycles upper address.
- #define DM7820_BAR1_DMADA1 0xB8

  PLX DMA channel 1 PCI dual address cycles upper address.

4.28.1 Detailed Description

DM7820_Register_BAR0_Offsets

4.29 DM7820 register BAR2 (memory-mapped FPGA registers) offsets

Modules

- DM7820 register board control offsets
- DM7820 register standard I/O offsets
- DM7820 register 8254 timer/counter offsets
- DM7820 register FIFO 0 offsets
- DM7820 register FIFO 1 offsets
- DM7820 register programmable clock 0 offsets
- DM7820 register programmable clock 1 offsets
- DM7820 register programmable clock 2 offsets
- DM7820 register programmable clock 3 offsets
- DM7820 register advanced interrupt 0 offsets
- DM7820 register advanced interrupt 1 offsets
- DM7820 register incremental encoder 0 offsets
- DM7820 register incremental encoder 1 offsets
- DM7820 register pulse width modulator 0 offsets
- DM7820 register pulse width modulator 1 offsets
- DM7820 register 8254 timer/counter A offsets
- DM7820 register 8254 timer/counter B offsets

Macros

- #define PLX9056_DMA_WIDTH_MASK 0x00000003
- #define PLX9056_DMA_WIDTH_8 0x00000000
- #define PLX9056_DMA_WIDTH_16 0x00000001
- #define PLX9056_DMA_WIDTH_32 0x00000002
- #define PLX9056_DMA_WAITSTATES_MASK 0x0000003c
- #define PLX9056_DMA_READY 0x00000040
- #define PLX9056_DMA_BURST 0x00000080
- #define PLX9056_DMA_LOCAL_BURST 0x00000100
- #define PLX9056_DMA_SCATTERGATHER 0x00000200
- #define PLX9056_DMA_DONE_INTERRUPT 0x00000400
- #define PLX9056_DMA_LOCAL_ADDRESSING_MODE 0x00000800
- #define PLX9056_DMA_DEMAND_MODE 0x00001000
- #define PLX9056_DMA_MEMWRITE_INV 0x00002000
- #define PLX9056_DMA_EOT_ENABLE 0x00004000
- #define PLX9056_DMA_FAST_SLOW_TERM 0x00008000
- #define PLX9056_DMA_CLEAR_COUNT 0x00010000
- #define PLX9056_DMA_INTERRUPT_SEL 0x00020000
- #define PLX9056_DMA_DAC_CHAIN 0x00040000
- #define PLX9056_DMA_EOT_END 0x00080000
- #define PLX9056_DMA_RING_MODE 0x00100000
- #define PLX9056_DMA_RING_CONTROL 0x00200000

4.29.1 Detailed Description

DM7820_Register_BAR1_Offsets

4.29.2 Macro Definition Documentation

4.29.2.1 #define PLX9056_DMA_BURST 0x00000080

Continuous burst enable (use as mask and control)

Definition at line 560 of file dm7820_registers.h.

4.29.2.2 #define PLX9056_DMA_CLEAR_COUNT 0x00010000

Clear count mode (use as mask and control) When enabled this mdoe will clear our the length field of the scatter/gather descriptor table entry when the DMA is completed. Interrupts are delayed until this clearing operation is comlpeted.

Definition at line 629 of file dm7820_registers.h.

4.29.2.3 #define PLX9056_DMA_DAC_CHAIN 0x00040000

DAC Chain load (not supported most chipsets)

Definition at line 643 of file dm7820_registers.h.

4.29.2.4 #define PLX9056_DMA_DEMAND_MODE 0x00001000

DMA Demand mode (use as mask and control) Enabling this mode causes the PLX to transfer data when the DREQ# input is asserted.

Definition at line 601 of file dm7820_registers.h.

4.29.2.5 #define PLX9056_DMA_DONE_INTERRUPT 0x00000400

DMA done interrupt enable (use as mask and control) At the time the DMA cycle is completed an interrupt will occur on the PCI bus if the master interrupt is enabled IF PLX9056_DMA_CLEAR_COUNT is enabled then the interrupt will not occur until the byte count is cleared in the PCI memory table (scatter/gather mode is enabled in this case).

Definition at line 585 of file dm7820_registers.h.

4.29.2.6 #define PLX9056_DMA_EOT_ENABLE 0x00004000

EOT# Enable

Definition at line 613 of file dm7820_registers.h.

4.29.2.7 #define PLX9056_DMA_EOT_END 0x00080000

EOT# End Link

Definition at line 649 of file dm7820_registers.h.

4.29.2.8 #define PLX9056_DMA_FAST_SLOW_TERM 0x00008000

Fast/Slow Terminate mode selected

Definition at line 619 of file dm7820_registers.h.

4.29.2.9 #define PLX9056_DMA_INTERRUPT_SEL 0x00020000

Interrupt Select when set, the DMA channel's interrupt is routed to the PCI buss

Definition at line 637 of file dm7820_registers.h.

4.29.2.10 #define PLX9056_DMA_LOCAL_ADDRESSING_MODE 0x00000800

DMA local addressing mode (use as mask and control) Enabling this bit holds the local address bus pointer constant, disabling causes it to autoincrement.

Definition at line 593 of file dm7820_registers.h.

4.29.2.11 #define PLX9056_DMA_LOCAL_BURST 0x00000100

Lecal burst enable (use as mask and control)

Definition at line 566 of file dm7820_registers.h.

4.29.2.12 #define PLX9056_DMA_MEMWRITE_INV 0x00002000

Memory write and invalidate mode.

Definition at line 607 of file dm7820_registers.h.

4.29.2.13 #define PLX9056_DMA_READY 0x00000040

TA#/READY# Input enable (use as mask and control)

Definition at line 554 of file dm7820_registers.h.

4.29.2.14 #define PLX9056_DMA_RING_CONTROL 0x00200000

Ring Management Void Stop Control Valid when PLX9056_DMA_RING_MODE is set When this bit is 0 the Scatter/-Gather controller will load the descriptor and execute the DMA. If this bit is 1 then the Scatter/Gather controller will stop polling when the valid bit goes 0. The host will need to restart the DMA controller by setting the DMACSRx[1] to 1

Definition at line 672 of file dm7820_registers.h.

4.29.2.15 #define PLX9056_DMA_RING_MODE 0x00100000

Ring Management Valid Mode Enabled When set the Ring Management Valid bit in DMASIZx[31] register bit controls the processing of DMA descriptors. If the valid bit is set, the transfer count is 0 and the descriptor is not the last descriptor in the chain then the DMA controller will move to the next descriptor in the chain

Definition at line 660 of file dm7820_registers.h.

4.29.2.16 #define PLX9056_DMA_SCATTERGATHER 0x00000200

Scatter/Gather mode enabled (use as mask and control) When this bit is enabled() the PLX will get address and length of data from a table in the PCI memory.

Definition at line 574 of file dm7820_registers.h.

4.29.2.17 #define PLX9056_DMA_WAITSTATES_MASK 0x0000003c

Internal wait state counter mask. This is a 4 bit value that is 0 by default.

Definition at line 548 of file dm7820_registers.h.

4.29.2.18 #define PLX9056_DMA_WIDTH_16 0x00000001

Local Bus Data Width 16 Bits.

Definition at line 535 of file dm7820_registers.h.

4.29.2.19 #define PLX9056_DMA_WIDTH_32 0x00000002

Local Bus Data Width 32 Bits.

Definition at line 541 of file dm7820_registers.h.

4.29.2.20 #define PLX9056_DMA_WIDTH_8 0x00000000

Local Bus Data Width 8 Bits.

Definition at line 529 of file dm7820_registers.h.

4.29.2.21 #define PLX9056_DMA_WIDTH_MASK 0x00000003

DM7820_Register_StdIO Local Bus Data Width Mask

Definition at line 523 of file dm7820_registers.h.

4.30 DM7820 register board control offsets

Macros

- #define DM7820_BAR2_FPGA_VERSION 0x0000

  FPGA version and type identifiers.
- #define DM7820_BAR2_SVN_VERSION 0x0002

  FPGA source code revision control version identifier.
- #define DM7820_BAR2_BOARD_RESET 0x0004

  Board reset.
- #define DM7820_BAR2_BRD_STAT 0x0008

  Board status.
- #define DM7820_BAR2_INTERRUPT_ENABLE 0x0010

  Local interrupt enable.
- #define DM7820_BAR2_INTERRUPT_STATUS 0x0012

  Local interrupt status.

4.30.1 Detailed Description

4.31 DM7820 register standard I/O offsets

Macros

- #define DM7820_BAR2_PORT0_OUTPUT 0x0040

  Port 0 output value.
- #define DM7820_BAR2_PORT0_INPUT 0x0042

  Port 0 input value.
- #define DM7820_BAR2_PORT0_TRISTATE 0x0044

  Port 0 direction.
- #define DM7820_BAR2_PORT0_MODE 0x0046

  Port 0 operating mode.
- #define DM7820_BAR2_PORT1_OUTPUT 0x0048

  Port 1 output value.
- #define DM7820_BAR2_PORT1_INPUT 0x004A

  Port 1 input value.
- #define DM7820_BAR2_PORT1_TRISTATE 0x004C

  Port 1 direction.
- #define DM7820_BAR2_PORT1_MODE 0x004E

  Port 1 operating mode.
- #define DM7820_BAR2_PORT2_OUTPUT 0x0050

  Port 2 output value.
- #define DM7820_BAR2_PORT2_INPUT 0x0052

  Port 2 input value.
- #define DM7820_BAR2_PORT2_TRISTATE 0x0054

  Port 2 direction.
- #define DM7820_BAR2_PORT2_MODE 0x0056

  Port 2 operating mode.
- #define DM7820_BAR2_STROBE_STATUS 0x0058

  Port 2 strobe signal status.
- #define DM7820_BAR2_PORT0_PERIPH_SEL_L 0x0060

  Port 0 bits 0-7 peripheral select.
- #define DM7820_BAR2_PORT0_PERIPH_SEL_H 0x0062

  Port 0 bits 8-15 peripheral select.
- #define DM7820_BAR2_PORT1_PERIPH_SEL_L 0x0064

  Port 1 bits 0-7 peripheral select.
- #define DM7820_BAR2_PORT1_PERIPH_SEL_H 0x0066

  Port 1 bits 8-15 peripheral select.
- #define DM7820_BAR2_PORT2_PERIPH_SEL_L 0x0068

  Port 2 bits 0-7 peripheral select.
- #define DM7820_BAR2_PORT2_PERIPH_SEL_H 0x006A

  Port 2 bits 8-15 peripheral select.

4.31.1 Detailed Description

DM7820_Register_General

4.32 DM7820 register 8254 timer/counter offsets

Macros

- #define DM7820_BAR2_TC_ID 0x0080

  8254 timer/counter block identifier
- #define DM7820_BAR2_TC_INT 0x0082

  8254 timer/counter interrupt control/status
- #define DM7820_BAR2_TC_A0_CONTROL 0x0084

  8254 timer/counter A0 control
- #define DM7820_BAR2_TC_A1_CONTROL 0x0086

  8254 timer/counter A1 control
- #define DM7820_BAR2_TC_A2_CONTROL 0x0088

  8254 timer/counter A2 control
- #define DM7820_BAR2_TC_B0_CONTROL 0x008A

  8254 timer/counter B0 control
- #define DM7820_BAR2_TC_B1_CONTROL 0x008C

  8254 timer/counter B1 control
- #define DM7820_BAR2_TC_B2_CONTROL 0x008E

  8254 timer/counter B2 control

4.32.1 Detailed Description

4.33 DM7820 register FIFO 0 offsets

Macros

- #define DM7820_BAR2_FIFO0_ID 0x00C0

  FIFO 0 block identifier.
- #define DM7820_BAR2_FIFO0_INT 0x00C2

  FIFO 0 interrupt control/status.
- #define DM7820_BAR2_FIFO0_IN_CLK 0x00C4

  FIFO 0 input clock.
- #define DM7820_BAR2_FIFO0_OUT_CLK 0x00C6

  FIFO 0 output clock.
- #define DM7820_BAR2_FIFO0_IN_DATA_DREQ 0x00C8

  FIFO 0 data input and PLX DMA request source.
- #define DM7820_BAR2_FIFO0_CON_STAT 0x00CA

  FIFO 0 control/status.
- #define DM7820_BAR2_FIFO0_RW_PORT 0x00CC

  FIFO 0 PCI read/write port.

4.33.1 Detailed Description

DM7820_Register_TmrCtr

4.34 DM7820 register FIFO 1 offsets

Macros

- #define DM7820_BAR2_FIFO1_ID 0x00D0

  FIFO 1 block identifier.
- #define DM7820_BAR2_FIFO1_INT 0x00D2

  FIFO 1 interrupt control/status.
- #define DM7820_BAR2_FIFO1_IN_CLK 0x00D4

  FIFO 1 input clock.
- #define DM7820_BAR2_FIFO1_OUT_CLK 0x00D6

  FIFO 1 output clock.
- #define DM7820_BAR2_FIFO1_IN_DATA_DREQ 0x00D8

  FIFO 1 data input and PLX DMA request source.
- #define DM7820_BAR2_FIFO1_CON_STAT 0x00DA

  FIFO 1 control/status.
- #define DM7820_BAR2_FIFO1_RW_PORT 0x00DC

  FIFO 1 PCI read/write port.

4.34.1 Detailed Description

DM7820_Register_FIFO_0

4.35 DM7820 register programmable clock 0 offsets

Macros

- #define DM7820_BAR2_PRGCLK0_ID 0x0100

  Programmable clock 0 block identifier.
- #define DM7820_BAR2_PRGCLK0_MODE 0x0102

  Programmable clock 0 operating mode.
- #define DM7820_BAR2_PRGCLK0_CLK 0x0104

  Programmable clock 0 master clock.
- #define DM7820_BAR2_PRGCLK0_START_STOP 0x0106

  Programmable clock 0 start/stop triggers.
- #define DM7820_BAR2_PRGCLK0_PERIOD 0x0108

  Programmable clock 0 period.

4.35.1 Detailed Description

DM7820_Register_FIFO_1

4.36 DM7820 register programmable clock 1 offsets

Macros

- #define DM7820_BAR2_PRGCLK1_ID 0x0140

  Programmable clock 1 block identifier.
- #define DM7820_BAR2_PRGCLK1_MODE 0x0142

  Programmable clock 1 operating mode.
- #define DM7820_BAR2_PRGCLK1_CLK 0x0144

  Programmable clock 1 master clock.
- #define DM7820_BAR2_PRGCLK1_START_STOP 0x0146

  Programmable clock 1 start/stop triggers.
- #define DM7820_BAR2_PRGCLK1_PERIOD 0x0148

  Programmable clock 1 period.

4.36.1 Detailed Description

DM7820_Register_PrgClk_0

4.37 DM7820 register programmable clock 2 offsets

Macros

- #define DM7820_BAR2_PRGCLK2_ID 0x0180

  Programmable clock 2 block identifier.
- #define DM7820_BAR2_PRGCLK2_MODE 0x0182

  Programmable clock 2 operating mode.
- #define DM7820_BAR2_PRGCLK2_CLK 0x0184

  Programmable clock 2 master clock.
- #define DM7820_BAR2_PRGCLK2_START_STOP 0x0186

  Programmable clock 2 start/stop triggers.
- #define DM7820_BAR2_PRGCLK2_PERIOD 0x0188

  Programmable clock 2 period.

4.37.1 Detailed Description

DM7820_Register_PrgClk_1

4.38 DM7820 register programmable clock 3 offsets

Macros

- #define DM7820_BAR2_PRGCLK3_ID 0x01C0

  Programmable clock 3 block identifier.
- #define DM7820_BAR2_PRGCLK3_MODE 0x01C2

  Programmable clock 3 operating mode.
- #define DM7820_BAR2_PRGCLK3_CLK 0x01C4

  Programmable clock 3 master clock.
- #define DM7820_BAR2_PRGCLK3_START_STOP 0x01C6

  Programmable clock 3 start/stop triggers.
- #define DM7820_BAR2_PRGCLK3_PERIOD 0x01C8

  Programmable clock 3 period.

4.38.1 Detailed Description

DM7820_Register_PrgClk_2

4.39 DM7820 register advanced interrupt 0 offsets

Macros

- #define DM7820_BAR2_ADVINT0_ID 0x0200

  Advanced interrupt 0 block identifier.
- #define DM7820_BAR2_ADVINT0_INT_MODE 0x0202

  Advanced interrupt 0 mode.
- #define DM7820_BAR2_ADVINT0_CLK 0x0204

  Advanced interrupt 0 master clock.
- #define DM7820_BAR2_ADVINT0_PORT0_MASK 0x0208

  Advanced interrupt 0 standard I/O port 0 mask.
- #define DM7820_BAR2_ADVINT0_PORT1_MASK 0x020A

  Advanced interrupt 0 standard I/O port 1 mask.
- #define DM7820_BAR2_ADVINT0_PORT2_MASK 0x020C

  Advanced interrupt 0 standard I/O port 2 mask.
- #define DM7820_BAR2_ADVINT0_PORT0_CMP 0x0210

  Advanced interrupt 0 standard I/O port 0 event mode compare.
- #define DM7820_BAR2_ADVINT0_PORT1_CMP 0x0212

  Advanced interrupt 0 standard I/O port 1 event mode compare.
- #define DM7820_BAR2_ADVINT0_PORT2_CMP 0x0214

  Advanced interrupt 0 standard I/O port 2 event mode compare.
- #define DM7820_BAR2_ADVINT0_PORT0_CAPT 0x0218

  Advanced interrupt 0 standard I/O port 0 value capture.
- #define DM7820_BAR2_ADVINT0_PORT1_CAPT 0x021A

  Advanced interrupt 0 standard I/O port 1 value capture.
- #define DM7820_BAR2_ADVINT0_PORT2_CAPT 0x021C

  Advanced interrupt 0 standard I/O port 2 value capture.

4.39.1 Detailed Description

DM7820_Register_PrgClk_3

4.40 DM7820 register advanced interrupt 1 offsets

Macros

- #define DM7820_BAR2_ADVINT1_ID 0x0240

  Advanced interrupt 1 block identifier.
- #define DM7820_BAR2_ADVINT1_INT_MODE 0x0242

  Advanced interrupt 1 mode.
- #define DM7820_BAR2_ADVINT1_CLK 0x0244

  Advanced interrupt 1 master clock.
- #define DM7820_BAR2_ADVINT1_PORT0_MASK 0x0248

  Advanced interrupt 1 standard I/O port 0 mask.
- #define DM7820_BAR2_ADVINT1_PORT1_MASK 0x024A

  Advanced interrupt 1 standard I/O port 1 mask.
- #define DM7820_BAR2_ADVINT1_PORT2_MASK 0x024C

  Advanced interrupt 1 standard I/O port 2 mask.
- #define DM7820_BAR2_ADVINT1_PORT0_CMP 0x0250

  Advanced interrupt 1 standard I/O port 0 event mode compare.
- #define DM7820_BAR2_ADVINT1_PORT1_CMP 0x0252

  Advanced interrupt 1 standard I/O port 1 event mode compare.
- #define DM7820_BAR2_ADVINT1_PORT2_CMP 0x0254

  Advanced interrupt 1 standard I/O port 2 event mode compare.
- #define DM7820_BAR2_ADVINT1_PORT0_CAPT 0x0258

  Advanced interrupt 1 standard I/O port 0 value capture.
- #define DM7820_BAR2_ADVINT1_PORT1_CAPT 0x025A

  Advanced interrupt 1 standard I/O port 1 value capture.
- #define DM7820_BAR2_ADVINT1_PORT2_CAPT 0x025C

  Advanced interrupt 1 standard I/O port 2 value capture.

4.40.1 Detailed Description

DM7820_Register_AdvInt_0

4.41 DM7820 register incremental encoder 0 offsets

Macros

- #define DM7820_BAR2_INCENC0_ID 0x0280

  Incremental encoder 0 block identifier.
- #define DM7820_BAR2_INCENC0_INT 0x0282

  Incremental encoder 0 interrupt control/status.
- #define DM7820_BAR2_INCENC0_CLOCK 0x0284

  Incremental encoder 0 master clock.
- #define DM7820_BAR2_INCENC0_MODE 0x0286

  Incremental encoder 0 operating mode.
- #define DM7820_BAR2_INCENC0_VALUEA 0x0288

  Incremental encoder 0 channel A value.
- #define DM7820_BAR2_INCENC0_VALUEB 0x028A

  Incremental encoder 0 channel B value.

4.41.1 Detailed Description

DM7820_Register_AdvInt_1

4.42 DM7820 register incremental encoder 1 offsets

Macros

- #define DM7820_BAR2_INCENC1_ID 0x02C0

  Incremental encoder 1 block identifier.
- #define DM7820_BAR2_INCENC1_INT 0x02C2

  Incremental encoder 1 interrupt control/status.
- #define DM7820_BAR2_INCENC1_CLOCK 0x02C4

  Incremental encoder 1 master clock.
- #define DM7820_BAR2_INCENC1_MODE 0x02C6

  Incremental encoder 1 operating mode.
- #define DM7820_BAR2_INCENC1_VALUEA 0x02C8

  Incremental encoder 1 channel A value.
- #define DM7820_BAR2_INCENC1_VALUEB 0x02CA

  Incremental encoder 1 channel B value.

4.42.1 Detailed Description

DM7820_Register_IncEnc_0

4.43 DM7820 register pulse width modulator 0 offsets

Macros

- #define DM7820_BAR2_PWM0_ID 0x0300

  Pulse width modulator 0 block identifier.
- #define DM7820_BAR2_PWM0_MODE 0x0302

  Pulse width modulator 0 mode.
- #define DM7820_BAR2_PWM0_CLK 0x0304

  Pulse width modulator 0 period/width clocks.
- #define DM7820_BAR2_PWM0_PERIOD 0x0308

  Pulse width modulator 0 period.
- #define DM7820_BAR2_PWM0_WIDTHA 0x0310

  Pulse width modulator 0 output A width.
- #define DM7820_BAR2_PWM0_WIDTHB 0x0314

  Pulse width modulator 0 output B width.
- #define DM7820_BAR2_PWM0_WIDTHC 0x0318

  Pulse width modulator 0 output C width.
- #define DM7820_BAR2_PWM0_WIDTHD 0x031C

  Pulse width modulator 0 output D width.

4.43.1 Detailed Description

DM7820_Register_IncEnc_1

4.44 DM7820 register pulse width modulator 1 offsets

Macros

- #define DM7820_BAR2_PWM1_ID 0x0340

  Pulse width modulator 1 block identifier.
- #define DM7820_BAR2_PWM1_MODE 0x0342

  Pulse width modulator 1 mode.
- #define DM7820_BAR2_PWM1_CLK 0x0344

  Pulse width modulator 1 period/width clocks.
- #define DM7820_BAR2_PWM1_PERIOD 0x0348

  Pulse width modulator 1 period.
- #define DM7820_BAR2_PWM1_WIDTHA 0x0350

  Pulse width modulator 1 output A width.
- #define DM7820_BAR2_PWM1_WIDTHB 0x0354

  Pulse width modulator 1 output B width.
- #define DM7820_BAR2_PWM1_WIDTHC 0x0358

  Pulse width modulator 1 output C width.
- #define DM7820_BAR2_PWM1_WIDTHD 0x035C

  Pulse width modulator 1 output D width.

4.44.1 Detailed Description

DM7820_Register_PWM_0

4.45 DM7820 register 8254 timer/counter A offsets

Macros

- #define DM7820_BAR2_TCA_COUNTER_0 0x1000

  8254 timer/counter A timer 0 value
- #define DM7820_BAR2_TCA_COUNTER_1 0x1004

  8254 timer/counter A timer 1 value
- #define DM7820_BAR2_TCA_COUNTER_2 0x1008

  8254 timer/counter A timer 2 value
- #define DM7820_BAR2_TCA_CON_WORD 0x100C

  8254 timer/counter A control word

4.45.1 Detailed Description

DM7820_Register_PWM_1

4.46 DM7820 register 8254 timer/counter B offsets

Macros

- #define DM7820_BAR2_TCB_COUNTER_0 0x1010

  8254 timer/counter B timer 0 value
- #define DM7820_BAR2_TCB_COUNTER_1 0x1014

  8254 timer/counter B timer 1 value
- #define DM7820_BAR2_TCB_COUNTER_2 0x1018

  8254 timer/counter B timer 2 value
- #define DM7820_BAR2_TCB_CON_WORD 0x101C

  8254 timer/counter B control word

4.46.1 Detailed Description

DM7820_Register_TmrCtr_A

4.47 DM7820 register PCI region lengths

Macros

- #define DM7820_BAR0_LENGTH 0x200

  Length in bytes of BAR0 (memory-mapped PLX registers)
- #define DM7820_BAR1_LENGTH 0x100

  Length in bytes of BAR1 (I/O-mapped PLX registers)
- #define DM7820_BAR2_LENGTH 0x2000

  Length in bytes of BAR2 (memory-mapped FPGA registers)

4.47.1 Detailed Description

DM7820_Register_BAR2_Offsets

4.48 DM7820 register functional block identifiers

Modules

- DM7820 register functional block identifier values
- DM7820 register functional block identifier offsets

4.48.1 Detailed Description

DM7820_Register_PCI_Region_Lengths

4.49 DM7820 register functional block identifier values

Macros

- #define DM7820_ID_TIMER_COUNTER 0x1001

  8254 timer/counter block identifier
- #define DM7820_ID_FIFO 0x2011

  FIFO block identifier.
- #define DM7820_ID_PROGRAMMABLE_CLOCK 0x1000

  Programmable clock block identifier.
- #define DM7820_ID_ADVANCED_INTERRUPT 0x0001

  Advanced interrupt block identifier.
- #define DM7820_ID_INCREMENTAL_ENCODER 0x0002

  Incremental encoder block identifier.
- #define DM7820_ID_PULSE_WIDTH_MODULATOR 0x0003

  Pulse width modulator block identifier.
- #define DM7820_ID_NONE 0x0000

  Empty block identifier.

4.49.1 Detailed Description

4.50 DM7820 register functional block identifier offsets

Macros

- #define DM7820_FIRST_ID_OFFSET 0x0080

  Offset of first possible block identifier.
- #define DM7820_LAST_ID_OFFSET 0x03C0

  Offset of last possible block identifier.

4.50.1 Detailed Description

DM7820_Register_Block_ID_Values

4.51 DM7820 register BAR2 macros

Modules

- DM7820 register BAR2 FPGA Version Register
- DM7820 register BAR2 Board Reset Register

4.51.1 Detailed Description

DM7820_Register_Block_IDs

4.52 DM7820 register BAR2 FPGA Version Register

Macros

- #define DM7820_FPGA_VERSION_TYPE_ID_MASK 0xFF00

  Bit mask to extract FPGA type identifier.
- #define DM7820_FPGA_VERSION_VERSION_MASK 0x00FF

  Bit mask to extract FPGA version identifier.

4.52.1 Detailed Description

4.53 DM7820 register BAR2 Board Reset Register

Macros

- #define DM7820_BOARD_RESET_DO_RESET 0xA5A5

  Value to write to cause a board reset.

4.53.1 Detailed Description

DM7820_Register_BAR2_FPGA_Version

4.54 DM7820 type definition header file

Modules

- DM7820 type enumerations
- DM7820 type definition structures
- DM7820 type definition typedefs

4.54.1 Detailed Description

4.55 DM7820 type enumerations

Modules

- DM7820 type PCI enumerations
- DM7820 type standard I/O enumerations
- DM7820 type timer/counter enumerations
- DM7820 type programmable clock enumerations
- DM7820 type pulse width modulator enumerations
- DM7820 type incremental encoder enumerations
- DM7820 type FIFO enumerations
- DM7820 type advanced interrupt enumerations
- DM7820 type interrupt enumerations

4.55.1 Detailed Description

4.56 DM7820 type PCI enumerations

Typedefs

- typedef enum dm7820_pci_region_num dm7820_pci_region_num_t

  Standard PCI region number type.
- typedef enum dm7820_pci_region_access_size dm7820_pci_region_access_size_t

  Standard PCI region access size type.

Enumerations

- enum dm7820_pci_region_num { DM7820_PCI_REGION_PLX_MEM = 0, DM7820_PCI_REGION_PLX_IO, DM7820_PCI_REGION_FPGA_MEM }

  Standard PCI region number.
- enum dm7820_pci_region_access_size { DM7820_PCI_REGION_ACCESS_8 = 0, DM7820_PCI_REGION_ACCESS_16, DM7820_PCI_REGION_ACCESS_32 }

  Desired size in bits of access to standard PCI region.

4.56.1 Detailed Description

4.56.2 Enumeration Type Documentation

4.56.2.1 enum dm7820_pci_region_access_size

Desired size in bits of access to standard PCI region.

Enumerator:

DM7820_PCI_REGION_ACCESS_8 8-bit access

DM7820_PCI_REGION_ACCESS_16 16-bit access

DM7820_PCI_REGION_ACCESS_32 32-bit access

Definition at line 94 of file dm7820_types.h.

4.56.2.2 enum dm7820_pci_region_num

Standard PCI region number.

Enumerator:

DM7820_PCI_REGION_PLX_MEM Memory-mapped PLX registers (BAR0)

DM7820_PCI_REGION_PLX_IO I/O-mapped PLX registers (BAR1)

DM7820_PCI_REGION_FPGA_MEM Memory-mapped FPGA registers (BAR2)

Definition at line 61 of file dm7820_types.h.

4.57 DM7820 type standard I/O enumerations

Typedefs

- typedef enum _DM7820_StdIO_Port DM7820_StdIO_Port
  Standard I/O port type.
- typedef enum _DM7820_StdIO_IO_Mode DM7820_StdIO_IO_Mode
  Standard I/O port mode type.
- typedef enum _DM7820_StdIO_Periph_Mode DM7820_StdIO_Periph_Mode
  Standard I/O port peripheral output mode type.
- typedef enum _DM7820_StdIO_Strobe DM7820_StdIO_Strobe
  Standard I/O port strobe signal type.

Enumerations

- enum _DM7820_StdIO_Port { DM7820_STDIO_PORT_0 = 0, DM7820_STDIO_PORT_1, DM7820_STDIO_PORT_2 }
  Standard I/O ports.
- enum _DM7820_StdIO_IO_Mode { DM7820_STDIO_MODE_INPUT = 0, DM7820_STDIO_MODE_OUTPUT, DM7820_STDIO_MODE_PER_OUT }
  Standard I/O port modes.
- enum _DM7820_StdIO_Periph_Mode { DM7820_STDIO_PERIPH_PWM = 0x0, DM7820_STDIO_PERIPH_CLK_OTHER, DM7820_STDIO_PERIPH_FIFO_0, DM7820_STDIO_PERIPH_FIFO_1 }
  Standard I/O port peripheral output modes.
- enum _DM7820_StdIO_Strobe { DM7820_STDIO_STROBE_1 = 0, DM7820_STDIO_STROBE_2 }
  Strobe signals.

4.57.1 Detailed Description

DM7820_Types_PCI_Enumerations

4.57.2 Enumeration Type Documentation

4.57.2.1 enum _DM7820_StdIO_IO_Mode

Standard I/O port modes.

Enumerator:

DM7820_STDIO_MODE_INPUT Input

DM7820_STDIO_MODE_OUTPUT Output

DM7820_STDIO_MODE_PER_OUT Peripheral output

Definition at line 174 of file dm7820_types.h.

4.57.2.2 enum _DM7820_StdIO_Periph_Mode

Standard I/O port peripheral output modes.

Enumerator:

DM7820_STDIO_PERIPH_PWM Pulse Width Modulator (PWM) mode; valid for port 2 only

DM7820_STDIO_PERIPH_CLK_OTHER Clock/other mode; valid for port 2 only

DM7820_STDIO_PERIPH_FIFO_0 FIFO 0 mode; valid for all ports

DM7820_STDIO_PERIPH_FIFO_1 FIFO 1 mode; valid for all ports

Definition at line 207 of file dm7820_types.h.

4.57.2.3 enum _DM7820_StdIO_Port

Standard I/O ports.

Enumerator:

DM7820_STDIO_PORT_0 Port 0

DM7820_STDIO_PORT_1 Port 1

DM7820_STDIO_PORT_2 Port 2

Definition at line 141 of file dm7820_types.h.

4.57.2.4 enum _DM7820_StdIO_Strobe

Strobe signals.

Enumerator:

DM7820_STDIO_STROBE_1 Strobe signal 1

DM7820_STDIO_STROBE_2 Strobe signal 2

Definition at line 246 of file dm7820_types.h.

4.58 DM7820 type timer/counter enumerations

Typedefs

- typedef enum _dm7820_tmrctr_timer dm7820_tmrctr_timer

  8254 timer/counter type
- typedef enum _dm7820_tmrctr_clock dm7820_tmrctr_clock

  8254 timer/counter clock selector type
- typedef enum _dm7820_tmrctr_gate dm7820_tmrctr_gate

  8254 timer/counter gate selector type
- typedef enum _dm7820_tmrctr_waveform dm7820_tmrctr_waveform

  8254 timer/counter waverform mode selector type
- typedef enum _dm7820_tmrctr_count_mode dm7820_tmrctr_count_mode

  8254 timer/counter count mode selector type

Enumerations

- enum _dm7820_tmrctr_timer { DM7820_TMRCTR_TIMER_A_0 = 0, DM7820_TMRCTR_TIMER_A_1, DM7820_TMRCTR_TIMER_A_2, DM7820_TMRCTR_TIMER_B_0, DM7820_TMRCTR_TIMER_B_1, DM7820_TMRCTR_TIMER_B_2 }

  8254 timers/counters
- enum _dm7820_tmrctr_clock { DM7820_TMRCTR_CLOCK_5_MHZ = 0, DM7820_TMRCTR_CLOCK_RESERVED, DM7820_TMRCTR_CLOCK_8254_A_0, DM7820_TMRCTR_CLOCK_8254_A_1, DM7820_TMRCTR_CLOCK_8254_A_2, DM7820_TMRCTR_CLOCK_8254_B_0, DM7820_TMRCTR_CLOCK_8254_B_1, DM7820_TMRCTR_CLOCK_8254_B_2, DM7820_TMRCTR_CLOCK_PROG_CLOCK_0, DM7820_TMRCTR_CLOCK_PROG_CLOCK_1, DM7820_TMRCTR_CLOCK_PROG_CLOCK_2, DM7820_TMRCTR_CLOCK_PROG_CLOCK_3, DM7820_TMRCTR_CLOCK_STROBE_1, DM7820_TMRCTR_CLOCK_STROBE_2, DM7820_TMRCTR_CLOCK_INV_STROBE_1, DM7820_TMRCTR_CLOCK_INV_STROBE_2 }

  8254 timer/counter clock selectors
- enum _dm7820_tmrctr_gate { DM7820_TMRCTR_GATE_LOGIC_0 = 0, DM7820_TMRCTR_GATE_LOGIC_1, DM7820_TMRCTR_GATE_8254_A_0, DM7820_TMRCTR_GATE_8254_A_1, DM7820_TMRCTR_GATE_8254_A_2, DM7820_TMRCTR_GATE_8254_B_0, DM7820_TMRCTR_GATE_8254_B_1, DM7820_TMRCTR_GATE_8254_B_2, DM7820_TMRCTR_GATE_PROG_CLOCK_0, DM7820_TMRCTR_GATE_PROG_CLOCK_1, DM7820_TMRCTR_GATE_PROG_CLOCK_2, DM7820_TMRCTR_GATE_PROG_CLOCK_3, DM7820_TMRCTR_GATE_STROBE_1, DM7820_TMRCTR_GATE_STROBE_2, DM7820_TMRCTR_GATE_INV_STROBE_1, DM7820_TMRCTR_GATE_INV_STROBE_2, DM7820_TMRCTR_GATE_PORT_2_BIT_0, DM7820_TMRCTR_GATE_PORT_2_BIT_1, DM7820_TMRCTR_GATE_PORT_2_BIT_2, DM7820_TMRCTR_GATE_PORT_2_BIT_3, DM7820_TMRCTR_GATE_PORT_2_BIT_4, DM7820_TMRCTR_GATE_PORT_2_BIT_5, DM7820_TMRCTR_GATE_PORT_2_BIT_6, DM7820_TMRCTR_GATE_PORT_2_BIT_7, DM7820_TMRCTR_GATE_PORT_2_BIT_8, DM7820_TMRCTR_GATE_PORT_2_BIT_9, DM7820_TMRCTR_GATE_PORT_2_BIT_10, DM7820_TMRCTR_GATE_PORT_2_BIT_11, DM7820_TMRCTR_GATE_PORT_2_BIT_12, DM7820_TMRCTR_GATE_PORT_2_BIT_13, DM7820_TMRCTR_GATE_PORT_2_BIT_14, DM7820_TMRCTR_GATE_PORT_2_BIT_15 }

  8254 timer/counter gate selectors

- enum _dm7820_tmrctr_waveform { DM7820_TMRCTR_WAVEFORM_EVENT_CTR = 0, DM7820_TMRCTR_WAVEFORM_PROG_ONE_SHOT, DM7820_TMRCTR_WAVEFORM_RATE_GENERATOR, DM7820_TMRCTR_WAVEFORM_SQUARE_WAVE, DM7820_TMRCTR_WAVEFORM_SOFTWARE_STROBE, DM7820_TMRCTR_WAVEFORM_HARDWARE_STROBE }

  8254 timer/counter waveform mode selectors
- enum _dm7820_tmrctr_count_mode { DM7820_TMRCTR_COUNT_MODE_BINARY = 0, DM7820_TMRCTR_COUNT_MODE_BCD }

  8254 timer/counter count mode selectors

4.58.1 Detailed Description

DM7820_Types_StdIO_Enumerations

4.58.2 Enumeration Type Documentation

4.58.2.1 enum _dm7820_tmrctr_clock

8254 timer/counter clock selectors

Enumerator:

DM7820_TMRCTR_CLOCK_5_MHZ 5 MHz clock

DM7820_TMRCTR_CLOCK_RESERVED Reserved; do not use

DM7820_TMRCTR_CLOCK_8254_A_0 8254 timer/counter A0

DM7820_TMRCTR_CLOCK_8254_A_1 8254 timer/counter A1

DM7820_TMRCTR_CLOCK_8254_A_2 8254 timer/counter A2

DM7820_TMRCTR_CLOCK_8254_B_0 8254 timer/counter B0

DM7820_TMRCTR_CLOCK_8254_B_1 8254 timer/counter B1

DM7820_TMRCTR_CLOCK_8254_B_2 8254 timer/counter B2

DM7820_TMRCTR_CLOCK_PROG_CLOCK_0 Programmable clock 0

DM7820_TMRCTR_CLOCK_PROG_CLOCK_1 Programmable clock 1

DM7820_TMRCTR_CLOCK_PROG_CLOCK_2 Programmable clock 2

DM7820_TMRCTR_CLOCK_PROG_CLOCK_3 Programmable clock 3

DM7820_TMRCTR_CLOCK_STROBE_1 Strobe signal 1

DM7820_TMRCTR_CLOCK_STROBE_2 Strobe signal 2

DM7820_TMRCTR_CLOCK_INV_STROBE_1 Inverted strobe signal 1

DM7820_TMRCTR_CLOCK_INV_STROBE_2 Inverted strobe signal 2

Definition at line 337 of file dm7820_types.h.

4.58.2.2 enum _dm7820_tmrctr_count_mode

8254 timer/counter count mode selectors

Enumerator:

DM7820_TMRCTR_COUNT_MODE_BINARY 16-bit binary mode

DM7820_TMRCTR_COUNT_MODE_BCD Binary Coded Decimal (BCD) mode

Definition at line 706 of file dm7820_types.h.

4.58.2.3 enum _dm7820_tmrctr_gate

8254 timer/counter gate selectors

Enumerator:

DM7820_TMRCTR_GATE_LOGIC_0 Logic 0
DM7820_TMRCTR_GATE_LOGIC_1 Logic 1
DM7820_TMRCTR_GATE_8254_A_0 8254 timer/counter A0
DM7820_TMRCTR_GATE_8254_A_1 8254 timer/counter A1
DM7820_TMRCTR_GATE_8254_A_2 8254 timer/counter A2
DM7820_TMRCTR_GATE_8254_B_0 8254 timer/counter B0
DM7820_TMRCTR_GATE_8254_B_1 8254 timer/counter B1
DM7820_TMRCTR_GATE_8254_B_2 8254 timer/counter B2
DM7820_TMRCTR_GATE_PROG_CLOCK_0 Programmable clock 0
DM7820_TMRCTR_GATE_PROG_CLOCK_1 Programmable clock 1
DM7820_TMRCTR_GATE_PROG_CLOCK_2 Programmable clock 2
DM7820_TMRCTR_GATE_PROG_CLOCK_3 Programmable clock 3
DM7820_TMRCTR_GATE_STROBE_1 Strobe signal 1
DM7820_TMRCTR_GATE_STROBE_2 Strobe signal 2
DM7820_TMRCTR_GATE_INV_STROBE_1 Inverted strobe signal 1
DM7820_TMRCTR_GATE_INV_STROBE_2 Inverted strobe signal 2
DM7820_TMRCTR_GATE_PORT_2_BIT_0 Digital I/O port 2 bit 0
DM7820_TMRCTR_GATE_PORT_2_BIT_1 Digital I/O port 2 bit 1
DM7820_TMRCTR_GATE_PORT_2_BIT_2 Digital I/O port 2 bit 2
DM7820_TMRCTR_GATE_PORT_2_BIT_3 Digital I/O port 2 bit 3
DM7820_TMRCTR_GATE_PORT_2_BIT_4 Digital I/O port 2 bit 4
DM7820_TMRCTR_GATE_PORT_2_BIT_5 Digital I/O port 2 bit 5
DM7820_TMRCTR_GATE_PORT_2_BIT_6 Digital I/O port 2 bit 6
DM7820_TMRCTR_GATE_PORT_2_BIT_7 Digital I/O port 2 bit 7
DM7820_TMRCTR_GATE_PORT_2_BIT_8 Digital I/O port 2 bit 8
DM7820_TMRCTR_GATE_PORT_2_BIT_9 Digital I/O port 2 bit 9
DM7820_TMRCTR_GATE_PORT_2_BIT_10 Digital I/O port 2 bit 10
DM7820_TMRCTR_GATE_PORT_2_BIT_11 Digital I/O port 2 bit 11
DM7820_TMRCTR_GATE_PORT_2_BIT_12 Digital I/O port 2 bit 12
DM7820_TMRCTR_GATE_PORT_2_BIT_13 Digital I/O port 2 bit 13
DM7820_TMRCTR_GATE_PORT_2_BIT_14 Digital I/O port 2 bit 14
DM7820_TMRCTR_GATE_PORT_2_BIT_15 Digital I/O port 2 bit 15

Definition at line 448 of file dm7820_types.h.

4.58.2.4 enum _dm7820_tmrctr_timer

8254 timers/counters

Enumerator:

DM7820_TMRCTR_TIMER_A_0 Timer 0 on first 8254 chip

DM7820_TMRCTR_TIMER_A_1 Timer 1 on first 8254 chip

DM7820_TMRCTR_TIMER_A_2 Timer 2 on first 8254 chip

DM7820_TMRCTR_TIMER_B_0 Timer 0 on second 8254 chip

DM7820_TMRCTR_TIMER_B_1 Timer 1 on second 8254 chip

DM7820_TMRCTR_TIMER_B_2 Timer 2 on second 8254 chip

Definition at line 286 of file dm7820_types.h.

4.58.2.5 enum _dm7820_tmrctr_waveform

8254 timer/counter waveform mode selectors

Enumerator:

DM7820_TMRCTR_WAVEFORM_EVENT_CTR Event counter

DM7820_TMRCTR_WAVEFORM_PROG_ONE_SHOT Programmable one shot

DM7820_TMRCTR_WAVEFORM_RATE_GENERATOR Rate generator

DM7820_TMRCTR_WAVEFORM_SQUARE_WAVE Square wave generator

DM7820_TMRCTR_WAVEFORM_SOFTWARE_STROBE Software triggered strobe

DM7820_TMRCTR_WAVEFORM_HARDWARE_STROBE Hardware triggered strobe

Definition at line 655 of file dm7820_types.h.

4.59 DM7820 type programmable clock enumerations

Typedefs

- typedef enum _dm7820_prgclk_clock dm7820_prgclk_clock

  Programmable clock type.
- typedef enum _dm7820_prgclk_mode dm7820_prgclk_mode

  Programmable clock mode type.
- typedef enum _dm7820_prgclk_master_clock dm7820_prgclk_master_clock

  Programmable clock master clock type.
- typedef enum _dm7820_prgclk_start_trigger dm7820_prgclk_start_trigger

  Programmable clock start trigger type.
- typedef enum _dm7820_prgclk_stop_trigger dm7820_prgclk_stop_trigger

  Programmable clock stop trigger type.

Enumerations

- enum _dm7820_prgclk_clock { DM7820_PRGCLK_CLOCK_0 = 0, DM7820_PRGCLK_CLOCK_1, DM7820_PRGCLK_CLOCK_2, DM7820_PRGCLK_CLOCK_3 }

  Programmable clocks.
- enum _dm7820_prgclk_mode { DM7820_PRGCLK_MODE_DISABLED = 0, DM7820_PRGCLK_MODE_CONTINUOUS, DM7820_PRGCLK_MODE_RESERVED, DM7820_PRGCLK_MODE_ONE_SHOT }

  Programmable clock modes.
- enum _dm7820_prgclk_master_clock { DM7820_PRGCLK_MASTER_25_MHZ = 0, DM7820_PRGCLK_MASTER_SAMPLE_CLOCK, DM7820_PRGCLK_MASTER_8254_A_0, DM7820_PRGCLK_MASTER_8254_A_1, DM7820_PRGCLK_MASTER_8254_A_2, DM7820_PRGCLK_MASTER_8254_B_0, DM7820_PRGCLK_MASTER_8254_B_1, DM7820_PRGCLK_MASTER_8254_B_2, DM7820_PRGCLK_MASTER_PROG_CLOCK_0, DM7820_PRGCLK_MASTER_PROG_CLOCK_1, DM7820_PRGCLK_MASTER_PROG_CLOCK_2, DM7820_PRGCLK_MASTER_PROG_CLOCK_3, DM7820_PRGCLK_MASTER_STROBE_1, DM7820_PRGCLK_MASTER_STROBE_2, DM7820_PRGCLK_MASTER_INV_STROBE_1, DM7820_PRGCLK_MASTER_INV_STROBE_2 }

  Programmable clock master clocks.
- enum _dm7820_prgclk_start_trigger { DM7820_PRGCLK_START_IMMEDIATE = 0, DM7820_PRGCLK_START_RESERVED_1, DM7820_PRGCLK_START_8254_A_0, DM7820_PRGCLK_START_8254_A_1, DM7820_PRGCLK_START_8254_A_2, DM7820_PRGCLK_START_8254_B_0, DM7820_PRGCLK_START_8254_B_1, DM7820_PRGCLK_START_8254_B_2, DM7820_PRGCLK_START_PROG_CLOCK_0, DM7820_PRGCLK_START_PROG_CLOCK_1, DM7820_PRGCLK_START_PROG_CLOCK_2, DM7820_PRGCLK_START_PROG_CLOCK_3, DM7820_PRGCLK_START_STROBE_1, DM7820_PRGCLK_START_STROBE_2, DM7820_PRGCLK_START_INV_STROBE_1, DM7820_PRGCLK_START_INV_STROBE_2, DM7820_PRGCLK_START_ADVANCED_INT_0, DM7820_PRGCLK_START_ADVANCED_INT_1, DM7820_PRGCLK_START_8254_INT, DM7820_PRGCLK_START_RESERVED_2, DM7820_PRGCLK_START_INC_ENCODER_0_INT, DM7820_PRGCLK_START_INC_ENCODER_1_INT, DM7820_PRGCLK_START_RESERVED_3, DM7820_PRGCLK_START_RESERVED_4, DM7820_PRGCLK_START_PWM_0_INT, DM7820_PRGCLK_START_PWM_1_INT, DM7820_PRGCLK_START_PROG_CLOCK_0_INT, DM7820_PRGCLK_START_PROG_CLOCK_1_INT, DM7820_PRGCLK_START_PROG_CLOCK_2_INT, DM7820_PRGCLK_START_PROG_CLOCK_3_INT, DM7820_PRGCLK_START_FIFO_0_INT, DM7820_PRGCLK_START_FIFO_1_INT }

  Programmable clock start triggers.

- enum _dm7820_prgclk_stop_trigger {
  DM7820_PRGCLK_STOP_NONE = 0, DM7820_PRGCLK_STOP_RESERVED_1, DM7820_PRGCLK_STOP_8254_A_0, DM7820_PRGCLK_STOP_8254_A_1,
  DM7820_PRGCLK_STOP_8254_A_2, DM7820_PRGCLK_STOP_8254_B_0, DM7820_PRGCLK_STOP_8254_B_1, DM7820_PRGCLK_STOP_8254_B_2,
  DM7820_PRGCLK_STOP_PROG_CLOCK_0, DM7820_PRGCLK_STOP_PROG_CLOCK_1, DM7820_PRGCLK_STOP_PROG_CLOCK_2, DM7820_PRGCLK_STOP_PROG_CLOCK_3,
  DM7820_PRGCLK_STOP_STROBE_1, DM7820_PRGCLK_STOP_STROBE_2, DM7820_PRGCLK_STOP_INV_STROBE_1, DM7820_PRGCLK_STOP_INV_STROBE_2,
  DM7820_PRGCLK_STOP_ADVANCED_INT_0, DM7820_PRGCLK_STOP_ADVANCED_INT_1, DM7820_PRGCLK_STOP_8254_INT, DM7820_PRGCLK_STOP_RESERVED_2,
  DM7820_PRGCLK_STOP_INC_ENCODER_0_INT, DM7820_PRGCLK_STOP_INC_ENCODER_1_INT, DM7820_PRGCLK_STOP_RESERVED_3, DM7820_PRGCLK_STOP_RESERVED_4,
  DM7820_PRGCLK_STOP_PWM_0_INT, DM7820_PRGCLK_STOP_PWM_1_INT, DM7820_PRGCLK_STOP_PROG_CLOCK_0_INT, DM7820_PRGCLK_STOP_PROG_CLOCK_1_INT,
  DM7820_PRGCLK_STOP_PROG_CLOCK_2_INT, DM7820_PRGCLK_STOP_PROG_CLOCK_3_INT, DM7820_PRGCLK_STOP_FIFO_0_INT, DM7820_PRGCLK_STOP_FIFO_1_INT }

  Programmable clock stop triggers.

4.59.1 Detailed Description

DM7820_Types_TmrCtr_Enumerations

4.59.2 Enumeration Type Documentation

4.59.2.1 enum _dm7820_prgclk_clock

Programmable clocks.

Enumerator:

DM7820_PRGCLK_CLOCK_0 Programmable clock 0

DM7820_PRGCLK_CLOCK_1 Programmable clock 1

DM7820_PRGCLK_CLOCK_2 Programmable clock 2

DM7820_PRGCLK_CLOCK_3 Programmable clock 3

Definition at line 746 of file dm7820_types.h.

4.59.2.2 enum _dm7820_prgclk_master_clock

Programmable clock master clocks.

Enumerator:

DM7820_PRGCLK_MASTER_25_MHZ 25 MHz clock

DM7820_PRGCLK_MASTER_SAMPLE_CLOCK Reserved; do not use

DM7820_PRGCLK_MASTER_8254_A_0 8254 timer/counter A0

DM7820_PRGCLK_MASTER_8254_A_1 8254 timer/counter A1

DM7820_PRGCLK_MASTER_8254_A_2 8254 timer/counter A2

DM7820_PRGCLK_MASTER_8254_B_0 8254 timer/counter B0

DM7820_PRGCLK_MASTER_8254_B_1 8254 timer/counter B1

DM7820_PRGCLK_MASTER_8254_B_2 8254 timer/counter B2

DM7820_PRGCLK_MASTER_PROG_CLOCK_0 Programmable clock 0

DM7820_PRGCLK_MASTER_PROG_CLOCK_1 Programmable clock 1
DM7820_PRGCLK_MASTER_PROG_CLOCK_2 Programmable clock 2
DM7820_PRGCLK_MASTER_PROG_CLOCK_3 Programmable clock 3
DM7820_PRGCLK_MASTER_STROBE_1 Strobe signal 1
DM7820_PRGCLK_MASTER_STROBE_2 Strobe signal 2
DM7820_PRGCLK_MASTER_INV_STROBE_1 Inverted strobe signal 1
DM7820_PRGCLK_MASTER_INV_STROBE_2 Inverted strobe signal 2

Definition at line 824 of file dm7820_types.h.

4.59.2.3 enum _dm7820_prgclk_mode

Programmable clock modes.

Enumerator:

DM7820_PRGCLK_MODE_DISABLED Disabled
DM7820_PRGCLK_MODE_CONTINUOUS Continuous mode
DM7820_PRGCLK_MODE_RESERVED Reserved; do not use
DM7820_PRGCLK_MODE_ONE_SHOT One shot mode

Definition at line 785 of file dm7820_types.h.

4.59.2.4 enum _dm7820_prgclk_start_trigger

Programmable clock start triggers.

Enumerator:

DM7820_PRGCLK_START_IMMEDIATE Start the clock immediately
DM7820_PRGCLK_START_RESERVED_1 Reserved; do not use
DM7820_PRGCLK_START_8254_A_0 8254 timer/counter A0
DM7820_PRGCLK_START_8254_A_1 8254 timer/counter A1
DM7820_PRGCLK_START_8254_A_2 8254 timer/counter A2
DM7820_PRGCLK_START_8254_B_0 8254 timer/counter B0
DM7820_PRGCLK_START_8254_B_1 8254 timer/counter B1
DM7820_PRGCLK_START_8254_B_2 8254 timer/counter B2
DM7820_PRGCLK_START_PROG_CLOCK_0 Programmable clock 0
DM7820_PRGCLK_START_PROG_CLOCK_1 Programmable clock 1
DM7820_PRGCLK_START_PROG_CLOCK_2 Programmable clock 2
DM7820_PRGCLK_START_PROG_CLOCK_3 Programmable clock 3
DM7820_PRGCLK_START_STROBE_1 Strobe signal 1
DM7820_PRGCLK_START_STROBE_2 Strobe signal 2
DM7820_PRGCLK_START_INV_STROBE_1 Inverted strobe signal 1
DM7820_PRGCLK_START_INV_STROBE_2 Inverted strobe signal 2
DM7820_PRGCLK_START_ADVANCED_INT_0 Advanced interrupt 0
DM7820_PRGCLK_START_ADVANCED_INT_1 Advanced interrupt 1
DM7820_PRGCLK_START_8254_INT 8254 timer/counter interrupt
DM7820_PRGCLK_START_RESERVED_2 Reserved; do not use

DM7820_PRGCLK_START_INC_ENCODER_0_INT Incremental Encoder 0 interrupt
DM7820_PRGCLK_START_INC_ENCODER_1_INT Incremental Encoder 1 interrupt
DM7820_PRGCLK_START_RESERVED_3 Reserved; do not use
DM7820_PRGCLK_START_RESERVED_4 Reserved; do not use
DM7820_PRGCLK_START_PWM_0_INT Pulse Width Modulator 0 interrupt
DM7820_PRGCLK_START_PWM_1_INT Pulse Width Modulator 1 interrupt
DM7820_PRGCLK_START_PROG_CLOCK_0_INT Programmable clock 0 interrupt
DM7820_PRGCLK_START_PROG_CLOCK_1_INT Programmable clock 1 interrupt
DM7820_PRGCLK_START_PROG_CLOCK_2_INT Programmable clock 2 interrupt
DM7820_PRGCLK_START_PROG_CLOCK_3_INT Programmable clock 3 interrupt
DM7820_PRGCLK_START_FIFO_0_INT FIFO 0 interrupt
DM7820_PRGCLK_START_FIFO_1_INT FIFO 1 interrupt

Definition at line 935 of file dm7820_types.h.

4.59.2.5 enum _dm7820_prgclk_stop_trigger

Programmable clock stop triggers.

Enumerator:

DM7820_PRGCLK_STOP_NONE Do not stop the clock
DM7820_PRGCLK_STOP_RESERVED_1 Reserved; do not use
DM7820_PRGCLK_STOP_8254_A_0 8254 timer/counter A0
DM7820_PRGCLK_STOP_8254_A_1 8254 timer/counter A1
DM7820_PRGCLK_STOP_8254_A_2 8254 timer/counter A2
DM7820_PRGCLK_STOP_8254_B_0 8254 timer/counter B0
DM7820_PRGCLK_STOP_8254_B_1 8254 timer/counter B1
DM7820_PRGCLK_STOP_8254_B_2 8254 timer/counter B2
DM7820_PRGCLK_STOP_PROG_CLOCK_0 Programmable clock 0
DM7820_PRGCLK_STOP_PROG_CLOCK_1 Programmable clock 1
DM7820_PRGCLK_STOP_PROG_CLOCK_2 Programmable clock 2
DM7820_PRGCLK_STOP_PROG_CLOCK_3 Programmable clock 3
DM7820_PRGCLK_STOP_STROBE_1 Strobe signal 1
DM7820_PRGCLK_STOP_STROBE_2 Strobe signal 2
DM7820_PRGCLK_STOP_INV_STROBE_1 Inverted strobe signal 1
DM7820_PRGCLK_STOP_INV_STROBE_2 Inverted strobe signal 2
DM7820_PRGCLK_STOP_ADVANCED_INT_0 Advanced interrupt 0
DM7820_PRGCLK_STOP_ADVANCED_INT_1 Advanced interrupt 1
DM7820_PRGCLK_STOP_8254_INT 8254 timer/counter interrupt
DM7820_PRGCLK_STOP_RESERVED_2 Reserved; do not use
DM7820_PRGCLK_STOP_INC_ENCODER_0_INT Incremental Encoder 0 interrupt
DM7820_PRGCLK_STOP_INC_ENCODER_1_INT Incremental Encoder 1 interrupt
DM7820_PRGCLK_STOP_RESERVED_3 Reserved; do not use
DM7820_PRGCLK_STOP_RESERVED_4 Reserved; do not use
DM7820_PRGCLK_STOP_PWM_0_INT Pulse Width Modulator 0 interrupt
DM7820_PRGCLK_STOP_PWM_1_INT Pulse Width Modulator 1 interrupt

DM7820_PRGCLK_STOP_PROG_CLOCK_0_INT Programmable clock 0 interrupt

DM7820_PRGCLK_STOP_PROG_CLOCK_1_INT Programmable clock 1 interrupt

DM7820_PRGCLK_STOP_PROG_CLOCK_2_INT Programmable clock 2 interrupt

DM7820_PRGCLK_STOP_PROG_CLOCK_3_INT Programmable clock 3 interrupt

DM7820_PRGCLK_STOP_FIFO_0_INT FIFO 0 interrupt

DM7820_PRGCLK_STOP_FIFO_1_INT FIFO 1 interrupt

Definition at line 1142 of file dm7820_types.h.

4.60 DM7820 type pulse width modulator enumerations

Typedefs

- typedef enum _dm7820_pwm_modulator dm7820_pwm_modulator

  Pulse width modulator type.

- typedef enum _dm7820_pwm_period_master_clock dm7820_pwm_period_master_clock

  Pulse width modulator period master clock type.

- typedef enum _dm7820_pwm_output dm7820_pwm_output

  Pulse width modulator output type.

- typedef enum _dm7820_pwm_width_master_clock dm7820_pwm_width_master_clock

  Pulse width modulator width master clock type.

Enumerations

- enum _dm7820_pwm_modulator

  Pulse width modulators.

- enum _dm7820_pwm_period_master_clock { DM7820_PWM_PERIOD_MASTER_25_MHZ = 0, DM7820_PWM_PERIOD_MASTER_RESERVED, DM7820_PWM_PERIOD_MASTER_8254_A_0, DM7820_PWM_PERIOD_MASTER_8254_A_1, DM7820_PWM_PERIOD_MASTER_8254_A_2, DM7820_PWM_PERIOD_MASTER_8254_B_0, DM7820_PWM_PERIOD_MASTER_8254_B_1, DM7820_PWM_PERIOD_MASTER_8254_B_2, DM7820_PWM_PERIOD_MASTER_PROG_CLOCK_0, DM7820_PWM_PERIOD_MASTER_PROG_CLOCK_1, DM7820_PWM_PERIOD_MASTER_PROG_CLOCK_2, DM7820_PWM_PERIOD_MASTER_PROG_CLOCK_3, DM7820_PWM_PERIOD_MASTER_STROBE_1, DM7820_PWM_PERIOD_MASTER_STROBE_2, DM7820_PWM_PERIOD_MASTER_INV_STROBE_1, DM7820_PWM_PERIOD_MASTER_INV_STROBE_2 }

  Pulse width modulator period master clocks.

- enum _dm7820_pwm_output { DM7820_PWM_OUTPUT_A = 0, DM7820_PWM_OUTPUT_B, DM7820_PWM_OUTPUT_C, DM7820_PWM_OUTPUT_D }

  Pulse width modulator outputs.

- enum _dm7820_pwm_width_master_clock { DM7820_PWM_WIDTH_MASTER_25_MHZ = 0, DM7820_PWM_WIDTH_MASTER_RESERVED, DM7820_PWM_WIDTH_MASTER_8254_A_0, DM7820_PWM_WIDTH_MASTER_8254_A_1, DM7820_PWM_WIDTH_MASTER_8254_A_2, DM7820_PWM_WIDTH_MASTER_8254_B_0, DM7820_PWM_WIDTH_MASTER_8254_B_1, DM7820_PWM_WIDTH_MASTER_8254_B_2, DM7820_PWM_WIDTH_MASTER_PROG_CLOCK_0, DM7820_PWM_WIDTH_MASTER_PROG_CLOCK_1, DM7820_PWM_WIDTH_MASTER_PROG_CLOCK_2, DM7820_PWM_WIDTH_MASTER_PROG_CLOCK_3, DM7820_PWM_WIDTH_MASTER_STROBE_1, DM7820_PWM_WIDTH_MASTER_STROBE_2, DM7820_PWM_WIDTH_MASTER_INV_STROBE_1, DM7820_PWM_WIDTH_MASTER_INV_STROBE_2 }

  Pulse width modulator width master clocks.

4.60.1 Detailed Description

DM7820_Types_PrgClk_Enumerations

4.60.2 Enumeration Type Documentation

4.60.2.1 enum _dm7820_pwm_output

Pulse width modulator outputs.

Enumerator:

DM7820_PWM_OUTPUT_A PWM output A
DM7820_PWM_OUTPUT_B PWM output B
DM7820_PWM_OUTPUT_C PWM output C
DM7820_PWM_OUTPUT_D PWM output D

Definition at line 1501 of file dm7820_types.h.

4.60.2.2 enum _dm7820_pwm_period_master_clock

Pulse width modulator period master clocks.

Enumerator:

DM7820_PWM_PERIOD_MASTER_25_MHZ 25 MHz clock
DM7820_PWM_PERIOD_MASTER_RESERVED Reserved; do not use
DM7820_PWM_PERIOD_MASTER_8254_A_0 8254 timer/counter A0
DM7820_PWM_PERIOD_MASTER_8254_A_1 8254 timer/counter A1
DM7820_PWM_PERIOD_MASTER_8254_A_2 8254 timer/counter A2
DM7820_PWM_PERIOD_MASTER_8254_B_0 8254 timer/counter B0
DM7820_PWM_PERIOD_MASTER_8254_B_1 8254 timer/counter B1
DM7820_PWM_PERIOD_MASTER_8254_B_2 8254 timer/counter B2
DM7820_PWM_PERIOD_MASTER_PROG_CLOCK_0 Programmable clock 0
DM7820_PWM_PERIOD_MASTER_PROG_CLOCK_1 Programmable clock 1
DM7820_PWM_PERIOD_MASTER_PROG_CLOCK_2 Programmable clock 2
DM7820_PWM_PERIOD_MASTER_PROG_CLOCK_3 Programmable clock 3
DM7820_PWM_PERIOD_MASTER_STROBE_1 Strobe signal 1
DM7820_PWM_PERIOD_MASTER_STROBE_2 Strobe signal 2
DM7820_PWM_PERIOD_MASTER_INV_STROBE_1 Inverted strobe signal 1
DM7820_PWM_PERIOD_MASTER_INV_STROBE_2 Inverted strobe signal 2

Definition at line 1389 of file dm7820_types.h.

4.60.2.3 enum _dm7820_pwm_width_master_clock

Pulse width modulator width master clocks.

Enumerator:

DM7820_PWM_WIDTH_MASTER_25_MHZ 25 MHz clock
DM7820_PWM_WIDTH_MASTER_RESERVED Reserved; do not use
DM7820_PWM_WIDTH_MASTER_8254_A_0 8254 timer/counter A0
DM7820_PWM_WIDTH_MASTER_8254_A_1 8254 timer/counter A1
DM7820_PWM_WIDTH_MASTER_8254_A_2 8254 timer/counter A2

DM7820_PWM_WIDTH_MASTER_8254_B_0 8254 timer/counter B0

DM7820_PWM_WIDTH_MASTER_8254_B_1 8254 timer/counter B1

DM7820_PWM_WIDTH_MASTER_8254_B_2 8254 timer/counter B2

DM7820_PWM_WIDTH_MASTER_PROG_CLOCK_0 Programmable clock 0

DM7820_PWM_WIDTH_MASTER_PROG_CLOCK_1 Programmable clock 1

DM7820_PWM_WIDTH_MASTER_PROG_CLOCK_2 Programmable clock 2

DM7820_PWM_WIDTH_MASTER_PROG_CLOCK_3 Programmable clock 3

DM7820_PWM_WIDTH_MASTER_STROBE_1 Strobe signal 1

DM7820_PWM_WIDTH_MASTER_STROBE_2 Strobe signal 2

DM7820_PWM_WIDTH_MASTER_INV_STROBE_1 Inverted strobe signal 1

DM7820_PWM_WIDTH_MASTER_INV_STROBE_2 Inverted strobe signal 2

Definition at line 1540 of file dm7820_types.h.

4.61 DM7820 type incremental encoder enumerations

Typedefs

- typedef enum _dm7820_incenc_encoder dm7820_incenc_encoder

  Incremental encoder type.
- typedef enum _dm7820_incenc_master_clock dm7820_incenc_master_clock

  Incremental encoder master clock type.
- typedef enum _dm7820_incenc_input_mode dm7820_incenc_input_mode

  Incremental encoder input mode type.
- typedef enum _dm7820_incenc_channel_mode dm7820_incenc_channel_mode

  Incremental encoder channel mode type.
- typedef enum _dm7820_incenc_phase_transition dm7820_incenc_phase_transition

  Incremental encoder phase filter transition type.
- typedef enum _dm7820_incenc_channel dm7820_incenc_channel

  Incremental encoder channel type.
- typedef enum _dm7820_incenc_status_condition dm7820_incenc_status_condition

  Incremental encoder status condition type.

Enumerations

- enum _dm7820_incenc_encoder { DM7820_INCENC_ENCODER_0 = 0, DM7820_INCENC_ENCODER_1 }

  Incremental encoders.
- enum _dm7820_incenc_master_clock { DM7820_INCENC_MASTER_25_MHZ = 0, DM7820_INCENC_MASTER_RESERVED, DM7820_INCENC_MASTER_8254_A_0, DM7820_INCENC_MASTER_8254_A_1, DM7820_INCENC_MASTER_8254_A_2, DM7820_INCENC_MASTER_8254_B_0, DM7820_INCENC_MASTER_8254_B_1, DM7820_INCENC_MASTER_8254_B_2, DM7820_INCENC_MASTER_PROG_CLOCK_0, DM7820_INCENC_MASTER_PROG_CLOCK_1, DM7820_INCENC_MASTER_PROG_CLOCK_2, DM7820_INCENC_MASTER_PROG_CLOCK_3, DM7820_INCENC_MASTER_STROBE_1, DM7820_INCENC_MASTER_STROBE_2, DM7820_INCENC_MASTER_INV_STROBE_1, DM7820_INCENC_MASTER_INV_STROBE_2 }

  Incremental encoder master clocks.
- enum _dm7820_incenc_input_mode { DM7820_INCENC_INPUT_SINGLE_ENDED = 0, DM7820_INCENC_INPUT_DIFFERENTIAL }

  Incremental encoder input modes.
- enum _dm7820_incenc_channel_mode { DM7820_INCENC_CHANNEL_INDEPENDENT = 0, DM7820_INCENC_CHANNEL_JOINED }

  Incremental encoder channel modes.
- enum _dm7820_incenc_phase_transition { DM7820_INCENC_PHASE_BA_00_TO_01_UP = 0, DM7820_INCENC_PHASE_BA_01_TO_11_UP, DM7820_INCENC_PHASE_BA_11_TO_10_UP, DM7820_INCENC_PHASE_BA_10_TO_00_UP, DM7820_INCENC_PHASE_BA_01_TO_00_DOWN, DM7820_INCENC_PHASE_BA_11_TO_01_DOWN, DM7820_INCENC_PHASE_BA_10_TO_11_DOWN, DM7820_INCENC_PHASE_BA_00_TO_10_DOWN }

  Incremental encoder phase filter transitions.
- enum _dm7820_incenc_channel { DM7820_INCENC_CHANNEL_A = 0, DM7820_INCENC_CHANNEL_B }

Incremental encoder channels.

- enum _dm7820_incenc_status_condition { DM7820_INCENC_STATUS_CHANNEL_A_POSITIVE_ROLLOVER = 0, DM7820_INCENC_STATUS_CHANNEL_A_NEGATIVE_ROLLOVER, DM7820_INCENC_STATUS_CHANNEL_B_POSITIVE_ROLLOVER, DM7820_INCENC_STATUS_CHANNEL_B_NEGATIVE_ROLLOVER }

  Incremental encoder status conditions.

4.61.1 Detailed Description

DM7820_Types_PWM_Enumerations

4.61.2 Enumeration Type Documentation

4.61.2.1 enum _dm7820_incenc_channel

Incremental encoder channels.

Enumerator:

DM7820_INCENC_CHANNEL_A Channel A

DM7820_INCENC_CHANNEL_B Channel B

Definition at line 1921 of file dm7820_types.h.

4.61.2.2 enum _dm7820_incenc_channel_mode

Incremental encoder channel modes.

Enumerator:

DM7820_INCENC_CHANNEL_INDEPENDENT Independent 16-bit channels

DM7820_INCENC_CHANNEL_JOINED Channels joined into single 32-bit channel

Definition at line 1830 of file dm7820_types.h.

4.61.2.3 enum _dm7820_incenc_encoder

Incremental encoders.

Enumerator:

DM7820_INCENC_ENCODER_0 Incremental encoder 0

DM7820_INCENC_ENCODER_1 Incremental encoder 1

Definition at line 1665 of file dm7820_types.h.

4.61.2.4 enum _dm7820_incenc_input_mode

Incremental encoder input modes.

Enumerator:

DM7820_INCENC_INPUT_SINGLE_ENDED Single ended

DM7820_INCENC_INPUT_DIFFERENTIAL Pseudo differential

Definition at line 1803 of file dm7820_types.h.

4.61.2.5 enum _dm7820_incenc_master_clock

Incremental encoder master clocks.

Enumerator:

DM7820_INCENC_MASTER_25_MHZ 25 MHz clock

DM7820_INCENC_MASTER_RESERVED Reserved; do not use

DM7820_INCENC_MASTER_8254_A_0 8254 timer/counter A0

DM7820_INCENC_MASTER_8254_A_1 8254 timer/counter A1

DM7820_INCENC_MASTER_8254_A_2 8254 timer/counter A2

DM7820_INCENC_MASTER_8254_B_0 8254 timer/counter B0

DM7820_INCENC_MASTER_8254_B_1 8254 timer/counter B1

DM7820_INCENC_MASTER_8254_B_2 8254 timer/counter B2

DM7820_INCENC_MASTER_PROG_CLOCK_0 Programmable clock 0

DM7820_INCENC_MASTER_PROG_CLOCK_1 Programmable clock 1

DM7820_INCENC_MASTER_PROG_CLOCK_2 Programmable clock 2

DM7820_INCENC_MASTER_PROG_CLOCK_3 Programmable clock 3

DM7820_INCENC_MASTER_STROBE_1 Strobe signal 1

DM7820_INCENC_MASTER_STROBE_2 Strobe signal 2

DM7820_INCENC_MASTER_INV_STROBE_1 Inverted strobe signal 1

DM7820_INCENC_MASTER_INV_STROBE_2 Inverted strobe signal 2

Definition at line 1692 of file dm7820_types.h.

4.61.2.6 enum _dm7820_incenc_phase_transition

Incremental encoder phase filter transitions.

Enumerator:

DM7820_INCENC_PHASE_BA_00_TO_01_UP Inputs B/A transition from 0/0 to 0/1 when counting up

DM7820_INCENC_PHASE_BA_01_TO_11_UP Inputs B/A transition from 0/1 to 1/1 when counting up

DM7820_INCENC_PHASE_BA_11_TO_10_UP Inputs B/A transition from 1/1 to 1/0 when counting up

DM7820_INCENC_PHASE_BA_10_TO_00_UP Inputs B/A transition from 1/0 to 0/0 when counting up

DM7820_INCENC_PHASE_BA_01_TO_00_DOWN Inputs B/A transition from 0/1 to 0/0 when counting down

DM7820_INCENC_PHASE_BA_11_TO_01_DOWN Inputs B/A transition from 1/1 to 0/1 when counting down

DM7820_INCENC_PHASE_BA_10_TO_11_DOWN Inputs B/A transition from 1/0 to 1/1 when counting down

DM7820_INCENC_PHASE_BA_00_TO_10_DOWN Inputs B/A transition from 0/0 to 1/0 when counting down

Definition at line 1857 of file dm7820_types.h.

4.61.2.7 enum _dm7820_incenc_status_condition

Incremental encoder status conditions.

Enumerator:

DM7820_INCENC_STATUS_CHANNEL_A_POSITIVE_ROLLOVER Channel A positive rollover

DM7820_INCENC_STATUS_CHANNEL_A_NEGATIVE_ROLLOVER Channel A negative rollover

DM7820_INCENC_STATUS_CHANNEL_B_POSITIVE_ROLLOVER Channel B positive rollover

DM7820_INCENC_STATUS_CHANNEL_B_NEGATIVE_ROLLOVER Channel B negative rollover

Definition at line 1948 of file dm7820_types.h.

4.62 DM7820 type FIFO enumerations

Typedefs

- typedef enum _dm7820_fifo_queue dm7820_fifo_queue

  FIFO type.
- typedef enum _dm7820_fifo_input_clock dm7820_fifo_input_clock

  FIFO input clock type.
- typedef enum _dm7820_fifo_output_clock dm7820_fifo_output_clock

  FIFO output clock type.
- typedef enum _dm7820_fifo_dma_request dm7820_fifo_dma_request

  FIFO DMA request source type.
- typedef enum _dm7820_fifo_data_input dm7820_fifo_data_input

  FIFO data input type.
- typedef enum _dm7820_fifo_status_condition dm7820_fifo_status_condition

  FIFO status condition type.

Enumerations

- enum _dm7820_fifo_queue { DM7820_FIFO_QUEUE_0 = 0, DM7820_FIFO_QUEUE_1 }

  FIFOs.
- enum _dm7820_fifo_input_clock {
  DM7820_FIFO_INPUT_CLOCK_25_MHZ = 0, DM7820_FIFO_INPUT_CLOCK_RESERVED_1, DM7820_FIFO_INPUT_CLOCK_8254_A_0, DM7820_FIFO_INPUT_CLOCK_8254_A_1,
  DM7820_FIFO_INPUT_CLOCK_8254_A_2, DM7820_FIFO_INPUT_CLOCK_8254_B_0, DM7820_FIFO_INPUT_CLOCK_8254_B_1, DM7820_FIFO_INPUT_CLOCK_8254_B_2,
  DM7820_FIFO_INPUT_CLOCK_PROG_CLOCK_0, DM7820_FIFO_INPUT_CLOCK_PROG_CLOCK_1, DM7820_FIFO_INPUT_CLOCK_PROG_CLOCK_2, DM7820_FIFO_INPUT_CLOCK_PROG_CLOCK_3,
  DM7820_FIFO_INPUT_CLOCK_STROBE_1, DM7820_FIFO_INPUT_CLOCK_STROBE_2, DM7820_FIFO_INPUT_CLOCK_INV_STROBE_1, DM7820_FIFO_INPUT_CLOCK_INV_STROBE_2,
  DM7820_FIFO_INPUT_CLOCK_ADVANCED_INT_0, DM7820_FIFO_INPUT_CLOCK_ADVANCED_INT_1, DM7820_FIFO_INPUT_CLOCK_8254_INT, DM7820_FIFO_INPUT_CLOCK_RESERVED_2,
  DM7280_FIFO_INPUT_CLOCK_INC_ENCODER_0_INT, DM7280_FIFO_INPUT_CLOCK_INC_ENCODER_1_INT, DM7820_FIFO_INPUT_CLOCK_RESERVED_3, DM7820_FIFO_INPUT_CLOCK_RESERVED_4,
  DM7820_FIFO_INPUT_CLOCK_PWM_0_INT, DM7820_FIFO_INPUT_CLOCK_PWM_1_INT, DM7820_FIFO_INPUT_CLOCK_PROG_CLOCK_0_INT, DM7820_FIFO_INPUT_CLOCK_PROG_CLOCK_1_INT,
  DM7820_FIFO_INPUT_CLOCK_PROG_CLOCK_2_INT, DM7820_FIFO_INPUT_CLOCK_PROG_CLOCK_3_INT, DM7820_FIFO_INPUT_CLOCK_PCI_READ, DM7820_FIFO_INPUT_CLOCK_PCI_WRITE }

  FIFO input clocks.
- enum _dm7820_fifo_output_clock {
  DM7820_FIFO_OUTPUT_CLOCK_25_MHZ = 0, DM7820_FIFO_OUTPUT_CLOCK_RESERVED_1, DM7820_FIFO_OUTPUT_CLOCK_8254_A_0, DM7820_FIFO_OUTPUT_CLOCK_8254_A_1,
  DM7820_FIFO_OUTPUT_CLOCK_8254_A_2, DM7820_FIFO_OUTPUT_CLOCK_8254_B_0, DM7820_FIFO_OUTPUT_CLOCK_8254_B_1, DM7820_FIFO_OUTPUT_CLOCK_8254_B_2,
  DM7820_FIFO_OUTPUT_CLOCK_PROG_CLOCK_0, DM7820_FIFO_OUTPUT_CLOCK_PROG_CLOCK_1, DM7820_FIFO_OUTPUT_CLOCK_PROG_CLOCK_2, DM7820_FIFO_OUTPUT_CLOCK_PROG_CLOCK_3,
  DM7820_FIFO_OUTPUT_CLOCK_STROBE_1, DM7820_FIFO_OUTPUT_CLOCK_STROBE_2, DM7820-

_FIFO_OUTPUT_CLOCK_INV_STROBE_1, DM7820_FIFO_OUTPUT_CLOCK_INV_STROBE_2, DM7820_FIFO_OUTPUT_CLOCK_ADVANCED_INT_0, DM7820_FIFO_OUTPUT_CLOCK_ADVANCED_INT_1, DM7820_FIFO_OUTPUT_CLOCK_8254_INT, DM7820_FIFO_OUTPUT_CLOCK_RESERVED_2, DM7280_FIFO_OUTPUT_CLOCK_INC_ENCODER_0_INT, DM7280_FIFO_OUTPUT_CLOCK_INC_ENCODER_1_INT, DM7820_FIFO_OUTPUT_CLOCK_RESERVED_3, DM7820_FIFO_OUTPUT_CLOCK_RESERVED_4, DM7820_FIFO_OUTPUT_CLOCK_PWM_0_INT, DM7820_FIFO_OUTPUT_CLOCK_PWM_1_INT, DM7820_FIFO_OUTPUT_CLOCK_PROG_CLOCK_0_INT, DM7820_FIFO_OUTPUT_CLOCK_PROG_CLOCK_1_INT, DM7820_FIFO_OUTPUT_CLOCK_PROG_CLOCK_2_INT, DM7820_FIFO_OUTPUT_CLOCK_PROG_CLOCK_3_INT, DM7820_FIFO_OUTPUT_CLOCK_PCI_READ, DM7820_FIFO_OUTPUT_CLOCK_PCI_WRITE }

FIFO output clocks.

- enum _dm7820_fifo_dma_request { DM7820_FIFO_DMA_REQUEST_READ = 0, DM7820_FIFO_DMA_REQUEST_NOT_EMPTY, DM7820_FIFO_DMA_REQUEST_WRITE, DM7820_FIFO_DMA_REQUEST_NOT_FULL }

  FIFO DMA request sources.

- enum _dm7820_fifo_data_input { DM7820_FIFO_0_DATA_INPUT_PCI_DATA = 0, DM7820_FIFO_0_DATA_INPUT_PORT_0, DM7820_FIFO_0_DATA_INPUT_PORT_2, DM7820_FIFO_0_DATA_INPUT_FIFO_0_OUTPUT, DM7820_FIFO_1_DATA_INPUT_PCI_DATA, DM7820_FIFO_1_DATA_INPUT_PORT_1, DM7820_FIFO_1_DATA_INPUT_INC_ENCODER_1_A, DM7820_FIFO_1_DATA_INPUT_INC_ENCODER_1_B }

  FIFO data inputs.

- enum _dm7820_fifo_status_condition { DM7820_FIFO_STATUS_READ_REQUEST = 0, DM7820_FIFO_STATUS_WRITE_REQUEST, DM7820_FIFO_STATUS_FULL, DM7820_FIFO_STATUS_EMPTY, DM7820_FIFO_STATUS_OVERFLOW, DM7820_FIFO_STATUS_UNDERFLOW }

  FIFO status conditions.

4.62.1 Detailed Description

DM7820_Types_IncEnc_Enumerations

4.62.2 Enumeration Type Documentation

4.62.2.1 enum _dm7820_fifo_data_input

FIFO data inputs.

Enumerator:

DM7820_FIFO_0_DATA_INPUT_PCI_DATA FIFO 0 data input is PCI data

DM7820_FIFO_0_DATA_INPUT_PORT_0 FIFO 0 data input is digital I/O port 0

DM7820_FIFO_0_DATA_INPUT_PORT_2 FIFO 0 data input is digital I/O port 2

DM7820_FIFO_0_DATA_INPUT_FIFO_0_OUTPUT FIFO 0 data input is FIFO 0 output

DM7820_FIFO_1_DATA_INPUT_PCI_DATA FIFO 1 data input is PCI data

DM7820_FIFO_1_DATA_INPUT_PORT_1 FIFO 1 data input is digital I/O port 1

DM7820_FIFO_1_DATA_INPUT_INC_ENCODER_1_A FIFO 1 data input is incremental encoder 1 channel A counter value

DM7820_FIFO_1_DATA_INPUT_INC_ENCODER_1_B FIFO 1 data input is incremental encoder 1 channel B counter value

Definition at line 2481 of file dm7820_types.h.

4.62.2.2 enum _dm7820_fifo_dma_request

FIFO DMA request sources.

Enumerator:

DM7820_FIFO_DMA_REQUEST_READ Read request
DM7820_FIFO_DMA_REQUEST_NOT_EMPTY Not empty
DM7820_FIFO_DMA_REQUEST_WRITE Write request
DM7820_FIFO_DMA_REQUEST_NOT_FULL Not full

Definition at line 2442 of file dm7820_types.h.

4.62.2.3 enum _dm7820_fifo_input_clock

FIFO input clocks.

Enumerator:

DM7820_FIFO_INPUT_CLOCK_25_MHZ 25 MHz clock
DM7820_FIFO_INPUT_CLOCK_RESERVED_1 Reserved; do not use
DM7820_FIFO_INPUT_CLOCK_8254_A_0 8254 timer/counter A0
DM7820_FIFO_INPUT_CLOCK_8254_A_1 8254 timer/counter A1
DM7820_FIFO_INPUT_CLOCK_8254_A_2 8254 timer/counter A2
DM7820_FIFO_INPUT_CLOCK_8254_B_0 8254 timer/counter B0
DM7820_FIFO_INPUT_CLOCK_8254_B_1 8254 timer/counter B1
DM7820_FIFO_INPUT_CLOCK_8254_B_2 8254 timer/counter B2
DM7820_FIFO_INPUT_CLOCK_PROG_CLOCK_0 Programmable clock 0
DM7820_FIFO_INPUT_CLOCK_PROG_CLOCK_1 Programmable clock 1
DM7820_FIFO_INPUT_CLOCK_PROG_CLOCK_2 Programmable clock 2
DM7820_FIFO_INPUT_CLOCK_PROG_CLOCK_3 Programmable clock 3
DM7820_FIFO_INPUT_CLOCK_STROBE_1 Strobe signal 1
DM7820_FIFO_INPUT_CLOCK_STROBE_2 Strobe signal 2
DM7820_FIFO_INPUT_CLOCK_INV_STROBE_1 Inverted strobe signal 1
DM7820_FIFO_INPUT_CLOCK_INV_STROBE_2 Inverted strobe signal 2
DM7820_FIFO_INPUT_CLOCK_ADVANCED_INT_0 Advanced interrupt 0
DM7820_FIFO_INPUT_CLOCK_ADVANCED_INT_1 Advanced interrupt 1
DM7820_FIFO_INPUT_CLOCK_8254_INT 8254 timer/counter interrupt
DM7820_FIFO_INPUT_CLOCK_RESERVED_2 Reserved; do not use
DM7280_FIFO_INPUT_CLOCK_INC_ENCODER_0_INT Incremental encoder 0 interrupt
DM7280_FIFO_INPUT_CLOCK_INC_ENCODER_1_INT Incremental encoder 1 interrupt
DM7820_FIFO_INPUT_CLOCK_RESERVED_3 Reserved; do not use
DM7820_FIFO_INPUT_CLOCK_RESERVED_4 Reserved; do not use
DM7820_FIFO_INPUT_CLOCK_PWM_0_INT PWM 0 interrupt
DM7820_FIFO_INPUT_CLOCK_PWM_1_INT PWM 1 interrupt
DM7820_FIFO_INPUT_CLOCK_PROG_CLOCK_0_INT Programmable clock 0 interrupt
DM7820_FIFO_INPUT_CLOCK_PROG_CLOCK_1_INT Programmable clock 1 interrupt
DM7820_FIFO_INPUT_CLOCK_PROG_CLOCK_2_INT Programmable clock 2 interrupt
DM7820_FIFO_INPUT_CLOCK_PROG_CLOCK_3_INT Programmable clock 3 interrupt
DM7820_FIFO_INPUT_CLOCK_PCI_READ PCI read from FIFO
DM7820_FIFO_INPUT_CLOCK_PCI_WRITE PCI write to FIFO

Definition at line 2028 of file dm7820_types.h.

4.62.2.4 enum _dm7820_fifo_output_clock

FIFO output clocks.

Enumerator:

DM7820_FIFO_OUTPUT_CLOCK_25_MHZ 25 MHz clock
DM7820_FIFO_OUTPUT_CLOCK_RESERVED_1 Reserved; do not use
DM7820_FIFO_OUTPUT_CLOCK_8254_A_0 8254 timer/counter A0
DM7820_FIFO_OUTPUT_CLOCK_8254_A_1 8254 timer/counter A1
DM7820_FIFO_OUTPUT_CLOCK_8254_A_2 8254 timer/counter A2
DM7820_FIFO_OUTPUT_CLOCK_8254_B_0 8254 timer/counter B0
DM7820_FIFO_OUTPUT_CLOCK_8254_B_1 8254 timer/counter B1
DM7820_FIFO_OUTPUT_CLOCK_8254_B_2 8254 timer/counter B2
DM7820_FIFO_OUTPUT_CLOCK_PROG_CLOCK_0 Programmable clock 0
DM7820_FIFO_OUTPUT_CLOCK_PROG_CLOCK_1 Programmable clock 1
DM7820_FIFO_OUTPUT_CLOCK_PROG_CLOCK_2 Programmable clock 2
DM7820_FIFO_OUTPUT_CLOCK_PROG_CLOCK_3 Programmable clock 3
DM7820_FIFO_OUTPUT_CLOCK_STROBE_1 Strobe signal 1
DM7820_FIFO_OUTPUT_CLOCK_STROBE_2 Strobe signal 2
DM7820_FIFO_OUTPUT_CLOCK_INV_STROBE_1 Inverted strobe signal 1
DM7820_FIFO_OUTPUT_CLOCK_INV_STROBE_2 Inverted strobe signal 2
DM7820_FIFO_OUTPUT_CLOCK_ADVANCED_INT_0 Advanced interrupt 0
DM7820_FIFO_OUTPUT_CLOCK_ADVANCED_INT_1 Advanced interrupt 1
DM7820_FIFO_OUTPUT_CLOCK_8254_INT 8254 timer/counter interrupt
DM7820_FIFO_OUTPUT_CLOCK_RESERVED_2 Reserved; do not use
DM7280_FIFO_OUTPUT_CLOCK_INC_ENCODER_0_INT Incremental encoder 0 interrupt
DM7280_FIFO_OUTPUT_CLOCK_INC_ENCODER_1_INT Incremental encoder 1 interrupt
DM7820_FIFO_OUTPUT_CLOCK_RESERVED_3 Reserved; do not use
DM7820_FIFO_OUTPUT_CLOCK_RESERVED_4 Reserved; do not use
DM7820_FIFO_OUTPUT_CLOCK_PWM_0_INT PWM 0 interrupt
DM7820_FIFO_OUTPUT_CLOCK_PWM_1_INT PWM 1 interrupt
DM7820_FIFO_OUTPUT_CLOCK_PROG_CLOCK_0_INT Programmable clock 0 interrupt
DM7820_FIFO_OUTPUT_CLOCK_PROG_CLOCK_1_INT Programmable clock 1 interrupt
DM7820_FIFO_OUTPUT_CLOCK_PROG_CLOCK_2_INT Programmable clock 2 interrupt
DM7820_FIFO_OUTPUT_CLOCK_PROG_CLOCK_3_INT Programmable clock 3 interrupt
DM7820_FIFO_OUTPUT_CLOCK_PCI_READ PCI read from FIFO
DM7820_FIFO_OUTPUT_CLOCK_PCI_WRITE PCI write to FIFO

Definition at line 2235 of file dm7820_types.h.

4.62.2.5 enum _dm7820_fifo_queue

FIFOs.

Enumerator:

DM7820_FIFO_QUEUE_0 FIFO 0
DM7820_FIFO_QUEUE_1 FIFO 1

Definition at line 2001 of file dm7820_types.h.

4.62.2.6 enum _dm7820_fifo_status_condition

FIFO status conditions.

Enumerator:

DM7820_FIFO_STATUS_READ_REQUEST FIFO read request

DM7820_FIFO_STATUS_WRITE_REQUEST FIFO read request

DM7820_FIFO_STATUS_FULL FIFO full

DM7820_FIFO_STATUS_EMPTY FIFO empty

DM7820_FIFO_STATUS_OVERFLOW FIFO overflow

DM7820_FIFO_STATUS_UNDERFLOW FIFO underflow

Definition at line 2544 of file dm7820_types.h.

4.63 DM7820 type advanced interrupt enumerations

Typedefs

- typedef enum _dm7820_advint_interrupt dm7820_advint_interrupt

  Advanced interrupt type.

- typedef enum _dm7820_advint_mode dm7820_advint_mode

  Advanced interrupt mode type.

- typedef enum _dm7820_advint_master_clock dm7820_advint_master_clock

  Advanced interrupt master clock type.

Enumerations

- enum _dm7820_advint_interrupt { DM7820_ADVINT_INTERRUPT_0 = 0, DM7820_ADVINT_INTERRUPT_1 }

  Advanced interrupts.

- enum _dm7820_advint_mode { DM7820_ADVINT_MODE_DISABLED = 0, DM7820_ADVINT_MODE_STROBE, DM7820_ADVINT_MODE_MATCH, DM7820_ADVINT_MODE_EVENT }

  Advanced interrupt modes.

- enum _dm7820_advint_master_clock { DM7820_ADVINT_MASTER_25_MHZ = 0, DM7820_ADVINT_MASTER_RESERVED, DM7820_ADVINT_MASTER_8254_A_0, DM7820_ADVINT_MASTER_8254_A_1, DM7820_ADVINT_MASTER_8254_A_2, DM7820_ADVINT_MASTER_8254_B_0, DM7820_ADVINT_MASTER_8254_B_1, DM7820_ADVINT_MASTER_8254_B_2, DM7820_ADVINT_MASTER_PROG_CLOCK_0, DM7820_ADVINT_MASTER_PROG_CLOCK_1, DM7820_ADVINT_MASTER_PROG_CLOCK_2, DM7820_ADVINT_MASTER_PROG_CLOCK_3, DM7820_ADVINT_MASTER_STROBE_1, DM7820_ADVINT_MASTER_STROBE_2, DM7820_ADVINT_MASTER_INV_STROBE_1, DM7820_ADVINT_MASTER_INV_STROBE_2 }

  Advanced interrupt master clocks.

4.63.1 Detailed Description

DM7820_Types_FIFO_Enumerations

4.63.2 Enumeration Type Documentation

4.63.2.1 enum _dm7820_advint_interrupt

Advanced interrupts.

Enumerator:

DM7820_ADVINT_INTERRUPT_0 Advanced interrupt 0

DM7820_ADVINT_INTERRUPT_1 Advanced interrupt 1

Definition at line 2608 of file dm7820_types.h.

4.63.2.2 enum _dm7820_advint_master_clock

Advanced interrupt master clocks.

Enumerator:

DM7820_ADVINT_MASTER_25_MHZ 25 MHz clock
DM7820_ADVINT_MASTER_RESERVED Reserved; do not use
DM7820_ADVINT_MASTER_8254_A_0 8254 timer/counter A0
DM7820_ADVINT_MASTER_8254_A_1 8254 timer/counter A1
DM7820_ADVINT_MASTER_8254_A_2 8254 timer/counter A2
DM7820_ADVINT_MASTER_8254_B_0 8254 timer/counter B0
DM7820_ADVINT_MASTER_8254_B_1 8254 timer/counter B1
DM7820_ADVINT_MASTER_8254_B_2 8254 timer/counter B2
DM7820_ADVINT_MASTER_PROG_CLOCK_0 Programmable clock 0
DM7820_ADVINT_MASTER_PROG_CLOCK_1 Programmable clock 1
DM7820_ADVINT_MASTER_PROG_CLOCK_2 Programmable clock 2
DM7820_ADVINT_MASTER_PROG_CLOCK_3 Programmable clock 3
DM7820_ADVINT_MASTER_STROBE_1 Strobe signal 1
DM7820_ADVINT_MASTER_STROBE_2 Strobe signal 2
DM7820_ADVINT_MASTER_INV_STROBE_1 Inverted strobe signal 1
DM7820_ADVINT_MASTER_INV_STROBE_2 Inverted strobe signal 2

Definition at line 2674 of file dm7820_types.h.

4.63.2.3 enum _dm7820_advint_mode

Advanced interrupt modes.

Enumerator:

DM7820_ADVINT_MODE_DISABLED Disabled
DM7820_ADVINT_MODE_STROBE Strobe mode
DM7820_ADVINT_MODE_MATCH Match mode
DM7820_ADVINT_MODE_EVENT Event mode

Definition at line 2635 of file dm7820_types.h.

4.64 DM7820 type interrupt enumerations

Data Structures

- struct _dm7820_interrupt_info

  Interrupt source information.

Typedefs

- typedef enum _dm7820_interrupt_source dm7820_interrupt_source

  Interrupt source type.
- typedef enum _dm7820_minor_interrupt_register dm7820_minor_interrupt_register

  Minor interrupt control/status register type.
- typedef struct _dm7820_interrupt_info dm7820_interrupt_info

  Interrupt source information type.

Enumerations

- enum _dm7820_interrupt_source {
  DM7820_INTERRUPT_ADVINT_0 = 0, DM7820_INTERRUPT_ADVINT_1, DM7820_INTERRUPT_FIFO_0_EMPTY, DM7820_INTERRUPT_FIFO_0_FULL,
  DM7820_INTERRUPT_FIFO_0_OVERFLOW, DM7820_INTERRUPT_FIFO_0_READ_REQUEST, DM7820_INTERRUPT_FIFO_0_UNDERFLOW, DM7820_INTERRUPT_FIFO_0_WRITE_REQUEST,
  DM7820_INTERRUPT_FIFO_1_EMPTY, DM7820_INTERRUPT_FIFO_1_FULL, DM7820_INTERRUPT_FIFO_1_OVERFLOW, DM7820_INTERRUPT_FIFO_1_READ_REQUEST,
  DM7820_INTERRUPT_FIFO_1_UNDERFLOW, DM7820_INTERRUPT_FIFO_1_WRITE_REQUEST, DM7820_INTERRUPT_INCENC_0_CHANNEL_A_NEGATIVE_ROLLOVER, DM7820_INTERRUPT_INCENC_0_CHANNEL_A_POSITIVE_ROLLOVER,
  DM7820_INTERRUPT_INCENC_0_CHANNEL_B_NEGATIVE_ROLLOVER, DM7820_INTERRUPT_INCENC_0_CHANNEL_B_POSITIVE_ROLLOVER, DM7820_INTERRUPT_INCENC_1_CHANNEL_A_NEGATIVE_ROLLOVER, DM7820_INTERRUPT_INCENC_1_CHANNEL_A_POSITIVE_ROLLOVER,
  DM7820_INTERRUPT_INCENC_1_CHANNEL_B_NEGATIVE_ROLLOVER, DM7820_INTERRUPT_INCENC_1_CHANNEL_B_POSITIVE_ROLLOVER, DM7820_INTERRUPT_PRGCLK_0, DM7820_INTERRUPT_PRGCLK_1,
  DM7820_INTERRUPT_PRGCLK_2, DM7820_INTERRUPT_PRGCLK_3, DM7820_INTERRUPT_PWM_0, DM7820_INTERRUPT_PWM_1,
  DM7820_INTERRUPT_TMRCTR_A_0, DM7820_INTERRUPT_TMRCTR_A_1, DM7820_INTERRUPT_TMRCTR_A_2, DM7820_INTERRUPT_TMRCTR_B_0,
  DM7820_INTERRUPT_TMRCTR_B_1, DM7820_INTERRUPT_TMRCTR_B_2, DM7820_INTERRUPT_FIFO_0_DMA_DONE, DM7820_INTERRUPT_FIFO_1_DMA_DONE,
  DM7820_INTERRUPT_NONE }

  Interrupt sources.
- enum _dm7820_minor_interrupt_register {
  DM7820_MINOR_INT_REG_FIFO_0_INT = 0, DM7820_MINOR_INT_REG_FIFO_1_INT, DM7820_MINOR_INT_REG_INCENC_0_INT, DM7820_MINOR_INT_REG_INCENC_1_INT,
  DM7820_MINOR_INT_REG_TMRCTR_INT, DM7820_MINOR_INT_REG_NONE }

  Minor interrupt control/status registers.

4.64.1 Detailed Description

DM7820_Types_AdvInt_Enumerations

4.64.2 Enumeration Type Documentation

4.64.2.1 enum _dm7820_interrupt_source

Interrupt sources.

Enumerator:

DM7820_INTERRUPT_ADVINT_0 Advanced interrupt block 0 interrupt

DM7820_INTERRUPT_ADVINT_1 Advanced interrupt block 1 interrupt (1)

DM7820_INTERRUPT_FIFO_0_EMPTY FIFO block FIFO 0 empty interrupt (2)

DM7820_INTERRUPT_FIFO_0_FULL FIFO block FIFO 0 full interrupt (3)

DM7820_INTERRUPT_FIFO_0_OVERFLOW FIFO block FIFO 0 overflow interrupt (4)

DM7820_INTERRUPT_FIFO_0_READ_REQUEST FIFO block FIFO 0 read request interrupt (5)

DM7820_INTERRUPT_FIFO_0_UNDERFLOW FIFO block FIFO 0 underflow interrupt (6)

DM7820_INTERRUPT_FIFO_0_WRITE_REQUEST FIFO block FIFO 0 write request interrupt (7)

DM7820_INTERRUPT_FIFO_1_EMPTY FIFO block FIFO 1 empty interrupt (8)

DM7820_INTERRUPT_FIFO_1_FULL FIFO block FIFO 1 full interrupt (9)

DM7820_INTERRUPT_FIFO_1_OVERFLOW FIFO block FIFO 1 overflow interrupt (10)

DM7820_INTERRUPT_FIFO_1_READ_REQUEST FIFO block FIFO 1 read request interrupt (11)

DM7820_INTERRUPT_FIFO_1_UNDERFLOW FIFO block FIFO 1 underflow interrupt (12)

DM7820_INTERRUPT_FIFO_1_WRITE_REQUEST FIFO block FIFO 1 write request interrupt (13)

DM7820_INTERRUPT_INCENC_0_CHANNEL_A_NEGATIVE_ROLLOVER Incremental encoder block 0 channel A negative rollover interrupt (14)

DM7820_INTERRUPT_INCENC_0_CHANNEL_A_POSITIVE_ROLLOVER Incremental encoder block 0 channel A positive rollover interrupt (15)

DM7820_INTERRUPT_INCENC_0_CHANNEL_B_NEGATIVE_ROLLOVER Incremental encoder block 0 channel B negative rollover interrupt (16)

DM7820_INTERRUPT_INCENC_0_CHANNEL_B_POSITIVE_ROLLOVER Incremental encoder block 0 channel B positive rollover interrupt (17)

DM7820_INTERRUPT_INCENC_1_CHANNEL_A_NEGATIVE_ROLLOVER Incremental encoder block 1 channel A negative rollover interrupt (18)

DM7820_INTERRUPT_INCENC_1_CHANNEL_A_POSITIVE_ROLLOVER Incremental encoder block 1 channel A positive rollover interrupt (19)

DM7820_INTERRUPT_INCENC_1_CHANNEL_B_NEGATIVE_ROLLOVER Incremental encoder block 1 channel B negative rollover interrupt (20)

DM7820_INTERRUPT_INCENC_1_CHANNEL_B_POSITIVE_ROLLOVER Incremental encoder block 1 channel B positive rollover interrupt (21)

DM7820_INTERRUPT_PRGCLK_0 Programmable clock block 0 interrupt (22)

DM7820_INTERRUPT_PRGCLK_1 Programmable clock block 1 interrupt (23)

DM7820_INTERRUPT_PRGCLK_2 Programmable clock block 2 interrupt (24)

DM7820_INTERRUPT_PRGCLK_3 Programmable clock block 3 interrupt (25)

DM7820_INTERRUPT_PWM_0 Pulse width modulator block 0 interrupt (26)

DM7820_INTERRUPT_PWM_1 Pulse width modulator block 1 interrupt (27)

DM7820_INTERRUPT_TMRCTR_A_0 8254 timer/counter A0 interrupt (28)

DM7820_INTERRUPT_TMRCTR_A_1 8254 timer/counter A1 interrupt (29)

DM7820_INTERRUPT_TMRCTR_A_2 8254 timer/counter A2 interrupt (30)

DM7820_INTERRUPT_TMRCTR_B_0 8254 timer/counter B0 interrupt (31)

DM7820_INTERRUPT_TMRCTR_B_1 8254 timer/counter B1 interrupt (32)

DM7820_INTERRUPT_TMRCTR_B_2 8254 timer/counter B2 interrupt (33)

DM7820_INTERRUPT_FIFO_0_DMA_DONE FIFO block FIFO 0 DMA done interrupt. Applications cannot control this interrupt but they can get its status. (34)

DM7820_INTERRUPT_FIFO_1_DMA_DONE FIFO block FIFO 1 DMA done interrupt. Applications cannot control this interrupt but they can get its status. (35)

DM7820_INTERRUPT_NONE Value which indicates no interrupt source. User level ignores this. The kernel uses this in the interrupt handler. This must be the last entry. (36)

Definition at line 2798 of file dm7820_types.h.

4.64.2.2 enum _dm7820_minor_interrupt_register

Minor interrupt control/status registers.

Enumerator:

DM7820_MINOR_INT_REG_FIFO_0_INT FIFO 0 Interrupt Register

DM7820_MINOR_INT_REG_FIFO_1_INT FIFO 1 Interrupt Register

DM7820_MINOR_INT_REG_INCENC_0_INT Incremental Encoder 0 Interrupt Register

DM7820_MINOR_INT_REG_INCENC_1_INT Incremental Encoder 1 Interrupt Register

DM7820_MINOR_INT_REG_TMRCTR_INT 8254 Timer/Counter Interrupt Register

DM7820_MINOR_INT_REG_NONE Value which indicates no minor interrupt register. This must be the last entry.

Definition at line 3038 of file dm7820_types.h.

4.65 DM7820 type definition structures

Data Structures

- struct dm7820_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.
- struct dm7820_interrupt_control

  Structure containing information needed to acknowledge, disable, and enable a particular interrupt source.
- struct dm7820_minor_int_reg_layout

  Minor interrupt register bit layout.

Typedefs

- typedef struct dm7820_pci_access_request dm7820_pci_access_request_t
- typedef struct dm7820_interrupt_control dm7820_interrupt_control_t

  Interrupt control information type.
- typedef struct dm7820_minor_int_reg_layout dm7820_minor_int_reg_layout_t

  Minor interrupt register bit layout type.

4.65.1 Detailed Description

DM7820_Types_Enumerations

4.65.2 Typedef Documentation

4.65.2.1 typedef struct dm7820_pci_access_request dm7820_pci_access_request_t

PCI region access request descriptor type

Definition at line 3191 of file dm7820_types.h.

4.66 DM7820 type definition typedefs

Typedefs

- typedef uint64_t dm7820_int_source_status_t

  Interrupt source status type.

4.66.1 Detailed Description

DM7820_Types_Structures

Chapter 5

Data Structure Documentation

5.1 _dm7820_interrupt_info Struct Reference

Interrupt source information.

`#include <dm7820_types.h>`

Data Fields

- dm7820_interrupt_source source
- int int_remaining
- int int_missed
- int error

5.1.1 Detailed Description

Interrupt source information.

Definition at line 3091 of file dm7820_types.h.

5.1.2 Field Documentation

5.1.2.1 int error

Error Code: 0 = success, -1 = failure

Definition at line 3107 of file dm7820_types.h.

Referenced by ISR().

5.1.2.2 int int_missed

The number of interrupt missed due to a full log in the driver.

Definition at line 3103 of file dm7820_types.h.

5.1.2.3 int int_remaining

The number of logged interrupt in the driver that still need attention

Definition at line 3099 of file dm7820_types.h.

5.1.2.4 dm7820_interrupt_source source

The interrupt sources for the last acknowledged interrupt

Definition at line 3095 of file dm7820_types.h.

Referenced by ISR(), and main().

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

- include/dm7820_types.h

5.2 DM7820_Board_Descriptor Struct Reference

DM7820 board descriptor. This structure holds information about a device needed by the library.

`#include <dm7820_library.h>`

Data Fields

- int file_descriptor
- void(* isr)(dm7820_interrupt_info status)
- pthread_t pid

5.2.1 Detailed Description

DM7820 board descriptor. This structure holds information about a device needed by the library.

Definition at line 214 of file dm7820_library.h.

5.2.2 Field Documentation

5.2.2.1 int file_descriptor

File descriptor for device returned from open()

Definition at line 220 of file dm7820_library.h.

Referenced by main().

5.2.2.2 void(* isr)(dm7820_interrupt_info status)

Function pointer to the user ISR callback function.

Definition at line 226 of file dm7820_library.h.

5.2.2.3 pthread_t pid

Process ID of the child process which will monitor DMA done interrupts.

Definition at line 232 of file dm7820_library.h.

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

- include/dm7820_library.h

5.3 dm7820_device_descriptor Struct Reference

DM7820 device descriptor. This structure holds information about a device needed by the kernel.

`#include <dm7820_driver.h>`

Data Fields

- char device_name [DM7820_DEVICE_NAME_LENGTH]
- dm7820_pci_region_t pci [PCI_ROM_RESOURCE]
- spinlock_t device_lock
- uint8_t reference_count
- unsigned int irq_number
- uint8_t interrupt_occurred
- uint8_t remove_isr_flag
- wait_queue_head_t int_wait_queue
- wait_queue_head_t dma_wait_queue
- dm7820_int_source_status_t int_source_status
- uint8_t dma_initialized [DM7820_FIFO_CHANNELS]
- uint32_t dma_size [DM7820_FIFO_CHANNELS]
- struct list_head dma_buffers_pre_transfer [DM7820_FIFO_CHANNELS]
- struct list_head dma_buffers_post_transfer [DM7820_FIFO_CHANNELS]
- uint8_t dma_in_read_direction [DM7820_FIFO_CHANNELS]
- dm7820_interrupt_source int_status [DM7820_INT_QUEUE_SIZE]
- unsigned int int_queue_in
- unsigned int int_queue_out
- unsigned int int_queue_missed
- unsigned int int_queue_count

5.3.1 Detailed Description

DM7820 device descriptor. This structure holds information about a device needed by the kernel.

Definition at line 305 of file dm7820_driver.h.

5.3.2 Field Documentation

5.3.2.1 spinlock_t device_lock

Concurrency control

Definition at line 324 of file dm7820_driver.h.

5.3.2.2 char device_name[DM7820_DEVICE_NAME_LENGTH]

Device name used when requesting resources; a NUL terminated string of the form rtd-dm7820-x where x is the device minor number.

Definition at line 312 of file dm7820_driver.h.

5.3.2.3 struct list_head dma_buffers_post_transfer[DM7820_FIFO_CHANNELS]

Per-FIFO channel linked list of DMA buffers containing data read from FIFO

Definition at line 397 of file dm7820_driver.h.

5.3.2.4 struct list_head dma_buffers_pre_transfer[DM7820_FIFO_CHANNELS]

Per-FIFO channel linked list of DMA buffers

Definition at line 390 of file dm7820_driver.h.

5.3.2.5 uint8_t dma_in_read_direction[DM7820_FIFO_CHANNELS]

Per-FIFO flag indicating direction of DMA, true if in read and false if in write

Definition at line 404 of file dm7820_driver.h.

5.3.2.6 uint8_t dma_initialized[DM7820_FIFO_CHANNELS]

Per-FIFO channel flag indicating whether or not DMA was initialized. A value of zero means DMA was not initialized. Any other value means DMA was initialized.

Definition at line 378 of file dm7820_driver.h.

5.3.2.7 uint32_t dma_size[DM7820_FIFO_CHANNELS]

Per-FIFO channel DMA transfer size

Definition at line 384 of file dm7820_driver.h.

5.3.2.8 wait_queue_head_t dma_wait_queue

Queue of processes waiting to be woken up when an interrupt occurs

Definition at line 362 of file dm7820_driver.h.

5.3.2.9 unsigned int int_queue_count

Number of interrupts currently in the queue

Definition at line 434 of file dm7820_driver.h.

5.3.2.10 unsigned int int_queue_in

Number of entries in the interrupt status queue

Definition at line 416 of file dm7820_driver.h.

5.3.2.11 unsigned int int_queue_missed

Number of interrupts missed because of a full queue

Definition at line 428 of file dm7820_driver.h.

5.3.2.12 unsigned int int_queue_out

Number of entries read from the interrupt status queue

Definition at line 422 of file dm7820_driver.h.

5.3.2.13 dm7820_int_source_status_t int_source_status

Bit mask indicating status of each interrupt source. A zero in a bit position means the corresponding interrupt source did not occur. A one in a bit position means the corresponding interrupt source did occur.

Definition at line 370 of file dm7820_driver.h.

5.3.2.14 dm7820_interrupt_source int_status[DM7820_INT_QUEUE_SIZE]

Interrupt status queue

Definition at line 410 of file dm7820_driver.h.

5.3.2.15 wait_queue_head_t int_wait_queue

Queue of processes waiting to be woken up when an interrupt occurs

Definition at line 356 of file dm7820_driver.h.

5.3.2.16 uint8_t interrupt_occurred

Flag indicating whether or not an interrupt occurred. Cleared when interrupt status is read. Set by interrupt handler.

Definition at line 344 of file dm7820_driver.h.

5.3.2.17 unsigned int irq_number

IRQ line number

Definition at line 337 of file dm7820_driver.h.

5.3.2.18 dm7820_pci_region_t pci[PCI_ROM_RESOURCE]

Information about each of the standard PCI regions

Definition at line 318 of file dm7820_driver.h.

5.3.2.19 uint8_t reference_count

Number of entities which have the device file open. Used to enforce single open semantics.

Definition at line 331 of file dm7820_driver.h.

5.3.2.20 uint8_t remove_isr_flag

Used to assist poll in shutting down the thread waiting for interrupts

Definition at line 350 of file dm7820_driver.h.

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

- include/dm7820_driver.h

5.4 dm7820_dma_descriptor_t Struct Reference

DM7820 DMA buffer descriptor. This structure holds allocation information for a single DMA buffer.

`#include <dm7820_driver.h>`

Data Fields

- dma_addr_t bus_address
- void * virtual_address

5.4.1 Detailed Description

DM7820 DMA buffer descriptor. This structure holds allocation information for a single DMA buffer.

Definition at line 264 of file dm7820_driver.h.

5.4.2 Field Documentation

5.4.2.1 dma_addr_t bus_address

Bus/physical address

Definition at line 270 of file dm7820_driver.h.

5.4.2.2 void* virtual_address

Virtual address

Definition at line 276 of file dm7820_driver.h.

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

- include/dm7820_driver.h

5.5 dm7820_dma_function_arguments Union Reference

Structure encapsulating arguments to all possible DMA functions.

`#include <dm7820_ioctl.h>`

Data Fields

- dm7820_dma_initialize_arguments_t dma_init

5.5.1 Detailed Description

Structure encapsulating arguments to all possible DMA functions.

Definition at line 236 of file dm7820_ioctl.h.

5.5.2 Field Documentation

5.5.2.1 dm7820_dma_initialize_arguments_t dma_init

DMA initialization

Definition at line 242 of file dm7820_ioctl.h.

Referenced by main().

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

- include/dm7820_ioctl.h

5.6 dm7820_dma_initialize_arguments Struct Reference

Arguments for DMA initialization function.

`#include <dm7820_ioctl.h>`

Data Fields

- uint32_t buffer_count
- uint32_t buffer_size

5.6.1 Detailed Description

Arguments for DMA initialization function.

Definition at line 216 of file dm7820_ioctl.h.

5.6.2 Field Documentation

5.6.2.1 uint32_t buffer_count

Number of DMA buffers to allocate

Definition at line 222 of file dm7820_ioctl.h.

Referenced by main().

5.6.2.2 uint32_t buffer_size

DMA buffer size in bytes

Definition at line 228 of file dm7820_ioctl.h.

Referenced by main().

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

- include/dm7820_ioctl.h

5.7 dm7820_dma_list_item_t Struct Reference

DM7820 DMA buffer list item.

`#include <dm7820_driver.h>`

Data Fields

- struct list_head list
- dm7820_dma_descriptor_t * dma_buffer

5.7.1 Detailed Description

DM7820 DMA buffer list item.

Definition at line 284 of file dm7820_driver.h.

5.7.2 Field Documentation

5.7.2.1 dm7820_dma_descriptor_t* dma_buffer

DMA buffer allocation information

Definition at line 296 of file dm7820_driver.h.

5.7.2.2 struct list_head list

Linked list management

Definition at line 290 of file dm7820_driver.h.

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

- include/dm7820_driver.h

5.8 dm7820_interrupt_control Struct Reference

Structure containing information needed to acknowledge, disable, and enable a particular interrupt source.

`#include <dm7820_types.h>`

Data Fields

- uint16_t int_enable_bit
- uint16_t int_status_bit
- dm7820_minor_interrupt_register minor_reg
- uint16_t minor_enable
- uint16_t minor_status

5.8.1 Detailed Description

Structure containing information needed to acknowledge, disable, and enable a particular interrupt source.

Definition at line 3199 of file dm7820_types.h.

5.8.2 Field Documentation

5.8.2.1 uint16_t int_enable_bit

Interrupt Enable Register mask that indicates which single bit controls the interrupt

Definition at line 3206 of file dm7820_types.h.

5.8.2.2 uint16_t int_status_bit

Interrupt Status Register mask that indicates which single bit gives the status of the interrupt

Definition at line 3213 of file dm7820_types.h.

5.8.2.3 uint16_t minor_enable

Minor interrupt enable register mask that indicates which single bit in the register controls the interrupt

Definition at line 3226 of file dm7820_types.h.

5.8.2.4 dm7820_minor_interrupt_register minor_reg

Minor interrupt register

Definition at line 3219 of file dm7820_types.h.

5.8.2.5 uint16_t minor_status

Minor interrupt enable register mask that indicates which single bit in the register gives the status of the interrupt

Definition at line 3233 of file dm7820_types.h.

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

- include/dm7820_types.h

5.9 dm7820_interrupt_status_source Struct Reference

Interrupt source information for a single Interrupt Status Register bit.

`#include <dm7820_driver.h>`

Data Fields

- dm7820_minor_interrupt_register minor_reg
- dm7820_interrupt_source source
- dm7820_interrupt_source * source_table

5.9.1 Detailed Description

Interrupt source information for a single Interrupt Status Register bit.

Definition at line 449 of file dm7820_driver.h.

5.9.2 Field Documentation

5.9.2.1 dm7820_minor_interrupt_register minor_reg

Minor interrupt register. If there is no minor interrupt register, this will be DM7820_MINOR_INT_REG_NONE.

Definition at line 456 of file dm7820_driver.h.

5.9.2.2 dm7820_interrupt_source source

Interrupt source for register bit. If there is a minor interrupt register, this will be DM7820_INTERRUPT_NONE.

Definition at line 463 of file dm7820_driver.h.

5.9.2.3 dm7820_interrupt_source* source_table

Table of interrupt sources for register bit. If there is no minor interrupt register, this will be NULL.

Definition at line 470 of file dm7820_driver.h.

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

- include/dm7820_driver.h

5.10 dm7820_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 <dm7820_ioctl.h>`

Data Fields

- dm7820_ioctl_region_readwrite_t readwrite
- dm7820_ioctl_region_modify_t modify
- dm7820_ioctl_interrupt_status_t int_status
- dm7820_ioctl_dma_function_t dma_function

5.10.1 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 307 of file dm7820_ioctl.h.

5.10.2 Field Documentation

5.10.2.1 dm7820_ioctl_dma_function_t dma_function

DMA management function

Definition at line 331 of file dm7820_ioctl.h.

Referenced by main().

5.10.2.2 dm7820_ioctl_interrupt_status_t int_status

Get interrupt status

Definition at line 325 of file dm7820_ioctl.h.

Referenced by main().

5.10.2.3 dm7820_ioctl_region_modify_t modify

PCI region read/modify/write

Definition at line 319 of file dm7820_ioctl.h.

Referenced by main().

5.10.2.4 dm7820_ioctl_region_readwrite_t readwrite

PCI region read and write

Definition at line 313 of file dm7820_ioctl.h.

Referenced by main().

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

- include/dm7820_ioctl.h

5.11 dm7820_ioctl_dma_function Struct Reference

ioctl() request structure for performing a DMA function

`#include <dm7820_ioctl.h>`

Data Fields

- void * user_buffer
- uint8_t direction
- uint32_t DMA_Transfer_Size
- uint32_t buffer_address
- uint32_t transfer_size
- dm7820_fifo_queue fifo
- dm7820_dma_manage_function_t function
- dm7820_dma_function_arguments_t arguments

5.11.1 Detailed Description

ioctl() request structure for performing a DMA function

Definition at line 250 of file dm7820_ioctl.h.

5.11.2 Field Documentation

5.11.2.1 dm7820_dma_function_arguments_t arguments

Arguments required by function

Definition at line 298 of file dm7820_ioctl.h.

Referenced by main().

5.11.2.2 uint32_t buffer_address

The address of a DMA buffer

Definition at line 274 of file dm7820_ioctl.h.

5.11.2.3 uint8_t direction

enumeration of the direction mode for the DMA channel

Definition at line 262 of file dm7820_ioctl.h.

5.11.2.4 uint32_t DMA_Transfer_Size

contains the transfer size for the DMA channel

Definition at line 268 of file dm7820_ioctl.h.

5.11.2.5 dm7820_fifo_queue fifo

DMA/FIFO channel to operate upon

Definition at line 286 of file dm7820_ioctl.h.

Referenced by main().

5.11.2.6 dm7820_dma_manage_function_t function

DMA function to perform

Definition at line 292 of file dm7820_ioctl.h.

Referenced by main().

5.11.2.7 uint32_t transfer_size

Size of the DMA transfer

Definition at line 280 of file dm7820_ioctl.h.

Referenced by main().

5.11.2.8 void* user_buffer

Buffer for data coming from user.

Definition at line 256 of file dm7820_ioctl.h.

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

- include/dm7820_ioctl.h

5.12 dm7820_ioctl_interrupt_status Struct Reference

ioctl() request structure for getting interrupt status and waiting for an interrupt to occur

`#include <dm7820_ioctl.h>`

Data Fields

- uint8_t wait_for_interrupt
- dm7820_interrupt_info int_source_info

5.12.1 Detailed Description

ioctl() request structure for getting interrupt status and waiting for an interrupt to occur

Definition at line 183 of file dm7820_ioctl.h.

5.12.2 Field Documentation

5.12.2.1 dm7820_interrupt_info int_source_info

Bit mask indicating status of each interrupt source. A zero in a bit position means the corresponding interrupt source did not occur. A one in a bit position means the corresponding interrupt source did occur.

Definition at line 200 of file dm7820_ioctl.h.

5.12.2.2 uint8_t wait_for_interrupt

Flag indicating whether or not to wait for an interrupt to occur before returning status. A value of zero means do not wait for an interrupt and just return whatever status is currently available. Any other value means wait for an interrupt to occur before returning status.

Definition at line 192 of file dm7820_ioctl.h.

Referenced by main().

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

- include/dm7820_ioctl.h

5.13 dm7820_ioctl_region_modify Struct Reference

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

```
#include <dm7820_ioctl.h>
```

Data Fields

- dm7820_pci_access_request_t access
- union {
  uint8_t mask8
  uint16_t mask16
  uint32_t mask32
  } mask

5.13.1 Detailed Description

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

Definition at line 128 of file dm7820_ioctl.h.

5.13.2 Field Documentation

5.13.2.1 dm7820_pci_access_request_t access

PCI region access request

Definition at line 134 of file dm7820_ioctl.h.

Referenced by main().

5.13.2.2 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.

Referenced by main().

5.13.2.3 uint16_t mask16

Mask for 16-bit operations

Definition at line 160 of file dm7820_ioctl.h.

Referenced by main().

5.13.2.4 uint32_t mask32

Mask for 32-bit operations

Definition at line 166 of file dm7820_ioctl.h.

Referenced by main().

5.13.2.5 uint8_t mask8

Mask for 8-bit operations

Definition at line 154 of file dm7820_ioctl.h.

Referenced by main().

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

- include/dm7820_ioctl.h

5.14 dm7820_ioctl_region_readwrite Struct Reference

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

`#include <dm7820_ioctl.h>`

Data Fields

- dm7820_pci_access_request_t access

5.14.1 Detailed Description

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

Definition at line 108 of file dm7820_ioctl.h.

5.14.2 Field Documentation

5.14.2.1 dm7820_pci_access_request_t access

PCI region access request

Definition at line 114 of file dm7820_ioctl.h.

Referenced by main().

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

- include/dm7820_ioctl.h

5.15 dm7820_minor_int_reg_layout Struct Reference

Minor interrupt register bit layout.

`#include <dm7820_types.h>`

Data Fields

- uint16_t offset
- uint16_t enable_mask
- uint16_t reserved_mask
- uint16_t status_mask

5.15.1 Detailed Description

Minor interrupt register bit layout.

Definition at line 3248 of file dm7820_types.h.

5.15.2 Field Documentation

5.15.2.1 uint16_t enable_mask

Bit mask that indicates which register bits are interrupt enable bits. A zero in a bit position means the corresponding register bit does not control an interrupt. A one in a bit position means the corresponding register bit controls an interrupt.

Definition at line 3263 of file dm7820_types.h.

5.15.2.2 uint16_t offset

BAR2 register offset

Definition at line 3254 of file dm7820_types.h.

5.15.2.3 uint16_t reserved_mask

Bit mask that indicates which register bits are reserved. A zero in a bit position means the corresponding register bit is not reserved. A one in a bit position means the corresponding register bit is reserved.

Definition at line 3271 of file dm7820_types.h.

5.15.2.4 uint16_t status_mask

Bit mask that indicates which register bits are interrupt status bits. A zero in a bit position means the corresponding register bit does not indicate interrupt status. A one in a bit position means the corresponding register bit indicates interrupt status.

Definition at line 3280 of file dm7820_types.h.

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

- include/dm7820_types.h

5.16 dm7820_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 <dm7820_types.h>`

Data Fields

- dm7820_pci_region_access_size_t size
- dm7820_pci_region_num_t region
- uint16_t offset
- union {
  uint8_t data8
  uint16_t data16
  uint32_t data32
  } data

5.16.1 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 3141 of file dm7820_types.h.

5.16.2 Field Documentation

5.16.2.1 union { ... } data

Data to write or the data read

Referenced by main().

5.16.2.2 uint16_t data16

16-bit value

Definition at line 3177 of file dm7820_types.h.

Referenced by main().

5.16.2.3 uint32_t data32

32-bit value

Definition at line 3183 of file dm7820_types.h.

Referenced by main().

5.16.2.4 uint8_t data8

8-bit value

Definition at line 3171 of file dm7820_types.h.

Referenced by main().

5.16.2.5 uint16_t offset

Offset within region to access

Definition at line 3159 of file dm7820_types.h.

Referenced by main().

5.16.2.6 dm7820_pci_region_num_t region

The PCI region to access

Definition at line 3153 of file dm7820_types.h.

Referenced by main().

5.16.2.7 dm7820_pci_region_access_size_t size

Size of access in bits

Definition at line 3147 of file dm7820_types.h.

Referenced by main().

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

- include/dm7820_types.h

5.17 dm7820_pci_region Struct Reference

DM7820 PCI region descriptor. This structure holds information about one of a device's PCI memory regions.

`#include <dm7820_driver.h>`

Data Fields

- unsigned long io_addr
- unsigned long length
- unsigned long phys_addr
- void * virt_addr
- uint8_t allocated

5.17.1 Detailed Description

DM7820 PCI region descriptor. This structure holds information about one of a device's PCI memory regions.

Definition at line 214 of file dm7820_driver.h.

5.17.2 Field Documentation

5.17.2.1 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 248 of file dm7820_driver.h.

5.17.2.2 unsigned long io_addr

I/O port number if I/O mapped

Definition at line 220 of file dm7820_driver.h.

5.17.2.3 unsigned long length

Length of region in bytes

Definition at line 226 of file dm7820_driver.h.

5.17.2.4 unsigned long phys_addr

Region's physical address if memory mapped or I/O port number if I/O mapped

Definition at line 233 of file dm7820_driver.h.

5.17.2.5 void* virt_addr

Address at which region is mapped in kernel virtual address space if memory mapped

Definition at line 240 of file dm7820_driver.h.

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

- include/dm7820_driver.h

5.18 register_read Struct Reference

Data Fields

- dm7820_pci_region_access_size_t size
- dm7820_pci_region_num_t region
- uint16_t offset
- uint32_t expected

5.18.1 Detailed Description

Register read test information

Definition at line 118 of file basic_test.c.

5.18.2 Field Documentation

5.18.2.1 uint32_t expected

Expected value of read; recast based upon size of read

Definition at line 142 of file basic_test.c.

Referenced by main().

5.18.2.2 uint16_t offset

Offset within region to read

Definition at line 136 of file basic_test.c.

Referenced by main().

5.18.2.3 dm7820_pci_region_num_t region

The PCI region to read

Definition at line 130 of file basic_test.c.

Referenced by main().

5.18.2.4 dm7820_pci_region_access_size_t size

Size of read in bits

Definition at line 124 of file basic_test.c.

Referenced by main().

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

- examples/basic_test.c

Chapter 6

File Documentation

6.1 examples/advint_event_interrupt.c File Reference

Example program which demonstrates how to use advanced interrupt block event mode interrupts.

```
#include <errno.h>
#include <error.h>
#include <getopt.h>
#include <limits.h>
#include <stdint.h>
#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <unistd.h>
#include "dm7820_library.h"
```

Functions

- void ISR (dm7820_interrupt_info interrupt_info)

  Userspace ISR.
- 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
- volatile unsigned int advint1_interrupts

6.1.1 Detailed Description

Example program which demonstrates how to use advanced interrupt block event mode interrupts.

```
Standard I/O block ports are configured as follows: 1) port 0 set to
output, 2) port 1 set to input, and 3) port 2 set to output.

Advanced interrupt 1 is configured as follows: 1) interrupt mode set to
event, 2) sampling clock set to 25 MHz clock, 3) port 0 mask register
set to ignore all bits, 4) port 1 mask register set to enable only bit 1
```

```
to generate an interrupt, and 5) port 2 mask register set to ignore all
bits.

Standard I/O block port 0 feeds values to port 1.  Therefore, each port
0 bit should be connected to the corresponding port 1 bit.  An initial
value of 0x0000 is written to port 0.

With the setup indicated above, an event interrupt should occur on port
1 whenever there is a value evenly divisible by 2 present except for
0x0000.

This program uses advanced interrupt block 1 interrupts and waits for
the interrupts to occur.

32767 event interrupts should occur.  65536 values are output but only
half of those (32768) should cause an interrupt.  The value 0x0000 does
not cause an interrupt since that value is already present on the input
port when it is written.  Thus, 32678 - 1 (32767) interrupts occur.
```

Note

This program does not use interrupt sleepy-wait. For this example, it is difficult to use sleep-waiting without making the program too complex and hard to understand.

```
-------------------------------------------------------------------------
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:

advint_event_interrupt.c 60252 2012-06-04 19:39:05Z rgroner

Definition in file advint_event_interrupt.c.

6.1.2 Function Documentation

6.1.2.1 void ISR (dm7820_interrupt_info *interrupt_info*)

Userspace ISR.

Parameters

interrupt_info	Information about the interrupt, returned by the kernel driver

Definition at line 98 of file advint_event_interrupt.c.

References advint1_interrupts, DM7820_INTERRUPT_ADVINT_1, DM7820_Return_Status, _dm7820_interrupt_info::error, and _dm7820_interrupt_info::source.

Referenced by main().

6.1.2.2 int main (int *argument_count*, char ** *arguments*)

Main program code.

Parameters

argument_count	Number of command line arguments passed to executable.
arguments	Address of array containing command line arguments.

Return values

EXIT_SUCCESS	Success.
EXIT_FAILURE	Failure.

Definition at line 168 of file advint_event_interrupt.c.

References advint1_interrupts, board, DM7820_ADVINT_INTERRUPT_0, DM7820_ADVINT_INTERRUPT_1, DM7820_ADVINT_MASTER_25_MHZ, DM7820_ADVINT_MODE_DISABLED, DM7820_ADVINT_MODE_EVENT, DM7820_AdvInt_Read_Capture(), DM7820_AdvInt_Set_Mask(), DM7820_AdvInt_Set_Master(), DM7820_AdvInt_Set_Mode(), DM7820_General_Close_Board(), DM7820_General_Enable_Interrupt(), DM7820_General_Open_Board(), DM7820_General_Reset(), DM7820_General_SetISRPriority(), DM7820_INTERRUPT_ADVINT_1, DM7820_Return_Status, DM7820_STDIO_MODE_INPUT, DM7820_STDIO_MODE_OUTPUT, DM7820_STDIO_PORT_0, DM7820_STDIO_PORT_1, DM7820_STDIO_PORT_2, DM7820_StdIO_Set_IO_Mode(), DM7820_StdIO_Set_Output(), ISR(), program_name, and usage().

6.1.3 Variable Documentation

6.1.3.1 volatile unsigned int advint1_interrupts

Counter for advint interrupts received

Definition at line 80 of file advint_event_interrupt.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 75 of file advint_event_interrupt.c.

Referenced by main(), and usage().

6.2 examples/advint_match_interrupt.c File Reference

Example program which demonstrates how to use advanced interrupt block match mode interrupts.

```
#include <errno.h>
#include <error.h>
#include <getopt.h>
#include <limits.h>
#include <stdint.h>
#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <unistd.h>
#include "dm7820_library.h"
```

Macros

- #define COMPARE_REGISTER_VALUE 0x8000

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

Example program which demonstrates how to use advanced interrupt block match mode interrupts.

```
        Standard I/O block ports are configured as follows: 1) port 0 set to
        output, 2) port 1 set to input, and 3) port 2 set to output.

        Advanced interrupt 0 is configured as follows: 1) interrupt mode set to
        match, 2) sampling clock set to 25 MHz clock, 3) port 0 mask register
        set to ignore all bits, 4) port 1 mask register set to enable all bits
        to generate an interrupt, 5) port 2 mask register set to ignore all
        bits, and 6) port 1 compare register set so that interrupt is generated
        when bit 15 is high and bits 14 through 0 are low.

    Standard I/O block port 0 feeds values to port 1.  Therefore, each port
        0 bit should be connected to the corresponding port 1 bit.

        This program uses advanced interrupt block 0 interrupts and waits for
        the interrupts to occur.  Only one such interrupt should occur.

    -----------------------------------------------------------------------
    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:

advint_match_interrupt.c 60252 2012-06-04 19:39:05Z rgroner

Definition in file advint_match_interrupt.c.

6.2.2 Macro Definition Documentation

6.2.2.1 #define COMPARE_REGISTER_VALUE 0x8000

Value to load into compare register

Definition at line 61 of file advint_match_interrupt.c.

Referenced by main().

6.2.3 Function Documentation

6.2.3.1 int main (int *argument_count*, char ** *arguments*)

Main program code.

Parameters

argument_count	Number of command line arguments passed to executable.
arguments	Address of array containing command line arguments.

Return values

EXIT_SUCCESS	Success.
EXIT_FAILURE	Failure.

Definition at line 127 of file advint_match_interrupt.c.

References board, COMPARE_REGISTER_VALUE, DM7820_ADVINT_INTERRUPT_0, DM7820_ADVINT_INTERRUPT_1, DM7820_ADVINT_MASTER_25_MHZ, DM7820_ADVINT_MODE_DISABLED, DM7820_ADVINT_MODE_MATCH, DM7820_AdvInt_Read_Capture(), DM7820_AdvInt_Set_Compare(), DM7820_AdvInt_Set_Mask(), DM7820_AdvInt_Set_Master(), DM7820_AdvInt_Set_Mode(), DM7820_General_Close_Board(), DM7820_General_Enable_Interrupt(), DM7820_General_Get_Interrupt_Status(), DM7820_General_Open_Board(), DM7820_General_Reset(), DM7820_INTERRUPT_ADVINT_0, DM7820_Return_Status, DM7820_STDIO_MODE_INPUT, DM7820_STDIO_MODE_OUTPUT, DM7820_STDIO_PORT_0, DM7820_STDIO_PORT_1, DM7820_STDIO_PORT_2, DM7820_StdIO_Set_IO_Mode(), DM7820_StdIO_Set_Output(), program_name, _dm7820_interrupt_info::source, and usage().

6.2.4 Variable Documentation

6.2.4.1 char* program_name [static]

Name of the program as invoked on the command line

Definition at line 71 of file advint_match_interrupt.c.

Referenced by main(), and usage().

6.3 examples/advint_status.c File Reference

Example program which demonstrates how to get advanced interrupt status when not using interrupts.

```
#include <errno.h>
#include <error.h>
#include <getopt.h>
#include <limits.h>
#include <stdint.h>
#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <unistd.h>
#include "dm7820_library.h"
```

Macros

- #define COMPARE_REGISTER_VALUE 0x8000

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

6.3.1 Detailed Description

Example program which demonstrates how to get advanced interrupt status when not using interrupts.

```
        Standard I/O block ports are configured as follows: 1) port 0 set to
        output, 2) port 1 set to input, and 3) port 2 set to output.

        Advanced interrupt 0 is configured as follows: 1) interrupt mode set to
        match, 2) sampling clock set to 25 MHz clock, 3) port 0 mask register
        set to ignore all bits, 4) port 1 mask register set to enable all bits
        to generate an interrupt, 5) port 2 mask register set to ignore all
        bits, and 6) port 1 compare register set so that interrupt is generated
        when bit 15 is high and bits 14 through 0 are low.

    Standard I/O block port 0 feeds values to port 1.  Therefore, each port
        0 bit should be connected to the corresponding port 1 bit.


    -----------------------------------------------------------------------
    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:

advint_status.c 60252 2012-06-04 19:39:05Z rgroner

Definition in file advint_status.c.

6.3.2 Macro Definition Documentation

6.3.2.1 #define COMPARE_REGISTER_VALUE 0x8000

Value to load into compare register

Definition at line 58 of file advint_status.c.

Referenced by main().

6.3.3 Function Documentation

6.3.3.1 int main (int *argument_count*, char ** *arguments*)

Main program code.

Parameters

argument_count	Number of command line arguments passed to executable.
arguments	Address of array containing command line arguments.

Return values

EXIT_SUCCESS	Success.
EXIT_FAILURE	Failure.

Definition at line 124 of file advint_status.c.

References board, COMPARE_REGISTER_VALUE, DM7820_AdvInt_Get_Status(), DM7820_ADVINT_INTERRUPT_0, DM7820_ADVINT_INTERRUPT_1, DM7820_ADVINT_MASTER_25_MHZ, DM7820_ADVINT_MODE_DISABLED, DM7820_ADVINT_MODE_MATCH, DM7820_AdvInt_Read_Capture(), DM7820_AdvInt_Set_Compare(), DM7820_AdvInt_Set_Mask(), DM7820_AdvInt_Set_Master(), DM7820_AdvInt_Set_Mode(), DM7820_General_Close_Board(), DM7820_General_Open_Board(), DM7820_Return_Status, DM7820_STDIO_MODE_INPUT, DM7820_STDIO_MODE_OUTPUT, DM7820_STDIO_PORT_0, DM7820_STDIO_PORT_1, DM7820_STDIO_PORT_2, DM7820_StdIO_Set_IO_Mode(), DM7820_StdIO_Set_Output(), program_name, and usage().

6.3.4 Variable Documentation

6.3.4.1 char* program_name [static]

Name of the program as invoked on the command line

Definition at line 68 of file advint_status.c.

Referenced by main(), and usage().

6.4 examples/advint_strobe_interrupt.c File Reference

Example program which demonstrates how to use advanced interrupt block strobe mode interrupts.

```
#include <errno.h>
#include <error.h>
#include <getopt.h>
#include <limits.h>
#include <stdint.h>
#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <unistd.h>
#include "dm7820_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

6.4.1 Detailed Description

Example program which demonstrates how to use advanced interrupt block strobe mode interrupts.

```
Standard I/O block ports are configured so that port 0 is set to output
and port 1 is set to input.  Port 2 is not used.

This program initializes advanced interrupt 0 by selecting strobe signal
1 as the strobe source and setting interrupt mode to strobe.

Standard I/O block port 0 feeds values to port 1.  Therefore, each port
0 bit should be connected to the corresponding port 1 bit.

This program initializes the strobe signals as follows: 1) strobe signal
1 set to input, 2) strobe signal 2 set to output, and 3) strobe signal 2
set low.

Strobe signal 2 provides the input for strobe signal 1.  Therefore, CN11
pin 2 should be connected to CN10 pin 2.

This program uses advanced interrupt block 0 interrupts and waits for
the interrupts to occur.  Only one such interrupt should occur.
```

Note

This program does not use interrupt sleepy-wait. For this example, it is difficult to use sleep-waiting without making the program too complex and hard to understand.

Id:

advint_strobe_interrupt.c 60252 2012-06-04 19:39:05Z rgroner

Definition in file advint_strobe_interrupt.c.

6.4.2 Function Documentation

6.4.2.1 int main (int *argument_count*, char ** *arguments*)

Main program code.

Parameters

argument_count	Number of command line arguments passed to executable.
arguments	Address of array containing command line arguments.

Return values

EXIT_SUCCESS	Success.
EXIT_FAILURE	Failure.

Definition at line 125 of file advint_strobe_interrupt.c.

References board, DM7820_ADVINT_INTERRUPT_0, DM7820_ADVINT_INTERRUPT_1, DM7820_ADVINT_MASTER_STROBE_1, DM7820_ADVINT_MODE_DISABLED, DM7820_ADVINT_MODE_STROBE, DM7820_AdvInt_Read_Capture(), DM7820_AdvInt_Set_Master(), DM7820_AdvInt_Set_Mode(), DM7820_General_Close_Board(), DM7820_General_Enable_Interrupt(), DM7820_General_Get_Interrupt_Status(), DM7820_General_Open_Board(), DM7820_General_Reset(), DM7820_INTERRUPT_ADVINT_0, DM7820_Return_Status, D-

M7820_STDIO_MODE_INPUT, DM7820_STDIO_MODE_OUTPUT, DM7820_STDIO_PORT_0, DM7820_STDIO_PORT_1, DM7820_StdIO_Set_IO_Mode(), DM7820_StdIO_Set_Output(), DM7820_STDIO_STROBE_1, DM7820_STDIO_STROBE_2, DM7820_StdIO_Strobe_Mode(), DM7820_StdIO_Strobe_Output(), program_name, _dm7820_interrupt_info::source, and usage().

6.4.3 Variable Documentation

6.4.3.1 char* program_name [static]

Name of the program as invoked on the command line

Definition at line 69 of file advint_strobe_interrupt.c.

Referenced by main(), and usage().

6.5 examples/basic_test.c File Reference

Program which tests the basic functionality of the driver.

```
#include <errno.h>
#include <error.h>
#include <fcntl.h>
#include <getopt.h>
#include <limits.h>
#include <stdint.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/ioctl.h>
#include <sys/stat.h>
#include <sys/sysmacros.h>
#include <sys/time.h>
#include <sys/types.h>
#include <sys/utsname.h>
#include <unistd.h>
#include "dm7820_ioctl.h"
#include "dm7820_registers.h"
```

Data Structures

- struct register_read

Macros

- #define DM7820_MAX_DMA_BUFFER_COUNT 32
- #define DM7820_MAX_DMA_BUFFER_SIZE 0x40000

Typedefs

- typedef struct register_read register_read_t
- typedef enum initialization_state initialization_state_t

Enumerations

- enum initialization_state {
  INIT_NO_INITIALIZATION = 0x00000000, INIT_BOARD_ARRAY_ALLOCATED = 0x00000004, INIT_BAD_DEVICE_FILE_CREATED = 0x00000008, INIT_BOARD_ARRAY_OPENED = 0x00000010,
  INIT_NO_INITIALIZATION = 0x00000000, INIT_BOARD_ARRAY_ALLOCATED = 0x00000004, INIT_BAD_DEVICE_FILE_CREATED = 0x00000008, INIT_BOARD_ARRAY_OPENED = 0x00000010 }

Functions

- static void cleanup (void)

  Perform actions necessary to clean up after the program encounters a fatal error.
- 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_success (int status)

  Verify that a function being tested succeeded.
- static void expect_failure_and_check (int status, int expected_errno)

  Verify that a function being tested failed with the given errno.
- int main (int argument_count, char **arguments)

  Main program code.

Variables

- static char * program_name
- static uint8_t access_sizes []
- static uint16_t region_sizes []
- static char * region_names []
- register_read_t register_reads []
- static volatile
  initialization_state_t init_state = INIT_NO_INITIALIZATION
- static volatile int * descriptors
- static volatile char bad_device_name [30]
- static volatile unsigned long device_count = 1

6.5.1 Detailed Description

Program which tests the basic functionality of 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.
-----------------------------------------------------------------------------
```

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.
This program uses sbrk() to determine what should be an invalid address for causing certain errors in the driver. The address returned by sbrk() may not cause failure in some circumstances but there seems to be no reliable and easy way to determine whether an arbitrary user address is actually mapped into a process' address space.

Id:

basic_test.c 89872 2015-07-08 16:05:17Z rgroner

Definition in file basic_test.c.

6.5.2 Macro Definition Documentation

6.5.2.1 #define DM7820_MAX_DMA_BUFFER_COUNT 32

Default driver limit on number of DMA buffers that can be allocated per DMA/FIFO channel

Definition at line 64 of file basic_test.c.

Referenced by main().

6.5.2.2 #define DM7820_MAX_DMA_BUFFER_SIZE 0x40000

Default driver limit in bytes on size of DMA buffers that can be allocated per DMA/FIFO channel

Definition at line 71 of file basic_test.c.

Referenced by main().

6.5.3 Typedef Documentation

6.5.3.1 typedef enum initialization_state initialization_state_t

Program initialization status type

Definition at line 159 of file basic_test.c.

6.5.3.2 typedef struct register_read register_read_t

Register read test case type

Definition at line 153 of file basic_test.c.

6.5.4 Enumeration Type Documentation

6.5.4.1 enum initialization_state

Status of program initialization. Used to indication which operations need to be undone when cleaning up after an error. This is a bit mask, so only one bit should be set for each enumeration value.

Enumerator:

INIT_NO_INITIALIZATION No initialization performed
INIT_BOARD_ARRAY_ALLOCATED File descriptor array allocated
INIT_BAD_DEVICE_FILE_CREATED /dev entry with invalid minor number created
INIT_BOARD_ARRAY_OPENED File descriptor array contains at least one opened file
INIT_NO_INITIALIZATION No initialization performed
INIT_BOARD_ARRAY_ALLOCATED Board descriptor array allocated
INIT_BAD_DEVICE_FILE_CREATED /dev entry with invalid minor number created
INIT_BOARD_ARRAY_OPENED Board descriptor array contains at least one opened file

Definition at line 83 of file basic_test.c.

6.5.5 Function Documentation

6.5.5.1 static void expect_failure_and_check (int *status,* int *expected_errno*) [static]

Verify that a function being tested failed with the given errno.

Parameters

status	Return code from function being tested.
expected_errno	Expected errno that function should have set.

Definition at line 651 of file basic_test.c.

Referenced by main().

6.5.5.2 static void expect_success (int *status*) [static]

Verify that a function being tested succeeded.

Parameters

status	Return code from function being tested.

Definition at line 624 of file basic_test.c.

Referenced by main().

6.5.5.3 int main (int *argument_count,* char ** *arguments*)

Main program code.

Parameters

argument_count	Number of command line arguments passed to executable.
arguments	Address of array containing command line arguments.

Return values

EXIT_SUCCESS	Success.
EXIT_FAILURE	Failure.

Definition at line 692 of file basic_test.c.

References dm7820_ioctl_region_readwrite::access, dm7820_ioctl_region_modify::access, access_sizes, dm7820_ioctl_dma_function::arguments, dm7820_dma_initialize_arguments::buffer_count, dm7820_dma_initialize_arguments::buffer_size, cleanup(), dm7820_pci_access_request::data, dm7820_pci_access_request::data16, dm7820_pci_access_request::data32, dm7820_pci_access_request::data8, device_count, DM7820_BAR0_DMAPADR0, DM7820_BAR0_LENGTH, DM7820_BAR2_BOARD_RESET, DM7820_BAR2_LENGTH, DM7820_BAR2_PORT0_OUTPUT, DM7820_BAR2_PORT0_PERIPH_SEL_L, DM7820_BOARD_RESET_DO_RESET, DM7820_DMA_FUNCTION_INITIALIZE, DM7820_DMA_FUNCTION_READ, DM7820_DMA_FUNCTION_STOP, DM7820_DMA_FUNCTION_WRITE, DM7820_FIFO_QUEUE_0, DM7820_FIFO_QUEUE_1, DM7820_IOCTL_DMA_FUNCTION, DM7820_IOCTL_GET_INTERRUPT_STATUS, DM7820_IOCTL_REGION_MODIFY, DM7820_IOCTL_REGION_READ, DM7820_IOCTL_REGION_WRITE, DM7820_MAX_DMA_BUFFER_COUNT, DM7820_MAX_DMA_BUFFER_SIZE, DM7820_PCI_REGION_ACCESS_16, DM7820_PCI_REGION_ACCESS_32, DM7820_PCI_REGION_ACCESS_8, DM7820_PCI_REGION_FPGA_MEM, DM7820_PCI_REGION_PLX_MEM, dm7820_ioctl_argument::dma_function, dm7820_dma_function_arguments::dma_init, expect_failure_and_check(), expect_success(), register_read::expected, dm7820_ioctl_dma_function::fifo, dm7820_ioctl_dma_function::function, INIT_BAD_DEVICE_FILE_CREATED, INIT_BOARD_ARRAY_ALLOCATED, INIT_BOARD_A-

RRAY_OPENED, dm7820_ioctl_argument::int_status, dm7820_ioctl_region_modify::mask, dm7820_ioctl_region_modify::mask16, dm7820_ioctl_region_modify::mask32, dm7820_ioctl_region_modify::mask8, dm7820_ioctl_argument::modify, register_read::offset, dm7820_pci_access_request::offset, program_name, dm7820_ioctl_argument::readwrite, register_read::region, dm7820_pci_access_request::region, region_names, region_sizes, register_reads, register_read::size, dm7820_pci_access_request::size, dm7820_ioctl_dma_function::transfer_size, usage(), and dm7820_ioctl_interrupt_status::wait_for_interrupt.

6.5.6 Variable Documentation

6.5.6.1 uint8_t access_sizes[ ] [static]

Initial value:

```
{

        8,

        16,

        32
}
```

Size in bits for each of the access sizes

Definition at line 175 of file basic_test.c.

Referenced by main().

6.5.6.2 volatile char bad_device_name[30] [static]

Path name of DM7820 device file with invalid minor number

Definition at line 534 of file basic_test.c.

6.5.6.3 volatile int* descriptors [static]

Array of DM7820 device file descriptors

Definition at line 528 of file basic_test.c.

6.5.6.4 volatile unsigned long device_count = 1 [static]

Number of devices found when driver was loaded

Definition at line 540 of file basic_test.c.

Referenced by cleanup(), and main().

6.5.6.5 volatile initialization_state_t init_state = INIT_NO_INITIALIZATION [static]

Program initialization state

Definition at line 522 of file basic_test.c.

6.5.6.6 char* program_name [static]

Name of the program as invoked on the command line

Definition at line 169 of file basic_test.c.

Referenced by main(), and usage().

6.5.6.7 char* region_names[] [static]

Initial value:

```
{

        "BAR0",

        "BAR1",

        "BAR2"
}
```

PCI region names

Definition at line 225 of file basic_test.c.

Referenced by main().

6.5.6.8 uint16_t region_sizes[] [static]

Initial value:

```
{

        DM7820_BAR0_LENGTH,

        DM7820_BAR1_LENGTH,

        DM7820_BAR2_LENGTH
}
```

Length in bytes for each of the PCI regions

Definition at line 200 of file basic_test.c.

Referenced by main().

6.5.6.9 register_read_t register_reads[]

Data for register read tests. The board module ID registers are used because they have known values.

Note

These registers are 16 bits wide, therefore a 32-bit read of them must mask off the most significant 16 bits.

Definition at line 255 of file basic_test.c.

Referenced by main().

6.6 examples/brdctl_device_info.c File Reference

Example program which demonstrates how to obtain version information and PCI master capable status from a device.

```
#include <errno.h>
#include <error.h>
#include <getopt.h>
#include <limits.h>
#include <stdint.h>
#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <unistd.h>
#include "dm7820_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

6.6.1 Detailed Description

Example program which demonstrates how to obtain version information and PCI master capable status from a device.

```
    ----------------------------------------------------------------------
    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: brdctl_device_info.c 60252 2012-06-04 19:39:05Z rgroner $
```

Definition in file brdctl_device_info.c.

6.6.2 Function Documentation

6.6.2.1 int main (int *argument_count*, char ** *arguments*)

Main program code.

Parameters

argument_count	Number of command line arguments passed to executable.
arguments	Address of array containing command line arguments.

Return values

EXIT_SUCCESS	Success.
EXIT_FAILURE	Failure.

Definition at line 95 of file brdctl_device_info.c.

References board, DM7820_General_Close_Board(), DM7820_General_Get_Version_Info(), DM7820_General_Is_PCI_Master(), DM7820_General_Open_Board(), DM7820_Return_Status, program_name, and usage().

6.6.3 Variable Documentation

6.6.3.1 char* program_name [static]

Name of the program as invoked on the command line

Definition at line 43 of file brdctl_device_info.c.

Referenced by main(), and usage().

6.7 examples/dma_capture_buffers.c File Reference

Example program which demonstrates how to use DMA to continuously acquire samples.

```
#include <errno.h>
#include <error.h>
#include <getopt.h>
#include <limits.h>
#include <stdint.h>
#include <stdio.h>
#include <stdlib.h>
#include <signal.h>
#include <strings.h>
#include <string.h>
#include <unistd.h>
#include <fcntl.h>
#include "dm7820_library.h"
```

Macros

- #define BUF_SIZE 0x40000
- #define BUF_NUM 8
- #define SAMPLES ((BUF_SIZE * BUF_NUM)/2)
- #define DMA_BUF_SIZE (BUF_SIZE * BUF_NUM)

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 (dm7820_interrupt_info interrupt_info)

  Userspace ISR.
- void clean_up ()

  Disable and clean up.
- int main (int argument_count, char **arguments)

  Main program code.

Variables

- static char * program_name
- int interrupts = 0
- DM7820_Board_Descriptor * board

6.7.1 Detailed Description

Example program which demonstrates how to use DMA to continuously acquire samples.

```
    This example program demonstrates the DM7820's ability to continuously
    sample data from FIFO 0 by using DMA.  This example will sample 8Mb of
    data (4 Million Samples) at a rate of 2.50 Mhz.  We read DMA samples from
    the device then read them into a 8M buffer.


    Warning:  This example program is only intended to run on a system with
    enough RAM to support the large buffer's which will be allocated.
    Preferably 128MB minimum.


-----------------------------------------------------------------------------
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_capture_buffers.c 60276 2012-06-05 16:04:15Z rgroner

Definition in file dma_capture_buffers.c.

6.7.2 Macro Definition Documentation

6.7.2.1 #define BUF_NUM 8

The number of buffers we want for each DMA channel.

Definition at line 89 of file dma_capture_buffers.c.

Referenced by main().

6.7.2.2 #define BUF_SIZE 0x40000

The size of the individual buffers we are asking to driver to create for each DMA channel. If you receive a memory error when trying to run this example, decrease this size.

Definition at line 83 of file dma_capture_buffers.c.

Referenced by main().

6.7.2.3 #define DMA_BUF_SIZE (BUF_SIZE * BUF_NUM)

The total space available in the device for DMA at any one time (4MB).

Definition at line 101 of file dma_capture_buffers.c.

Referenced by main().

6.7.2.4 #define SAMPLES ((BUF_SIZE * BUF_NUM)/2)

The number of samples that can be held by the list of buffers we created.

Definition at line 95 of file dma_capture_buffers.c.

6.7.3 Function Documentation

6.7.3.1 void ISR (dm7820_interrupt_info *interrupt_info*)

Userspace ISR.

Parameters

interrupt_info	Information about the interrupt, returned by the kernel driver

Definition at line 144 of file dma_capture_buffers.c.

References DM7820_INTERRUPT_FIFO_0_DMA_DONE, DM7820_Return_Status, _dm7820_interrupt_info::error, interrupts, and _dm7820_interrupt_info::source.

6.7.3.2 int main (int *argument_count,* char ** *arguments*)

Main program code.

Parameters

argument_count	Number of command line arguments passed to executable.
arguments	Address of array containing command line arguments.

Return values

EXIT_SUCCESS	Success.
EXIT_FAILURE	Failure.

Definition at line 229 of file dma_capture_buffers.c.

References BUF_NUM, BUF_SIZE, clean_up(), DM7820_DMA_DEMAND_ON_DM7820_TO_PCI, DM7820_FIFO_0_DATA_INPUT_PORT_0, DM7820_FIFO_DMA_Configure(), DM7820_FIFO_DMA_Create_Buffer(), DM7820_FIFO_DMA_Enable(), DM7820_FIFO_DMA_Free_Buffer(), DM7820_FIFO_DMA_Initialize(), DM7820_FIFO_DMA_Read(), DM7820_FIFO_DMA_REQUEST_READ, DM7820_FIFO_Enable(), DM7820_FIFO_INPUT_CLOCK_PROG_CLOCK_0, DM7820_FIFO_OUTPUT_CLOCK_PCI_READ, DM7820_FIFO_QUEUE_0, DM7820_FIFO_Set_Data_Input(), DM7820_FIFO_Set_DMA_Request(), DM7820_FIFO_Set_Input_Clock(), DM7820_FIFO_Set_Output_Clock(), DM7820_General_Enable_Interrupt(), DM7820_General_Open_Board(), DM7820_General_RemoveISR(), DM7820_General_Reset(), DM7820_General_SetISRPriority(), DM7820_INTERRUPT_FIFO_0_EMPTY, DM7820_PRGCLK_CLOCK_0, DM7820_PRGCLK_MASTER_25_MHZ, DM7820_PRGCLK_MODE_CONTINUOUS, DM7820_PRGCLK_MODE_DISABLED, DM7820_PrgClk_Set_Master(), DM7820_PrgClk_Set_Mode(), DM7820_PrgClk_Set_Period(), DM7820_PrgClk_Set_Start_Trigger(), DM7820_PrgClk_Set_Stop_Trigger(), DM7820_PRGCLK_START_IMMEDIATE, DM7820_PRGCLK_STOP_NONE, DM7820_Return_Status, DM7820_STDIO_MODE_INPUT, DM7820_STDIO_PORT_0, DM7820_StdIO_Set_IO_Mode(), DMA_BUF_SIZE, interrupts, ISR(), program_name, and usage().

6.7.4 Variable Documentation

6.7.4.1 DM7820_Board_Descriptor* board

Device descriptor

Definition at line 75 of file dma_capture_buffers.c.

Referenced by main().

6.7.4.2 int interrupts = 0

Variable to count the number of DMA Done interrupts occurring

Definition at line 69 of file dma_capture_buffers.c.

Referenced by ISR(), and main().

6.7.4.3 char* program_name [static]

Name of the program as invoked on the command line

Definition at line 63 of file dma_capture_buffers.c.

Referenced by main(), and usage().

6.8 examples/dma_capture_file.c File Reference

Example program which demonstrates how to use DMA to continuously acquire samples.

```
#include <errno.h>
#include <error.h>
#include <getopt.h>
#include <limits.h>
#include <stdint.h>
#include <stdio.h>
#include <stdlib.h>
#include <signal.h>
#include <strings.h>
#include <string.h>
#include <unistd.h>
#include <fcntl.h>
#include "dm7820_library.h"
```

Macros

- #define DAT_FILE "./test.dat"
- #define BUF_SIZE 0x40000
- #define BUF_NUM 8
- #define SAMPLES ((BUF_SIZE * BUF_NUM)/2)
- #define DMA_BUF_SIZE (BUF_SIZE * BUF_NUM)

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 (dm7820_interrupt_info interrupt_info)

  Userspace ISR.
- void clean_up ()

  Disable and clean up.

- int main (int argument_count, char **arguments)

  Main program code.

Variables

- static char * program_name
- volatile int dma_done_interrupts = 0
- DM7820_Board_Descriptor * board
- int test_data

6.8.1 Detailed Description

Example program which demonstrates how to use DMA to continuously acquire samples.

```
    This example program demonstrates the DM7820's ability to continuously
    sample data from FIFO 0 by using DMA.  This example will sample 8Mb of
    data (4 Million Samples) at a rate of 1 Mhz.  We read DMA samples from
    the device then dump them to a file on disk.  We use FILE IO here as
    attempting to create buffers large enough to hold all the data we expect
    to gather can cause failures.  Because we are using FILE I/O the speed
    at which we can continuously sample is restricted by how fast the I/O
    operations can finish.

    Warning:  This example program is only intended to run on a system with
    enough RAM to support the large buffer's which will be allocated.
    Preferably 128MB minimum.

-----------------------------------------------------------------------------
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_capture_file.c 79285 2014-05-21 20:37:42Z rgroner

Definition in file dma_capture_file.c.

6.8.2 Macro Definition Documentation

6.8.2.1 #define BUF_NUM 8

The number of buffers we want for each DMA channel.

Definition at line 105 of file dma_capture_file.c.

Referenced by main().

6.8.2.2 #define BUF_SIZE 0x40000

The size of the individual buffers we are asking to driver to create for each DMA channel. If you receive a memory error when trying to run this example, decrease this size.

Definition at line 99 of file dma_capture_file.c.

Referenced by main().

6.8.2.3 #define DAT_FILE "./test.dat"

Constant pathname and filename for the data file.

Definition at line 91 of file dma_capture_file.c.

Referenced by clean_up(), and main().

6.8.2.4 #define DMA_BUF_SIZE (BUF_SIZE * BUF_NUM)

The total space available in the device for DMA at any one time (4MB).

Definition at line 117 of file dma_capture_file.c.

Referenced by main().

6.8.2.5 #define SAMPLES ((BUF_SIZE * BUF_NUM)/2)

The number of samples that can be held by the list of buffers we created.

Definition at line 111 of file dma_capture_file.c.

6.8.3 Function Documentation

6.8.3.1 void ISR (dm7820_interrupt_info *interrupt_info*)

Userspace ISR.

Parameters

interrupt_info	Information about the interrupt, returned by the kernel driver

Definition at line 160 of file dma_capture_file.c.

References DM7820_INTERRUPT_FIFO_0_DMA_DONE, DM7820_Return_Status, dma_done_interrupts, _dm7820_interrupt_info::error, and _dm7820_interrupt_info::source.

6.8.3.2 int main (int *argument_count,* char ** *arguments*)

Main program code.

Parameters

argument_count	Number of command line arguments passed to executable.
arguments	Address of array containing command line arguments.

Return values

EXIT_SUCCESS	Success.
EXIT_FAILURE	Failure.

Definition at line 251 of file dma_capture_file.c.

References BUF_NUM, BUF_SIZE, clean_up(), DAT_FILE, DM7820_DMA_DEMAND_ON_DM7820_TO_PCI, DM7820_FIFO_0_DATA_INPUT_PORT_0, DM7820_FIFO_DMA_Configure(), DM7820_FIFO_DMA_Create_Buffer(), DM7820_FIFO_DMA_Enable(), DM7820_FIFO_DMA_Free_Buffer(), DM7820_FIFO_DMA_Initialize(), DM7820_FIFO_DMA_Read(), DM7820_FIFO_DMA_REQUEST_READ, DM7820_FIFO_Enable(), DM7820_FIFO_Get_Status(), DM7820_FIFO_INPUT_CLOCK_PROG_CLOCK_0, DM7820_FIFO_OUTPUT_CLOCK_PCI_READ, DM7820_FIFO_QUEUE_0, DM7820_FIFO_Set_Data_Input(), DM7820_FIFO_Set_DMA_Request(), DM7820_FI-

FO_Set_Input_Clock(), DM7820_FIFO_Set_Output_Clock(), DM7820_FIFO_STATUS_EMPTY, DM7820_General_Enable_Interrupt(), DM7820_General_Open_Board(), DM7820_General_RemoveISR(), DM7820_General_Reset(), DM7820_General_SetISRPriority(), DM7820_INTERRUPT_FIFO_0_EMPTY, DM7820_PRGCLK_CLOCK_0, DM7820_PRGCLK_MASTER_25_MHZ, DM7820_PRGCLK_MODE_CONTINUOUS, DM7820_PRGCLK_MODE_DISABLED, DM7820_PrgClk_Set_Master(), DM7820_PrgClk_Set_Mode(), DM7820_PrgClk_Set_Period(), DM7820_PrgClk_Set_Start_Trigger(), DM7820_PrgClk_Set_Stop_Trigger(), DM7820_PRGCLK_START_IMMEDIATE, DM7820_PRGCLK_STOP_NONE, DM7820_Return_Status, DM7820_STDIO_MODE_INPUT, DM7820_STDIO_PORT_0, DM7820_StdIO_Set_IO_Mode(), DMA_BUF_SIZE, dma_done_interrupts, ISR(), program_name, test_data, and usage().

6.8.4 Variable Documentation

6.8.4.1 DM7820_Board_Descriptor* board

Device descriptor

Definition at line 79 of file dma_capture_file.c.

6.8.4.2 volatile int dma_done_interrupts = 0

Variable to count the number of DMA Done interrupts occurring

Definition at line 73 of file dma_capture_file.c.

Referenced by ISR(), and main().

6.8.4.3 char* program_name [static]

Name of the program as invoked on the command line

Definition at line 67 of file dma_capture_file.c.

Referenced by main(), and usage().

6.8.4.4 int test_data

File handle for the data file.

Definition at line 85 of file dma_capture_file.c.

Referenced by main().

6.9 examples/fifo_cascaded.c File Reference

Example program which demonstrates how to use the FIFO block FIFOs such that the output of one FIFO serves as the input of another FIFO.

```
#include <errno.h>
#include <error.h>
#include <getopt.h>
#include <limits.h>
#include <stdint.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/types.h>
#include <unistd.h>
#include "dm7820_library.h"
```

Functions

- static void get_fifo_status (DM7820_Board_Descriptor *board, dm7820_fifo_queue fifo, dm7820_fifo_status_condition condition, uint8_t *status)

  Determine whether or not the specified status condition has occurred for the given FIFO.
- 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

6.9.1 Detailed Description

Example program which demonstrates how to use the FIFO block FIFOs such that the output of one FIFO serves as the input of another FIFO.

```
This program transfers FIFO 0 data out standard I/O port 0, in standard
I/O port 1, and into FIFO 1.

FIFO 0 is configured as follows: 1) input clock set to PCI write
request, 2) output clock set to programmable clock 1, and 3) data input
set to PCI data.

FIFO 1 is configured as follows: 1) input clock set to programmable
clock 1, 2) output clock set to PCI read request, and 3) data input set
to standard I/O port 1.

The programmable clocks are set up as follows:

          Master      Start       Stop
    Clock Clock       Trigger     Trigger     Mode        Frequency
    ================================================================
    0     25 MHz      immediate   none        continuous  100 KHz
    1     Clock 0     immediate   none        continuous  2 Hz

Standard I/O ports 0 and 1 should be connected as follows:

    CN10 Pin    CN11 Pin
    ====================
    17          17
    19          19
    21          21
    23          23
    25          25
    27          27
    29          29
    31          31
    33          33
    35          35
    37          37
    39          39
    41          41
    43          43
    45          45
    47          47

A character string is written to FIFO 0.  Once the programmable clocks
are started, the string is transferred to FIFO 1 at the rate of two
characters per second.  While the characters are being transferred, the
program reads the characters from FIFO 1 and reassembles the string.
```

```
--------------------------------------------------------------------------
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:

fifo_cascaded.c 60252 2012-06-04 19:39:05Z rgroner

Definition in file fifo_cascaded.c.

6.9.2 Function Documentation

6.9.2.1 static void get_fifo_status (DM7820_Board_Descriptor * *board*, dm7820_fifo_queue *fifo*, dm7820_fifo_status_condition *condition*, uint8_t * *status*) [static]

Determine whether or not the specified status condition has occurred for the given FIFO.

Parameters

board	Address of device's library board descriptor.
fifo	The FIFO to determine status of.
condition	The status condition to check for.
status	Address where occurrence flag should be stored.

Definition at line 126 of file fifo_cascaded.c.

References DM7820_FIFO_Get_Status().

Referenced by main().

6.9.2.2 int main (int *argument_count*, char ** *arguments*)

Main program code.

Parameters

argument_count	Number of command line arguments passed to executable.
arguments	Address of array containing command line arguments.

Return values

EXIT_SUCCESS	Success.
EXIT_FAILURE	Failure.

Definition at line 186 of file fifo_cascaded.c.

References board, DM7820_FIFO_0_DATA_INPUT_PCI_DATA, DM7820_FIFO_1_DATA_INPUT_PORT_1, DM7820_FIFO_Enable(), DM7820_FIFO_INPUT_CLOCK_PCI_WRITE, DM7820_FIFO_INPUT_CLOCK_PROG_CLOCK_1, DM7820_FIFO_OUTPUT_CLOCK_PCI_READ, DM7820_FIFO_OUTPUT_CLOCK_PROG_CLOCK_1, DM7820_FIFO_QUEUE_0, DM7820_FIFO_QUEUE_1, DM7820_FIFO_Read(), DM7820_FIFO_Set_Data_Input(), DM7820_FIFO_Set_Input_Clock(), DM7820_FIFO_Set_Output_Clock(), DM7820_FIFO_STATUS_EMPTY, DM7820_FIFO_STATUS_FULL, DM7820_FIFO_STATUS_OVERFLOW, DM7820_FIFO_STATUS_UNDERFLOW, DM7820_FIFO_Write(), DM7820_General_Close_Board(), DM7820_General_Open_Board(), DM7820_PRGCLK_CLOCK_0, DM7820_PRGCLK_CLOCK_1, DM7820_PRGCLK_CLOCK_2, DM7820_PRGCLK_CLOCK_3,

DM7820_PRGCLK_MASTER_25_MHZ, DM7820_PRGCLK_MASTER_PROG_CLOCK_0, DM7820_PRGCLK_MODE_CONTINUOUS, DM7820_PRGCLK_MODE_DISABLED, DM7820_PrgClk_Set_Master(), DM7820_PrgClk_Set_Mode(), DM7820_PrgClk_Set_Period(), DM7820_PrgClk_Set_Start_Trigger(), DM7820_PrgClk_Set_Stop_Trigger(), DM7820_PRGCLK_START_IMMEDIATE, DM7820_PRGCLK_STOP_NONE, DM7820_Return_Status, DM7820_STDIO_MODE_INPUT, DM7820_STDIO_MODE_PER_OUT, DM7820_STDIO_PERIPH_FIFO_0, DM7820_STDIO_PORT_0, DM7820_STDIO_PORT_1, DM7820_StdIO_Set_IO_Mode(), DM7820_StdIO_Set_Periph_Mode(), get_fifo_status(), program_name, and usage().

6.9.3 Variable Documentation

6.9.3.1 char* program_name [static]

Name of the program as invoked on the command line

Definition at line 91 of file fifo_cascaded.c.

Referenced by main(), and usage().

6.10 examples/fifo_interrupt.c File Reference

Example program which demonstrates a loopback accross the FIFO's using DMA.

```
#include <errno.h>
#include <error.h>
#include <getopt.h>
#include <limits.h>
#include <stdint.h>
#include <stdio.h>
#include <stdlib.h>
#include <strings.h>
#include <unistd.h>
#include "dm7820_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

6.10.1 Detailed Description

Example program which demonstrates a loopback accross the FIFO's using DMA.

```
This program uses FIFO 0, which is configured as follows: 1) input clock
set to PCI write request, 2) output clock set to programmable clock 1,
and 3) data input set to PCI data.

The programmable clocks are set up as follows:

            Master      Start       Stop
```

```
    Clock   Clock       Trigger     Trigger     Mode        Frequency
    ================================================================
    0       25 MHz      immediate   none        continuous  500 Hz
    1       Clock 0     immediate   none        continuous  1 Hz

With the setup indicated above, programmable clock 1 will cause a value
to be read from FIFO 0 once every second.

Any value clocked out of FIFO 0 is discarded.

The program loops performing the following sequence of actions:
1) writing 5 values to FIFO 0, 2) enabling FIFO 0 empty interrupt,
3) starting programmable clocks 0 and 1, 4) waiting for a FIFO 0 empty
interrupt, and 5) disabling programmable clocks 0 and 1.

The above sequence of actions is performed 4 times and then the program
exits.


-----------------------------------------------------------------------
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:

fifo_interrupt.c 60252 2012-06-04 19:39:05Z rgroner

Definition in file fifo_interrupt.c.

6.10.2 Function Documentation

6.10.2.1 int main (int *argument_count*, char ** *arguments*)

Main program code.

Parameters

argument_count	Number of command line arguments passed to executable.
arguments	Address of array containing command line arguments.

Return values

EXIT_SUCCESS	Success.
EXIT_FAILURE	Failure.

Definition at line 134 of file fifo_interrupt.c.

References board, DM7820_FIFO_0_DATA_INPUT_PCI_DATA, DM7820_FIFO_Enable(), DM7820_FIFO_INPUT_CLOCK_PCI_WRITE, DM7820_FIFO_OUTPUT_CLOCK_PROG_CLOCK_1, DM7820_FIFO_QUEUE_0, DM7820_FIFO_Set_Data_Input(), DM7820_FIFO_Set_Input_Clock(), DM7820_FIFO_Set_Output_Clock(), DM7820_FIFO_Write(), DM7820_General_Close_Board(), DM7820_General_Enable_Interrupt(), DM7820_General_Get_Interrupt_Status(), DM7820_General_Open_Board(), DM7820_General_Reset(), DM7820_INTERRUPT_FIFO_0_EMPTY, DM7820_PRGCLK_CLOCK_0, DM7820_PRGCLK_CLOCK_1, DM7820_PRGCLK_MASTER_25_MHZ, DM7820_PRGCLK_MASTER_PROG_CLOCK_0, DM7820_PRGCLK_MODE_CONTINUOUS, DM7820_PRGCLK_MODE_DISABLED, DM7820_PrgClk_Set_Master(), DM7820_PrgClk_Set_Mode(), DM7820_PrgClk_Set_Period(), DM7820_PrgClk_Set_Start_Trigger(), DM7820_PrgClk_Set_Stop_Trigger(), DM7820_PRGCLK_START_IMMEDIATE, DM7820_PRGCLK_STOP_NONE, DM7820_Return_Status, program_name, _dm7820_interrupt_info::source, and usage().

6.10.3 Variable Documentation

6.10.3.1 char* program_name [static]

Name of the program as invoked on the command line

Definition at line 72 of file fifo_interrupt.c.

Referenced by main(), and usage().

6.11 examples/fifo_pci_access.c File Reference

Example program which demonstrates how to use the FIFO block FIFOs with PCI read requests clocking data out of the FIFO and PCI write requests clocking data into the FIFO.

```
#include <errno.h>
#include <error.h>
#include <getopt.h>
#include <limits.h>
#include <stdint.h>
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include "dm7820_library.h"
```

Macros

- #define SDRAM_FIFO_ELEMENTS ((1024 * 1024 * 2) - 1)
- #define INPUT_FIFO_ELEMENTS (256 - 1)
- #define OUTPUT_FIFO_ELEMENTS (256 - 1)
- #define EXPECTED_FIFO_ELEMENTS
- #define MODULUS_DIVISOR 4321

Functions

- static void get_fifo_status (DM7820_Board_Descriptor *board, dm7820_fifo_queue fifo, dm7820_fifo_status_condition condition, uint8_t *status)

  Determine whether or not the specified status condition has occurred for the given FIFO.
- 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

6.11.1 Detailed Description

Example program which demonstrates how to use the FIFO block FIFOs with PCI read requests clocking data out of the FIFO and PCI write requests clocking data into the FIFO.

Note

This program shows how to access FIFOs at the most basic level.
Because this program does not use interrupts, it also demonstrates how to check FIFO status when interrupts are not used.

Warning

This program takes a long time to complete.

```
    This program uses FIFO 0, which is configured as follows: 1) input clock
    set to PCI write request, 2) output clock set to PCI read request, and
    3) data input set to PCI data.

    Values are written to FIFO 0 until the FIFO is full.  Then, values are
    read from FIFO 0 until the FIFO is empty.  While reading values, a check
    is made to see that the values are read back in the order they were
    written.

-----------------------------------------------------------------------------
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: fifo_pci_access.c 60252 2012-06-04 19:39:05Z rgroner $
```

Definition in file fifo_pci_access.c.

6.11.2 Macro Definition Documentation

6.11.2.1 #define EXPECTED_FIFO_ELEMENTS

Value:

```
( \
    SDRAM_FIFO_ELEMENTS + INPUT_FIFO_ELEMENTS +
      OUTPUT_FIFO_ELEMENTS \
)
```

Expected number of 16-bit FIFO elements

Definition at line 86 of file fifo_pci_access.c.

Referenced by main().

6.11.2.2 #define INPUT_FIFO_ELEMENTS (256 - 1)

256 16-bit elements in input FIFO; one element is lost for hardware to determine full status

Definition at line 73 of file fifo_pci_access.c.

6.11.2.3 #define MODULUS_DIVISOR 4321

Divisor for modulus operator; used when writing data to FIFO that can be easily recalculated when reading data from FIFO

Definition at line 95 of file fifo_pci_access.c.

Referenced by main().

6.11.2.4 #define OUTPUT_FIFO_ELEMENTS (256 - 1)

256 16-bit elements in output FIFO; one element is lost for hardware to determine full status

Definition at line 80 of file fifo_pci_access.c.

6.11.2.5 #define SDRAM_FIFO_ELEMENTS ((1024 * 1024 * 2) - 1)

2 megabyte 16-bit elements in SDRAM FIFO; one element is lost for hardware to determine full status

Definition at line 65 of file fifo_pci_access.c.

6.11.3 Function Documentation

6.11.3.1 static void get_fifo_status (DM7820_Board_Descriptor * *board*, dm7820_fifo_queue *fifo*, dm7820_fifo_status_condition *condition*, uint8_t * *status*) [static]

Determine whether or not the specified status condition has occurred for the given FIFO.

Parameters

board	Address of device's library board descriptor.
fifo	The FIFO to determine status of.
condition	The status condition to check for.
status	Address where occurrence flag should be stored.

Definition at line 140 of file fifo_pci_access.c.

References DM7820_FIFO_Get_Status().

Referenced by main().

6.11.3.2 int main (int *argument_count*, char ** *arguments*)

Main program code.

Parameters

argument_count	Number of command line arguments passed to executable.
arguments	Address of array containing command line arguments.

Return values

EXIT_SUCCESS	Success.
EXIT_FAILURE	Failure.

Definition at line 200 of file fifo_pci_access.c.

References board, DM7820_FIFO_0_DATA_INPUT_PCI_DATA, DM7820_FIFO_Enable(), DM7820_FIFO_INPUT_CLOCK_PCI_WRITE, DM7820_FIFO_OUTPUT_CLOCK_PCI_READ, DM7820_FIFO_QUEUE_0, DM7820_FIFO_QUEUE_1, DM7820_FIFO_Read(), DM7820_FIFO_Set_Data_Input(), DM7820_FIFO_Set_Input_Clock(), DM7820_FIFO_Set_Output_Clock(), DM7820_FIFO_STATUS_EMPTY, DM7820_FIFO_STATUS_FULL, DM7820_FIFO_STATUS_OVERFLOW, DM7820_FIFO_STATUS_UNDERFLOW, DM7820_FIFO_Write(), DM7820_General_Close_Board(), DM7820_General_Open_Board(), DM7820_Return_Status, EXPECTED_FIFO_ELEMENTS, get_fifo_status(), MODULUS_DIVISOR, program_name, and usage().

6.11.4 Variable Documentation

6.11.4.1 char* program_name [static]

Name of the program as invoked on the command line

Definition at line 105 of file fifo_pci_access.c.

Referenced by main(), and usage().

6.12 examples/incenc_encoders.c File Reference

Example program which demonstrates how to use the incremental encoder block encoders.

```
#include <errno.h>
#include <error.h>
#include <getopt.h>
#include <limits.h>
#include <signal.h>
#include <stdint.h>
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include "dm7820_library.h"
```

Functions

- static void sigint_handler (int signal_number)

  Signal handler for SIGINT Control-C keyboard interrupt.
- 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
- static volatile sig_atomic_t exit_program = 0

6.12.1 Detailed Description

Example program which demonstrates how to use the incremental encoder block encoders.

```
Incremental encoders 0 and 1 are configured as follows: 1) master clock
set to 25 MHz clock, 2) value register hold disabled, 3) single-ended
inputs, 4) input filter disabled, 5) independent A/B channels, and 6)
index input disabled.

Incremental encoder 0 is further configured with an initial value of
0xFFFF and a phase filter to disable counter change on up transitions.

Incremental encoder 1 is further configured with an initial value of
0x0000 and a phase filter to disable counter change on down transitions.

This program assumes a single incremental encoder is connected to
incremental encoder 0 channel A and to incremental encoder 1 channel A.
The incremental encoder should be connected as follows:

    Encoder Pin     CN11 Pin    CN10 Pin
```

```
==================================
A               47          47
B               43          43
-               N/A         50
+               N/A         49

With the setup indicated above, encoder 0 channel A counts down only
when the incremental encoder is rotated in one direction and encoder 1
channel A counts up only when the incremental encoder is rotated in the
opposite direction.

Channel B on incremental encoders 0 and 1 is not used.
```

Note

Because all four phase filter transitions are enabled in a particular counting direction (up or down), the counter value will update at four times the incremental encoder "cycles per revolution" rate.

Id:

incenc_encoders.c 79285 2014-05-21 20:37:42Z rgroner

Definition in file incenc_encoders.c.

6.12.2 Function Documentation

6.12.2.1 int main (int *argument_count*, char ** *arguments*)

Main program code.

Parameters

argument_count	Number of command line arguments passed to executable.
arguments	Address of array containing command line arguments.

Return values

EXIT_SUCCESS	Success.
EXIT_FAILURE	Failure.

Definition at line 179 of file incenc_encoders.c.

References board, DM7820_General_Close_Board(), DM7820_General_Open_Board(), DM7820_INCENC_CHANNEL_A, DM7820_INCENC_CHANNEL_INDEPENDENT, DM7820_IncEnc_Configure(), DM7820_INCENC_DISABLE_PHASE_FILTER_TRANSITION, DM7820_IncEnc_Enable(), DM7820_IncEnc_Enable_Hold(), DM7820_INCENC_ENCODER_0, DM7820_INCENC_ENCODER_1, DM7820_IncEnc_Get_Independent_Value(), DM7820_INCENC_INPUT_SINGLE_ENDED, DM7820_INCENC_MASTER_25_MHZ, DM7820_INCENC_PHASE_BA_00_TO_01_UP, DM7820_INCENC_PHASE_BA_00_TO_10_DOWN, DM7820_INCENC_PHASE_BA_01_TO_00_DOWN, DM7820_INCENC_PHASE_BA_01_TO_11_UP, DM7820_INCENC_PHASE_BA_10_TO_00_UP, DM7820_INCENC_PHASE_BA_10_TO_11_DOWN, DM7820_INCENC_PHASE_BA_11_TO_01_DOWN, DM7820_INCENC_PHASE_BA_11_TO_10_UP, DM7820_INCENC_RESET_PHASE_FILTER, DM7820_IncEnc_Set_Independent_Value(), DM7820_IncEnc_Set_Master(), DM7820_Return_Status, DM7820_STDIO_MODE_INPUT, DM7820_ST-

DIO_PORT_0, DM7820_STDIO_PORT_1, DM7820_StdIO_Set_IO_Mode(), exit_program, program_name, sigint-_handler(), and usage().

6.12.2.2 static void sigint_handler (int *signal_number*) [static]

Signal handler for SIGINT Control-C keyboard interrupt.

Parameters

signal_number	Signal number passed in from kernel.

Warning

One must be extremely careful about what functions are called from a signal handler. printf() and related functions are considered unsafe for use in signal handlers. Therefore, this function uses write() instead.

Definition at line 109 of file incenc_encoders.c.

References exit_program.

Referenced by main().

6.12.3 Variable Documentation

6.12.3.1 volatile sig_atomic_t exit_program = 0 [static]

Flag used by SIGINT signal handler to tell main program to exit because the user hit Control-C

Definition at line 86 of file incenc_encoders.c.

Referenced by main(), and sigint_handler().

6.12.3.2 char* program_name [static]

Name of the program as invoked on the command line

Definition at line 79 of file incenc_encoders.c.

Referenced by main(), and usage().

6.13 examples/incenc_interrupt.c File Reference

Example program which demonstrates how to use the incremental encoder block interrupts.

```
#include <errno.h>
#include <error.h>
#include <getopt.h>
#include <limits.h>
#include <stdint.h>
#include <stdio.h>
#include <stdlib.h>
#include <strings.h>
#include <unistd.h>
#include "dm7820_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

6.13.1 Detailed Description

Example program which demonstrates how to use the incremental encoder block interrupts.

```
Incremental encoder 0 is configured as follows: 1) master clock set to
25 MHz clock, 2) value register hold disabled, 3) single-ended inputs,
4) input filter disabled, 5) independent A/B channels, and 6) index
input disabled.

Incremental encoder 0 is further configured with an initial value of
0x8000 and a phase filter to enable counter change on all transitions.

This program assumes an incremental encoder is connected to incremental
encoder 0 channel A.  The incremental encoder should be connected as
follows:

    Encoder Pin     CN10 Pin
    ========================
    A               47
    B               43
    -               50
    +               49

Channel B on incremental encoder 0 is not used.

This program uses incremental encoder 0 channel A negative & positive
rollover interrupts and waits for the interrupts to occur.  One such
interrupt is waited on and then the program exits.
```

Note

Because all phase filter transitions are enabled in a particular counting direction (up or down), the counter value will update at four times the incremental encoder "cycles per revolution" rate.
This program does not display the channel A value because sleepy waiting may be used.

Id:

incenc_interrupt.c 60252 2012-06-04 19:39:05Z rgroner

Definition in file incenc_interrupt.c.

6.13.2 Function Documentation

6.13.2.1 int main (int *argument_count*, char ** *arguments*)

Main program code.

Parameters

argument_count	Number of command line arguments passed to executable.
arguments	Address of array containing command line arguments.

Return values

EXIT_SUCCESS	Success.
EXIT_FAILURE	Failure.

Definition at line 142 of file incenc_interrupt.c.

References board, DM7820_General_Close_Board(), DM7820_General_Enable_Interrupt(), DM7820_General_Get_Interrupt_Status(), DM7820_General_Open_Board(), DM7820_General_Reset(), DM7820_INCENC_CHANNEL_A, DM7820_INCENC_CHANNEL_INDEPENDENT, DM7820_IncEnc_Configure(), DM7820_IncEnc_Enable(), DM7820_IncEnc_Enable_Hold(), DM7820_INCENC_ENCODER_0, DM7820_INCENC_INPUT_SINGLE_ENDED, DM7820_INCENC_MASTER_25_MHZ, DM7820_INCENC_RESET_PHASE_FILTER, DM7820_IncEnc_Set_Independent_Value(), DM7820_IncEnc_Set_Master(), DM7820_INTERRUPT_INCENC_0_CHANNEL_A_NEGATIVE_ROLLOVER, DM7820_INTERRUPT_INCENC_0_CHANNEL_A_POSITIVE_ROLLOVER, DM7820_Return_Status, DM7820_STDIO_MODE_INPUT, DM7820_STDIO_PORT_0, DM7820_StdIO_Set_IO_Mode(), program_name, _dm7820_interrupt_info::source, and usage().

6.13.3 Variable Documentation

6.13.3.1 char* program_name [static]

Name of the program as invoked on the command line

Definition at line 79 of file incenc_interrupt.c.

Referenced by main(), and usage().

6.14 examples/incenc_status.c File Reference

Example program which demonstrates how to get incremental encoder status when not using interrupts.

```
#include <errno.h>
#include <error.h>
#include <getopt.h>
#include <limits.h>
#include <stdint.h>
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include "dm7820_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

Example program which demonstrates how to get incremental encoder status when not using interrupts.

```
Incremental encoder 0 is configured as follows: 1) master clock set to
25 MHz clock, 2) value register hold disabled, 3) single-ended inputs,
4) input filter disabled, 5) independent A/B channels, and 6) index
input disabled.

Incremental encoder 0 is further configured with an initial value of
0x8000 and a phase filter to enable counter change on all transitions.

This program assumes an incremental encoder is connected to incremental
encoder 0 channel A.  The incremental encoder should be connected as
follows:

    Encoder Pin     CN10 Pin
    ========================
    A               47
    B               43
    -               50
    +               49

Encoder 0 channel A values are read and printed until either negative
rollover or positive rollover occurs.

Channel B on incremental encoder 0 is not used.
```

Note

Because all phase filter transitions are enabled in a particular counting direction (up or down), the counter value will update at four times the incremental encoder "cycles per revolution" rate.

```
-------------------------------------------------------------------------
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:

incenc_status.c 60252 2012-06-04 19:39:05Z rgroner

Definition in file incenc_status.c.

6.14.2 Function Documentation

6.14.2.1 int main (int *argument_count*, char ** *arguments*)

Main program code.

Parameters

argument_count	Number of command line arguments passed to executable.
arguments	Address of array containing command line arguments.

Return values

EXIT_SUCCESS	Success.
EXIT_FAILURE	Failure.

Definition at line 129 of file incenc_status.c.

References board, DM7820_General_Close_Board(), DM7820_General_Open_Board(), DM7820_INCENC_CHANNEL_A, DM7820_INCENC_CHANNEL_INDEPENDENT, DM7820_IncEnc_Configure(), DM7820_IncEnc_Enable(), DM7820_IncEnc_Enable_Hold(), DM7820_INCENC_ENCODER_0, DM7820_IncEnc_Get_Independent_Value(), DM7820_IncEnc_Get_Status(), DM7820_INCENC_INPUT_SINGLE_ENDED, DM7820_INCENC_MASTER_25_MHZ, DM7820_INCENC_RESET_PHASE_FILTER, DM7820_IncEnc_Set_Independent_Value(), DM7820_IncEnc_Set_Master(), DM7820_INCENC_STATUS_CHANNEL_A_NEGATIVE_ROLLOVER, DM7820_INCENC_STATUS_CHANNEL_A_POSITIVE_ROLLOVER, DM7820_Return_Status, DM7820_STDIO_MODE_INPUT, DM7820_STDIO_PORT_0, DM7820_StdIO_Set_IO_Mode(), program_name, and usage().

6.14.3 Variable Documentation

6.14.3.1 char* program_name [static]

Name of the program as invoked on the command line

Definition at line 73 of file incenc_status.c.

Referenced by main(), and usage().

6.15 examples/library_test.c File Reference

Program which tests the basic functionality of the user library.

```
#include <errno.h>
#include <error.h>
#include <fcntl.h>
#include <getopt.h>
#include <limits.h>
#include <stdint.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/ioctl.h>
#include <sys/stat.h>
#include <sys/sysmacros.h>
#include <sys/time.h>
#include <sys/types.h>
#include <sys/utsname.h>
#include <unistd.h>
#include "dm7820_ioctl.h"
#include "dm7820_library.h"
#include "dm7820_registers.h"
```

Macros

- #define DM7820_MAX_DMA_BUFFER_COUNT 32
- #define DM7820_MAX_DMA_BUFFER_SIZE 0x20000

Typedefs

- typedef enum initialization_state initialization_state_t

Enumerations

- enum initialization_state {
  INIT_NO_INITIALIZATION = 0x00000000, INIT_BOARD_ARRAY_ALLOCATED = 0x00000004, INIT_BAD-
  _DEVICE_FILE_CREATED = 0x00000008, INIT_BOARD_ARRAY_OPENED = 0x00000010,
  INIT_NO_INITIALIZATION = 0x00000000, INIT_BOARD_ARRAY_ALLOCATED = 0x00000004, INIT_BAD-
  _DEVICE_FILE_CREATED = 0x00000008, INIT_BOARD_ARRAY_OPENED = 0x00000010 }

Functions

- static void cleanup (void)

  Perform actions necessary to clean up after the program encounters a fatal error.
- static void expect_failure_and_check (DM7820_Error status, int expected_errno)

  Verify that a function being tested failed with the given errno.
- static void expect_success (DM7820_Error status)

  Verify that a function being tested succeeded.
- 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
- static volatile
  initialization_state_t init_state = INIT_NO_INITIALIZATION
- static volatile
  DM7820_Board_Descriptor ** boards
- static volatile char bad_device_name [30]
- static volatile unsigned long device_count = 1

6.15.1 Detailed Description

Program which tests the basic functionality of the user library.

Id:

library_test.c 86306 2015-03-05 15:27:53Z rgroner

Definition in file library_test.c.

6.15.2 Macro Definition Documentation

6.15.2.1 #define DM7820_MAX_DMA_BUFFER_COUNT 32

Default driver limit on number of DMA buffers that can be allocated per DMA/FIFO channel

Definition at line 52 of file library_test.c.

Referenced by main().

6.15.2.2 #define DM7820_MAX_DMA_BUFFER_SIZE 0x20000

Default driver limit in bytes on size of DMA buffers that can be allocated per DMA/FIFO channel

Definition at line 59 of file library_test.c.

Referenced by main().

6.15.3 Typedef Documentation

6.15.3.1 typedef enum initialization_state initialization_state_t

Program initialization status type

Definition at line 106 of file library_test.c.

6.15.4 Enumeration Type Documentation

6.15.4.1 enum initialization_state

Status of program initialization. Used to indication which operations need to be undone when cleaning up after an error. This is a bit mask, so only one bit should be set for each enumeration value.

Enumerator:

INIT_NO_INITIALIZATION No initialization performed

INIT_BOARD_ARRAY_ALLOCATED File descriptor array allocated

INIT_BAD_DEVICE_FILE_CREATED /dev entry with invalid minor number created

INIT_BOARD_ARRAY_OPENED File descriptor array contains at least one opened file

INIT_NO_INITIALIZATION No initialization performed

INIT_BOARD_ARRAY_ALLOCATED Board descriptor array allocated

INIT_BAD_DEVICE_FILE_CREATED /dev entry with invalid minor number created

INIT_BOARD_ARRAY_OPENED Board descriptor array contains at least one opened file

Definition at line 71 of file library_test.c.

6.15.5 Function Documentation

6.15.5.1 static void expect_failure_and_check (DM7820_Error *status,* int *expected_errno*) [static]

Verify that a function being tested failed with the given errno.

Parameters

status	Return code from function being tested.
expected_errno	Expected errno that function should have set.

Definition at line 190 of file library_test.c.

References cleanup().

Referenced by main().

6.15.5.2 static void expect_success (DM7820_Error *status*) [static]

Verify that a function being tested succeeded.

Parameters

status	Return code from function being tested.

Definition at line 218 of file library_test.c.

References cleanup().

Referenced by main().

6.15.5.3 int main (int *argument_count,* char ** *arguments*)

Main program code.

Parameters

argument_count	Number of command line arguments passed to executable.
arguments	Address of array containing command line arguments.

Return values

EXIT_SUCCESS	Success.
EXIT_FAILURE	Failure.

Definition at line 276 of file library_test.c.

References dm7820_ioctl_region_readwrite::access, board, cleanup(), dm7820_pci_access_request::data, dm7820_pci_access_request::data16, device_count, DM7820_AdvInt_Get_Status(), DM7820_ADVINT_INTERRUPT_0, DM7820_ADVINT_INTERRUPT_1, DM7820_ADVINT_MASTER_25_MHZ, DM7820_ADVINT_MASTER_INV_STROBE_2, DM7820_ADVINT_MASTER_RESERVED, DM7820_ADVINT_MODE_DISABLED, DM7820_ADVINT_MODE_EVENT, DM7820_AdvInt_Read_Capture(), DM7820_AdvInt_Set_Compare(), DM7820_AdvInt_Set_Mask(), DM7820_AdvInt_Set_Master(), DM7820_AdvInt_Set_Mode(), DM7820_BAR2_BRD_STAT, DM7820_BAR2_PORT0_OUTPUT, DM7820_DMA_DEMAND_ON_PCI_TO_DM7820, DM7820_FIFO_0_DATA_INPUT_PCI_DATA, DM7820_FIFO_1_DATA_INPUT_INC_ENCODER_1_B, DM7820_FIFO_1_DATA_INPUT_PCI_DATA, DM7820_FIFO_DMA_Configure(), DM7820_FIFO_DMA_Initialize(), DM7820_FIFO_DMA_REQUEST_NOT_FULL, DM7820_FIFO_DMA_REQUEST_READ, DM7820_FIFO_Enable(), DM7820_FIFO_Get_Status(), DM7820_FIFO_INPUT_CLOCK_25_MHZ, DM7820_FIFO_INPUT_CLOCK_PCI_WRITE, DM7820_FIFO_INPUT_CLOCK_RESERVED_1, DM7820_FIFO_INPUT_CLOCK_RESERVED_2, DM7820_FIFO_INPUT_CLOCK_-

RESERVED_3, DM7820_FIFO_INPUT_CLOCK_RESERVED_4, DM7820_FIFO_OUTPUT_CLOCK_25_MHZ, DM7820_FIFO_OUTPUT_CLOCK_PCI_WRITE, DM7820_FIFO_OUTPUT_CLOCK_RESERVED_1, DM7820_FIFO_OUTPUT_CLOCK_RESERVED_2, DM7820_FIFO_OUTPUT_CLOCK_RESERVED_3, DM7820_FIFO_OUTPUT_CLOCK_RESERVED_4, DM7820_FIFO_QUEUE_0, DM7820_FIFO_QUEUE_1, DM7820_FIFO_Read(), DM7820_FIFO_Set_Data_Input(), DM7820_FIFO_Set_DMA_Request(), DM7820_FIFO_Set_Input_Clock(), DM7820_FIFO_Set_Output_Clock(), DM7820_FIFO_STATUS_EMPTY, DM7820_FIFO_STATUS_READ_REQUEST, DM7820_FIFO_STATUS_UNDERFLOW, DM7820_FIFO_Write(), DM7820_General_Close_Board(), DM7820_General_Enable_Interrupt(), DM7820_General_Get_Interrupt_Status(), DM7820_General_Get_Version_Info(), DM7820_General_Is_PCI_Master(), DM7820_General_Open_Board(), DM7820_General_Reset(), DM7820_INCENC_CHANNEL_A, DM7820_INCENC_CHANNEL_B, DM7820_INCENC_CHANNEL_INDEPENDENT, DM7820_INCENC_CHANNEL_JOINED, DM7820_IncEnc_Configure(), DM7820_IncEnc_Enable(), DM7820_IncEnc_Enable_Hold(), DM7820_INCENC_ENCODER_0, DM7820_INCENC_ENCODER_1, DM7820_IncEnc_Get_Independent_Value(), DM7820_IncEnc_Get_Joined_Value(), DM7820_IncEnc_Get_Status(), DM7820_INCENC_INPUT_DIFFERENTIAL, DM7820_INCENC_INPUT_SINGLE_ENDED, DM7820_INCENC_MASTER_25_MHZ, DM7820_INCENC_MASTER_INV_STROBE_2, DM7820_INCENC_MASTER_RESERVED, DM7820_IncEnc_Set_Independent_Value(), DM7820_IncEnc_Set_Joined_Value(), DM7820_IncEnc_Set_Master(), DM7820_INCENC_STATUS_CHANNEL_A_POSITIVE_ROLLOVER, DM7820_INCENC_STATUS_CHANNEL_B_NEGATIVE_ROLLOVER, DM7820_INTERRUPT_ADVINT_0, DM7820_INTERRUPT_FIFO_0_DMA_DONE, DM7820_INTERRUPT_FIFO_1_DMA_DONE, DM7820_IOCTL_REGION_READ, DM7820_IOCTL_REGION_WRITE, DM7820_MAX_DMA_BUFFER_COUNT, DM7820_MAX_DMA_BUFFER_SIZE, DM7820_PCI_REGION_ACCESS_16, DM7820_PCI_REGION_FPGA_MEM, DM7820_PRGCLK_CLOCK_0, DM7820_PRGCLK_CLOCK_3, DM7820_PRGCLK_MASTER_25_MHZ, DM7820_PRGCLK_MASTER_INV_STROBE_2, DM7820_PRGCLK_MODE_CONTINUOUS, DM7820_PRGCLK_MODE_DISABLED, DM7820_PRGCLK_MODE_ONE_SHOT, DM7820_PRGCLK_MODE_RESERVED, DM7820_PrgClk_Set_Master(), DM7820_PrgClk_Set_Mode(), DM7820_PrgClk_Set_Period(), DM7820_PrgClk_Set_Start_Trigger(), DM7820_PrgClk_Set_Stop_Trigger(), DM7820_PRGCLK_START_FIFO_1_INT, DM7820_PRGCLK_START_IMMEDIATE, DM7820_PRGCLK_START_RESERVED_1, DM7820_PRGCLK_START_RESERVED_2, DM7820_PRGCLK_START_RESERVED_3, DM7820_PRGCLK_START_RESERVED_4, DM7820_PRGCLK_STOP_FIFO_1_INT, DM7820_PRGCLK_STOP_NONE, DM7820_PRGCLK_STOP_RESERVED_1, DM7820_PRGCLK_STOP_RESERVED_2, DM7820_PRGCLK_STOP_RESERVED_3, DM7820_PRGCLK_STOP_RESERVED_4, DM7820_PWM_Enable(), DM7820_PWM_OUTPUT_A, DM7820_PWM_OUTPUT_D, DM7820_PWM_PERIOD_MASTER_25_MHZ, DM7820_PWM_PERIOD_MASTER_INV_STROBE_2, DM7820_PWM_PERIOD_MASTER_RESERVED, DM7820_PWM_Set_Period(), DM7820_PWM_Set_Period_Master(), DM7820_PWM_Set_Width(), DM7820_PWM_Set_Width_Master(), DM7820_PWM_WIDTH_MASTER_25_MHZ, DM7820_PWM_WIDTH_MASTER_INV_STROBE_2, DM7820_PWM_WIDTH_MASTER_RESERVED, DM7820_StdIO_Get_Input(), DM7820_STDIO_MODE_INPUT, DM7820_STDIO_MODE_PER_OUT, DM7820_STDIO_PERIPH_CLK_OTHER, DM7820_STDIO_PERIPH_FIFO_0, DM7820_STDIO_PERIPH_FIFO_1, DM7820_STDIO_PERIPH_PWM, DM7820_STDIO_PORT_0, DM7820_STDIO_PORT_1, DM7820_STDIO_PORT_2, DM7820_StdIO_Set_IO_Mode(), DM7820_StdIO_Set_Output(), DM7820_StdIO_Set_Periph_Mode(), DM7820_STDIO_STROBE_1, DM7820_STDIO_STROBE_2, DM7820_StdIO_Strobe_Input(), DM7820_StdIO_Strobe_Mode(), DM7820_StdIO_Strobe_Output(), DM7820_TMRCTR_CLOCK_5_MHZ, DM7820_TMRCTR_CLOCK_INV_STROBE_2, DM7820_TMRCTR_CLOCK_RESERVED, DM7820_TMRCTR_COUNT_MODE_BCD, DM7820_TMRCTR_COUNT_MODE_BINARY, DM7820_TMRCTR_GATE_LOGIC_0, DM7820_TMRCTR_GATE_PORT_2_BIT_15, DM7820_TmrCtr_Get_Status(), DM7820_TmrCtr_Program(), DM7820_TmrCtr_Read(), DM7820_TmrCtr_Select_Clock(), DM7820_TmrCtr_Select_Gate(), DM7820_TMRCTR_TIMER_A_0, DM7820_TMRCTR_TIMER_B_2, DM7820_TMRCTR_WAVEFORM_EVENT_CTR, DM7820_TMRCTR_WAVEFORM_HARDWARE_STROBE, expect_failure_and_check(), expect_success(), DM7820_Board_Descriptor::file_descriptor, INIT_BOARD_ARRAY_ALLOCATED, INIT_BOARD_ARRAY_OPENED, init_state, dm7820_pci_access_request::offset, program_name, dm7820_ioctl_argument::readwrite, dm7820_pci_access_request::region, dm7820_pci_access_request::size, and usage().

6.15.6 Variable Documentation

6.15.6.1 volatile char bad_device_name[30] [static]

Path name of DM7820 device file with invalid minor number

Definition at line 134 of file library_test.c.

Referenced by cleanup().

6.15.6.2 volatile DM7820_Board_Descriptor** boards [static]

Array of DM7820 library device descriptors

Definition at line 128 of file library_test.c.

6.15.6.3 volatile unsigned long device_count = 1 [static]

Number of devices found when driver was loaded

Definition at line 140 of file library_test.c.

Referenced by main().

6.15.6.4 volatile initialization_state_t init_state = INIT_NO_INITIALIZATION [static]

Program initialization state

Definition at line 122 of file library_test.c.

Referenced by cleanup(), and main().

6.15.6.5 char* program_name [static]

Name of the program as invoked on the command line

Definition at line 116 of file library_test.c.

Referenced by main(), and usage().

6.16 examples/loopback.c File Reference

Example program which demonstrates how to use the FIFO block interrupts. Note: This example requires a loopback cable!

```
#include <errno.h>
#include <error.h>
#include <getopt.h>
#include <limits.h>
#include <stdint.h>
#include <stdio.h>
#include <stdlib.h>
#include <signal.h>
#include <strings.h>
#include <string.h>
#include <unistd.h>
#include "dm7820_library.h"
```

Macros

- #define BUF_SIZE 0x40000
- #define BUF_NUM 16
- #define SAMPLES ((BUF_SIZE * BUF_NUM) / 2)
- #define DMA_BUF_SIZE (BUF_SIZE * BUF_NUM)

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 (dm7820_interrupt_info interrupt_info)

  Userspace ISR.
- void clean_up ()

  Disable and clean up.
- int main (int argument_count, char **arguments)

  Main program code.

Variables

- static char * program_name
- volatile int dma_done_interrupts = 0
- volatile int fifo0_empty_interrupts = 0
- DM7820_Board_Descriptor * board

6.16.1 Detailed Description

Example program which demonstrates how to use the FIFO block interrupts. Note: This example requires a loopback cable!

```
    The Program does the following
    1) Allocate a 4MB Block of output data.
    2) Allocate a 4MB input buffer.
    3) FIFO 0:
        a. Input Clock = PCI Write
        b. Output Clock = Prog Clock 0
        c. In Data = PCI Data
        d. DREQ = Write Request
    4) FIFO 1:
        a. Input Clock = Prog Clock 0
        b. Out Clock = PCI Read
        c. In Data = Port 1
        d. DREQ = Read Request
    5) Port 0:
        a. All Outputs
        b. Select FIFO 0 as peripheral
    6) Prog Clock 0:
        a. Continuous
        b. Clock Source = 25MHz
        c. Period = 3 (results in 6.25 MHz Clock)
    7) Setup and starts both DMA's.
    8) Wait until FIFO 0 is not empty.
    9) Clock first data value to loopback
        a. Change FIFO 0 output clock to PCI Read
        b. Read from FIFO 0 data register
        c. Change FIFO 0 output clock to Prog Clock 0
    10) Start Prog Clock 0
    11) Wait until DMA1 is complete
    13) Test the Data
    12) Turn off Prog Clock 0 and the FIFO's


    Warning:  This example program is only intended to run on a system with
    enough RAM to support the large buffer's which will be allocated.
    Preferably 128MB minimum.

-------------------------------------------------------------------------
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:

loopback.c 60252 2012-06-04 19:39:05Z rgroner

Definition in file loopback.c.

6.16.2 Macro Definition Documentation

6.16.2.1 #define BUF_NUM 16

The number of buffers we want for each DMA channel.

Definition at line 119 of file loopback.c.

Referenced by main().

6.16.2.2 #define BUF_SIZE 0x40000

The size of the individual buffers we are asking to driver to create for each DMA channel.

Definition at line 113 of file loopback.c.

Referenced by main().

6.16.2.3 #define DMA_BUF_SIZE (BUF_SIZE * BUF_NUM)

The size of the buffer required to handle SAMPLES number of samples.

Definition at line 131 of file loopback.c.

Referenced by main().

6.16.2.4 #define SAMPLES ((BUF_SIZE * BUF_NUM) / 2)

The number of samples that can be held by the list of buffers we created.

Definition at line 125 of file loopback.c.

Referenced by main().

6.16.3 Function Documentation

6.16.3.1 void ISR (dm7820_interrupt_info *interrupt_info*)

Userspace ISR.

Parameters

interrupt_info	Information about the interrupt, returned by the kernel driver

Definition at line 174 of file loopback.c.

References DM7820_INTERRUPT_FIFO_0_EMPTY, DM7820_INTERRUPT_FIFO_1_DMA_DONE, DM7820_Return_Status, dma_done_interrupts, _dm7820_interrupt_info::error, fifo0_empty_interrupts, and _dm7820_interrupt_info::source.

6.16.3.2 int main (int *argument_count,* char ** *arguments*)

Main program code.

Parameters

argument_count	Number of command line arguments passed to executable.
arguments	Address of array containing command line arguments.

Return values

EXIT_SUCCESS	Success.
EXIT_FAILURE	Failure.

Definition at line 276 of file loopback.c.

References BUF_NUM, BUF_SIZE, clean_up(), DM7820_DMA_DEMAND_ON_DM7820_TO_PCI, DM7820_DMA_DEMAND_ON_PCI_TO_DM7820, DM7820_FIFO_0_DATA_INPUT_PCI_DATA, DM7820_FIFO_1_DATA_INPUT_PORT_1, DM7820_FIFO_DMA_Configure(), DM7820_FIFO_DMA_Create_Buffer(), DM7820_FIFO_DMA_Enable(), DM7820_FIFO_DMA_Free_Buffer(), DM7820_FIFO_DMA_Initialize(), DM7820_FIFO_DMA_Read(), DM7820_FIFO_DMA_REQUEST_READ, DM7820_FIFO_DMA_REQUEST_WRITE, DM7820_FIFO_DMA_Write(), DM7820_FIFO_Enable(), DM7820_FIFO_Get_Status(), DM7820_FIFO_INPUT_CLOCK_PCI_READ, DM7820_FIFO_INPUT_CLOCK_PCI_WRITE, DM7820_FIFO_INPUT_CLOCK_PROG_CLOCK_0, DM7820_FIFO_OUTPUT_CLOCK_PROG_CLOCK_0, DM7820_FIFO_QUEUE_0, DM7820_FIFO_QUEUE_1, DM7820_FIFO_Read(), DM7820_FIFO_Set_Data_Input(), DM7820_FIFO_Set_DMA_Request(), DM7820_FIFO_Set_Input_Clock(), DM7820_FIFO_Set_Output_Clock(), DM7820_FIFO_STATUS_EMPTY, DM7820_General_Enable_Interrupt(), DM7820_General_Open_Board(), DM7820_General_RemoveISR(), DM7820_General_Reset(), DM7820_General_SetISRPriority(), DM7820_INTERRUPT_FIFO_0_EMPTY, DM7820_INTERRUPT_FIFO_1_EMPTY, DM7820_PRGCLK_CLOCK_0, DM7820_PRGCLK_MASTER_25_MHZ, DM7820_PRGCLK_MODE_CONTINUOUS, DM7820_PRGCLK_MODE_DISABLED, DM7820_PrgClk_Set_Master(), DM7820_PrgClk_Set_Mode(), DM7820_PrgClk_Set_Period(), DM7820_PrgClk_Set_Start_Trigger(), DM7820_PrgClk_Set_Stop_Trigger(), DM7820_PRGCLK_START_IMMEDIATE, DM7820_PRGCLK_STOP_NONE, DM7820_Return_Status, DM7820_STDIO_MODE_INPUT, DM7820_STDIO_MODE_PER_OUT, DM7820_STDIO_PERIPH_FIFO_0, DM7820_STDIO_PERIPH_FIFO_1, DM7820_STDIO_PORT_0, DM7820_STDIO_PORT_1, DM7820_StdIO_Set_IO_Mode(), DM7820_StdIO_Set_Periph_Mode(), DMA_BUF_SIZE, dma_done_interrupts, fifo0_empty_interrupts, ISR(), program_name, SAMPLES, and usage().

6.16.4 Variable Documentation

6.16.4.1 DM7820_Board_Descriptor* board

Device descriptor

Definition at line 106 of file loopback.c.

6.16.4.2 volatile int dma_done_interrupts = 0

Variable to count the number of DMA Done interrupts occurring

Definition at line 94 of file loopback.c.

6.16.4.3 volatile int fifo0_empty_interrupts = 0

Variable to count the number of FIFO0 empty interrupts occurring

Definition at line 100 of file loopback.c.

Referenced by ISR(), and main().

6.16.4.4 char* program_name [static]

Name of the program as invoked on the command line

Definition at line 88 of file loopback.c.

Referenced by main(), and usage().

6.17 examples/prgclk_clocks.c File Reference

Example program which demonstrates how to use the programmable clock block clocks.

```
#include <errno.h>
#include <error.h>
#include <getopt.h>
#include <limits.h>
#include <stdint.h>
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include "dm7820_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

6.17.1 Detailed Description

Example program which demonstrates how to use the programmable clock block clocks.

```
This program sets up programmable clocks 0 and 1 so that they run at the
same frequency but approximately 180 degrees out of phase.  The phase
shift is only approximate because the delays inherent in starting clock
2 via clock 0 and starting clock 1 via clock 2 must be considered when
setting clock 2's period.  Clock 2 is used to delay clock 1 relative to
clock 0.

The programmable clocks are set up as follows:

              Start       Stop
    Clock     Trigger     Trigger     Mode        Frequency
    ========================================================
    0         immediate   none        continuous  50 KHz
    1         clock 2     none        continuous  50 KHz
    2         clock 0     none        one-shot    100 KHz

The above clocks are set to use the 25 MHz clock as their master clocks.

The standard I/O block port 2 pins are set up to output the programmable
clocks so that their frequencies may be measured.
```

```
    ------------------------------------------------------------------------
    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: prgclk_clocks.c 60252 2012-06-04 19:39:05Z rgroner $
```

Definition in file prgclk_clocks.c.

6.17.2 Function Documentation

6.17.2.1 int main (int *argument_count*, char ** *arguments*)

Main program code.

Parameters

argument_count	Number of command line arguments passed to executable.
arguments	Address of array containing command line arguments.

Return values

EXIT_SUCCESS	Success.
EXIT_FAILURE	Failure.

Definition at line 121 of file prgclk_clocks.c.

References board, DM7820_General_Close_Board(), DM7820_General_Open_Board(), DM7820_PRGCLK_CLOCK_0, DM7820_PRGCLK_CLOCK_1, DM7820_PRGCLK_CLOCK_2, DM7820_PRGCLK_CLOCK_3, DM7820_PRGCLK_MASTER_25_MHZ, DM7820_PRGCLK_MODE_CONTINUOUS, DM7820_PRGCLK_MODE_DISABLED, DM7820_PRGCLK_MODE_ONE_SHOT, DM7820_PrgClk_Set_Master(), DM7820_PrgClk_Set_Mode(), DM7820_PrgClk_Set_Period(), DM7820_PrgClk_Set_Start_Trigger(), DM7820_PrgClk_Set_Stop_Trigger(), DM7820_PRGCLK_START_IMMEDIATE, DM7820_PRGCLK_START_PROG_CLOCK_0, DM7820_PRGCLK_START_PROG_CLOCK_2, DM7820_PRGCLK_STOP_NONE, DM7820_Return_Status, DM7820_STDIO_MODE_PER_OUT, DM7820_STDIO_PERIPH_CLK_OTHER, DM7820_STDIO_PORT_2, DM7820_StdIO_Set_IO_Mode(), DM7820_StdIO_Set_Periph_Mode(), program_name, and usage().

6.17.3 Variable Documentation

6.17.3.1 char* program_name [static]

Name of the program as invoked on the command line

Definition at line 65 of file prgclk_clocks.c.

Referenced by main(), and usage().

6.18 examples/pwm_interrupt.c File Reference

Example program which demonstrates how to use the pulse width modulator block interrupts.

```
#include <errno.h>
#include <error.h>
#include <getopt.h>
#include <limits.h>
#include <stdint.h>
#include <stdio.h>
#include <stdlib.h>
#include <strings.h>
#include <unistd.h>
#include "dm7820_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

6.18.1 Detailed Description

Example program which demonstrates how to use the pulse width modulator block interrupts.

```
    Programmable clock 0 is set up as follows:

        Master      Start       Stop
        Clock       Trigger     Trigger     Mode        Frequency
        ==========================================================
        25 MHz      immediate   none        continuous  500 Hz

    This program configures pulse width modulator 0 as follows: 1) master
    clock set to programmable clock 0, 2) period set to obtain frequency of
    1 Hz, 3) width master clock set to programmable clock 0, and 4) output A
    width initially set to obtain 20% positive duty cycle.

    The standard I/O block port 2 pins are set up to output the pulse width
    modulators so that PWM 0 frequency and duty cycle may be measured.

    With the setup indicated about, PWM 0 generates an interrupt once a
    second.

    This program uses pulse width modulator 0 interrupts and waits for the
    interrupts to occur.  20 such interrupts are waited on and then the
    program exits.  After 10 interrupts occur, the PWM 0 output A width is
    changed to obtain 80% positive duty cycle.

-------------------------------------------------------------------------
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:

pwm_interrupt.c 60252 2012-06-04 19:39:05Z rgroner

Definition in file pwm_interrupt.c.

6.18.2 Function Documentation

6.18.2.1 int main (int *argument_count*, char ** *arguments*)

Main program code.

Parameters

argument_count	Number of command line arguments passed to executable.
arguments	Address of array containing command line arguments.

Return values

EXIT_SUCCESS	Success.
EXIT_FAILURE	Failure.

Definition at line 131 of file pwm_interrupt.c.

References board, DM7820_General_Close_Board(), DM7820_General_Enable_Interrupt(), DM7820_General_Get_Interrupt_Status(), DM7820_General_Open_Board(), DM7820_General_Reset(), DM7820_INTERRUPT_PWM_0, DM7820_PRGCLK_CLOCK_0, DM7820_PRGCLK_MASTER_25_MHZ, DM7820_PRGCLK_MODE_CONTINUOUS, DM7820_PRGCLK_MODE_DISABLED, DM7820_PrgClk_Set_Master(), DM7820_PrgClk_Set_Mode(), DM7820_PrgClk_Set_Period(), DM7820_PrgClk_Set_Start_Trigger(), DM7820_PrgClk_Set_Stop_Trigger(), DM7820_PRGCLK_START_IMMEDIATE, DM7820_PRGCLK_STOP_NONE, DM7820_PWM_Enable(), DM7820_PWM_OUTPUT_A, DM7820_PWM_PERIOD_MASTER_PROG_CLOCK_0, DM7820_PWM_Set_Period(), DM7820_PWM_Set_Period_Master(), DM7820_PWM_Set_Width(), DM7820_PWM_Set_Width_Master(), DM7820_PWM_WIDTH_MASTER_PROG_CLOCK_0, DM7820_Return_Status, DM7820_STDIO_MODE_PER_OUT, DM7820_STDIO_PERIPH_PWM, DM7820_STDIO_PORT_2, DM7820_StdIO_Set_IO_Mode(), DM7820_StdIO_Set_Periph_Mode(), program_name, _dm7820_interrupt_info::source, and usage().

6.18.3 Variable Documentation

6.18.3.1 char* program_name [static]

Name of the program as invoked on the command line

Definition at line 68 of file pwm_interrupt.c.

Referenced by main(), and usage().

6.19 examples/pwm_measure.c File Reference

This program demonstrates how an incoming pulse width can be measured using the DM7820.

```
#include <errno.h>
#include <error.h>
#include <getopt.h>
#include <limits.h>
#include <stdint.h>
#include <stdio.h>
#include <stdlib.h>
#include <signal.h>
#include <strings.h>
#include <string.h>
#include <unistd.h>
#include <fcntl.h>
#include "dm7820_library.h"
```

Macros

- #define BUF_SIZE 0x4000
- #define BUF_NUM 8
- #define SAMPLES ((BUF_SIZE * BUF_NUM)/2)
- #define DMA_BUF_SIZE (BUF_SIZE * BUF_NUM)
- #define UTC_RATE 2

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 (dm7820_interrupt_info interrupt_info)

  Userspace ISR.
- void clean_up ()

  Disable and clean up.
- static void sigint_handler (int signal_number)

  Signal handler for SIGINT Control-C keyboard interrupt.
- void do_digital_io ()

  Perform all digital I/O port initialization.
- void do_incenc ()

  Perform all incremental encoder initialization.
- void do_8254 ()

  Perform all 8254 Timer/Counter initialization.
- void do_pwm ()

  Perform all PWM initialization.
- void do_fifo ()

  Perform all FIFO initialization.
- void do_dma ()

  Perform all DMA initialization.
- int main (int argument_count, char **arguments)

  Main program code.

Variables

- static char * program_name
- volatile int dma_done_interrupts = 0
- DM7820_Board_Descriptor * board
- uint8_t exit_program = 0

6.19.1 Detailed Description

This program demonstrates how an incoming pulse width can be measured using the DM7820.

```
The example program implements the block diagram drawn below

                      5MHz
                     __|_
                    |    | out
              |-gate-|8254|------------o PIN -----o
              |      |____|   ______              | CONNECT
              |               |     A|--o PIN -----o
    PWM_IN ---|      |-count-|IncEnc|
   (strobe1)  |      |       |_____B|---o PIN -- connect to ground
              |      |        ______
              |      |-input-|      |
              |--------clock-| FIFO |
                             |______|

The pulse width coming from the position sensor should be connected to
strobe1 (Pin 2 on CN11).  This example uses PWM0 Output A (Pin 15 CN10) to
emulate the position sensor input on strobe1.

The strobe1 input is then internally used as the gate intput for
Timer/Counter A0.  Timer/Counter A0 out is then externally used as the input
for Incremental Encoder 1 Channel A (connect Pin 11 CN10 to Pin 47 CN11).
The incremental encoder's count value is then inserted into the FIFO.

Encoder count values are then read from the FIFO.  Taking a difference
from those values will give the number of 8254 User Timer/Counter periods
that have occured.  If that period is known value, the length of the Duty
Cycle can then be calculated.


    Warning:  This example program is only intended to run on a system with
    enough RAM to support the large buffer's which will be allocated.
    Preferably 128MB minimum.


--------------------------------------------------------------------------
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_capture_buffers.c 33597 2008-11-26 16:46:30Z wtate

Definition in file pwm_measure.c.

6.19.2 Macro Definition Documentation

6.19.2.1 #define BUF_NUM 8

The number of buffers we want for each DMA channel.

Definition at line 118 of file pwm_measure.c.

Referenced by do_dma().

6.19.2.2 #define BUF_SIZE 0x4000

The size of the individual buffers we are asking to driver to create for each DMA channel.

Definition at line 112 of file pwm_measure.c.

Referenced by do_dma(), and main().

6.19.2.3 #define DMA_BUF_SIZE (BUF_SIZE * BUF_NUM)

The total space available in the device for DMA at any one time (4MB).

Definition at line 130 of file pwm_measure.c.

Referenced by main().

6.19.2.4 #define SAMPLES ((BUF_SIZE * BUF_NUM)/2)

The number of samples that can be held by the list of buffers we created.

Definition at line 124 of file pwm_measure.c.

6.19.2.5 #define UTC_RATE 2

User Timer/Counter divisor, the smaller this number is the better your resolution for measuring PW is.

Definition at line 135 of file pwm_measure.c.

Referenced by do_8254(), and main().

6.19.3 Function Documentation

6.19.3.1 void ISR (dm7820_interrupt_info *interrupt_info*)

Userspace ISR.

Parameters

interrupt_info	Information about the interrupt, returned by the kernel driver

Definition at line 178 of file pwm_measure.c.

References DM7820_INTERRUPT_FIFO_1_DMA_DONE, DM7820_Return_Status, dma_done_interrupts, _dm7820_interrupt_info::error, and _dm7820_interrupt_info::source.

6.19.3.2 int main (int *argument_count*, char ** *arguments*)

Main program code.

Parameters

argument_count	Number of command line arguments passed to executable.
arguments	Address of array containing command line arguments.

Return values

EXIT_SUCCESS	Success.
EXIT_FAILURE	Failure.

Definition at line 582 of file pwm_measure.c.

References BUF_SIZE, clean_up(), DM7820_FIFO_DMA_Create_Buffer(), DM7820_FIFO_DMA_Enable(), DM7820_FIFO_DMA_Free_Buffer(), DM7820_FIFO_DMA_Read(), DM7820_FIFO_Enable(), DM7820_FIFO_QUEUE_1, DM7820_General_Enable_Interrupt(), DM7820_General_Open_Board(), DM7820_General_Remove-ISR(), DM7820_General_Reset(), DM7820_General_SetISRPriority(), DM7820_INTERRUPT_FIFO_1_EMPTY, DM7820_Return_Status, DMA_BUF_SIZE, dma_done_interrupts, do_8254(), do_digital_io(), do_dma(), do_fifo(), do_incenc(), do_pwm(), exit_program, ISR(), program_name, sigint_handler(), usage(), and UTC_RATE.

6.19.3.3 static void sigint_handler (int *signal_number*) [static]

Signal handler for SIGINT Control-C keyboard interrupt.

Parameters

signal_number	Signal number passed in from the kernel.

Warning

One must be extremely careful about what functions are called from a signal handler.

Definition at line 246 of file pwm_measure.c.

References exit_program.

Referenced by main().

6.19.4 Variable Documentation

6.19.4.1 DM7820_Board_Descriptor* board

Device descriptor

Definition at line 99 of file pwm_measure.c.

6.19.4.2 volatile int dma_done_interrupts = 0

Variable to count the number of DMA Done interrupts occurring

Definition at line 93 of file pwm_measure.c.

6.19.4.3 uint8_t exit_program = 0

Flag indicating user intent to exit the program

Definition at line 105 of file pwm_measure.c.

Referenced by main(), and sigint_handler().

6.19.4.4 char* program_name [static]

Name of the program as invoked on the command line

Definition at line 87 of file pwm_measure.c.

Referenced by main(), and usage().

6.20 examples/pwm_modulators.c File Reference

Example program which demonstrates how to use the pulse width modulator block modulators.

```
#include <errno.h>
#include <error.h>
#include <getopt.h>
#include <limits.h>
#include <stdint.h>
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include "dm7820_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

6.20.1 Detailed Description

Example program which demonstrates how to use the pulse width modulator block modulators.

```
        This program sets up the pulse width modulators as follows:

            PWM     Output    Duty Cyle
            ============================
            0       A         20%
                    B         40%
                    C         60%
                    D         80%
            1       A         20%
                    B         40%
                    C         60%
                    D         80%

        PWMs 0 and 1 are set to use the 25 MHz clock as their period and width
        master clocks.  The period of PWM 0 is set to provide a frequency of 1
        MHz.  The period of PWM 1 is set to provide a frequency of 100 KHz.

        The standard I/O block port 2 pins are set up to output the pulse width
        modulators so that their frequencies and duty cycles may be measured.


    -------------------------------------------------------------------------
    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: pwm_modulators.c 60252 2012-06-04 19:39:05Z rgroner $
```

Definition in file pwm_modulators.c.

6.20.2 Function Documentation

6.20.2.1 int main (int *argument_count*, char ** *arguments*)

Main program code.

Parameters

argument_count	Number of command line arguments passed to executable.
arguments	Address of array containing command line arguments.

Return values

EXIT_SUCCESS	Success.
EXIT_FAILURE	Failure.

Definition at line 120 of file pwm_modulators.c.

References board, DM7820_General_Close_Board(), DM7820_General_Open_Board(), DM7820_PWM_Enable(), DM7820_PWM_OUTPUT_A, DM7820_PWM_OUTPUT_B, DM7820_PWM_OUTPUT_C, DM7820_PWM_OUTPUT_D, DM7820_PWM_PERIOD_MASTER_25_MHZ, DM7820_PWM_Set_Period(), DM7820_PWM_Set_Period_Master(), DM7820_PWM_Set_Width(), DM7820_PWM_Set_Width_Master(), DM7820_PWM_WIDTH_MASTER_25_MHZ, DM7820_Return_Status, DM7820_STDIO_MODE_PER_OUT, DM7820_STDIO_PERIPH_PWM, DM7820_STDIO_PORT_2, DM7820_StdIO_Set_IO_Mode(), DM7820_StdIO_Set_Periph_Mode(), program_name, and usage().

6.20.3 Variable Documentation

6.20.3.1 char* program_name [static]

Name of the program as invoked on the command line

Definition at line 64 of file pwm_modulators.c.

Referenced by main(), and usage().

6.21 examples/stdio_digital_io.c File Reference

Example program which demonstrates how to use the standard I/O block digital I/O.

```
#include <errno.h>
#include <error.h>
#include <getopt.h>
#include <limits.h>
#include <stdint.h>
#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <unistd.h>
#include "dm7820_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

6.21.1 Detailed Description

Example program which demonstrates how to use the standard I/O block digital I/O.

```
--------------------------------------------------------------------------
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:

stdio_digital_io.c 60252 2012-06-04 19:39:05Z rgroner

Definition in file stdio_digital_io.c.

6.21.2 Function Documentation

6.21.2.1 int main (int *argument_count*, char ** *arguments*)

Main program code.

Parameters

argument_count	Number of command line arguments passed to executable.
arguments	Address of array containing command line arguments.

Return values

EXIT_SUCCESS	Success.
EXIT_FAILURE	Failure.

Definition at line 107 of file stdio_digital_io.c.

References board, DM7820_General_Close_Board(), DM7820_General_Open_Board(), DM7820_Return_Status, DM7820_StdIO_Get_Input(), DM7820_STDIO_MODE_INPUT, DM7820_STDIO_MODE_OUTPUT, DM7820_StdIO_Set_IO_Mode(), DM7820_StdIO_Set_Output(), program_name, and usage().

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 43 of file stdio_digital_io.c.

Referenced by main(), and usage().

6.22 examples/stdio_peripheral_output.c File Reference

Example program which demonstrates how to select peripheral outputs for standard I/O block digital I/O ports.

```
#include <errno.h>
#include <error.h>
#include <getopt.h>
#include <limits.h>
#include <stdint.h>
#include <stdio.h>
#include <stdlib.h>
#include <strings.h>
#include <sys/types.h>
#include <unistd.h>
#include "dm7820_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

6.22.1 Detailed Description

Example program which demonstrates how to select peripheral outputs for standard I/O block digital I/O ports.

Note

This program does not set up the selected peripheral output. It merely programs the digital I/O ports so that a peripheral can be output.

```
-------------------------------------------------------------------------
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: stdio_peripheral_output.c 60252 2012-06-04 19:39:05Z rgroner $
```

Definition in file stdio_peripheral_output.c.

6.22.2 Function Documentation

6.22.2.1 int main (int *argument_count*, char ** *arguments*)

Main program code.

Parameters

argument_count	Number of command line arguments passed to executable.
arguments	Address of array containing command line arguments.

Return values

EXIT_SUCCESS	Success.
EXIT_FAILURE	Failure.

Definition at line 125 of file stdio_peripheral_output.c.

References board, DM7820_General_Close_Board(), DM7820_General_Open_Board(), DM7820_Return_Status, DM7820_STDIO_MODE_PER_OUT, DM7820_STDIO_PERIPH_CLK_OTHER, DM7820_STDIO_PERIPH_FIFO_0, DM7820_STDIO_PERIPH_FIFO_1, DM7820_STDIO_PERIPH_PWM, DM7820_StdIO_Set_IO_Mode(), DM7820_StdIO_Set_Periph_Mode(), program_name, and usage().

6.22.3 Variable Documentation

6.22.3.1 char* program_name [static]

Name of the program as invoked on the command line

Definition at line 48 of file stdio_peripheral_output.c.

Referenced by main(), and usage().

6.23 examples/stdio_strobe_signal.c File Reference

Example program which demonstrates how to use the standard I/O block strobe signals.

```
#include <errno.h>
#include <error.h>
#include <getopt.h>
#include <limits.h>
#include <stdint.h>
#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <unistd.h>
#include "dm7820_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

6.23.1 Detailed Description

Example program which demonstrates how to use the standard I/O block strobe signals.

Note

This program uses standard I/O block strobe signals 1 and 2. Each signal is used to change the state of the other. Therefore, CN11 pin 2 should be connected to CN10 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: stdio_strobe_signal.c 60252 2012-06-04 19:39:05Z rgroner $
```

Definition in file stdio_strobe_signal.c.

6.23.2 Function Documentation

6.23.2.1 int main (int *argument_count*, char ** *arguments*)

Main program code.

Parameters

argument_count	Number of command line arguments passed to executable.
arguments	Address of array containing command line arguments.

Return values

EXIT_SUCCESS	Success.
EXIT_FAILURE	Failure.

Definition at line 104 of file stdio_strobe_signal.c.

References board, DM7820_General_Close_Board(), DM7820_General_Open_Board(), DM7820_Return_Status, DM7820_STDIO_STROBE_1, DM7820_STDIO_STROBE_2, DM7820_StdIO_Strobe_Input(), DM7820_StdIO_Strobe_Mode(), DM7820_StdIO_Strobe_Output(), program_name, and usage().

6.23.3 Variable Documentation

6.23.3.1 char* program_name [static]

Name of the program as invoked on the command line

Definition at line 48 of file stdio_strobe_signal.c.

Referenced by main(), and usage().

6.24 examples/tmrctr_interrupt.c File Reference

Example program which demonstrates how to use 8254 timer/counter block timer interrupts.

```
#include <errno.h>
#include <error.h>
#include <getopt.h>
#include <limits.h>
#include <stdint.h>
#include <stdio.h>
#include <stdlib.h>
#include <strings.h>
#include <sys/types.h>
#include <unistd.h>
#include "dm7820_library.h"
```

Functions

- static void sigint_handler (int signal_number)

  Signal handler for SIGINT Control-C keyboard interrupt.
- 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 (dm7820_interrupt_info interrupt_info)

  Userspace ISR.
- int main (int argument_count, char **arguments)

  Main program code.

Variables

- static char * program_name
- static volatile int interrupts = 0
- static volatile sig_atomic_t exit_program = 0

6.24.1 Detailed Description

Example program which demonstrates how to use 8254 timer/counter block timer interrupts.

```
    Timers A0 and A1 are configured as follows:

              Input   Count   Waveform
        Timer Clock   Mode    Mode                Frequency
        ======================================================
        A0    5 MHz   binary  square wave         100 Hz
        A1    A0      binary  square wave         .5 Hz

    Each timer has its gate set to logic 1 to enable counting.

    With the setup indicated above, timers A0 and A1 are cascaded.  Timer A1
    generates an interrupt once every two seconds.

The program uses timer A1 interrupts and waits for the interrupts to
    occur.  Ten such interrupts are waited on and then the program exits.


-------------------------------------------------------------------------
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:

tmrctr_interrupt.c 79285 2014-05-21 20:37:42Z rgroner

Definition in file tmrctr_interrupt.c.

6.24.2 Function Documentation

6.24.2.1 void ISR (dm7820_interrupt_info *interrupt_info*)

Userspace ISR.

Parameters

interrupt_info	Information about the interrupt, returned by the kernel driver

Definition at line 155 of file tmrctr_interrupt.c.

References DM7820_INTERRUPT_TMRCTR_A_1, DM7820_Return_Status, _dm7820_interrupt_info::error, interrupts, and _dm7820_interrupt_info::source.

6.24.2.2 int main (int *argument_count*, char ** *arguments*)

Main program code.

Parameters

argument_count	Number of command line arguments passed to executable.
arguments	Address of array containing command line arguments.

Return values

EXIT_SUCCESS	Success.
EXIT_FAILURE	Failure.

Definition at line 201 of file tmrctr_interrupt.c.

References board, DM7820_General_Close_Board(), DM7820_General_Enable_Interrupt(), DM7820_General_Open_Board(), DM7820_General_RemoveISR(), DM7820_General_Reset(), DM7820_General_SetISRPriority(), DM7820_INTERRUPT_TMRCTR_A_1, DM7820_Return_Status, DM7820_TMRCTR_CLOCK_5_MHZ, DM7820_TMRCTR_CLOCK_8254_A_0, DM7820_TMRCTR_COUNT_MODE_BINARY, DM7820_TMRCTR_GATE_LOGIC_0, DM7820_TMRCTR_GATE_LOGIC_1, DM7820_TmrCtr_Program(), DM7820_TmrCtr_Select_Clock(), DM7820_TmrCtr_Select_Gate(), DM7820_TMRCTR_TIMER_A_0, DM7820_TMRCTR_TIMER_A_1, DM7820_TMRCTR_WAVEFORM_SQUARE_WAVE, exit_program, interrupts, ISR(), program_name, sigint_handler(), and usage().

6.24.2.3 static void sigint_handler (int *signal_number*) [static]

Signal handler for SIGINT Control-C keyboard interrupt.

Parameters

signal_number	Signal number passed in from kernel.

Warning

One must be extremely careful about what functions are called from a signal handler. printf() and related functions are considered unsafe for use in signal handlers. Therefore, this function uses write() instead.

Definition at line 98 of file tmrctr_interrupt.c.

References exit_program.

Referenced by main().

6.24.3 Variable Documentation

6.24.3.1 volatile sig_atomic_t exit_program = 0 [static]

Flag used by SIGINT signal handler to tell main program to exit because the user hit Control-C

Definition at line 75 of file tmrctr_interrupt.c.

6.24.3.2 volatile int interrupts = 0 [static]

Variable to count the number of DMA Done interrupts occurring

Definition at line 68 of file tmrctr_interrupt.c.

6.24.3.3 char* program_name [static]

Name of the program as invoked on the command line

Definition at line 62 of file tmrctr_interrupt.c.

Referenced by main(), and usage().

6.25 examples/tmrctr_status.c File Reference

Example program which demonstrates how to get 8254 timer/counter block timer status when not using interrupts.

```
#include <errno.h>
#include <error.h>
#include <getopt.h>
#include <limits.h>
#include <stdint.h>
#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <unistd.h>
#include "dm7820_library.h"
```

Macros

- #define TIMER_A0_DIVISOR 50000
- #define TIMER_A1_DIVISOR 2
- #define TIMER_A1_FREQUENCY 50
- #define TIMER_A2_DIVISOR 1000

Functions

- static void disable_timers (DM7820_Board_Descriptor *board)

  Disable all 8254 timer/counters by setting their gates to logic zero.
- 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

6.25.1 Detailed Description

Example program which demonstrates how to get 8254 timer/counter block timer status when not using interrupts.

```
        Timers A0, A1, and A2 are configured as follows:

                    Input   Count   Waveform
            Timer   Clock   Mode    Mode                Frequency
            ========================================================
            A0      5 MHz   binary  square wave         100 Hz
            A1      A0      binary  square wave         50 Hz
            A2      A1      binary  event counter

        Each timer has its gate set to logic 1 to enable counting.

        With the setup indicated above, timers A0, A1, and A2 are cascaded and
        timer A2 is decremented every tick of timer A1, i.e. every 20
        milliseconds.

        Timer A2 is loaded with a divisor value of 1000.  With this value, it
        will take 20 seconds for timer A2 to count down to zero.

    -----------------------------------------------------------------------------
    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:

tmrctr_status.c 60252 2012-06-04 19:39:05Z rgroner

Definition in file tmrctr_status.c.

6.25.2 Macro Definition Documentation

6.25.2.1 #define TIMER_A0_DIVISOR 50000

Timer/counter A0 divisor

Definition at line 63 of file tmrctr_status.c.

Referenced by main().

6.25.2.2 #define TIMER_A1_DIVISOR 2

Timer/counter A1 divisor

Definition at line 69 of file tmrctr_status.c.

Referenced by main().

6.25.2.3 #define TIMER_A1_FREQUENCY 50

Timer/counter A1 frequency in Hertz

Definition at line 75 of file tmrctr_status.c.

Referenced by main().

6.25.2.4 #define TIMER_A2_DIVISOR 1000

Timer/counter A2 divisor

Definition at line 81 of file tmrctr_status.c.

Referenced by main().

6.25.3 Function Documentation

6.25.3.1 static void disable_timers (DM7820_Board_Descriptor * *board*) [static]

Disable all 8254 timer/counters by setting their gates to logic zero.

Parameters

board	Address of device's library board descriptor.

Definition at line 109 of file tmrctr_status.c.

References DM7820_Return_Status, DM7820_TMRCTR_GATE_LOGIC_0, DM7820_TmrCtr_Select_Gate(), DM7820_TMRCTR_TIMER_A_0, DM7820_TMRCTR_TIMER_A_1, DM7820_TMRCTR_TIMER_A_2, DM7820_TMRCTR_TIMER_B_0, DM7820_TMRCTR_TIMER_B_1, and DM7820_TMRCTR_TIMER_B_2.

Referenced by main().

6.25.3.2 int main (int *argument_count*, char ** *arguments*)

Main program code.

Parameters

argument_count	Number of command line arguments passed to executable.
arguments	Address of array containing command line arguments.

Return values

EXIT_SUCCESS	Success.
EXIT_FAILURE	Failure.

Definition at line 209 of file tmrctr_status.c.

References board, disable_timers(), DM7820_General_Close_Board(), DM7820_General_Open_Board(), DM7820_Return_Status, DM7820_TMRCTR_CLOCK_5_MHZ, DM7820_TMRCTR_CLOCK_8254_A_0, DM7820_TMRCTR_CLOCK_8254_A_1, DM7820_TMRCTR_COUNT_MODE_BINARY, DM7820_TMRCTR_GATE_LOGIC_1, DM7820_TmrCtr_Get_Status(), DM7820_TmrCtr_Program(), DM7820_TmrCtr_Read(), DM7820_TmrCtr_Select_Clock(), DM7820_TmrCtr_Select_Gate(), DM7820_TMRCTR_TIMER_A_0, DM7820_TMRCTR_TIMER_A_1, DM7820_TMRCTR_TIMER_A_2, DM7820_TMRCTR_WAVEFORM_EVENT_CTR, DM7820_TMRCTR_WAVEFORM_SQUARE_WAVE, program_name, TIMER_A0_DIVISOR, TIMER_A1_DIVISOR, TIMER_A1_FREQUENCY, TIMER_A2_DIVISOR, and usage().

6.25.4 Variable Documentation

6.25.4.1 char* program_name [static]

Name of the program as invoked on the command line

Definition at line 91 of file tmrctr_status.c.

Referenced by main(), and usage().

6.26 examples/tmrctr_timers.c File Reference

Example program which demonstrates how to use the 8254 timer/counter block timers.

```
#include <errno.h>
#include <error.h>
#include <getopt.h>
#include <limits.h>
#include <signal.h>
#include <stdint.h>
#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <unistd.h>
#include "dm7820_library.h"
```

Functions

- static void sigint_handler (int signal_number)

  Signal handler for SIGINT Control-C keyboard interrupt.
- 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
- static volatile sig_atomic_t exit_program = 0

6.26.1 Detailed Description

Example program which demonstrates how to use the 8254 timer/counter block timers.

```
All timers are set to count in binary mode.  Timers A0, A1, A2, B0, and
B1 are set to square wave generator waveform mode.  Timer B2 is set to
event counter waveform mode.  Each timer has its gate set to logic 1 to
enable counting.  The input clocks and frequencies are set as follows:

    Timer       Input Clock    Frequency
    ===================================
    A0          5 MHz          500 KHz
    A1          A0             50 KHz
    A2          A1             5 KHz
    B0          A2             500 Hz
    B1          B0             50 Hz
```

```
          B2          B1

    With the setup indicated above, all six timers are cascaded and timer B2
    is decremented every tick of timer B1, i.e. every 20 milliseconds.
    Thus, timer B2 indicates the length of time that timer B1 has been
    running.


-------------------------------------------------------------------------
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:

tmrctr_timers.c 79285 2014-05-21 20:37:42Z rgroner

Definition in file tmrctr_timers.c.

6.26.2 Function Documentation

6.26.2.1 int main (int *argument_count*, char ** *arguments*)

Main program code.

Parameters

argument_count	Number of command line arguments passed to executable.
arguments	Address of array containing command line arguments.

Return values

EXIT_SUCCESS	Success.
EXIT_FAILURE	Failure.

Definition at line 165 of file tmrctr_timers.c.

References board, DM7820_General_Close_Board(), DM7820_General_Open_Board(), DM7820_Return_Status, DM7820_TMRCTR_CLOCK_5_MHZ, DM7820_TMRCTR_CLOCK_8254_A_0, DM7820_TMRCTR_CLOCK_8254_A_1, DM7820_TMRCTR_CLOCK_8254_A_2, DM7820_TMRCTR_CLOCK_8254_B_0, DM7820_TMRCTR_CLOCK_8254_B_1, DM7820_TMRCTR_COUNT_MODE_BINARY, DM7820_TMRCTR_GATE_LOGIC_0, DM7820_TMRCTR_GATE_LOGIC_1, DM7820_TmrCtr_Program(), DM7820_TmrCtr_Read(), DM7820_TmrCtr_Select_Clock(), DM7820_TmrCtr_Select_Gate(), DM7820_TMRCTR_TIMER_A_0, DM7820_TMRCTR_TIMER_A_1, DM7820_TMRCTR_TIMER_A_2, DM7820_TMRCTR_TIMER_B_0, DM7820_TMRCTR_TIMER_B_1, DM7820_TMRCTR_TIMER_B_2, DM7820_TMRCTR_WAVEFORM_EVENT_CTR, DM7820_TMRCTR_WAVEFORM_SQUARE_WAVE, exit_program, program_name, sigint_handler(), and usage().

6.26.2.2 static void sigint_handler (int *signal_number*) [static]

Signal handler for SIGINT Control-C keyboard interrupt.

Parameters

signal_number	Signal number passed in from kernel.

Warning

One must be extremely careful about what functions are called from a signal handler. printf() and related functions are considered unsafe for use in signal handlers. Therefore, this function uses write() instead.

Definition at line 95 of file tmrctr_timers.c.

References exit_program.

Referenced by main().

6.26.3 Variable Documentation

6.26.3.1 volatile sig_atomic_t exit_program = 0 [static]

Flag used by SIGINT signal handler to tell main program to exit because the user hit Control-C

Definition at line 72 of file tmrctr_timers.c.

6.26.3.2 char* program_name [static]

Name of the program as invoked on the command line

Definition at line 65 of file tmrctr_timers.c.

Referenced by main(), and usage().

6.27 include/dm7820_driver.h File Reference

Definitions for the DM7820 driver.

```
#include <linux/fs.h>
#include <linux/list.h>
#include <linux/pci.h>
#include <linux/spinlock.h>
#include <linux/types.h>
#include "dm7820_ioctl.h"
#include "dm7820_types.h"
```

Data Structures

- struct dm7820_pci_region

  DM7820 PCI region descriptor. This structure holds information about one of a device's PCI memory regions.
- struct dm7820_dma_descriptor_t

  DM7820 DMA buffer descriptor. This structure holds allocation information for a single DMA buffer.
- struct dm7820_dma_list_item_t

  DM7820 DMA buffer list item.
- struct dm7820_device_descriptor

  DM7820 device descriptor. This structure holds information about a device needed by the kernel.
- struct dm7820_interrupt_status_source

  Interrupt source information for a single Interrupt Status Register bit.

Macros

- #define DM7820_DEVICE_NAME_LENGTH 22

  Maximum number of characters in device's name.
- #define DM7820_PCI_DEVICE_ID 0x7820

  DM7820 PCI device ID.
- #define RTD_PCI_VENDOR_ID 0x1435

  RTD Embedded Technologies PCI vendor ID.
- #define DM7820_PCI_REGIONS PCI_ROM_RESOURCE

  Number of standard PCI regions.
- #define DM7820_FIFO_CHANNELS 2

  Number of FIFO channels per device.
- #define DM7820_MAX_DMA_BUFFER_SIZE 0x40000

  Maximum size in bytes of any DMA buffer.
- #define DM7820_MAX_DMA_BUFFER_COUNT 16

  Maximum number of DMA buffers per DMA/FIFO channel.
- #define DM7820_INT_QUEUE_SIZE 0x10

  Maximum number of entries in the interrupt status queue;.

Typedefs

- typedef enum dm7820_pci_region_access_dir dm7820_pci_region_access_dir_t

  Standard PCI region access direction type.
- typedef struct dm7820_pci_region dm7820_pci_region_t

  DM7820 PCI region descriptor type.
- typedef struct dm7820_device_descriptor dm7820_device_descriptor_t

  DM7820 device descriptor type.
- typedef struct dm7820_interrupt_status_source dm7820_interrupt_status_source_t

  Interrupt Status Register bit interrupt source information type.

Enumerations

- enum dm7820_pci_region_access_dir { DM7820_PCI_REGION_ACCESS_READ = 0, DM7820_PCI_REGION_ACCESS_WRITE }

  Direction of access to standard PCI region.

Functions

- static void dm7820_access_pci_region (const dm7820_device_descriptor_t *dm7820_device, dm7820_pci_access_request_t *pci_request, dm7820_pci_region_access_dir_t direction)

  Read from or write to one of the standard PCI regions.
- static int dm7820_allocate_irq (dm7820_device_descriptor_t *dm7820_device, const struct pci_dev *pci_device)

  Allocate an interrupt line for a DM7820 device.
- static void dm7820_disable_all_interrupts (const dm7820_device_descriptor_t *dm7820_device)

  Disable all non-PLX interrupts for the specified DM7820 device.
- static void dm7820_enable_plx_interrupts (const dm7820_device_descriptor_t *dm7820_device, uint8_t enable)

Disable or enable PLX interrupts for the specified DM7820 device.

- static void dm7820_free_dma_mappings (dm7820_device_descriptor_t *dm7820_device, dm7820_fifo_queue fifo)

  Free all coherent/consistent DMA mappings for the given DMA/FIFO channel on the specified DM7820 device.
- static int dm7820_get_interrupt_status (dm7820_device_descriptor_t *dm7820_device, unsigned long ioctl_param)

  Get interrupt status for the specified DM7820 device, optionally waiting for an interrupt to occur before returning the status.
- static void dm7820_get_pci_master_status (dm7820_device_descriptor_t *dm7820_device, uint8_t *pci_master)

  Determine whether or not a device is PCI master capable.
- static void dm7820_initialize_device_descriptor (dm7820_device_descriptor_t *dm7820_device)

  Initialize the device descriptor for the specified DM7820 device.
- static int dm7820_initialize_dma (dm7820_device_descriptor_t *dm7820_device, dm7820_ioctl_argument_t *ioctl_argument)

  Initialize DMA for the specified DM7820 device.
- dma_addr_t dm7820_get_buffer_phy_addr (dm7820_device_descriptor_t *dm7820_device, dm7820_fifo_queue fifo)

  Returns the physical address of the next available DMA buffer.
- static int dm7820_dma_read (dm7820_device_descriptor_t *dm7820_device, dm7820_ioctl_argument_t *ioctl_argument)

  Read from DMA buffer to copy to user.
- static int dm7820_dma_write (dm7820_device_descriptor_t *dm7820_device, dm7820_ioctl_argument_t *ioctl_argument)

  Write to DMA buffer.
- static int dm7820_dma_stop (dm7820_device_descriptor_t *dm7820_device, dm7820_fifo_queue fifo)

  Stops a DMA transfer on the specified channel if one is currently running.
- static int dm7820_dma_pause (dm7820_device_descriptor_t *dm7820_device, dm7820_fifo_queue fifo)

  Pause DMA – Used by the STOP_DMA function.
- static int dm7820_dma_check_xfer (dm7820_device_descriptor_t *dm7820_device, dm7820_fifo_queue fifo)

  Checks if there is a transfer currently underway for the specified DMA/FIFO channel.
- static void dm7820_initialize_hardware (const dm7820_device_descriptor_t *dm7820_device)

  Initialize the specified DM7820 device.
- static void dm7820_int_queue_add (dm7820_device_descriptor_t *dm7820_device, dm7820_interrupt_source source)

  Add an interrupt source to the queue.
- static dm7820_interrupt_info dm7820_dequeue_interrupt (dm7820_device_descriptor_t *dm7820_device)

  Remove an interrupt from the front of the queue.
- static irqreturn_t dm7820_interrupt_handler (int irq_number, void *device_id)

  DM7820 device interrupt handler.
- static long dm7820_ioctl (struct file *file, unsigned int request_code, unsigned long ioctl_param)

  Process ioctl(2) system calls directed toward a DM7820 device file.
- int dm7820_load (void)

  Perform all actions necessary to initialize the DM7820 driver and devices.
- static int dm7820_modify_pci_region (dm7820_device_descriptor_t *dm7820_device, 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 dm7820_open (struct inode *inode, struct file *file)

  Prepare a DM7820 device file to be opened and used.
- static unsigned int dm7820_poll (struct file *file, struct poll_table_struct *poll_table)

Determine whether or not a DM7820 device is readable. This function supports the poll(2) and select(2) system calls.

- static int dm7820_probe_device_blocks (dm7820_device_descriptor_t *dm7820_device)

  Probe and set up all functional blocks on a device.
- static int dm7820_probe_devices (uint32_t *device_count, dm7820_device_descriptor_t **device_descriptors)

  Probe and set up all DM7820 devices.
- static int dm7820_process_pci_regions (dm7820_device_descriptor_t *dm7820_device, 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 dm7820_read_pci_region (dm7820_device_descriptor_t *dm7820_device, unsigned long ioctl_param)

  Read an unsigned value from one of a device's PCI regions.
- static int dm7820_register_char_device (int *major)

  Register the DM7820 character device and request dynamic allocation of a character device major number.
- static int dm7820_release (struct inode *inode, struct file *file)

  Do all processing necessary after the last reference to a DM7820 device file is released elsewhere in the kernel.
- static void dm7820_release_resources (void)

  Release any resources allocated by the driver.
- static int dm7820_service_dma_function (dm7820_device_descriptor_t *dm7820_device, unsigned long ioctl_param)

  Process user space DMA function request.
- void dm7820_unload (void)

  Perform all actions necessary to deinitialize the DM7820 driver and devices.
- static int dm7820_unregister_char_device (void)

  Unregister the DM7820 character device and free the character device major number.
- static int dm7820_validate_device (const dm7820_device_descriptor_t *dm7820_device)

  Given what is assumed to be the address of a DM7820 device descriptor, make sure it corresponds to a valid DM7820 device descriptor.
- static int dm7820_validate_pci_access (const dm7820_device_descriptor_t *dm7820_device, const dm7820_pci_access_request_t *pci_request)

  Validate a user-space access to one of a device's PCI regions.
- static int dm7820_write_pci_region (dm7820_device_descriptor_t *dm7820_device, unsigned long ioctl_param)

  Write an unsigned value to one of a device's PCI regions.

Variables

- static struct file_operations dm7820_file_ops

  File operations supported by driver.

6.27.1 Detailed Description

Definitions for the DM7820 driver.

```
//-----------------------------------------------------------------------------
//  COPYRIGHT (C) RTD EMBEDDED TECHNOLOGIES, INC.  ALL RIGHTS RESERVED.
//
//  This software package is dual-licensed.  Source code that is compiled for
//  kernel mode execution is licensed under the GNU General Public License
//  version 2.  For a copy of this license, refer to the file
//  LICENSE_GPLv2.TXT (which should be included with this software) or contact
//  the Free Software Foundation.  Source code that is compiled for user mode
//  execution is licensed under the RTD End-User Software License Agreement.
//  For a copy of this license, refer to LICENSE.TXT or contact RTD Embedded
```

```
//  Technologies, Inc.  Using this software indicates agreement with the
//  license terms listed above.
//-----------------------------------------------------------------------------
```

Id:

dm7820_driver.h 86294 2015-03-04 21:36:57Z rgroner

Definition in file dm7820_driver.h.

6.28 include/dm7820_globals.h File Reference

Global variables used both in the kernel and in the library.

```
#include "dm7820_registers.h"
#include "dm7820_types.h"
```

Variables

- static dm7820_interrupt_control_t dm7820_interrupt_control []
  Table of information needed to acknowledge, disable, and enable all interrupt sources.
- dm7820_minor_int_reg_layout_t dm7820_minor_int_reg_layout []
  Table of information providing layout of all minor interrupt registers.

6.28.1 Detailed Description

Global variables used both in the kernel and in the library.

```
@warning
    Only the driver and library source code should include this file.

//-----------------------------------------------------------------------------
//  COPYRIGHT (C) RTD EMBEDDED TECHNOLOGIES, INC.  ALL RIGHTS RESERVED.
//
//  This software package is dual-licensed.  Source code that is compiled for
//  kernel mode execution is licensed under the GNU General Public License
//  version 2.  For a copy of this license, refer to the file
//  LICENSE_GPLv2.TXT (which should be included with this software) or contact
//  the Free Software Foundation.  Source code that is compiled for user mode
//  execution is licensed under the RTD End-User Software License Agreement.
//  For a copy of this license, refer to LICENSE.TXT or contact RTD Embedded
//  Technologies, Inc.  Using this software indicates agreement with the
//  license terms listed above.
//-----------------------------------------------------------------------------
```

Id:

dm7820_globals.h 89872 2015-07-08 16:05:17Z rgroner

Definition in file dm7820_globals.h.

6.29 include/dm7820_ioctl.h File Reference

Low level ioctl() request descriptor structure and request code definitions.

```
#include <linux/ioctl.h>
#include <linux/types.h>
#include "dm7820_types.h"
```

Data Structures

- struct dm7820_ioctl_region_readwrite

  ioctl() request structure for read from or write to PCI region
- struct dm7820_ioctl_region_modify

  ioctl() request structure for PCI region read/modify/write
- struct dm7820_ioctl_interrupt_status

  ioctl() request structure for getting interrupt status and waiting for an interrupt to occur
- struct dm7820_dma_initialize_arguments

  Arguments for DMA initialization function.
- union dm7820_dma_function_arguments

  Structure encapsulating arguments to all possible DMA functions.
- struct dm7820_ioctl_dma_function

  ioctl() request structure for performing a DMA function
- union dm7820_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 DM7820_IOCTL_MAGIC 'D'

  Unique 8-bit value used to generate unique ioctl() request codes.
- #define DM7820_IOCTL_REQUEST_BASE 0x00

  First ioctl() request number.
- #define DM7820_IOCTL_REGION_READ

  ioctl() request code for reading from a PCI region
- #define DM7820_IOCTL_REGION_WRITE

  ioctl() request code for writing to a PCI region
- #define DM7820_IOCTL_REGION_MODIFY

  ioctl() request code for PCI region read/modify/write
- #define DM7820_IOCTL_GET_INTERRUPT_STATUS

  ioctl() request code for getting interrupt status and waiting for an interrupt to occur
- #define DM7820_IOCTL_DMA_FUNCTION

  ioctl() request code for DMA function
- #define DM7820_IOCTL_WAKEUP

  ioctl() request code for User ISR thread wake up
- #define DM7820_IOCTL_INTERRUPT_INFO

  ioctl() request code to retrieve interrupt status information

Typedefs

- typedef enum dm7820_dma_manage_function dm7820_dma_manage_function_t

  Functions supported by driver DMA management system.
- typedef struct dm7820_ioctl_region_readwrite dm7820_ioctl_region_readwrite_t
- typedef struct dm7820_ioctl_region_modify dm7820_ioctl_region_modify_t

  ioctl() PCI region read/modify/write request descriptor type
- typedef struct dm7820_ioctl_interrupt_status dm7820_ioctl_interrupt_status_t

ioctl() interrupt status request descriptor type

- typedef struct dm7820_dma_initialize_arguments dm7820_dma_initialize_arguments_t

  Arguments for DMA initialization function.

- typedef union dm7820_dma_function_arguments dm7820_dma_function_arguments_t

  Structure encapsulating arguments to all possible DMA functions.

- typedef struct dm7820_ioctl_dma_function dm7820_ioctl_dma_function_t

  ioctl() request structure for performing a DMA function

- typedef union dm7820_ioctl_argument dm7820_ioctl_argument_t

  ioctl() request descriptor type

Enumerations

- enum dm7820_dma_manage_function { DM7820_DMA_FUNCTION_INITIALIZE = 0, DM7820_DMA_FUNCTION_STOP, DM7820_DMA_FUNCTION_READ, DM7820_DMA_FUNCTION_WRITE, DM7820_DMA_GET_BUFFER_ADDR }

  Functions supported by driver DMA management system.

6.29.1 Detailed Description

Low level ioctl() request descriptor structure and request code definitions.

```
//-----------------------------------------------------------------------------
//  COPYRIGHT (C) RTD EMBEDDED TECHNOLOGIES, INC.  ALL RIGHTS RESERVED.
//
//  This software package is dual-licensed.  Source code that is compiled for
//  kernel mode execution is licensed under the GNU General Public License
//  version 2.  For a copy of this license, refer to the file
//  LICENSE_GPLv2.TXT (which should be included with this software) or contact
//  the Free Software Foundation.  Source code that is compiled for user mode
//  execution is licensed under the RTD End-User Software License Agreement.
//  For a copy of this license, refer to LICENSE.TXT or contact RTD Embedded
//  Technologies, Inc.  Using this software indicates agreement with the
//  license terms listed above.
//-----------------------------------------------------------------------------
```

Id:

dm7820_ioctl.h 86294 2015-03-04 21:36:57Z rgroner

Definition in file dm7820_ioctl.h.

6.30 include/dm7820_library.h File Reference

DM7820 user library definitions.

```
#include <stdint.h>
#include "dm7820_types.h"
#include <pthread.h>
#include <sys/wait.h>
```

Data Structures

- struct DM7820_Board_Descriptor

  DM7820 board descriptor. This structure holds information about a device needed by the library.

Macros

- #define DM7820_Return_Status(status, string) if(status != 0) { error(EXIT_FAILURE,errno, "ERROR: %s FAILED\n",string); }
- #define DM7820_DMA_DEMAND_OFF_PCI_TO_DM7820 0x00
- #define DM7820_DMA_DEMAND_OFF_DM7820_TO_PCI 0x01
- #define DM7820_DMA_DEMAND_ON_PCI_TO_DM7820 0x02
- #define DM7820_DMA_DEMAND_ON_DM7820_TO_PCI 0x03
- #define DM7820_INCENC_DISABLE_PHASE_FILTER_TRANSITION(filter, transition) ((filter) |= (1 << (transition)))

  Configure an incremental encoder phase filter so that counter value update is disabled for the given transition.
- #define DM7820_INCENC_ENABLE_PHASE_FILTER_TRANSITION(filter, transition) ((filter) &= ~(1 << (transition)))

  Configure an incremental encoder phase filter so that counter value update is enabled for the given transition.
- #define DM7820_INCENC_RESET_PHASE_FILTER(filter) ((filter) = 0x00)

  Reset an incremental encoder phase filter.
- #define DM7820_INTERRUPT_STATUS_IS_SOURCE_PENDING(status, source) (((status) & (0x1LL << (source))) ? 0xFF : 0x00)

  Determine whether or not the specified interrupt source is pending in the interrupt status obtained via DM7820_General_Get_Interrupt_Status().

Typedefs

- typedef int DM7820_Error

  DM7820 user library error code type.
- typedef uint8_t dm7820_incenc_phase_filter

  Incremental encoder phase filter type.
- typedef struct DM7820_Board_Descriptor DM7820_Board_Descriptor

Functions

- DM7820_Error DM7820_AdvInt_Get_Status (DM7820_Board_Descriptor *handle, dm7820_advint_interrupt interrupt, uint8_t *occurred)

  Determine whether or not a status condition has occurred for the given advanced interrupt.
- DM7820_Error DM7820_AdvInt_Read_Capture (DM7820_Board_Descriptor *handle, dm7820_advint_interrupt interrupt, DM7820_StdIO_Port port, uint16_t *value)

  Read the capture register value for the given advanced interrupt and standard I/O port.
- DM7820_Error DM7820_AdvInt_Set_Compare (DM7820_Board_Descriptor *handle, dm7820_advint_interrupt interrupt, DM7820_StdIO_Port port, uint16_t value)

  Load the compare register for the given advanced interrupt and standard I/O port.
- DM7820_Error DM7820_AdvInt_Set_Mask (DM7820_Board_Descriptor *handle, dm7820_advint_interrupt interrupt, DM7820_StdIO_Port port, uint16_t value)

  Load the mask register for the given advanced interrupt and standard I/O port.
- DM7820_Error DM7820_AdvInt_Set_Master (DM7820_Board_Descriptor *handle, dm7820_advint_interrupt interrupt, dm7820_advint_master_clock master)

  Select the master clock for the given advanced interrupt.

- DM7820_Error DM7820_AdvInt_Set_Mode (DM7820_Board_Descriptor *handle, dm7820_advint_interrupt interrupt, dm7820_advint_mode mode)

  Set the mode for the given advanced interrupt.
- DM7820_Error DM7820_FIFO_Enable (DM7820_Board_Descriptor *handle, dm7820_fifo_queue fifo, uint8_t enable)

  Enable or disable the given FIFO.
- DM7820_Error DM7820_FIFO_Get_Status (DM7820_Board_Descriptor *handle, dm7820_fifo_queue fifo, dm7820_fifo_status_condition condition, uint8_t *occurred)

  Determine whether or not the specified status condition has occurred for the given FIFO.
- DM7820_Error DM7820_FIFO_DMA_Initialize (DM7820_Board_Descriptor *handle, dm7820_fifo_queue fifo, uint32_t buffer_count, uint32_t buffer_size)

  Set up direct memory access (DMA) for the given DMA/FIFO channel.
- DM7820_Error DM7820_FIFO_DMA_Create_Buffer (uint16_t **buf, uint32_t size)

  Creates a user space DMA buffer.
- DM7820_Error DM7820_FIFO_DMA_Free_Buffer (uint16_t **buf, uint32_t size)

  Frees a previously created user space buffer.
- DM7820_Error DM7820_FIFO_DMA_Read (DM7820_Board_Descriptor *handle, dm7820_fifo_queue fifo, void *user_buffer, uint32_t num_bufs)

  Reads the DMA buffers in the driver.
- DM7820_Error DM7820_FIFO_DMA_Write (DM7820_Board_Descriptor *handle, dm7820_fifo_queue fifo, void *user_buffer, uint32_t num_bufs)

  Copies a user buffer to DMA buffers to be sent into a FIFO.
- DM7820_Error DM7820_Stop_DMA (DM7820_Board_Descriptor *handle, dm7820_fifo_queue fifo)

  Aborts a DMA transfer on a given channel.
- DM7820_Error DM7820_FIFO_DMA_Configure (DM7820_Board_Descriptor *handle, dm7820_fifo_queue fifo, uint8_t direction, uint32_t transfer_size)

  Configure the specified DMA channel.
- DM7820_Error DM7820_FIFO_DMA_Enable (DM7820_Board_Descriptor *handle, dm7820_fifo_queue fifo, uint8_t enable, uint8_t start)

  Enable and/or Start a DMA channel.
- DM7820_Error DM7820_FIFO_Read (DM7820_Board_Descriptor *handle, dm7820_fifo_queue fifo, uint16_t *data)

  Read a single value from the given FIFO.
- DM7820_Error DM7820_FIFO_Set_DMA_Request (DM7820_Board_Descriptor *handle, dm7820_fifo_queue fifo, dm7820_fifo_dma_request source)

  Set DMA request source for the given FIFO.
- DM7820_Error DM7820_FIFO_Set_Data_Input (DM7820_Board_Descriptor *handle, dm7820_fifo_queue fifo, dm7820_fifo_data_input input)

  Set data input for the given FIFO.
- DM7820_Error DM7820_FIFO_Set_Input_Clock (DM7820_Board_Descriptor *handle, dm7820_fifo_queue fifo, dm7820_fifo_input_clock clock)

  Set input clock for the given FIFO.
- DM7820_Error DM7820_FIFO_Set_Output_Clock (DM7820_Board_Descriptor *handle, dm7820_fifo_queue fifo, dm7820_fifo_output_clock clock)

  Set output clock for the given FIFO.
- DM7820_Error DM7820_FIFO_Write (DM7820_Board_Descriptor *handle, dm7820_fifo_queue fifo, uint16_t data)

  Write a single value to the given FIFO.
- DM7820_Error DM7820_General_Close_Board (DM7820_Board_Descriptor *handle)

  Close a DM7820 device file.
- DM7820_Error DM7820_General_Enable_Interrupt (DM7820_Board_Descriptor *handle, dm7820_interrupt_source source, uint8_t enable)

  Enable or disable the given interrupt source.

- DM7820_Error DM7820_General_Get_Interrupt_Status (DM7820_Board_Descriptor *handle, dm7820_interrupt_info *interrupt_info, uint8_t wait_for_interrupt)

  Get a device's interrupt status, optionally waiting for an interrupt to occur.
- DM7820_Error DM7820_General_Open_Board (uint8_t dev_num, DM7820_Board_Descriptor **handle)

  Open a DM7820 device file.
- DM7820_Error DM7820_General_Get_Version_Info (DM7820_Board_Descriptor *handle, uint8_t *fpga_type_id, uint8_t *fpga_version, uint16_t *svn_version)

  Read a device's FPGA and source code revision control versions.
- DM7820_Error DM7820_General_Is_PCI_Master (DM7820_Board_Descriptor *handle, uint8_t *pci_master)

  Determine whether or not a device is PCI master capable.
- DM7820_Error DM7820_General_Reset (DM7820_Board_Descriptor *handle)

  Reset a DM7820 device.
- DM7820_Error DM7820_General_RemoveISR (DM7820_Board_Descriptor *handle)

  Uninstall userspace ISR.
- DM7820_Error DM7820_General_StartThread (int(*fnct)(void *), void *data)

  Creates thread to watch for interrupts and call userspace ISR.
- void * DM7820_General_WaitForInterrupt (void *ptr)

  Waits for DMA Done interrupts.
- DM7820_Error DM7820_General_SetISRPriority (DM7820_Board_Descriptor *handle, int priority)

  Changes the Priority for the ISR thread.
- DM7820_Error DM7820_IncEnc_Configure (DM7820_Board_Descriptor *handle, dm7820_incenc_encoder encoder, dm7820_incenc_phase_filter phase_filter, dm7820_incenc_input_mode input_mode, uint8_t enable_input_filter, dm7820_incenc_channel_mode channel_mode, uint8_t enable_index)

  Configure the given incremental encoder.
- DM7820_Error DM7820_IncEnc_Enable (DM7820_Board_Descriptor *handle, dm7820_incenc_encoder encoder, uint8_t enable)

  Enable or disable the given incremental encoder.
- DM7820_Error DM7820_IncEnc_Enable_Hold (DM7820_Board_Descriptor *handle, dm7820_incenc_encoder encoder, uint8_t enable)

  Enable or disable value register hold for the given incremental encoder.
- DM7820_Error DM7820_IncEnc_Get_Independent_Value (DM7820_Board_Descriptor *handle, dm7820_incenc_encoder encoder, dm7820_incenc_channel channel, uint16_t *value)

  Get 16-bit counter value of the given independent incremental encoder channel.
- DM7820_Error DM7820_IncEnc_Get_Joined_Value (DM7820_Board_Descriptor *handle, dm7820_incenc_encoder encoder, uint32_t *value)

  Get 32-bit counter value of the given independent incremental encoder whose channels are joined.
- DM7820_Error DM7820_IncEnc_Get_Status (DM7820_Board_Descriptor *handle, dm7820_incenc_encoder encoder, dm7820_incenc_status_condition condition, uint8_t *occurred)

  Determine whether or not the specified status condition has occurred for the given incremental encoder.
- DM7820_Error DM7820_IncEnc_Set_Independent_Value (DM7820_Board_Descriptor *handle, dm7820_incenc_encoder encoder, dm7820_incenc_channel channel, uint16_t value)

  Set 16-bit counter value for the given independent incremental encoder channel.
- DM7820_Error DM7820_IncEnc_Set_Joined_Value (DM7820_Board_Descriptor *handle, dm7820_incenc_encoder encoder, uint32_t value)

  Set 32-bit counter value for the given incremental encoder whose channels are joined.
- DM7820_Error DM7820_IncEnc_Set_Master (DM7820_Board_Descriptor *handle, dm7820_incenc_encoder encoder, dm7820_incenc_master_clock master)

  Set the master clock for the given incremental encoder.
- DM7820_Error DM7820_PWM_Enable (DM7820_Board_Descriptor *handle, dm7820_pwm_modulator pwm, uint8_t enable)

  Enable or disable the given pulse width modulator (PWM).

- DM7820_Error DM7820_PWM_Set_Period (DM7820_Board_Descriptor *handle, dm7820_pwm_modulator pwm, uint32_t period)

  Set the period for the given pulse width modulator (PWM).
- DM7820_Error DM7820_PWM_Set_Period_Master (DM7820_Board_Descriptor *handle, dm7820_pwm_modulator pwm, dm7820_pwm_period_master_clock master)

  Set the period master clock for the given pulse width modulator (PWM).
- DM7820_Error DM7820_PWM_Set_Width (DM7820_Board_Descriptor *handle, dm7820_pwm_modulator pwm, dm7820_pwm_output output, uint16_t width)

  Set the width for the specified output on the given pulse width modulator (PWM).
- DM7820_Error DM7820_PWM_Set_Width_Master (DM7820_Board_Descriptor *handle, dm7820_pwm_modulator pwm, dm7820_pwm_width_master_clock master)

  Set the width master clock for the given pulse width modulator (PWM).
- DM7820_Error DM7820_PrgClk_Set_Master (DM7820_Board_Descriptor *handle, dm7820_prgclk_clock clock, dm7820_prgclk_master_clock master)

  Select the master clock for the given programmable clock.
- DM7820_Error DM7820_PrgClk_Set_Mode (DM7820_Board_Descriptor *handle, dm7820_prgclk_clock clock, dm7820_prgclk_mode mode)

  Select the mode for the given programmable clock.
- DM7820_Error DM7820_PrgClk_Set_Period (DM7820_Board_Descriptor *handle, dm7820_prgclk_clock clock, uint32_t period)

  Set the period for the given programmable clock.
- DM7820_Error DM7820_PrgClk_Set_Start_Trigger (DM7820_Board_Descriptor *handle, dm7820_prgclk_clock clock, dm7820_prgclk_start_trigger start)

  Set the start trigger for the given programmable clock.
- DM7820_Error DM7820_PrgClk_Set_Stop_Trigger (DM7820_Board_Descriptor *handle, dm7820_prgclk_clock clock, dm7820_prgclk_stop_trigger stop)

  Set the stop trigger for the given programmable clock.
- DM7820_Error DM7820_StdIO_Get_Input (DM7820_Board_Descriptor *handle, DM7820_StdIO_Port port, uint16_t *value)

  Read a value from the given standard I/O port.
- DM7820_Error DM7820_StdIO_Set_IO_Mode (DM7820_Board_Descriptor *handle, DM7820_StdIO_Port port, uint16_t bits, DM7820_StdIO_IO_Mode mode)

  Set the mode for specific bits in the given standard I/O port.
- DM7820_Error DM7820_StdIO_Set_Output (DM7820_Board_Descriptor *handle, DM7820_StdIO_Port port, uint16_t value)

  Write a value to the given standard I/O port.
- DM7820_Error DM7820_StdIO_Set_Periph_Mode (DM7820_Board_Descriptor *handle, DM7820_StdIO_Port port, uint16_t bits, DM7820_StdIO_Periph_Mode mode)

  Set the peripheral output mode for specific bits in the given standard I/O port.
- DM7820_Error DM7820_StdIO_Strobe_Input (DM7820_Board_Descriptor *handle, DM7820_StdIO_Strobe strobe, uint8_t *state)

  Determine state of given strobe signal.
- DM7820_Error DM7820_StdIO_Strobe_Mode (DM7820_Board_Descriptor *handle, DM7820_StdIO_Strobe strobe, uint8_t output)

  Set the direction (input or output) for the given strobe signal.
- DM7820_Error DM7820_StdIO_Strobe_Output (DM7820_Board_Descriptor *handle, DM7820_StdIO_Strobe strobe, uint8_t state)

  Set state of given strobe signal.
- DM7820_Error DM7820_TmrCtr_Get_Status (DM7820_Board_Descriptor *handle, dm7820_tmrctr_timer timer, uint8_t *occurred)

  Determine whether or not a status condition has occurred for the given timer/counter.
- DM7820_Error DM7820_TmrCtr_Program (DM7820_Board_Descriptor *handle, dm7820_tmrctr_timer timer, dm7820_tmrctr_waveform waveform, dm7820_tmrctr_count_mode count_mode, uint16_t divisor)

Program the given 8254 timer/counter. This will 1) set the waveform mode, 2) set the count mode, and 3) load a divisor into the timer.

- DM7820_Error DM7820_TmrCtr_Read (DM7820_Board_Descriptor *handle, dm7820_tmrctr_timer timer, uint16_t *value)

  Read the value of the given 8254 timer/counter.
- DM7820_Error DM7820_TmrCtr_Select_Clock (DM7820_Board_Descriptor *handle, dm7820_tmrctr_timer timer, dm7820_tmrctr_clock clock)

  Select the clock input for the given 8254 timer/counter.
- DM7820_Error DM7820_TmrCtr_Select_Gate (DM7820_Board_Descriptor *handle, dm7820_tmrctr_timer timer, dm7820_tmrctr_gate gate)

  Select the gate input for the given 8254 timer/counter.

6.30.1 Detailed Description

DM7820 user library definitions.

```
//-----------------------------------------------------------------------------
//  COPYRIGHT (C) RTD EMBEDDED TECHNOLOGIES, INC.  ALL RIGHTS RESERVED.
//
//  This software package is dual-licensed.  Source code that is compiled for
//  kernel mode execution is licensed under the GNU General Public License
//  version 2.  For a copy of this license, refer to the file
//  LICENSE_GPLv2.TXT (which should be included with this software) or contact
//  the Free Software Foundation.  Source code that is compiled for user mode
//  execution is licensed under the RTD End-User Software License Agreement.
//  For a copy of this license, refer to LICENSE.TXT or contact RTD Embedded
//  Technologies, Inc.  Using this software indicates agreement with the
//  license terms listed above.
//-----------------------------------------------------------------------------
```

Id:

dm7820_library.h 86275 2015-03-04 15:53:23Z rgroner

Definition in file dm7820_library.h.

6.31 include/dm7820_registers.h File Reference

Register definitions for DM7820 devices.

Macros

- #define DM7820_BAR0_INTCSR 0x68

  PLX interrupt control/status.
- #define DM7820_BAR0_DMAMODE0 0x80

  PLX DMA channel 0 mode.
- #define DM7820_BAR0_DMAPADR0 0x84

  PLX DMA channel 0 PCI address.
- #define DM7820_BAR0_DMALADR0 0x88

  PLX DMA channel 0 local address.
- #define DM7820_BAR0_DMASIZ0 0x8C

  PLX DMA channel 0 transfer size.
- #define DM7820_BAR0_DMADPR0 0x90

  PLX DMA channel 0 descriptor pointer.
- #define DM7820_BAR0_DMAMODE1 0x94

PLX DMA channel 1 mode.

- #define DM7820_BAR0_DMAPADR1 0x98

  PLX DMA channel 1 PCI address.
- #define DM7820_BAR0_DMALADR1 0x9C

  PLX DMA channel 1 local address.
- #define DM7820_BAR0_DMASIZ1 0xA0

  PLX DMA channel 1 transfer size.
- #define DM7820_BAR0_DMADPR1 0xA4

  PLX DMA channel 1 descriptor pointer.
- #define DM7820_BAR0_DMACSR0 0xA8

  PLX DMA channel 0 command/status.
- #define DM7820_BAR0_DMACSR1 0xA9

  PLX DMA channel 1 command/status.
- #define DM7820_BAR0_DMAARB 0xAC

  PLX DMA arbitration.
- #define DM7820_BAR0_DMATHR 0xB0

  PLX DMA threshold.
- #define DM7820_BAR0_DMADA0 0xB4

  PLX DMA channel 0 PCI dual address cycles upper address.
- #define DM7820_BAR0_DMADA1 0xB8

  PLX DMA channel 1 PCI dual address cycles upper address.
- #define DM7820_BAR1_INTCSR 0x68

  PLX interrupt control/status.
- #define DM7820_BAR1_DMAMODE0 0x80

  PLX DMA channel 0 mode.
- #define DM7820_BAR1_DMAPADR0 0x84

  PLX DMA channel 0 PCI address.
- #define DM7820_BAR1_DMALADR0 0x88

  PLX DMA channel 0 local address.
- #define DM7820_BAR1_DMASIZ0 0x8C

  PLX DMA channel 0 transfer size.
- #define DM7820_BAR1_DMADPR0 0x90

  PLX DMA channel 0 descriptor pointer.
- #define DM7820_BAR1_DMAMODE1 0x94

  PLX DMA channel 1 mode.
- #define DM7820_BAR1_DMAPADR1 0x98

  PLX DMA channel 1 PCI address.
- #define DM7820_BAR1_DMALADR1 0x9C

  PLX DMA channel 1 local address.
- #define DM7820_BAR1_DMASIZ1 0xA0

  PLX DMA channel 1 transfer size.
- #define DM7820_BAR1_DMADPR1 0xA4

  PLX DMA channel 1 descriptor pointer.
- #define DM7820_BAR1_DMACSR0 0xA8

  PLX DMA channel 0 command/status.
- #define DM7820_BAR1_DMACSR1 0xA9

  PLX DMA channel 1 command/status.
- #define DM7820_BAR1_DMAARB 0xAC

  PLX DMA arbitration.
- #define DM7820_BAR1_DMATHR 0xB0

  PLX DMA threshold.

- #define DM7820_BAR1_DMADA0 0xB4

  PLX DMA channel 0 PCI dual address cycles upper address.
- #define DM7820_BAR1_DMADA1 0xB8

  PLX DMA channel 1 PCI dual address cycles upper address.
- #define DM7820_BAR2_FPGA_VERSION 0x0000

  FPGA version and type identifiers.
- #define DM7820_BAR2_SVN_VERSION 0x0002

  FPGA source code revision control version identifier.
- #define DM7820_BAR2_BOARD_RESET 0x0004

  Board reset.
- #define DM7820_BAR2_BRD_STAT 0x0008

  Board status.
- #define DM7820_BAR2_INTERRUPT_ENABLE 0x0010

  Local interrupt enable.
- #define DM7820_BAR2_INTERRUPT_STATUS 0x0012

  Local interrupt status.
- #define DM7820_BAR2_PORT0_OUTPUT 0x0040

  Port 0 output value.
- #define DM7820_BAR2_PORT0_INPUT 0x0042

  Port 0 input value.
- #define DM7820_BAR2_PORT0_TRISTATE 0x0044

  Port 0 direction.
- #define DM7820_BAR2_PORT0_MODE 0x0046

  Port 0 operating mode.
- #define DM7820_BAR2_PORT1_OUTPUT 0x0048

  Port 1 output value.
- #define DM7820_BAR2_PORT1_INPUT 0x004A

  Port 1 input value.
- #define DM7820_BAR2_PORT1_TRISTATE 0x004C

  Port 1 direction.
- #define DM7820_BAR2_PORT1_MODE 0x004E

  Port 1 operating mode.
- #define DM7820_BAR2_PORT2_OUTPUT 0x0050

  Port 2 output value.
- #define DM7820_BAR2_PORT2_INPUT 0x0052

  Port 2 input value.
- #define DM7820_BAR2_PORT2_TRISTATE 0x0054

  Port 2 direction.
- #define DM7820_BAR2_PORT2_MODE 0x0056

  Port 2 operating mode.
- #define DM7820_BAR2_STROBE_STATUS 0x0058

  Port 2 strobe signal status.
- #define DM7820_BAR2_PORT0_PERIPH_SEL_L 0x0060

  Port 0 bits 0-7 peripheral select.
- #define DM7820_BAR2_PORT0_PERIPH_SEL_H 0x0062

  Port 0 bits 8-15 peripheral select.
- #define DM7820_BAR2_PORT1_PERIPH_SEL_L 0x0064

  Port 1 bits 0-7 peripheral select.
- #define DM7820_BAR2_PORT1_PERIPH_SEL_H 0x0066

  Port 1 bits 8-15 peripheral select.
- #define DM7820_BAR2_PORT2_PERIPH_SEL_L 0x0068

Port 2 bits 0-7 peripheral select.

- #define DM7820_BAR2_PORT2_PERIPH_SEL_H 0x006A

  Port 2 bits 8-15 peripheral select.
- #define PLX9056_DMA_WIDTH_MASK 0x00000003
- #define PLX9056_DMA_WIDTH_8 0x00000000
- #define PLX9056_DMA_WIDTH_16 0x00000001
- #define PLX9056_DMA_WIDTH_32 0x00000002
- #define PLX9056_DMA_WAITSTATES_MASK 0x0000003c
- #define PLX9056_DMA_READY 0x00000040
- #define PLX9056_DMA_BURST 0x00000080
- #define PLX9056_DMA_LOCAL_BURST 0x00000100
- #define PLX9056_DMA_SCATTERGATHER 0x00000200
- #define PLX9056_DMA_DONE_INTERRUPT 0x00000400
- #define PLX9056_DMA_LOCAL_ADDRESSING_MODE 0x00000800
- #define PLX9056_DMA_DEMAND_MODE 0x00001000
- #define PLX9056_DMA_MEMWRITE_INV 0x00002000
- #define PLX9056_DMA_EOT_ENABLE 0x00004000
- #define PLX9056_DMA_FAST_SLOW_TERM 0x00008000
- #define PLX9056_DMA_CLEAR_COUNT 0x00010000
- #define PLX9056_DMA_INTERRUPT_SEL 0x00020000
- #define PLX9056_DMA_DAC_CHAIN 0x00040000
- #define PLX9056_DMA_EOT_END 0x00080000
- #define PLX9056_DMA_RING_MODE 0x00100000
- #define PLX9056_DMA_RING_CONTROL 0x00200000
- #define DM7820_BAR2_TC_ID 0x0080

  8254 timer/counter block identifier
- #define DM7820_BAR2_TC_INT 0x0082

  8254 timer/counter interrupt control/status
- #define DM7820_BAR2_TC_A0_CONTROL 0x0084

  8254 timer/counter A0 control
- #define DM7820_BAR2_TC_A1_CONTROL 0x0086

  8254 timer/counter A1 control
- #define DM7820_BAR2_TC_A2_CONTROL 0x0088

  8254 timer/counter A2 control
- #define DM7820_BAR2_TC_B0_CONTROL 0x008A

  8254 timer/counter B0 control
- #define DM7820_BAR2_TC_B1_CONTROL 0x008C

  8254 timer/counter B1 control
- #define DM7820_BAR2_TC_B2_CONTROL 0x008E

  8254 timer/counter B2 control
- #define DM7820_BAR2_FIFO0_ID 0x00C0

  FIFO 0 block identifier.
- #define DM7820_BAR2_FIFO0_INT 0x00C2

  FIFO 0 interrupt control/status.
- #define DM7820_BAR2_FIFO0_IN_CLK 0x00C4

  FIFO 0 input clock.
- #define DM7820_BAR2_FIFO0_OUT_CLK 0x00C6

  FIFO 0 output clock.
- #define DM7820_BAR2_FIFO0_IN_DATA_DREQ 0x00C8

  FIFO 0 data input and PLX DMA request source.
- #define DM7820_BAR2_FIFO0_CON_STAT 0x00CA

  FIFO 0 control/status.

- #define DM7820_BAR2_FIFO0_RW_PORT 0x00CC

  FIFO 0 PCI read/write port.
- #define DM7820_BAR2_FIFO1_ID 0x00D0

  FIFO 1 block identifier.
- #define DM7820_BAR2_FIFO1_INT 0x00D2

  FIFO 1 interrupt control/status.
- #define DM7820_BAR2_FIFO1_IN_CLK 0x00D4

  FIFO 1 input clock.
- #define DM7820_BAR2_FIFO1_OUT_CLK 0x00D6

  FIFO 1 output clock.
- #define DM7820_BAR2_FIFO1_IN_DATA_DREQ 0x00D8

  FIFO 1 data input and PLX DMA request source.
- #define DM7820_BAR2_FIFO1_CON_STAT 0x00DA

  FIFO 1 control/status.
- #define DM7820_BAR2_FIFO1_RW_PORT 0x00DC

  FIFO 1 PCI read/write port.
- #define DM7820_BAR2_PRGCLK0_ID 0x0100

  Programmable clock 0 block identifier.
- #define DM7820_BAR2_PRGCLK0_MODE 0x0102

  Programmable clock 0 operating mode.
- #define DM7820_BAR2_PRGCLK0_CLK 0x0104

  Programmable clock 0 master clock.
- #define DM7820_BAR2_PRGCLK0_START_STOP 0x0106

  Programmable clock 0 start/stop triggers.
- #define DM7820_BAR2_PRGCLK0_PERIOD 0x0108

  Programmable clock 0 period.
- #define DM7820_BAR2_PRGCLK1_ID 0x0140

  Programmable clock 1 block identifier.
- #define DM7820_BAR2_PRGCLK1_MODE 0x0142

  Programmable clock 1 operating mode.
- #define DM7820_BAR2_PRGCLK1_CLK 0x0144

  Programmable clock 1 master clock.
- #define DM7820_BAR2_PRGCLK1_START_STOP 0x0146

  Programmable clock 1 start/stop triggers.
- #define DM7820_BAR2_PRGCLK1_PERIOD 0x0148

  Programmable clock 1 period.
- #define DM7820_BAR2_PRGCLK2_ID 0x0180

  Programmable clock 2 block identifier.
- #define DM7820_BAR2_PRGCLK2_MODE 0x0182

  Programmable clock 2 operating mode.
- #define DM7820_BAR2_PRGCLK2_CLK 0x0184

  Programmable clock 2 master clock.
- #define DM7820_BAR2_PRGCLK2_START_STOP 0x0186

  Programmable clock 2 start/stop triggers.
- #define DM7820_BAR2_PRGCLK2_PERIOD 0x0188

  Programmable clock 2 period.
- #define DM7820_BAR2_PRGCLK3_ID 0x01C0

  Programmable clock 3 block identifier.
- #define DM7820_BAR2_PRGCLK3_MODE 0x01C2

  Programmable clock 3 operating mode.
- #define DM7820_BAR2_PRGCLK3_CLK 0x01C4

Programmable clock 3 master clock.

- #define DM7820_BAR2_PRGCLK3_START_STOP 0x01C6

  Programmable clock 3 start/stop triggers.
- #define DM7820_BAR2_PRGCLK3_PERIOD 0x01C8

  Programmable clock 3 period.
- #define DM7820_BAR2_ADVINT0_ID 0x0200

  Advanced interrupt 0 block identifier.
- #define DM7820_BAR2_ADVINT0_INT_MODE 0x0202

  Advanced interrupt 0 mode.
- #define DM7820_BAR2_ADVINT0_CLK 0x0204

  Advanced interrupt 0 master clock.
- #define DM7820_BAR2_ADVINT0_PORT0_MASK 0x0208

  Advanced interrupt 0 standard I/O port 0 mask.
- #define DM7820_BAR2_ADVINT0_PORT1_MASK 0x020A

  Advanced interrupt 0 standard I/O port 1 mask.
- #define DM7820_BAR2_ADVINT0_PORT2_MASK 0x020C

  Advanced interrupt 0 standard I/O port 2 mask.
- #define DM7820_BAR2_ADVINT0_PORT0_CMP 0x0210

  Advanced interrupt 0 standard I/O port 0 event mode compare.
- #define DM7820_BAR2_ADVINT0_PORT1_CMP 0x0212

  Advanced interrupt 0 standard I/O port 1 event mode compare.
- #define DM7820_BAR2_ADVINT0_PORT2_CMP 0x0214

  Advanced interrupt 0 standard I/O port 2 event mode compare.
- #define DM7820_BAR2_ADVINT0_PORT0_CAPT 0x0218

  Advanced interrupt 0 standard I/O port 0 value capture.
- #define DM7820_BAR2_ADVINT0_PORT1_CAPT 0x021A

  Advanced interrupt 0 standard I/O port 1 value capture.
- #define DM7820_BAR2_ADVINT0_PORT2_CAPT 0x021C

  Advanced interrupt 0 standard I/O port 2 value capture.
- #define DM7820_BAR2_ADVINT1_ID 0x0240

  Advanced interrupt 1 block identifier.
- #define DM7820_BAR2_ADVINT1_INT_MODE 0x0242

  Advanced interrupt 1 mode.
- #define DM7820_BAR2_ADVINT1_CLK 0x0244

  Advanced interrupt 1 master clock.
- #define DM7820_BAR2_ADVINT1_PORT0_MASK 0x0248

  Advanced interrupt 1 standard I/O port 0 mask.
- #define DM7820_BAR2_ADVINT1_PORT1_MASK 0x024A

  Advanced interrupt 1 standard I/O port 1 mask.
- #define DM7820_BAR2_ADVINT1_PORT2_MASK 0x024C

  Advanced interrupt 1 standard I/O port 2 mask.
- #define DM7820_BAR2_ADVINT1_PORT0_CMP 0x0250

  Advanced interrupt 1 standard I/O port 0 event mode compare.
- #define DM7820_BAR2_ADVINT1_PORT1_CMP 0x0252

  Advanced interrupt 1 standard I/O port 1 event mode compare.
- #define DM7820_BAR2_ADVINT1_PORT2_CMP 0x0254

  Advanced interrupt 1 standard I/O port 2 event mode compare.
- #define DM7820_BAR2_ADVINT1_PORT0_CAPT 0x0258

  Advanced interrupt 1 standard I/O port 0 value capture.
- #define DM7820_BAR2_ADVINT1_PORT1_CAPT 0x025A

  Advanced interrupt 1 standard I/O port 1 value capture.

- #define DM7820_BAR2_ADVINT1_PORT2_CAPT 0x025C

  Advanced interrupt 1 standard I/O port 2 value capture.
- #define DM7820_BAR2_INCENC0_ID 0x0280

  Incremental encoder 0 block identifier.
- #define DM7820_BAR2_INCENC0_INT 0x0282

  Incremental encoder 0 interrupt control/status.
- #define DM7820_BAR2_INCENC0_CLOCK 0x0284

  Incremental encoder 0 master clock.
- #define DM7820_BAR2_INCENC0_MODE 0x0286

  Incremental encoder 0 operating mode.
- #define DM7820_BAR2_INCENC0_VALUEA 0x0288

  Incremental encoder 0 channel A value.
- #define DM7820_BAR2_INCENC0_VALUEB 0x028A

  Incremental encoder 0 channel B value.
- #define DM7820_BAR2_INCENC1_ID 0x02C0

  Incremental encoder 1 block identifier.
- #define DM7820_BAR2_INCENC1_INT 0x02C2

  Incremental encoder 1 interrupt control/status.
- #define DM7820_BAR2_INCENC1_CLOCK 0x02C4

  Incremental encoder 1 master clock.
- #define DM7820_BAR2_INCENC1_MODE 0x02C6

  Incremental encoder 1 operating mode.
- #define DM7820_BAR2_INCENC1_VALUEA 0x02C8

  Incremental encoder 1 channel A value.
- #define DM7820_BAR2_INCENC1_VALUEB 0x02CA

  Incremental encoder 1 channel B value.
- #define DM7820_BAR2_PWM0_ID 0x0300

  Pulse width modulator 0 block identifier.
- #define DM7820_BAR2_PWM0_MODE 0x0302

  Pulse width modulator 0 mode.
- #define DM7820_BAR2_PWM0_CLK 0x0304

  Pulse width modulator 0 period/width clocks.
- #define DM7820_BAR2_PWM0_PERIOD 0x0308

  Pulse width modulator 0 period.
- #define DM7820_BAR2_PWM0_WIDTHA 0x0310

  Pulse width modulator 0 output A width.
- #define DM7820_BAR2_PWM0_WIDTHB 0x0314

  Pulse width modulator 0 output B width.
- #define DM7820_BAR2_PWM0_WIDTHC 0x0318

  Pulse width modulator 0 output C width.
- #define DM7820_BAR2_PWM0_WIDTHD 0x031C

  Pulse width modulator 0 output D width.
- #define DM7820_BAR2_PWM1_ID 0x0340

  Pulse width modulator 1 block identifier.
- #define DM7820_BAR2_PWM1_MODE 0x0342

  Pulse width modulator 1 mode.
- #define DM7820_BAR2_PWM1_CLK 0x0344

  Pulse width modulator 1 period/width clocks.
- #define DM7820_BAR2_PWM1_PERIOD 0x0348

  Pulse width modulator 1 period.
- #define DM7820_BAR2_PWM1_WIDTHA 0x0350

Pulse width modulator 1 output A width.

- #define DM7820_BAR2_PWM1_WIDTHB 0x0354

  Pulse width modulator 1 output B width.
- #define DM7820_BAR2_PWM1_WIDTHC 0x0358

  Pulse width modulator 1 output C width.
- #define DM7820_BAR2_PWM1_WIDTHD 0x035C

  Pulse width modulator 1 output D width.
- #define DM7820_BAR2_TCA_COUNTER_0 0x1000

  8254 timer/counter A timer 0 value
- #define DM7820_BAR2_TCA_COUNTER_1 0x1004

  8254 timer/counter A timer 1 value
- #define DM7820_BAR2_TCA_COUNTER_2 0x1008

  8254 timer/counter A timer 2 value
- #define DM7820_BAR2_TCA_CON_WORD 0x100C

  8254 timer/counter A control word
- #define DM7820_BAR2_TCB_COUNTER_0 0x1010

  8254 timer/counter B timer 0 value
- #define DM7820_BAR2_TCB_COUNTER_1 0x1014

  8254 timer/counter B timer 1 value
- #define DM7820_BAR2_TCB_COUNTER_2 0x1018

  8254 timer/counter B timer 2 value
- #define DM7820_BAR2_TCB_CON_WORD 0x101C

  8254 timer/counter B control word
- #define DM7820_BAR0_LENGTH 0x200

  Length in bytes of BAR0 (memory-mapped PLX registers)
- #define DM7820_BAR1_LENGTH 0x100

  Length in bytes of BAR1 (I/O-mapped PLX registers)
- #define DM7820_BAR2_LENGTH 0x2000

  Length in bytes of BAR2 (memory-mapped FPGA registers)
- #define DM7820_ID_TIMER_COUNTER 0x1001

  8254 timer/counter block identifier
- #define DM7820_ID_FIFO 0x2011

  FIFO block identifier.
- #define DM7820_ID_PROGRAMMABLE_CLOCK 0x1000

  Programmable clock block identifier.
- #define DM7820_ID_ADVANCED_INTERRUPT 0x0001

  Advanced interrupt block identifier.
- #define DM7820_ID_INCREMENTAL_ENCODER 0x0002

  Incremental encoder block identifier.
- #define DM7820_ID_PULSE_WIDTH_MODULATOR 0x0003

  Pulse width modulator block identifier.
- #define DM7820_ID_NONE 0x0000

  Empty block identifier.
- #define DM7820_FIRST_ID_OFFSET 0x0080

  Offset of first possible block identifier.
- #define DM7820_LAST_ID_OFFSET 0x03C0

  Offset of last possible block identifier.
- #define DM7820_FPGA_VERSION_TYPE_ID_MASK 0xFF00

  Bit mask to extract FPGA type identifier.
- #define DM7820_FPGA_VERSION_VERSION_MASK 0x00FF

  Bit mask to extract FPGA version identifier.
- #define DM7820_BOARD_RESET_DO_RESET 0xA5A5

  Value to write to cause a board reset.

6.31.1 Detailed Description

Register definitions for DM7820 devices.

```
//------------------------------------------------------------------------------
//  COPYRIGHT (C) RTD EMBEDDED TECHNOLOGIES, INC.  ALL RIGHTS RESERVED.
//
//  This software package is dual-licensed.  Source code that is compiled for
//  kernel mode execution is licensed under the GNU General Public License
//  version 2.  For a copy of this license, refer to the file
//  LICENSE_GPLv2.TXT (which should be included with this software) or contact
//  the Free Software Foundation.  Source code that is compiled for user mode
//  execution is licensed under the RTD End-User Software License Agreement.
//  For a copy of this license, refer to LICENSE.TXT or contact RTD Embedded
//  Technologies, Inc.  Using this software indicates agreement with the
//  license terms listed above.
//------------------------------------------------------------------------------
```

Id:

dm7820_registers.h 86275 2015-03-04 15:53:23Z rgroner

Definition in file dm7820_registers.h.

6.32 include/dm7820_types.h File Reference

Type definitions used both in kernel and user space.

Data Structures

- struct _dm7820_interrupt_info

  Interrupt source information.
- struct dm7820_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.
- struct dm7820_interrupt_control

  Structure containing information needed to acknowledge, disable, and enable a particular interrupt source.
- struct dm7820_minor_int_reg_layout

  Minor interrupt register bit layout.

Typedefs

- typedef enum dm7820_pci_region_num dm7820_pci_region_num_t

  Standard PCI region number type.
- typedef enum dm7820_pci_region_access_size dm7820_pci_region_access_size_t

  Standard PCI region access size type.
- typedef enum _DM7820_StdIO_Port DM7820_StdIO_Port

  Standard I/O port type.
- typedef enum _DM7820_StdIO_IO_Mode DM7820_StdIO_IO_Mode

  Standard I/O port mode type.
- typedef enum _DM7820_StdIO_Periph_Mode DM7820_StdIO_Periph_Mode

  Standard I/O port peripheral output mode type.
- typedef enum _DM7820_StdIO_Strobe DM7820_StdIO_Strobe

Standard I/O port strobe signal type.

- typedef enum _dm7820_tmrctr_timer dm7820_tmrctr_timer

  8254 timer/counter type
- typedef enum _dm7820_tmrctr_clock dm7820_tmrctr_clock

  8254 timer/counter clock selector type
- typedef enum _dm7820_tmrctr_gate dm7820_tmrctr_gate

  8254 timer/counter gate selector type
- typedef enum _dm7820_tmrctr_waveform dm7820_tmrctr_waveform

  8254 timer/counter waverform mode selector type
- typedef enum _dm7820_tmrctr_count_mode dm7820_tmrctr_count_mode

  8254 timer/counter count mode selector type
- typedef enum _dm7820_prgclk_clock dm7820_prgclk_clock

  Programmable clock type.
- typedef enum _dm7820_prgclk_mode dm7820_prgclk_mode

  Programmable clock mode type.
- typedef enum _dm7820_prgclk_master_clock dm7820_prgclk_master_clock

  Programmable clock master clock type.
- typedef enum _dm7820_prgclk_start_trigger dm7820_prgclk_start_trigger

  Programmable clock start trigger type.
- typedef enum _dm7820_prgclk_stop_trigger dm7820_prgclk_stop_trigger

  Programmable clock stop trigger type.
- typedef enum _dm7820_pwm_modulator dm7820_pwm_modulator

  Pulse width modulator type.
- typedef enum _dm7820_pwm_period_master_clock dm7820_pwm_period_master_clock

  Pulse width modulator period master clock type.
- typedef enum _dm7820_pwm_output dm7820_pwm_output

  Pulse width modulator output type.
- typedef enum _dm7820_pwm_width_master_clock dm7820_pwm_width_master_clock

  Pulse width modulator width master clock type.
- typedef enum _dm7820_incenc_encoder dm7820_incenc_encoder

  Incremental encoder type.
- typedef enum _dm7820_incenc_master_clock dm7820_incenc_master_clock

  Incremental encoder master clock type.
- typedef enum _dm7820_incenc_input_mode dm7820_incenc_input_mode

  Incremental encoder input mode type.
- typedef enum _dm7820_incenc_channel_mode dm7820_incenc_channel_mode

  Incremental encoder channel mode type.
- typedef enum _dm7820_incenc_phase_transition dm7820_incenc_phase_transition

  Incremental encoder phase filter transition type.
- typedef enum _dm7820_incenc_channel dm7820_incenc_channel

  Incremental encoder channel type.

- typedef enum _dm7820_incenc_status_condition dm7820_incenc_status_condition

  Incremental encoder status condition type.
- typedef enum _dm7820_fifo_queue dm7820_fifo_queue

  FIFO type.
- typedef enum _dm7820_fifo_input_clock dm7820_fifo_input_clock

  FIFO input clock type.
- typedef enum _dm7820_fifo_output_clock dm7820_fifo_output_clock

  FIFO output clock type.
- typedef enum _dm7820_fifo_dma_request dm7820_fifo_dma_request

  FIFO DMA request source type.
- typedef enum _dm7820_fifo_data_input dm7820_fifo_data_input

  FIFO data input type.
- typedef enum _dm7820_fifo_status_condition dm7820_fifo_status_condition

  FIFO status condition type.
- typedef enum _dm7820_advint_interrupt dm7820_advint_interrupt

  Advanced interrupt type.
- typedef enum _dm7820_advint_mode dm7820_advint_mode

  Advanced interrupt mode type.
- typedef enum _dm7820_advint_master_clock dm7820_advint_master_clock

  Advanced interrupt master clock type.
- typedef enum _dm7820_interrupt_source dm7820_interrupt_source

  Interrupt source type.
- typedef enum _dm7820_minor_interrupt_register dm7820_minor_interrupt_register

  Minor interrupt control/status register type.
- typedef struct _dm7820_interrupt_info dm7820_interrupt_info

  Interrupt source information type.
- typedef struct dm7820_pci_access_request dm7820_pci_access_request_t
- typedef struct dm7820_interrupt_control dm7820_interrupt_control_t

  Interrupt control information type.
- typedef struct dm7820_minor_int_reg_layout dm7820_minor_int_reg_layout_t

  Minor interrupt register bit layout type.
- typedef uint64_t dm7820_int_source_status_t

  Interrupt source status type.

Enumerations

- enum dm7820_pci_region_num { DM7820_PCI_REGION_PLX_MEM = 0, DM7820_PCI_REGION_PLX_IO, DM7820_PCI_REGION_FPGA_MEM }

  Standard PCI region number.
- enum dm7820_pci_region_access_size { DM7820_PCI_REGION_ACCESS_8 = 0, DM7820_PCI_REGION_ACCESS_16, DM7820_PCI_REGION_ACCESS_32 }

  Desired size in bits of access to standard PCI region.
- enum _DM7820_StdIO_Port { DM7820_STDIO_PORT_0 = 0, DM7820_STDIO_PORT_1, DM7820_STDIO_PORT_2 }

  Standard I/O ports.
- enum _DM7820_StdIO_IO_Mode { DM7820_STDIO_MODE_INPUT = 0, DM7820_STDIO_MODE_OUTPUT, DM7820_STDIO_MODE_PER_OUT }

  Standard I/O port modes.
- enum _DM7820_StdIO_Periph_Mode { DM7820_STDIO_PERIPH_PWM = 0x0, DM7820_STDIO_PERIPH_CLK_OTHER, DM7820_STDIO_PERIPH_FIFO_0, DM7820_STDIO_PERIPH_FIFO_1 }

  Standard I/O port peripheral output modes.
- enum _DM7820_StdIO_Strobe { DM7820_STDIO_STROBE_1 = 0, DM7820_STDIO_STROBE_2 }

  Strobe signals.
- enum _dm7820_tmrctr_timer { DM7820_TMRCTR_TIMER_A_0 = 0, DM7820_TMRCTR_TIMER_A_1, DM7820_TMRCTR_TIMER_A_2, DM7820_TMRCTR_TIMER_B_0, DM7820_TMRCTR_TIMER_B_1, DM7820_TMRCTR_TIMER_B_2 }

  8254 timers/counters
- enum _dm7820_tmrctr_clock { DM7820_TMRCTR_CLOCK_5_MHZ = 0, DM7820_TMRCTR_CLOCK_RESERVED, DM7820_TMRCTR_CLOCK_8254_A_0, DM7820_TMRCTR_CLOCK_8254_A_1, DM7820_TMRCTR_CLOCK_8254_A_2, DM7820_TMRCTR_CLOCK_8254_B_0, DM7820_TMRCTR_CLOCK_8254_B_1, DM7820_TMRCTR_CLOCK_8254_B_2, DM7820_TMRCTR_CLOCK_PROG_CLOCK_0, DM7820_TMRCTR_CLOCK_PROG_CLOCK_1, DM7820_TMRCTR_CLOCK_PROG_CLOCK_2, DM7820_TMRCTR_CLOCK_PROG_CLOCK_3, DM7820_TMRCTR_CLOCK_STROBE_1, DM7820_TMRCTR_CLOCK_STROBE_2, DM7820_TMRCTR_CLOCK_INV_STROBE_1, DM7820_TMRCTR_CLOCK_INV_STROBE_2 }

  8254 timer/counter clock selectors
- enum _dm7820_tmrctr_gate { DM7820_TMRCTR_GATE_LOGIC_0 = 0, DM7820_TMRCTR_GATE_LOGIC_1, DM7820_TMRCTR_GATE_8254_A_0, DM7820_TMRCTR_GATE_8254_A_1, DM7820_TMRCTR_GATE_8254_A_2, DM7820_TMRCTR_GATE_8254_B_0, DM7820_TMRCTR_GATE_8254_B_1, DM7820_TMRCTR_GATE_8254_B_2, DM7820_TMRCTR_GATE_PROG_CLOCK_0, DM7820_TMRCTR_GATE_PROG_CLOCK_1, DM7820_TMRCTR_GATE_PROG_CLOCK_2, DM7820_TMRCTR_GATE_PROG_CLOCK_3, DM7820_TMRCTR_GATE_STROBE_1, DM7820_TMRCTR_GATE_STROBE_2, DM7820_TMRCTR_GATE_INV_STROBE_1, DM7820_TMRCTR_GATE_INV_STROBE_2, DM7820_TMRCTR_GATE_PORT_2_BIT_0, DM7820_TMRCTR_GATE_PORT_2_BIT_1, DM7820_TMRCTR_GATE_PORT_2_BIT_2, DM7820_TMRCTR_GATE_PORT_2_BIT_3, DM7820_TMRCTR_GATE_PORT_2_BIT_4, DM7820_TMRCTR_GATE_PORT_2_BIT_5, DM7820_TMRCTR_GATE_PORT_2_BIT_6, DM7820_TMRCTR_GATE_PORT_2_BIT_7, DM7820_TMRCTR_GATE_PORT_2_BIT_8, DM7820_TMRCTR_GATE_PORT_2_BIT_9, DM7820_TMRCTR_GATE_PORT_2_BIT_10, DM7820_TMRCTR_GATE_PORT_2_BIT_11, DM7820_TMRCTR_GATE_PORT_2_BIT_12, DM7820_TMRCTR_GATE_PORT_2_BIT_13, DM7820_TMRCTR_GATE_PORT_2_BIT_14, DM7820_TMRCTR_GATE_PORT_2_BIT_15 }

  8254 timer/counter gate selectors
- enum _dm7820_tmrctr_waveform { DM7820_TMRCTR_WAVEFORM_EVENT_CTR = 0, DM7820_TMRCTR_WAVEFORM_PROG_ONE_SHOT, DM7820_TMRCTR_WAVEFORM_RATE_GENERATOR, DM7820_TMRCTR_WAVEFORM_SQUARE-

_WAVE,
DM7820_TMRCTR_WAVEFORM_SOFTWARE_STROBE, DM7820_TMRCTR_WAVEFORM_HARDWARE_STROBE }

8254 timer/counter waveform mode selectors

- enum _dm7820_tmrctr_count_mode { DM7820_TMRCTR_COUNT_MODE_BINARY = 0, DM7820_TMRCTR_COUNT_MODE_BCD }

  8254 timer/counter count mode selectors

- enum _dm7820_prgclk_clock { DM7820_PRGCLK_CLOCK_0 = 0, DM7820_PRGCLK_CLOCK_1, DM7820_PRGCLK_CLOCK_2, DM7820_PRGCLK_CLOCK_3 }

  Programmable clocks.

- enum _dm7820_prgclk_mode { DM7820_PRGCLK_MODE_DISABLED = 0, DM7820_PRGCLK_MODE_CONTINUOUS, DM7820_PRGCLK_MODE_RESERVED, DM7820_PRGCLK_MODE_ONE_SHOT }

  Programmable clock modes.

- enum _dm7820_prgclk_master_clock {
  DM7820_PRGCLK_MASTER_25_MHZ = 0, DM7820_PRGCLK_MASTER_SAMPLE_CLOCK, DM7820_PRGCLK_MASTER_8254_A_0, DM7820_PRGCLK_MASTER_8254_A_1,
  DM7820_PRGCLK_MASTER_8254_A_2, DM7820_PRGCLK_MASTER_8254_B_0, DM7820_PRGCLK_MASTER_8254_B_1, DM7820_PRGCLK_MASTER_8254_B_2,
  DM7820_PRGCLK_MASTER_PROG_CLOCK_0, DM7820_PRGCLK_MASTER_PROG_CLOCK_1, DM7820_PRGCLK_MASTER_PROG_CLOCK_2, DM7820_PRGCLK_MASTER_PROG_CLOCK_3,
  DM7820_PRGCLK_MASTER_STROBE_1, DM7820_PRGCLK_MASTER_STROBE_2, DM7820_PRGCLK_MASTER_INV_STROBE_1, DM7820_PRGCLK_MASTER_INV_STROBE_2 }

  Programmable clock master clocks.

- enum _dm7820_prgclk_start_trigger {
  DM7820_PRGCLK_START_IMMEDIATE = 0, DM7820_PRGCLK_START_RESERVED_1, DM7820_PRGCLK_START_8254_A_0, DM7820_PRGCLK_START_8254_A_1,
  DM7820_PRGCLK_START_8254_A_2, DM7820_PRGCLK_START_8254_B_0, DM7820_PRGCLK_START_8254_B_1, DM7820_PRGCLK_START_8254_B_2,
  DM7820_PRGCLK_START_PROG_CLOCK_0, DM7820_PRGCLK_START_PROG_CLOCK_1, DM7820_PRGCLK_START_PROG_CLOCK_2, DM7820_PRGCLK_START_PROG_CLOCK_3,
  DM7820_PRGCLK_START_STROBE_1, DM7820_PRGCLK_START_STROBE_2, DM7820_PRGCLK_START_INV_STROBE_1, DM7820_PRGCLK_START_INV_STROBE_2,
  DM7820_PRGCLK_START_ADVANCED_INT_0, DM7820_PRGCLK_START_ADVANCED_INT_1, DM7820_PRGCLK_START_8254_INT, DM7820_PRGCLK_START_RESERVED_2,
  DM7820_PRGCLK_START_INC_ENCODER_0_INT, DM7820_PRGCLK_START_INC_ENCODER_1_INT,
  DM7820_PRGCLK_START_RESERVED_3, DM7820_PRGCLK_START_RESERVED_4,
  DM7820_PRGCLK_START_PWM_0_INT, DM7820_PRGCLK_START_PWM_1_INT, DM7820_PRGCLK_START_PROG_CLOCK_0_INT, DM7820_PRGCLK_START_PROG_CLOCK_1_INT,
  DM7820_PRGCLK_START_PROG_CLOCK_2_INT, DM7820_PRGCLK_START_PROG_CLOCK_3_INT,
  DM7820_PRGCLK_START_FIFO_0_INT, DM7820_PRGCLK_START_FIFO_1_INT }

  Programmable clock start triggers.

- enum _dm7820_prgclk_stop_trigger {
  DM7820_PRGCLK_STOP_NONE = 0, DM7820_PRGCLK_STOP_RESERVED_1, DM7820_PRGCLK_STOP_8254_A_0, DM7820_PRGCLK_STOP_8254_A_1,
  DM7820_PRGCLK_STOP_8254_A_2, DM7820_PRGCLK_STOP_8254_B_0, DM7820_PRGCLK_STOP_8254_B_1, DM7820_PRGCLK_STOP_8254_B_2,
  DM7820_PRGCLK_STOP_PROG_CLOCK_0, DM7820_PRGCLK_STOP_PROG_CLOCK_1, DM7820_PRGCLK_STOP_PROG_CLOCK_2, DM7820_PRGCLK_STOP_PROG_CLOCK_3,
  DM7820_PRGCLK_STOP_STROBE_1, DM7820_PRGCLK_STOP_STROBE_2, DM7820_PRGCLK_STOP_INV_STROBE_1, DM7820_PRGCLK_STOP_INV_STROBE_2,
  DM7820_PRGCLK_STOP_ADVANCED_INT_0, DM7820_PRGCLK_STOP_ADVANCED_INT_1, DM7820_PRGCLK_STOP_8254_INT, DM7820_PRGCLK_STOP_RESERVED_2,
  DM7820_PRGCLK_STOP_INC_ENCODER_0_INT, DM7820_PRGCLK_STOP_INC_ENCODER_1_INT, DM7820_PRGCLK_STOP_RESERVED_3, DM7820_PRGCLK_STOP_RESERVED_4,
  DM7820_PRGCLK_STOP_PWM_0_INT, DM7820_PRGCLK_STOP_PWM_1_INT, DM7820_PRGCLK_ST-

OP_PROG_CLOCK_0_INT, DM7820_PRGCLK_STOP_PROG_CLOCK_1_INT, DM7820_PRGCLK_STOP_PROG_CLOCK_2_INT, DM7820_PRGCLK_STOP_PROG_CLOCK_3_INT, DM7820_PRGCLK_STOP_FIFO_0_INT, DM7820_PRGCLK_STOP_FIFO_1_INT }

Programmable clock stop triggers.

- enum _dm7820_pwm_modulator

  Pulse width modulators.

- enum _dm7820_pwm_period_master_clock { DM7820_PWM_PERIOD_MASTER_25_MHZ = 0, DM7820_PWM_PERIOD_MASTER_RESERVED, DM7820_PWM_PERIOD_MASTER_8254_A_0, DM7820_PWM_PERIOD_MASTER_8254_A_1, DM7820_PWM_PERIOD_MASTER_8254_A_2, DM7820_PWM_PERIOD_MASTER_8254_B_0, DM7820_PWM_PERIOD_MASTER_8254_B_1, DM7820_PWM_PERIOD_MASTER_8254_B_2, DM7820_PWM_PERIOD_MASTER_PROG_CLOCK_0, DM7820_PWM_PERIOD_MASTER_PROG_CLOCK_1, DM7820_PWM_PERIOD_MASTER_PROG_CLOCK_2, DM7820_PWM_PERIOD_MASTER_PROG_CLOCK_3, DM7820_PWM_PERIOD_MASTER_STROBE_1, DM7820_PWM_PERIOD_MASTER_STROBE_2, DM7820_PWM_PERIOD_MASTER_INV_STROBE_1, DM7820_PWM_PERIOD_MASTER_INV_STROBE_2 }

  Pulse width modulator period master clocks.

- enum _dm7820_pwm_output { DM7820_PWM_OUTPUT_A = 0, DM7820_PWM_OUTPUT_B, DM7820_PWM_OUTPUT_C, DM7820_PWM_OUTPUT_D }

  Pulse width modulator outputs.

- enum _dm7820_pwm_width_master_clock { DM7820_PWM_WIDTH_MASTER_25_MHZ = 0, DM7820_PWM_WIDTH_MASTER_RESERVED, DM7820_PWM_WIDTH_MASTER_8254_A_0, DM7820_PWM_WIDTH_MASTER_8254_A_1, DM7820_PWM_WIDTH_MASTER_8254_A_2, DM7820_PWM_WIDTH_MASTER_8254_B_0, DM7820_PWM_WIDTH_MASTER_8254_B_1, DM7820_PWM_WIDTH_MASTER_8254_B_2, DM7820_PWM_WIDTH_MASTER_PROG_CLOCK_0, DM7820_PWM_WIDTH_MASTER_PROG_CLOCK_1, DM7820_PWM_WIDTH_MASTER_PROG_CLOCK_2, DM7820_PWM_WIDTH_MASTER_PROG_CLOCK_3, DM7820_PWM_WIDTH_MASTER_STROBE_1, DM7820_PWM_WIDTH_MASTER_STROBE_2, DM7820_PWM_WIDTH_MASTER_INV_STROBE_1, DM7820_PWM_WIDTH_MASTER_INV_STROBE_2 }

  Pulse width modulator width master clocks.

- enum _dm7820_incenc_encoder { DM7820_INCENC_ENCODER_0 = 0, DM7820_INCENC_ENCODER_1 }

  Incremental encoders.

- enum _dm7820_incenc_master_clock { DM7820_INCENC_MASTER_25_MHZ = 0, DM7820_INCENC_MASTER_RESERVED, DM7820_INCENC_MASTER_8254_A_0, DM7820_INCENC_MASTER_8254_A_1, DM7820_INCENC_MASTER_8254_A_2, DM7820_INCENC_MASTER_8254_B_0, DM7820_INCENC_MASTER_8254_B_1, DM7820_INCENC_MASTER_8254_B_2, DM7820_INCENC_MASTER_PROG_CLOCK_0, DM7820_INCENC_MASTER_PROG_CLOCK_1, DM7820_INCENC_MASTER_PROG_CLOCK_2, DM7820_INCENC_MASTER_PROG_CLOCK_3, DM7820_INCENC_MASTER_STROBE_1, DM7820_INCENC_MASTER_STROBE_2, DM7820_INCENC_MASTER_INV_STROBE_1, DM7820_INCENC_MASTER_INV_STROBE_2 }

  Incremental encoder master clocks.

- enum _dm7820_incenc_input_mode { DM7820_INCENC_INPUT_SINGLE_ENDED = 0, DM7820_INCENC_INPUT_DIFFERENTIAL }

  Incremental encoder input modes.

- enum _dm7820_incenc_channel_mode { DM7820_INCENC_CHANNEL_INDEPENDENT = 0, DM7820_INCENC_CHANNEL_JOINED }

  Incremental encoder channel modes.

- enum _dm7820_incenc_phase_transition { DM7820_INCENC_PHASE_BA_00_TO_01_UP = 0, DM7820_INCENC_PHASE_BA_01_TO_11_UP, DM7820_INCENC_PHASE_BA_11_TO_10_UP, DM7820_INCENC_PHASE_BA_10_TO_00_UP, DM7820_INCENC_PHASE_BA_01_TO_00_DOWN, DM7820_INCENC_PHASE_BA_11_TO_01_DOWN,

DM7820_INCENC_PHASE_BA_10_TO_11_DOWN, DM7820_INCENC_PHASE_BA_00_TO_10_DOWN }

Incremental encoder phase filter transitions.

- enum _dm7820_incenc_channel { DM7820_INCENC_CHANNEL_A = 0, DM7820_INCENC_CHANNEL_B }

  Incremental encoder channels.

- enum _dm7820_incenc_status_condition { DM7820_INCENC_STATUS_CHANNEL_A_POSITIVE_ROLLOVER = 0, DM7820_INCENC_STATUS_CHANNEL_A_NEGATIVE_ROLLOVER, DM7820_INCENC_STATUS_CHANNEL_B_POSITIVE_ROLLOVER, DM7820_INCENC_STATUS_CHANNEL_B_NEGATIVE_ROLLOVER }

  Incremental encoder status conditions.

- enum _dm7820_fifo_queue { DM7820_FIFO_QUEUE_0 = 0, DM7820_FIFO_QUEUE_1 }

  FIFOs.

- enum _dm7820_fifo_input_clock {
  DM7820_FIFO_INPUT_CLOCK_25_MHZ = 0, DM7820_FIFO_INPUT_CLOCK_RESERVED_1, DM7820_FIFO_INPUT_CLOCK_8254_A_0, DM7820_FIFO_INPUT_CLOCK_8254_A_1,
  DM7820_FIFO_INPUT_CLOCK_8254_A_2, DM7820_FIFO_INPUT_CLOCK_8254_B_0, DM7820_FIFO_INPUT_CLOCK_8254_B_1, DM7820_FIFO_INPUT_CLOCK_8254_B_2,
  DM7820_FIFO_INPUT_CLOCK_PROG_CLOCK_0, DM7820_FIFO_INPUT_CLOCK_PROG_CLOCK_1, DM7820_FIFO_INPUT_CLOCK_PROG_CLOCK_2, DM7820_FIFO_INPUT_CLOCK_PROG_CLOCK_3,
  DM7820_FIFO_INPUT_CLOCK_STROBE_1, DM7820_FIFO_INPUT_CLOCK_STROBE_2, DM7820_FIFO_INPUT_CLOCK_INV_STROBE_1, DM7820_FIFO_INPUT_CLOCK_INV_STROBE_2,
  DM7820_FIFO_INPUT_CLOCK_ADVANCED_INT_0, DM7820_FIFO_INPUT_CLOCK_ADVANCED_INT_1, DM7820_FIFO_INPUT_CLOCK_8254_INT, DM7820_FIFO_INPUT_CLOCK_RESERVED_2,
  DM7280_FIFO_INPUT_CLOCK_INC_ENCODER_0_INT, DM7280_FIFO_INPUT_CLOCK_INC_ENCODER_1_INT, DM7820_FIFO_INPUT_CLOCK_RESERVED_3, DM7820_FIFO_INPUT_CLOCK_RESERVED_4,
  DM7820_FIFO_INPUT_CLOCK_PWM_0_INT, DM7820_FIFO_INPUT_CLOCK_PWM_1_INT, DM7820_FIFO_INPUT_CLOCK_PROG_CLOCK_0_INT, DM7820_FIFO_INPUT_CLOCK_PROG_CLOCK_1_INT,
  DM7820_FIFO_INPUT_CLOCK_PROG_CLOCK_2_INT, DM7820_FIFO_INPUT_CLOCK_PROG_CLOCK_3_INT, DM7820_FIFO_INPUT_CLOCK_PCI_READ, DM7820_FIFO_INPUT_CLOCK_PCI_WRITE }

  FIFO input clocks.

- enum _dm7820_fifo_output_clock {
  DM7820_FIFO_OUTPUT_CLOCK_25_MHZ = 0, DM7820_FIFO_OUTPUT_CLOCK_RESERVED_1, DM7820_FIFO_OUTPUT_CLOCK_8254_A_0, DM7820_FIFO_OUTPUT_CLOCK_8254_A_1,
  DM7820_FIFO_OUTPUT_CLOCK_8254_A_2, DM7820_FIFO_OUTPUT_CLOCK_8254_B_0, DM7820_FIFO_OUTPUT_CLOCK_8254_B_1, DM7820_FIFO_OUTPUT_CLOCK_8254_B_2,
  DM7820_FIFO_OUTPUT_CLOCK_PROG_CLOCK_0, DM7820_FIFO_OUTPUT_CLOCK_PROG_CLOCK_1, DM7820_FIFO_OUTPUT_CLOCK_PROG_CLOCK_2, DM7820_FIFO_OUTPUT_CLOCK_PROG_CLOCK_3,
  DM7820_FIFO_OUTPUT_CLOCK_STROBE_1, DM7820_FIFO_OUTPUT_CLOCK_STROBE_2, DM7820_FIFO_OUTPUT_CLOCK_INV_STROBE_1, DM7820_FIFO_OUTPUT_CLOCK_INV_STROBE_2,
  DM7820_FIFO_OUTPUT_CLOCK_ADVANCED_INT_0, DM7820_FIFO_OUTPUT_CLOCK_ADVANCED_INT_1, DM7820_FIFO_OUTPUT_CLOCK_8254_INT, DM7820_FIFO_OUTPUT_CLOCK_RESERVED_2,
  DM7280_FIFO_OUTPUT_CLOCK_INC_ENCODER_0_INT, DM7280_FIFO_OUTPUT_CLOCK_INC_ENCODER_1_INT, DM7820_FIFO_OUTPUT_CLOCK_RESERVED_3, DM7820_FIFO_OUTPUT_CLOCK_RESERVED_4,
  DM7820_FIFO_OUTPUT_CLOCK_PWM_0_INT, DM7820_FIFO_OUTPUT_CLOCK_PWM_1_INT, DM7820_FIFO_OUTPUT_CLOCK_PROG_CLOCK_0_INT, DM7820_FIFO_OUTPUT_CLOCK_PROG_CLOCK_1_INT,
  DM7820_FIFO_OUTPUT_CLOCK_PROG_CLOCK_2_INT, DM7820_FIFO_OUTPUT_CLOCK_PROG_CLOCK_3_INT, DM7820_FIFO_OUTPUT_CLOCK_PCI_READ, DM7820_FIFO_OUTPUT_CLOCK_PCI_WRITE }

  FIFO output clocks.

- enum _dm7820_fifo_dma_request { DM7820_FIFO_DMA_REQUEST_READ = 0, DM7820_FIFO_DMA_REQUEST_NOT_EMPTY, DM7820_FIFO_DMA_REQUEST_WRITE, DM7820_FIFO_DMA_REQUEST_NOT_FULL }

FIFO DMA request sources.

- enum _dm7820_fifo_data_input { DM7820_FIFO_0_DATA_INPUT_PCI_DATA = 0, DM7820_FIFO_0_DATA_INPUT_PORT_0, DM7820_FIFO_0_DATA_INPUT_PORT_2, DM7820_FIFO_0_DATA_INPUT_FIFO_0_OUTPUT, DM7820_FIFO_1_DATA_INPUT_PCI_DATA, DM7820_FIFO_1_DATA_INPUT_PORT_1, DM7820_FIFO_1_DATA_INPUT_INC_ENCODER_1_A, DM7820_FIFO_1_DATA_INPUT_INC_ENCODER_1_B }

  FIFO data inputs.

- enum _dm7820_fifo_status_condition { DM7820_FIFO_STATUS_READ_REQUEST = 0, DM7820_FIFO_STATUS_WRITE_REQUEST, DM7820_FIFO_STATUS_FULL, DM7820_FIFO_STATUS_EMPTY, DM7820_FIFO_STATUS_OVERFLOW, DM7820_FIFO_STATUS_UNDERFLOW }

  FIFO status conditions.

- enum _dm7820_advint_interrupt { DM7820_ADVINT_INTERRUPT_0 = 0, DM7820_ADVINT_INTERRUPT_1 }

  Advanced interrupts.

- enum _dm7820_advint_mode { DM7820_ADVINT_MODE_DISABLED = 0, DM7820_ADVINT_MODE_STROBE, DM7820_ADVINT_MODE_MATCH, DM7820_ADVINT_MODE_EVENT }

  Advanced interrupt modes.

- enum _dm7820_advint_master_clock { DM7820_ADVINT_MASTER_25_MHZ = 0, DM7820_ADVINT_MASTER_RESERVED, DM7820_ADVINT_MASTER_8254_A_0, DM7820_ADVINT_MASTER_8254_A_1, DM7820_ADVINT_MASTER_8254_A_2, DM7820_ADVINT_MASTER_8254_B_0, DM7820_ADVINT_MASTER_8254_B_1, DM7820_ADVINT_MASTER_8254_B_2, DM7820_ADVINT_MASTER_PROG_CLOCK_0, DM7820_ADVINT_MASTER_PROG_CLOCK_1, DM7820_ADVINT_MASTER_PROG_CLOCK_2, DM7820_ADVINT_MASTER_PROG_CLOCK_3, DM7820_ADVINT_MASTER_STROBE_1, DM7820_ADVINT_MASTER_STROBE_2, DM7820_ADVINT_MASTER_INV_STROBE_1, DM7820_ADVINT_MASTER_INV_STROBE_2 }

  Advanced interrupt master clocks.

- enum _dm7820_interrupt_source { DM7820_INTERRUPT_ADVINT_0 = 0, DM7820_INTERRUPT_ADVINT_1, DM7820_INTERRUPT_FIFO_0_EMPTY, DM7820_INTERRUPT_FIFO_0_FULL, DM7820_INTERRUPT_FIFO_0_OVERFLOW, DM7820_INTERRUPT_FIFO_0_READ_REQUEST, DM7820_INTERRUPT_FIFO_0_UNDERFLOW, DM7820_INTERRUPT_FIFO_0_WRITE_REQUEST, DM7820_INTERRUPT_FIFO_1_EMPTY, DM7820_INTERRUPT_FIFO_1_FULL, DM7820_INTERRUPT_FIFO_1_OVERFLOW, DM7820_INTERRUPT_FIFO_1_READ_REQUEST, DM7820_INTERRUPT_FIFO_1_UNDERFLOW, DM7820_INTERRUPT_FIFO_1_WRITE_REQUEST, DM7820_INTERRUPT_INCENC_0_CHANNEL_A_NEGATIVE_ROLLOVER, DM7820_INTERRUPT_INCENC_0_CHANNEL_A_POSITIVE_ROLLOVER, DM7820_INTERRUPT_INCENC_0_CHANNEL_B_NEGATIVE_ROLLOVER, DM7820_INTERRUPT_INCENC_0_CHANNEL_B_POSITIVE_ROLLOVER, DM7820_INTERRUPT_INCENC_1_CHANNEL_A_NEGATIVE_ROLLOVER, DM7820_INTERRUPT_INCENC_1_CHANNEL_A_POSITIVE_ROLLOVER, DM7820_INTERRUPT_INCENC_1_CHANNEL_B_NEGATIVE_ROLLOVER, DM7820_INTERRUPT_INCENC_1_CHANNEL_B_POSITIVE_ROLLOVER, DM7820_INTERRUPT_PRGCLK_0, DM7820_INTERRUPT_PRGCLK_1, DM7820_INTERRUPT_PRGCLK_2, DM7820_INTERRUPT_PRGCLK_3, DM7820_INTERRUPT_PWM_0, DM7820_INTERRUPT_PWM_1, DM7820_INTERRUPT_TMRCTR_A_0, DM7820_INTERRUPT_TMRCTR_A_1, DM7820_INTERRUPT_TMRCTR_A_2, DM7820_INTERRUPT_TMRCTR_B_0, DM7820_INTERRUPT_TMRCTR_B_1, DM7820_INTERRUPT_TMRCTR_B_2, DM7820_INTERRUPT_FIFO_0_DMA_DONE, DM7820_INTERRUPT_FIFO_1_DMA_DONE, DM7820_INTERRUPT_NONE }

  Interrupt sources.

- enum _dm7820_minor_interrupt_register { DM7820_MINOR_INT_REG_FIFO_0_INT = 0, DM7820_MINOR_INT_REG_FIFO_1_INT, DM7820_MINOR_INT_REG_INCENC_0_INT, DM7820_MINOR_INT_REG_INCENC_1_INT, DM7820_MINOR_INT_REG_TMRCTR_INT, DM7820_MINOR_INT_REG_NONE }

  Minor interrupt control/status registers.

6.32.1 Detailed Description

Type definitions used both in kernel and user space.

```
//-----------------------------------------------------------------------------
//  COPYRIGHT (C) RTD EMBEDDED TECHNOLOGIES, INC.  ALL RIGHTS RESERVED.
//
//  This software package is dual-licensed.  Source code that is compiled for
//  kernel mode execution is licensed under the GNU General Public License
//  version 2.  For a copy of this license, refer to the file
//  LICENSE_GPLv2.TXT (which should be included with this software) or contact
//  the Free Software Foundation.  Source code that is compiled for user mode
//  execution is licensed under the RTD End-User Software License Agreement.
//  For a copy of this license, refer to LICENSE.TXT or contact RTD Embedded
//  Technologies, Inc.  Using this software indicates agreement with the
//  license terms listed above.
//-----------------------------------------------------------------------------
```

Id:

dm7820_types.h 86294 2015-03-04 21:36:57Z rgroner

Definition in file dm7820_types.h.

Index