OmniTek NetViz Driver

Generated by Doxygen 1.7.1

Fri Oct 29 2010 16:07:11

Contents

Chapter 1

OmniTek NetViz DMA Driver

1.1 Introduction

This is the documentation for the OmniTek DMA Driver supplied to support the Barco NetViz platform.

1.1.1 Copyright

1.2 Guides

Chapter 2

NetViz Driver DMA API Documentation

2.1 Introduction

The following details the driver level API for DMA operation

The DMA functionality of the driver is provided in several layers. The lowest layer, the DMA core is responsible for programming the required scatter gather entries etc. for the device, and operating the device DMA Channels.

On top of the core layer is the DMA Request layer. This processes DMA requests, handling the necessary completion, cancellation and channel management operations. The request layer makes use of queues to manage multiple outstanding requests. Essentially this is the API layer.

The DMA request layer is designed to accept requests from multiple sources, e.g. blocking IO calls, async IO calls and IOCtl IO calls.

2.2 Implementation

As an example we can look at OmniTekDma.c This uses DMA to provide blocking IO functionality.

2.2.1 Request Creation

DMA Transfers are represented as transaction requests. In this case we provide the DMA channel, the user buffer (buf), size of transfer (count) local address (f_pos) and direction (write). We also specify the transaction type, and a callback function. The routine will allocate memory for and initialize a request with the supplied information, and submit it to the queue for processing.

```
//Create a transaction
result = OmniTekDMACreateRequest(
  pChannel,
  buf,
  count,
  *f_pos,
  write,
  OMNITEK_TRANSACTION_SYNC_IO,
  NULL,   //We don't need to store any extra information
```

```
    &OmniTek_MDMA_dev_complete,
    &pTransaction
  );
```

The result will either be a 0 (success) indicating the transaction has been accepted for the hardware, -EBUSY if the hardware is busy and the transaction has been added to a queue, or another code indicating that some error occurred.

2.2.2 Completion

Once created the DMA request will be processed, and once the transfer has occurred, the callback routine will be called, with a status indicating whether the transaction completed succesfully, or if it was cancelled.

In the OmniTekDma code the process requesting the transfer is put to sleep after the request is created, and placed on a wait queue. The callback function wakes up this wait queue, causing any processes that have completed requests to resume. The requesting process can then continue, in the OmnitekDma case returning to the user application the transfer status.

Once the process no longer needs the request it should call OmniTekDMARequestRelease() which will free the memory allocated for the request.

2.2.3 Cancellation

Requests may need cancellation. This is achieved through calling OmniTekDMARequestCancel() with the transaction to be cancelled.

Cancellation is complicated by the nature of DMA - the current state of the hardware is not known. If the supplied transaction is pending and has not been programmed for the hardware then it can be cancelled immediately. If, however, it has been programmed to hardware it is not possible to know whether the hardware has already begun this request. In this case cancellation requires stopping the DMA channel. This will cancel *ALL* transactions for the channel, whether pending or active, and hard abort the channel. This may leave the DMA controller in an unstable state, and in practice there should be no need to cancel requests except in exceptional situations (shutdown, application crash etc.).

The OmniTekDma.c code cancels transactions that do not complete within a specified time. The blocking IO process waits until either the transaction completes or until a timeout occurs. If the timeout occurs the request is cancelled and the process sleeps again. Once the request is cancelled the associated completion callback will be called, which will cause the process to wake up.

This second wait is important as it is possible that the request completes during the cancel call. In any case OmniTekDMARequestRelease() cannot be called until the transaction is finished with - e.g. after the completion callback has occurred.

Chapter 3

Data Structure Index

3.1 Data Structures

Here are the data structures with brief descriptions:

Chapter 4

File Index

4.1 File List

Here is a list of all files with brief descriptions:

Chapter 5

Data Structure Documentation

5.1 _DmaSglBuffer::_Allocated Struct Reference

`#include <OmniTekResources_linux.h>`

Data Fields

- u8 * Free
- void * Memory
- size_t Size

5.1.1 Detailed Description

buffer size

Details for SGL Buffer allocated in commonbuffer

5.1.2 Field Documentation

5.1.2.1 u8* Free

Used space array from buffer - we're going to do this using a slab buffer, so hopefully this will be unnecessary!

5.1.2.2 void* Memory

Pointer to allocated memory from the commonbuffer

5.1.2.3 size_t Size

Size of allocated SGL Buffer

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

- driver/OmniTekResources_linux.h

5.2 _OMNITEK_INTERFACE_EXTENSION::_bar_registers Struct Reference

details of the registers in each BAR

`#include <OmniTek_linux.h>`

Data Fields

- u32 * regs
- u32 num_regs

5.2.1 Detailed Description

details of the registers in each BAR

5.2.2 Field Documentation

5.2.2.1 u32 num_regs

Number of registers in this BAR

5.2.2.2 u32* regs

Pointer to registers

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

- driver/OmniTek_linux.h

5.3 _dma_interrupt_info::_chan_int_counts Struct Reference

Data Fields

- atomic_t n_event_ints
- atomic_t n_sg_ints

5.3.1 Field Documentation

5.3.1.1 atomic_t n_event_ints

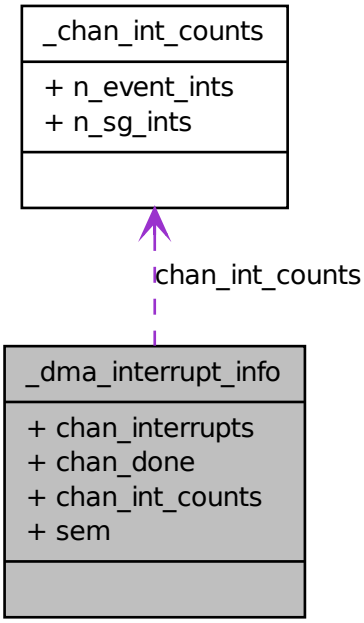
5.3.1.2 atomic_t n_sg_ints

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

- driver/OmniTekDmaIsr_linux.c

5.4 _dma_interrupt_info Struct Reference

Collaboration diagram for _dma_interrupt_info:



Data Structures

- struct _chan_int_counts

Data Fields

- u16 chan_interrupts
- u16 chan_done
- struct _dma_interrupt_info::_chan_int_counts chan_int_counts [16]
- struct semaphore sem

5.4.1 Field Documentation

5.4.1.1 u16 chan_done

5.4.1.2 struct _dma_interrupt_info::_chan_int_counts chan_int_counts[16]

5.4.1.3 u16 chan_interrupts

5.4.1.4 struct semaphore sem

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

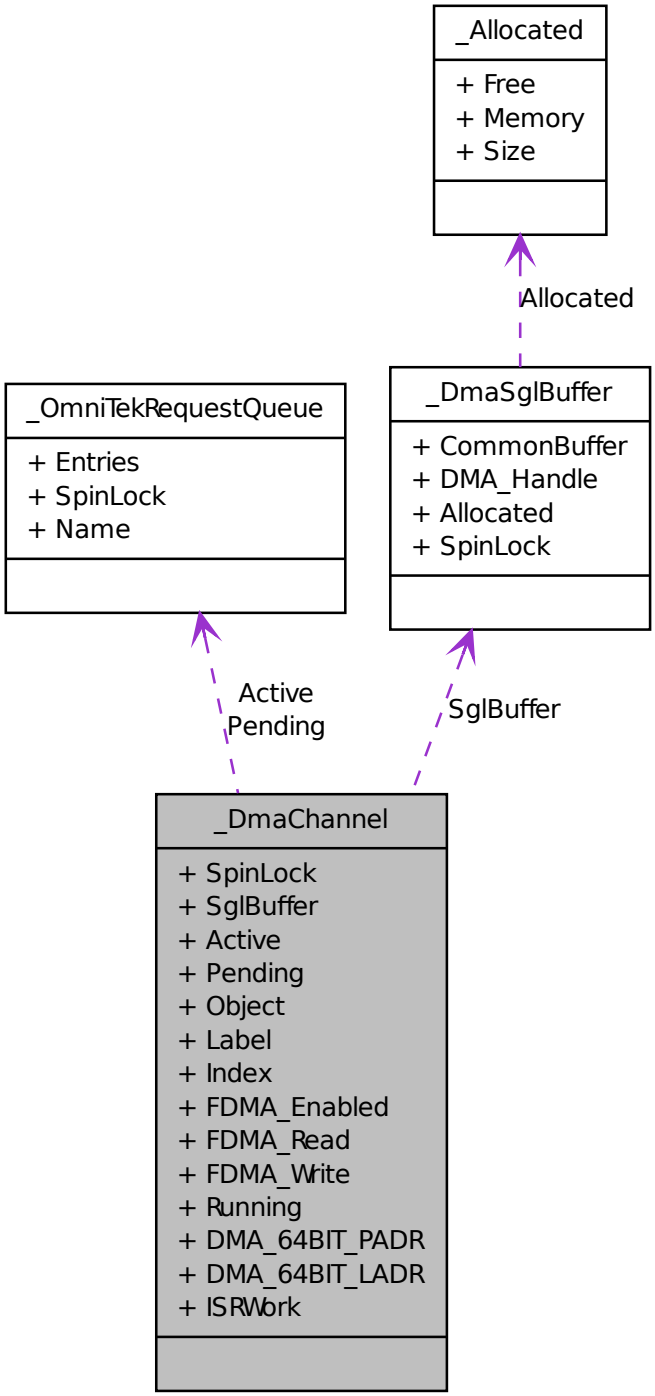
- driver/OmniTekDmaIsr_linux.c

5.5 _DmaChannel Struct Reference

Data structure containing details of a DMA Channel Resource.

`#include <OmniTekResources_linux.h>`

Collaboration diagram for _DmaChannel:



Data Fields

- spinlock_t SpinLock
- DmaSglBuffer SglBuffer
- struct _OmniTekRequestQueue Active
- struct _OmniTekRequestQueue Pending
- struct list_head Object
- u8 Label
- u8 Index
- bool FDMA_Enabled
- bool FDMA_Read
- bool FDMA_Write
- bool Running
- bool DMA_64BIT_PADR
- bool DMA_64BIT_LADR
- struct work_struct ISRWork

5.5.1 Detailed Description

Data structure containing details of a DMA Channel Resource. This data structure contains the details of each individual DMA channel.

5.5.2 Field Documentation

5.5.2.1 struct _OmniTekRequestQueue Active

Active Queue (max 1 entry for MDMA)

5.5.2.2 bool DMA_64BIT_LADR

Channel supports 64 bit local Addresses (>4GB device memory)

5.5.2.3 bool DMA_64BIT_PADR

Channel supports 64 bit PCIE Addresses (>4GB Host memory)

5.5.2.4 bool FDMA_Enabled

Channel is FDMA capable

5.5.2.5 bool FDMA_Read

Channel is FDMA Read

5.5.2.6 bool FDMA_Write

Channel is FDMA Write

5.5.2.7 u8 Index

Channel Index

5.5.2.8 struct work_struct ISRWork

Work item for ISR

5.5.2.9 u8 Label

Channel label

5.5.2.10 struct list_head Object

Object for DmaCtrl channels

5.5.2.11 struct _OmniTekRequestQueue Pending

Pending Queue

5.5.2.12 bool Running

Channel is Active

5.5.2.13 DmaSglBuffer SglBuffer

Buffer for scatter gather entries

5.5.2.14 spinlock_t SpinLock

Spin lock for this buffer

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

- driver/OmniTekResources_linux.h

5.6 _OmniTekDmaTransactionContext::_DMACoreInfo Struct Reference

`#include <OmniTekDma.h>`

Data Fields

- struct page ** pages
- struct sg_table sgt
- unsigned long first_page

- unsigned long last_page
- unsigned long num_pages
- off_t offset

5.6.1 Field Documentation

5.6.1.1 unsigned long first_page

Address for first reserved page

5.6.1.2 unsigned long last_page

Address for last reserved page

5.6.1.3 unsigned long num_pages

Number of reserved pages

5.6.1.4 off_t offset

Offset of data into first page

5.6.1.5 struct page** pages

< Data structure for DMA Core Info Reserved memory pages data structure

5.6.1.6 struct sg_table sgt

Scatter gather mappings table

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

- driver/OmniTekDma.h

5.7 _DmaCtrl Struct Reference

DMA Control Resource.

```
#include <OmniTekResources_linux.h>
```

Data Fields

- struct list_head Channels
- u8 nMDMA
- u8 nFDMARead
- u8 nFDMAWrite
- u8 nFDMABoth

- u8 nChannels
- bool DMA_64BIT_PADR
- bool DMA_64BIT_LADR
- wait_queue_head_t DMA_Wait_Queue
- struct workqueue_struct * DMA_Work_Queue
- bool DmaInterrupts

5.7.1 Detailed Description

DMA Control Resource.

5.7.2 Field Documentation

5.7.2.1 struct list_head Channels

channels associated with this controller

5.7.2.2 bool DMA_64BIT_LADR

channels support 64 bit PCIE Addresses (>4GB Host memory)

5.7.2.3 bool DMA_64BIT_PADR

5.7.2.4 wait_queue_head_t DMA_Wait_Queue

channels support 64 bit local Addresses (>4GB device memory)

5.7.2.5 struct workqueue_struct* DMA_Work_Queue

Wait queue for sleeping blocking operations

5.7.2.6 bool DmaInterrupts

Work queue for Completions Set to indicate the interrupt status is in the DMA controller

5.7.2.7 u8 nChannels

Total number of channels

5.7.2.8 u8 nFDMABoth

Number of FDMA bidir channels

5.7.2.9 u8 nFDMARead

Number of FDMA read channels

5.7.2.10 u8 nFDMAWrite

Number of FDMA write channels

5.7.2.11 u8 nMDMA

Number of MDMA Channels

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

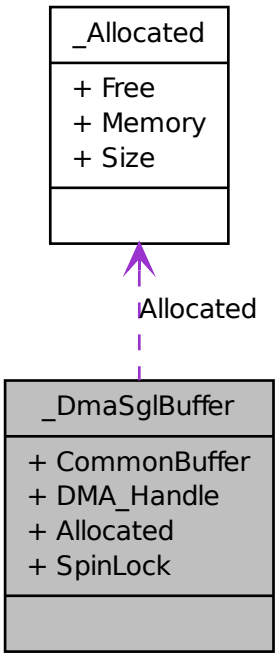
- driver/OmniTekResources_linux.h

5.8 _DmaSglBuffer Struct Reference

DMA Scatter Gather List buffer.

`#include <OmniTekResources_linux.h>`

Collaboration diagram for _DmaSglBuffer:



Data Structures

- struct _Allocated

Data Fields

- u32 * CommonBuffer
- dma_addr_t DMA_Handle
- struct _DmaSglBuffer::_Allocated Allocated
- spinlock_t SpinLock

5.8.1 Detailed Description

DMA Scatter Gather List buffer. This is the common buffer (accessible by both driver and DMA device) that stores scatter gather entries.

5.8.2 Field Documentation

5.8.2.1 struct _DmaSglBuffer::_Allocated Allocated

5.8.2.2 u32* CommonBuffer

Pointer to coherent DMA Memory for SGL Buffer

5.8.2.3 dma_addr_t DMA_Handle

5.8.2.4 spinlock_t SpinLock

Spin lock for this buffer

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

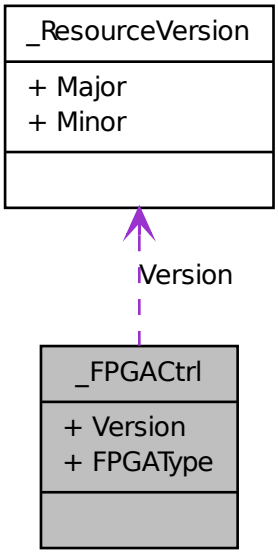
- driver/OmniTekResources_linux.h

5.9 _FPGACtrl Struct Reference

FPGA Control Resource.

`#include <OmniTekResources_linux.h>`

Collaboration diagram for _FPGACtrl:



Data Fields

- ResourceVersion Version
- u32 FPGAType

5.9.1 Detailed Description

FPGA Control Resource.

5.9.2 Field Documentation

5.9.2.1 u32 FPGAType

FPGA Identification

5.9.2.2 ResourceVersion Version

Resource Version information

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

- driver/OmniTekResources_linux.h

5.10 _GeneralCtrl Struct Reference

Data structure for PCIE BAR.

`#include <OmniTekResources_linux.h>`

Data Fields

- u8 Bar
- u32 RegisterOffset
- bool Initialised

5.10.1 Detailed Description

Data structure for PCIE BAR. This contains basic information about each of the PCIE BARs for the device

5.10.2 Field Documentation

5.10.2.1 u8 Bar

BAR Number

5.10.2.2 bool Initialised

Has this BAR been initialised

5.10.2.3 u32 RegisterOffset

Offset of registers into BAR

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

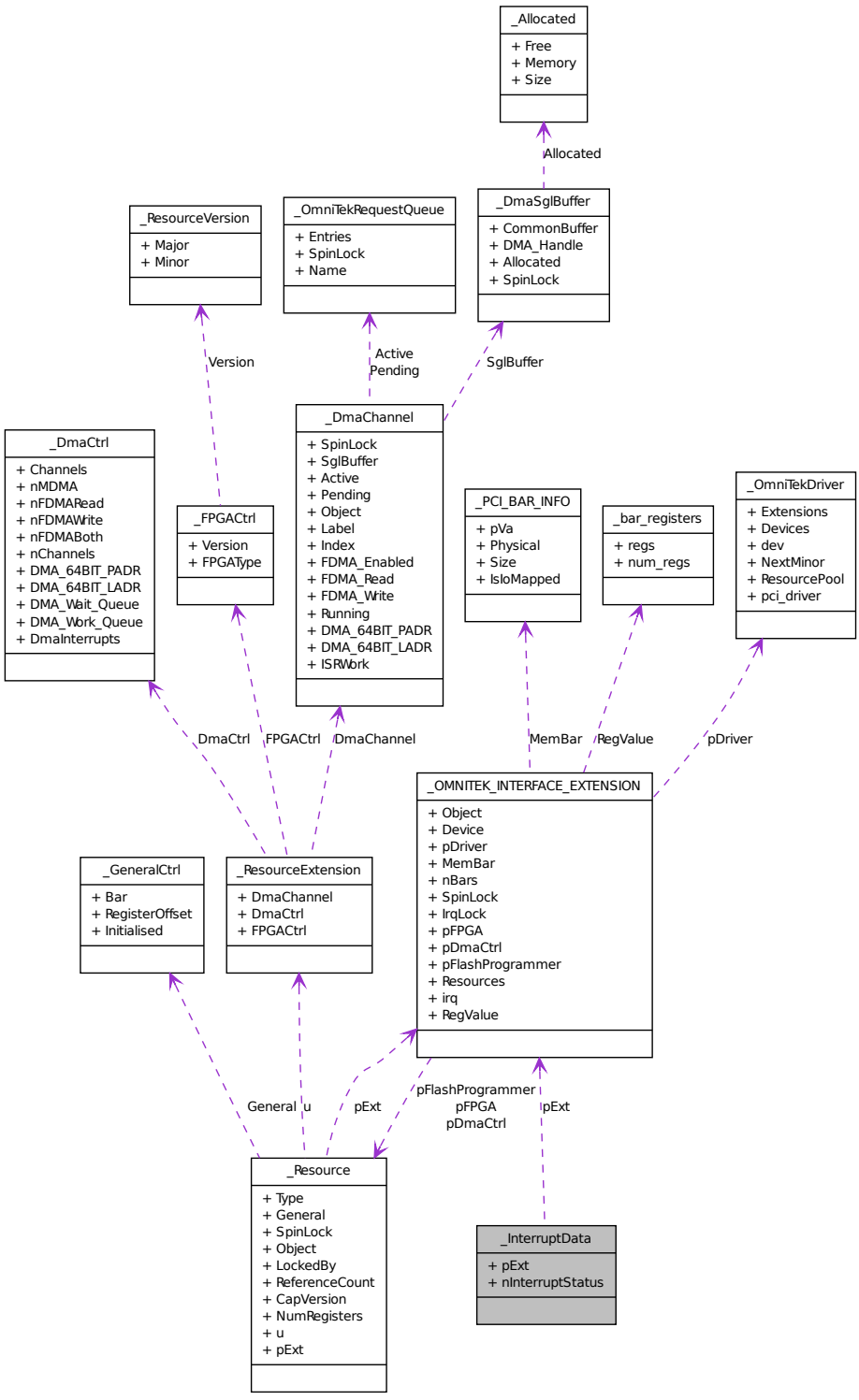
- driver/OmniTekResources_linux.h

5.11 _InterruptData Struct Reference

data about interrupts

`#include <OmniTek_linux.h>`

Collaboration diagram for _InterruptData:



Data Fields

- POMNITEK_INTERFACE_EXTENSION pExt
- u32 nInterruptStatus

5.11.1 Detailed Description

data about interrupts

5.11.2 Field Documentation

5.11.2.1 u32 nInterruptStatus

Interrupt status

5.11.2.2 POMNITEK_INTERFACE_EXTENSION pExt

Pointer to device extension

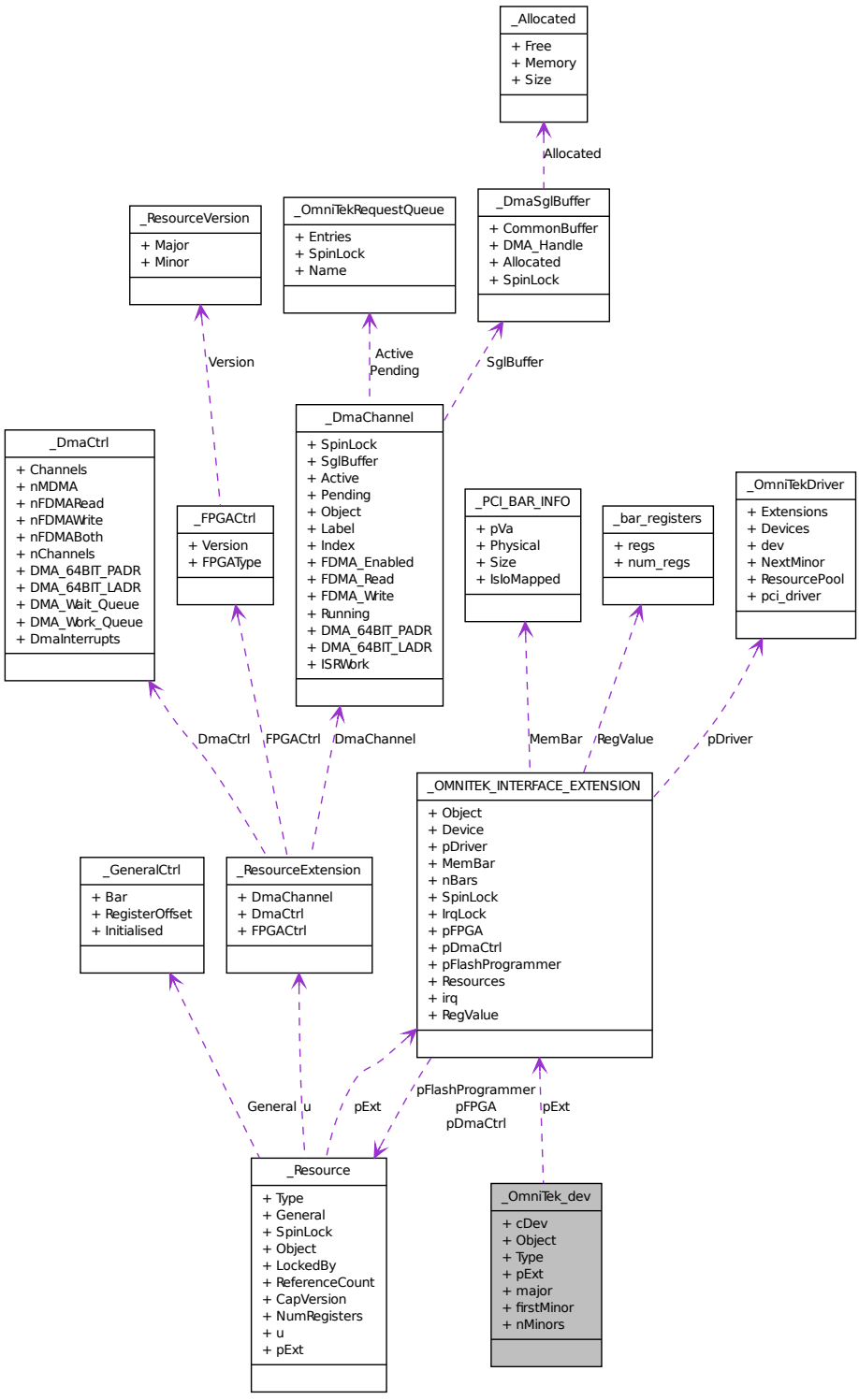
The documentation for this struct was generated from the following file:

- driver/OmniTek_linux.h

5.12 _OmniTek_dev Struct Reference

```
#include <OmniTekFops_linux.h>
```

Collaboration diagram for _OmniTek_dev:



Data Fields

- struct cdev cDev
- struct list_head Object
- OmniTek_DevTypes Type
- struct _OMNITEK_INTERFACE_EXTENSION * pExt
- int major
- int firstMinor
- int nMinors

5.12.1 Field Documentation

5.12.1.1 struct cdev cDev

cdev struct for the BAR device

5.12.1.2 int firstMinor

If there are several devices with this type, then this is minor number of the first in the group

5.12.1.3 int major

Device's major number

5.12.1.4 int nMinors

If there are several devices with this type, then this is the total number of them

5.12.1.5 struct list_head Object

For adding to the driver list of devices

5.12.1.6 struct _OMNITEK_INTERFACE_EXTENSION* pExt

Pointer to the interface extension for this BAR

5.12.1.7 OmniTek_DevTypes Type

This device's type

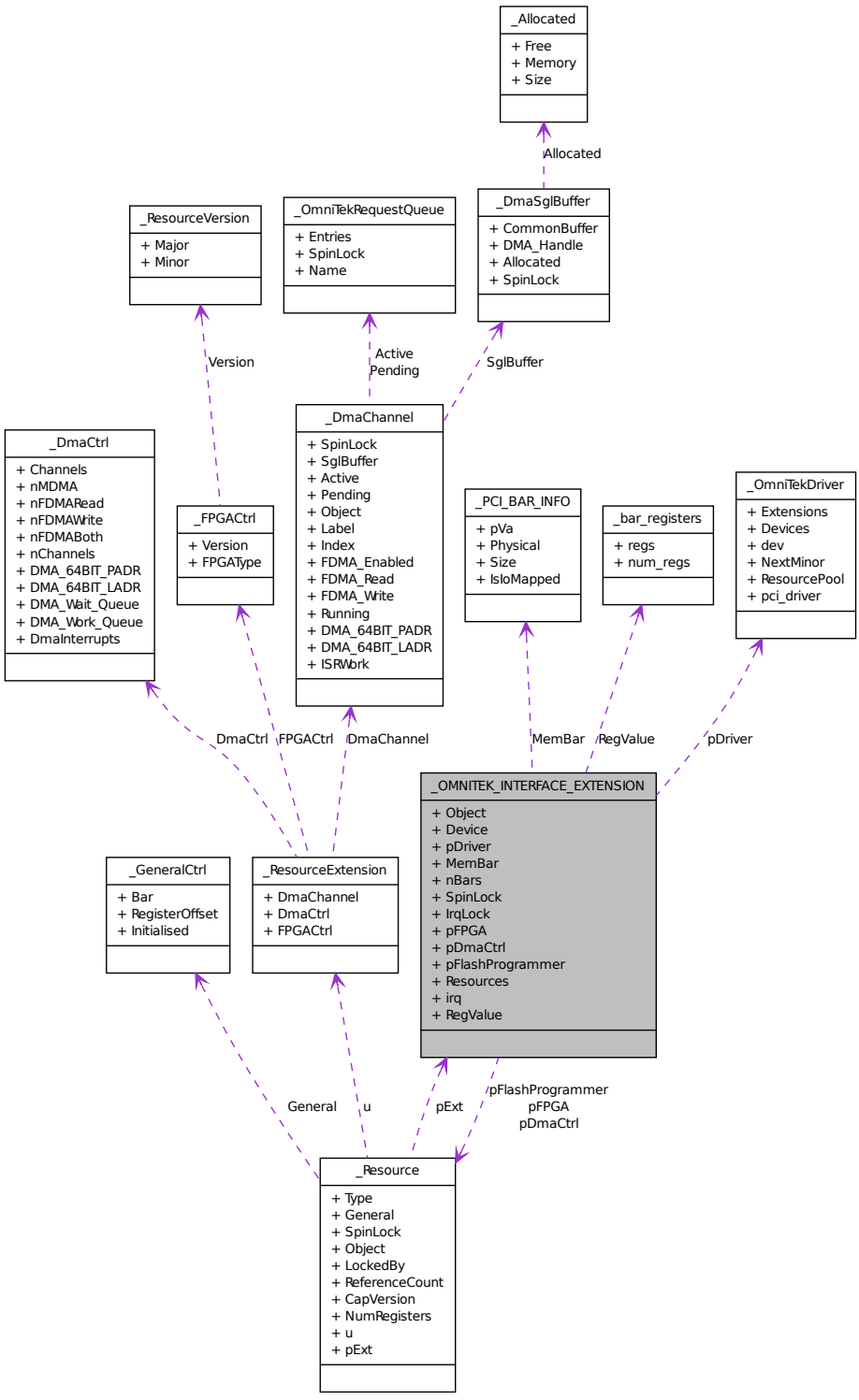
The documentation for this struct was generated from the following file:

- driver/OmniTekFops_linux.h

5.13 _OMNITEK_INTERFACE_EXTENSION Struct Reference

```
#include <OmniTek_linux.h>
```

Collaboration diagram for _OMNITEK_INTERFACE_EXTENSION:



Data Structures

- struct _bar_registers

  details of the registers in each BAR

Data Fields

- struct list_head Object
- struct pci_dev * Device
- struct _OmniTekDriver * pDriver
- PCI_BAR_INFO MemBar [MAX_NUM_MEM_BARS]
- u8 nBars
- spinlock_t SpinLock
- spinlock_t IrqLock
- PResource pFPGA
- PResource pDmaCtrl
- PResource pFlashProgrammer
- struct list_head Resources
- int irq
- struct _OMNITEK_INTERFACE_EXTENSION::_bar_registers RegValue [4]

5.13.1 Field Documentation

5.13.1.1 struct pci_dev* Device

Pointer to the underlying PCI device

5.13.1.2 int irq

IRQ Number for device

5.13.1.3 spinlock_t IrqLock

SpinLock used to disable interrupts (on current processor)

5.13.1.4 PCI_BAR_INFO MemBar[MAX_NUM_MEM_BARS]

Details of the device BARs

5.13.1.5 u8 nBars

Number of BARs available

5.13.1.6 struct list_head Object

For tracking the extension

5.13.1.7 PResource pDmaCtrl

Pointer to DMA resource

5.13.1.8 struct _OmniTekDriver* pDriver

Pointer to the OmniTekDriver struct for this device

5.13.1.9 PResource pFlashProgrammer

Pointer to Flash programmer resource

5.13.1.10 PResource pFPGA

Pointer to FPGA Resource

5.13.1.11 struct _OMNITEK_INTERFACE_EXTENSION::_bar_registers RegValue[4]

5.13.1.12 struct list_head Resources

Linked list of resources in extension

5.13.1.13 spinlock_t SpinLock

Prevent simultaneous access to the context

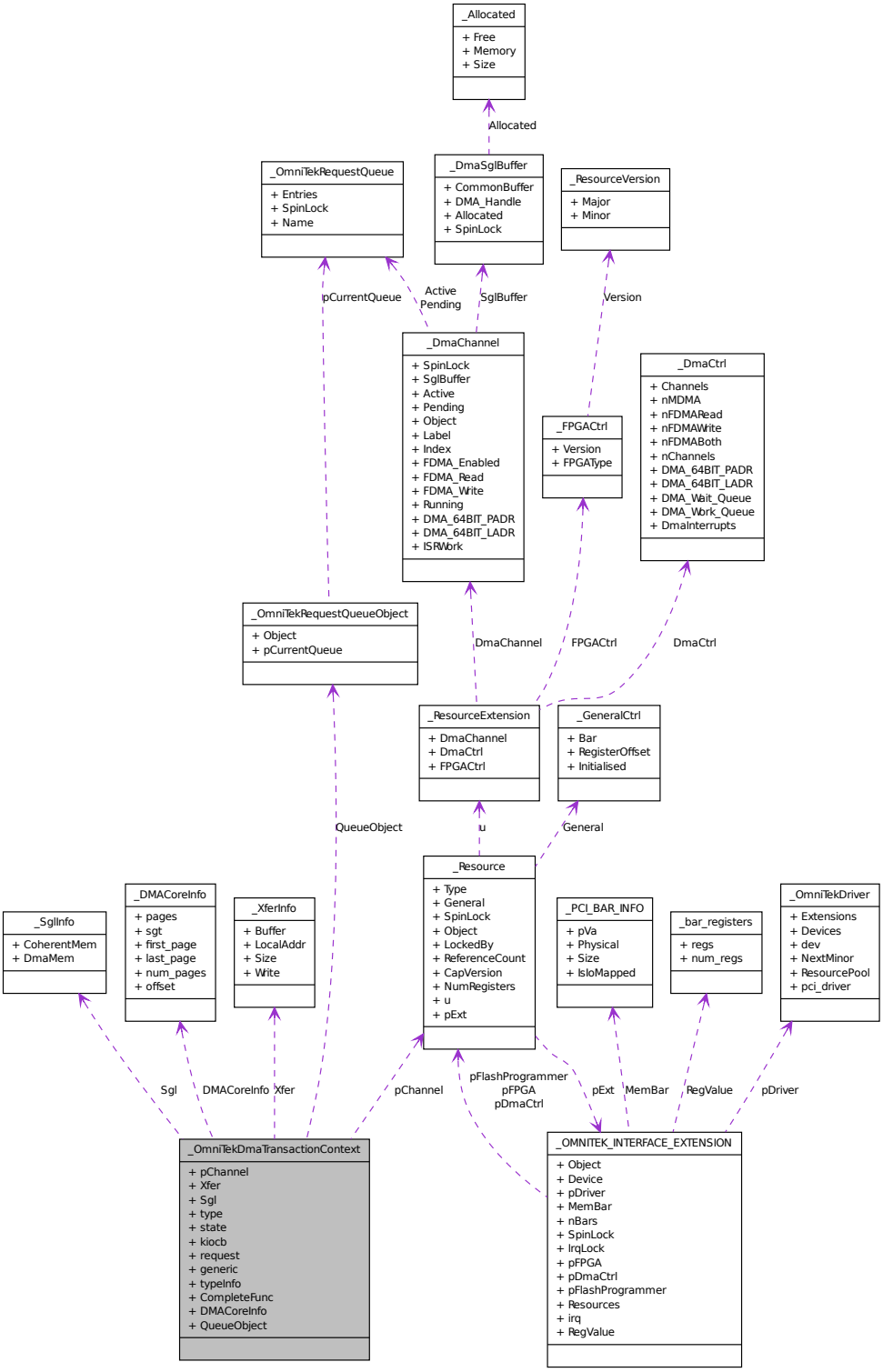
The documentation for this struct was generated from the following file:

- driver/OmniTek_linux.h

5.14 _OmniTekDmaTransactionContext Struct Reference

`#include <OmniTekDma.h>`

Collaboration diagram for _OmniTekDmaTransactionContext:



Data Structures

- struct _DMACoreInfo
- struct _SglInfo
- struct _XferInfo

Data Fields

- struct _Resource * pChannel
- struct _OmniTekDmaTransactionContext::_XferInfo Xfer
- struct _OmniTekDmaTransactionContext::_SglInfo Sgl
- OmniTekTransactionType type
- OmniTekTransactionState state
- union {
  struct kiocb * kiocb
  struct OmniTekRequest * request
  void * generic
  } typeInfo

- OmniTekTransactionCompleteCb CompleteFunc
- struct _OmniTekDmaTransactionContext::_DMACoreInfo DMACoreInfo
- struct _OmniTekRequestQueueObject QueueObject

5.14.1 Field Documentation

5.14.1.1 OmniTekTransactionCompleteCb CompleteFunc

Complete function call back pointer

5.14.1.2 struct _OmniTekDmaTransactionContext::_DMACoreInfo DMACoreInfo

5.14.1.3 void* generic

5.14.1.4 struct kiocb* kiocb

5.14.1.5 struct _Resource* pChannel

Channel for this struct

5.14.1.6 struct _OmniTekRequestQueueObject QueueObject

Request Queue object

5.14.1.7 struct OmniTekRequest* request

5.14.1.8 struct _OmniTekDmaTransactionContext::_SglInfo Sgl

5.14.1.9 OmniTekTransactionState state

Transaction state

5.14.1.10 OmniTekTransactionType type

Type of transaction

5.14.1.11 union { ... } typeInfo

5.14.1.12 struct _OmniTekDmaTransactionContext::_XferInfo Xfer

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

- driver/OmniTekDma.h

5.15 _OmniTekDriver Struct Reference

```
#include <OmniTek_linux.h>
```

Data Fields

- struct list_head Extensions
- struct list_head Devices
- dev_t dev
- u32 NextMinor
- struct kmem_cache * ResourcePool
- struct pci_driver pci_driver

5.15.1 Field Documentation

5.15.1.1 dev_t dev

5.15.1.2 struct list_head Devices

5.15.1.3 struct list_head Extensions

5.15.1.4 u32 NextMinor

5.15.1.5 struct pci_driver pci_driver

5.15.1.6 struct kmem_cache* ResourcePool

Lookaside buffer for resources

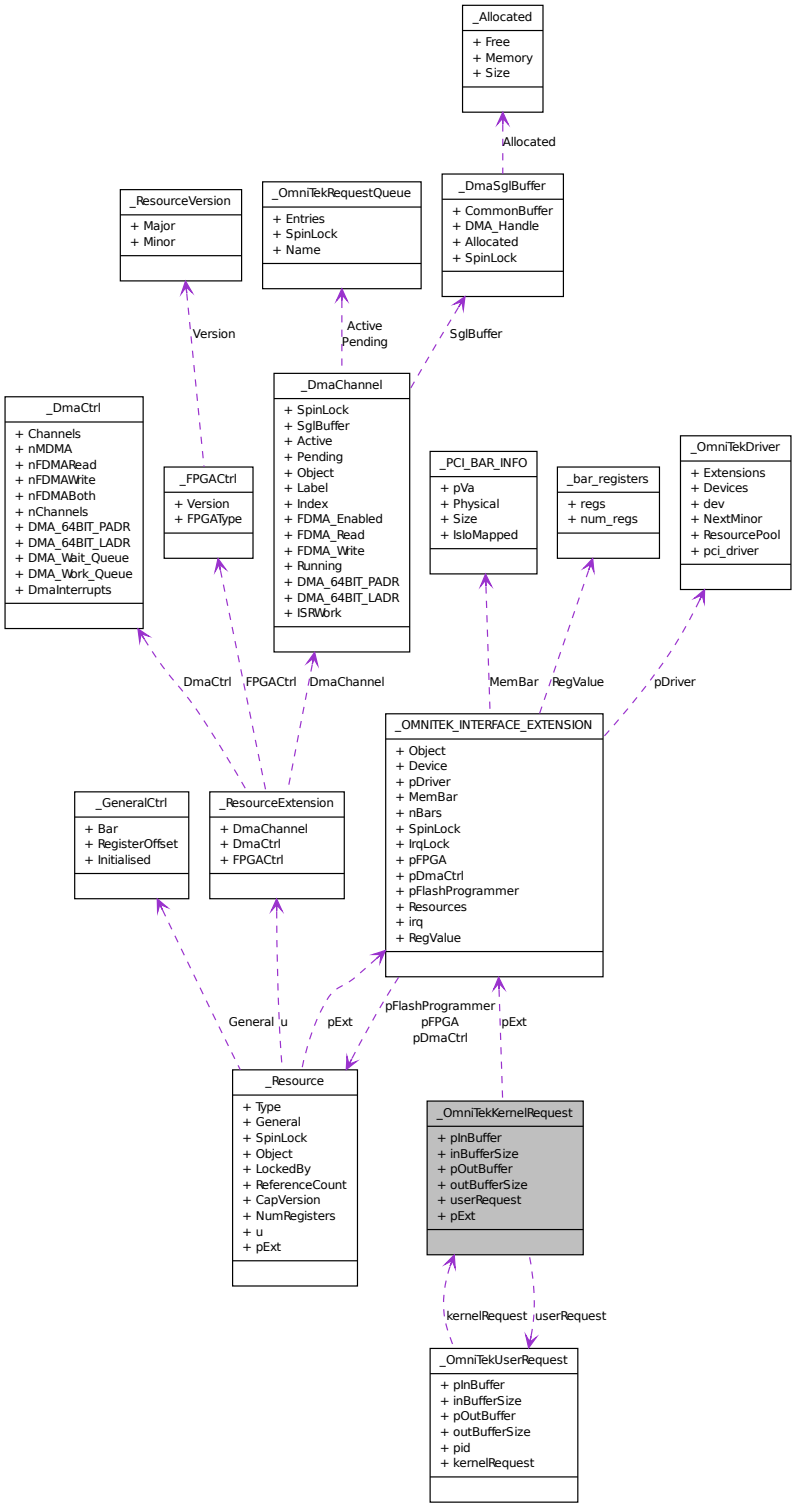
The documentation for this struct was generated from the following file:

- driver/OmniTek_linux.h

5.16 _OmniTekKernelRequest Struct Reference

```
#include <OmniTekRequest_linux.h>
```

Collaboration diagram for _OmniTekKernelRequest:



Data Fields

- u32 * pInBuffer
- u32 inBufferSize
- u32 * pOutBuffer
- u32 outBufferSize
- struct _OmniTekUserRequest * userRequest
- struct _OMNITEK_INTERFACE_EXTENSION * pExt

5.16.1 Field Documentation

5.16.1.1 u32 inBufferSize

Size of input buffer

5.16.1.2 u32 outBufferSize

Size of output buffer

5.16.1.3 struct _OMNITEK_INTERFACE_EXTENSION* pExt

Pointer to the interface extension that this request is targetting

5.16.1.4 u32* pInBuffer

Pointer to input buffer (user->kernel) in user space

5.16.1.5 u32* pOutBuffer

Pointer to output buffer (kernel->user) in user space

5.16.1.6 struct _OmniTekUserRequest* userRequest

The user request that this kernel request originated from

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

- driver/OmniTekRequest_linux.h

5.17 _OmniTekRequestQueue Struct Reference

`#include <OmniTekRequestQueue.h>`

Data Fields

- struct list_head Entries
- spinlock_t SpinLock
- char * Name

5.17.1 Field Documentation

5.17.1.1 struct list_head Entries

Queue Entries List

5.17.1.2 char* Name

Queue Name (for Debug)

5.17.1.3 spinlock_t SpinLock

Queue Spinlock

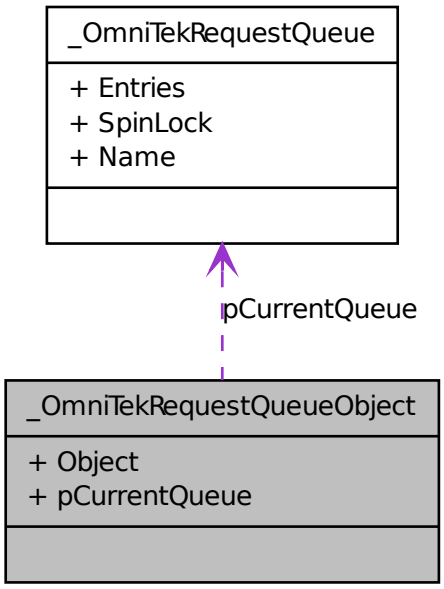
The documentation for this struct was generated from the following file:

- driver/OmniTekRequestQueue.h

5.18 _OmniTekRequestQueueObject Struct Reference

`#include <OmniTekRequestQueue.h>`

Collaboration diagram for _OmniTekRequestQueueObject:



Data Fields

- struct list_head Object
- struct _OmniTekRequestQueue * pCurrentQueue

5.18.1 Field Documentation

5.18.1.1 struct list_head Object

Object for placing in queue

5.18.1.2 struct _OmniTekRequestQueue* pCurrentQueue

Pointer to object's current queue

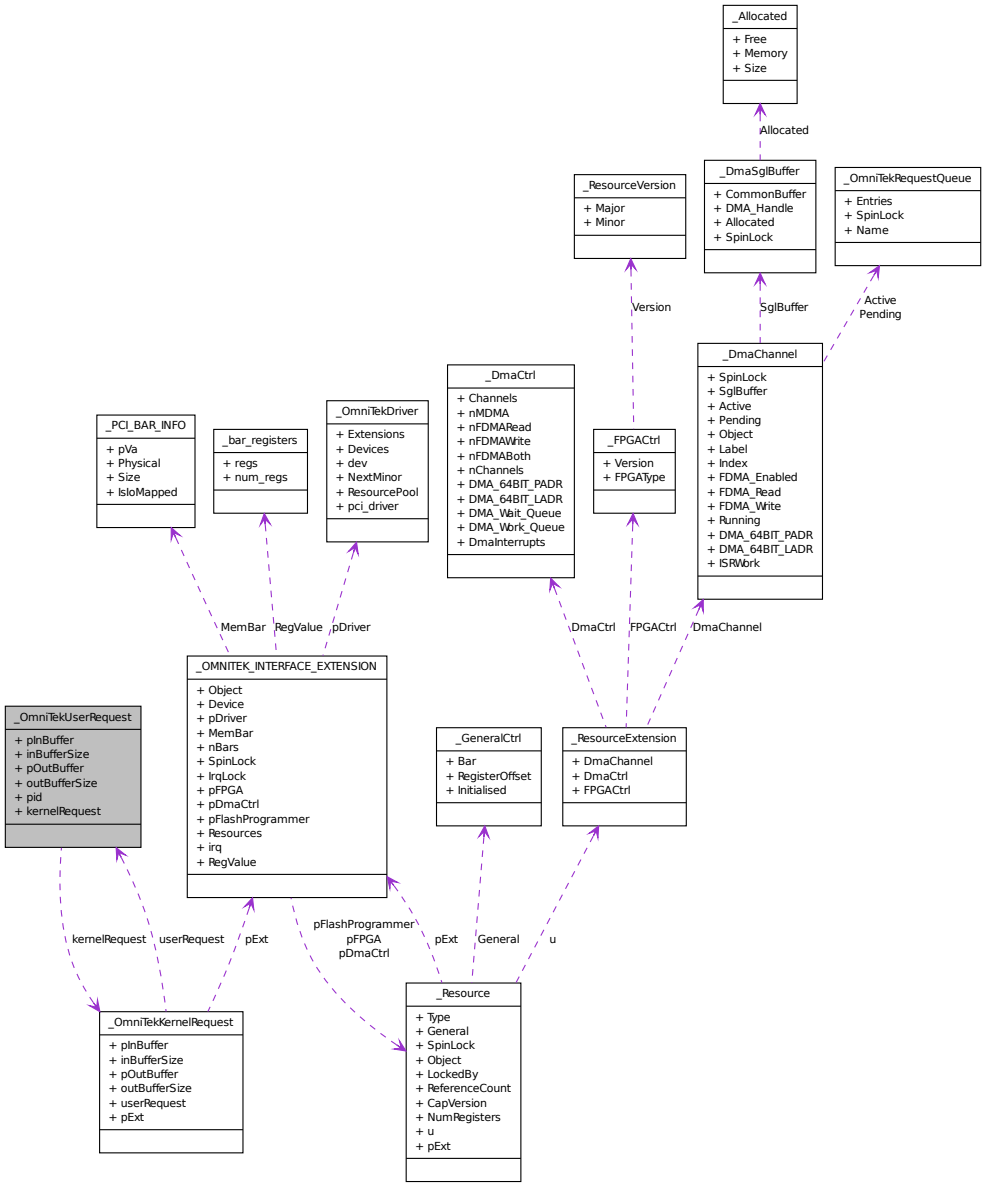
The documentation for this struct was generated from the following file:

- driver/OmniTekRequestQueue.h

5.19 _OmniTekUserRequest Struct Reference

```
#include <OmniTekRequest_linux.h>
```

Collaboration diagram for _OmniTekUserRequest:



Data Fields

- u32 * pInBuffer
- u32 inBufferSize
- u32 * pOutBuffer
- u32 outBufferSize
- pid_t pid
- struct _OmniTekKernelRequest * kernelRequest

5.19.1 Field Documentation

5.19.1.1 u32 inBufferSize

Size of input buffer

5.19.1.2 struct _OmniTekKernelRequest* kernelRequest

Associated kernel request

5.19.1.3 u32 outBufferSize

Size of output buffer

5.19.1.4 pid_t pid

Process ID that request originated from - will be signalled on completion

5.19.1.5 u32* pInBuffer

Pointer to input buffer (user->kernel) in user space

5.19.1.6 u32* pOutBuffer

Pointer to output buffer (kernel->user) in user space

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

- driver/OmniTekRequest_linux.h

5.20 _PCI_BAR_INFO Struct Reference

PCI BAR Space information.

`#include <OmniTek_linux.h>`

Data Fields

- u32 * pVa
- u32 * Physical
- u32 Size
- bool IsIoMapped

5.20.1 Detailed Description

PCI BAR Space information.

5.20.2 Field Documentation

5.20.2.1 bool IsIoMapped

Memory or I/O mapped?

5.20.2.2 u32* Physical

BAR Physical Address

5.20.2.3 u32* pVa

BAR Kernel Virtual Address

5.20.2.4 u32 Size

BAR size

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

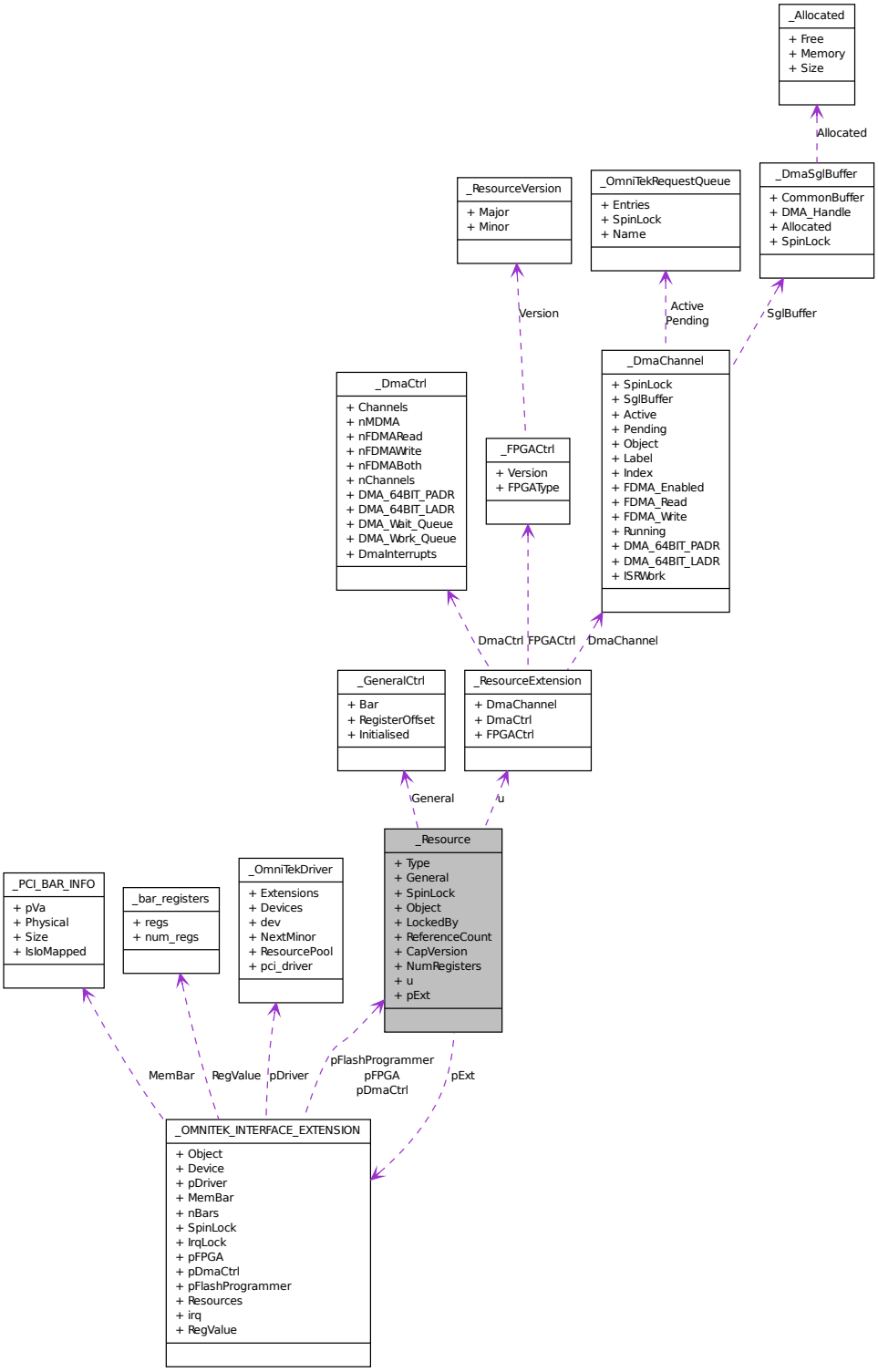
- driver/OmniTek_linux.h

5.21 _Resource Struct Reference

Generic resource Details.

`#include <OmniTekResources_linux.h>`

Collaboration diagram for _Resource:



Data Structures

- union _ResourceExtension

Data Fields

- ResourceType Type
- GeneralCtrl General
- spinlock_t SpinLock
- struct list_head Object
- u32 LockedBy
- u8 ReferenceCount
- u8 CapVersion
- u32 NumRegisters
- union _Resource::_ResourceExtension u
- struct _OMNITEK_INTERFACE_EXTENSION * pExt

5.21.1 Detailed Description

Generic resource Details.

5.21.2 Field Documentation

5.21.2.1 u8 CapVersion

5.21.2.2 GeneralCtrl General

5.21.2.3 u32 LockedBy

5.21.2.4 u32 NumRegisters

5.21.2.5 struct list_head Object

5.21.2.6 struct _OMNITEK_INTERFACE_EXTENSION* pExt

5.21.2.7 u8 ReferenceCount

5.21.2.8 spinlock_t SpinLock

5.21.2.9 ResourceType Type

5.21.2.10 union _Resource::_ResourceExtension u

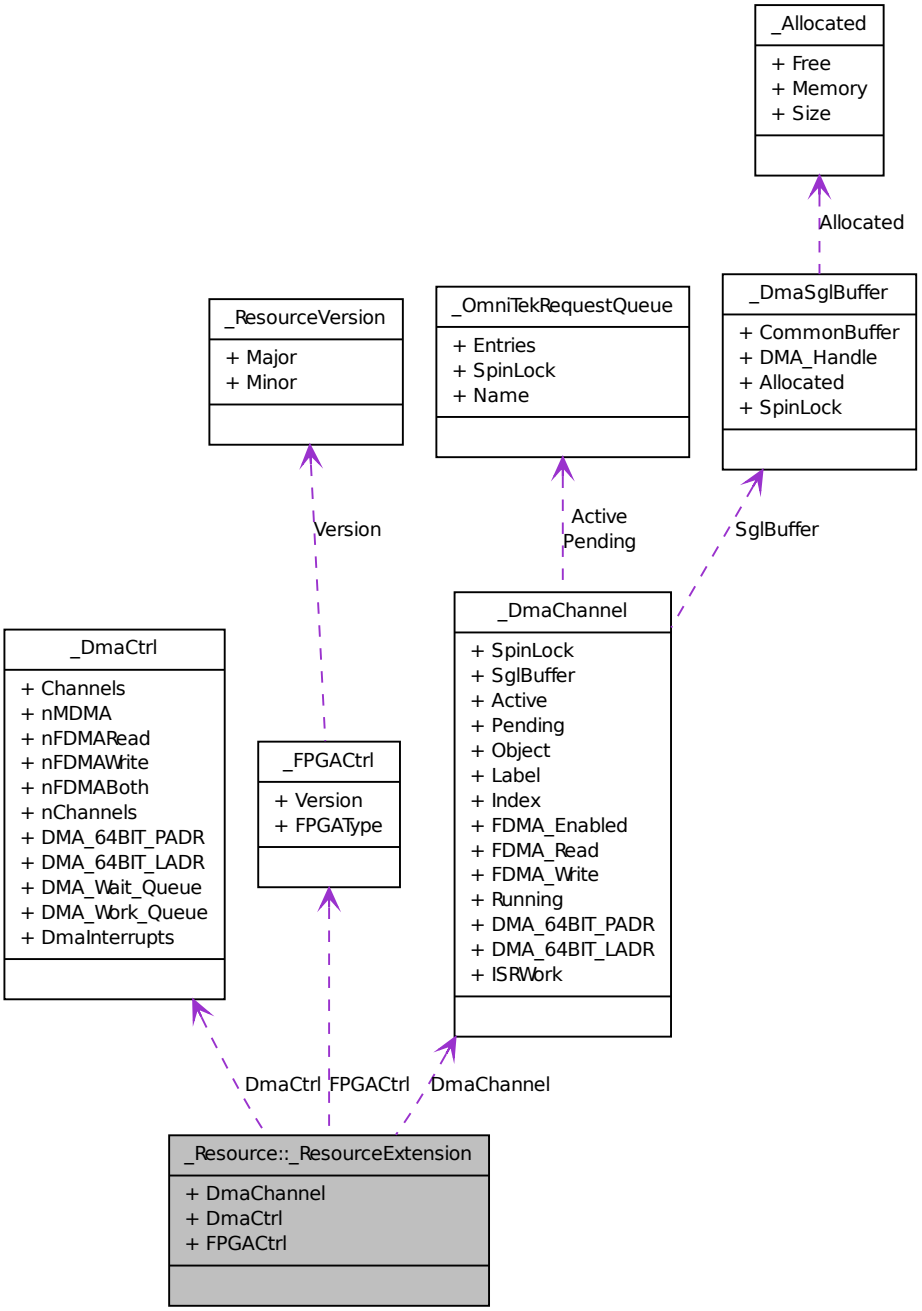
The documentation for this struct was generated from the following file:

- driver/OmniTekResources_linux.h

5.22 _Resource::_ResourceExtension Union Reference

`#include <OmniTekResources_linux.h>`

Collaboration diagram for _Resource::_ResourceExtension:



Data Fields

- DmaChannel DmaChannel
- DmaCtrl DmaCtrl
- FPGACtrl FPGACtrl

5.22.1 Detailed Description

The resource may be one of these types

5.22.2 Field Documentation

5.22.2.1 DmaChannel DmaChannel

5.22.2.2 DmaCtrl DmaCtrl

5.22.2.3 FPGACtrl FPGACtrl

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

- driver/OmniTekResources_linux.h

5.23 _ResourceVersion Struct Reference

Version details of a device resource.

`#include <OmniTekResources_linux.h>`

Data Fields

- u8 Major
- u8 Minor

5.23.1 Detailed Description

Version details of a device resource. This contains the major and minor version number of the device resource. Can be used to determine whether certain features are available

5.23.2 Field Documentation

5.23.2.1 u8 Major

Resource Major Version

5.23.2.2 u8 Minor

Resource Minor Version

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

- driver/OmniTekResources_linux.h

5.24 _OmniTekDmaTransactionContext::_SglInfo Struct Reference

`#include <OmniTekDma.h>`

Data Fields

- void * CoherentMem
- dma_addr_t DmaMem

5.24.1 Field Documentation

5.24.1.1 void* CoherentMem

< Scatter gather list information struct Pointer to DMA coherent memory for this transfer

5.24.1.2 dma_addr_t DmaMem

DMA Address for coherent memory (address for device)

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

- driver/OmniTekDma.h

5.25 _OmniTekDmaTransactionContext::_XferInfo Struct Reference

`#include <OmniTekDma.h>`

Data Fields

- char * Buffer
- loff_t LocalAddr
- size_t Size
- bool Write

5.25.1 Field Documentation

5.25.1.1 char* Buffer

< Transfer information struct User Buffer Pointer

5.25.1.2 loff_t LocalAddr

Local address (to DMA controller - e.g. device memory address)

5.25.1.3 size_t Size

Size of transfer

5.25.1.4 bool Write

Direction of transfer (if true then transfer is write to device)

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

- driver/OmniTekDma.h

5.26 OmniTek_dev Struct Reference

BAR Device data structure.

`#include <OmniTekFops_linux.h>`

5.26.1 Detailed Description

BAR Device data structure.

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

- driver/OmniTekFops_linux.h

5.27 OmniTekDriver Struct Reference

Used to keep track of the Extensions created by the driver.

`#include <OmniTek_linux.h>`

5.27.1 Detailed Description

Used to keep track of the Extensions created by the driver.

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

- driver/OmniTek_linux.h

5.28 OmniTekKernelRequest Struct Reference

`#include <OmniTekRequest_linux.h>`

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

- driver/OmniTekRequest_linux.h

5.29 OmniTekUserRequest Struct Reference

`#include <OmniTekRequest_linux.h>`

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

- driver/OmniTekRequest_linux.h

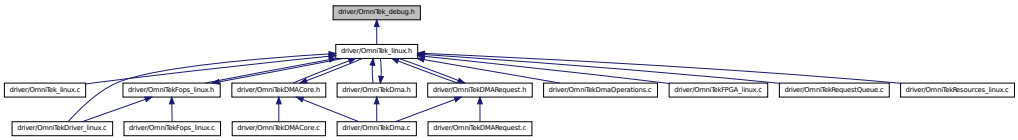
Chapter 6

File Documentation

6.1 driver/.OmniTekResource_linux.o.d File Reference

6.2 driver/OmniTek_debug.h File Reference

This graph shows which files directly or indirectly include this file:



Defines

- #define DMA 1
- #define DMA_OPS 2
- #define IRQ 4
- #define GENERAL 8
- #define RESOURCES 16
- #define FOPS 32
- #define DMA_CORE 64
- #define DMA_REQUEST 128
- #define REQUEST_QUEUE 256
- #define DMA_PAGES 512
- #define OMNITEK_DEBUG_CATEGORIES (DMA | GENERAL | RESOURCES | FOPS | DMA_-REQUEST | REQUEST_QUEUE)
- #define OmniTekDebug(category, level, format, Args...) ({if (category & OMNITEK_DEBUG_-CATEGORIES) printk(level "%s: " format ,__func__,##Args);})

6.2.1 Define Documentation

6.2.1.1 #define DMA 1

6.2.1.2 #define DMA_CORE 64

6.2.1.3 #define DMA_OPS 2

6.2.1.4 #define DMA_PAGES 512

6.2.1.5 #define DMA_REQUEST 128

6.2.1.6 #define FOPS 32

6.2.1.7 #define GENERAL 8

6.2.1.8 #define IRQ 4

6.2.1.9 #define OMNITEK_DEBUG_CATEGORIES (DMA | GENERAL | RESOURCES | FOPS | DMA_REQUEST | REQUEST_QUEUE)

6.2.1.10 #define OmniTekDebug(*category, level, format, Args...*) ({if (category & OMNITEK_DEBUG_CATEGORIES) printk(level "%s: " format ,__func__,##Args);})

6.2.1.11 #define REQUEST_QUEUE 256

6.2.1.12 #define RESOURCES 16

6.3 driver/OmniTek_Driver.mod.c File Reference

```
#include <linux/module.h>
#include <linux/vermagic.h>
#include <linux/compiler.h>
```

Functions

- MODULE_INFO (vermagic, VERMAGIC_STRING)
- struct module __this_module __attribute__ ((section(".gnu.linkonce.this_module")))
- static struct modversion_info ____versions[ ] __used __attribute__ ((section("__versions")))
- static const char __module_depends[ ] __used __attribute__ ((section(".modinfo")))
- MODULE_ALIAS ("pci:v00001AA3d00000001sv*sd*bc*sc*i*")
- MODULE_ALIAS ("pci:v00001AA3d00002002sv*sd*bc*sc*i*")
- MODULE_ALIAS ("pci:v00001AA3d00000010sv*sd*bc*sc*i*")
- MODULE_ALIAS ("pci:v000011A4d00000057sv*sd*bc*sc*i*")
- MODULE_ALIAS ("pci:v000011A4d00000058sv*sd*bc*sc*i*")
- MODULE_INFO (srcversion,"52136DEBA3F6C3A20C76FE8")

6.3.1 Function Documentation

6.3.1.1 struct module __this_module __attribute__ ((section(".gnu.linkonce.this_module"))) [read]

6.3.1.2 static const char __module_depends [ ] __used __attribute__ ((section(".modinfo"))) [static]

6.3.1.3 static struct modversion_info ____versions [ ] __used __attribute__ ((section("__versions"))) [static, read]

6.3.1.4 MODULE_ALIAS ("pci:v000011A4d00000057sv*sd*bc*sc*i*")

6.3.1.5 MODULE_ALIAS ("pci:v00001AA3d00000010sv*sd*bc*sc*i*")

6.3.1.6 MODULE_ALIAS ("pci:v000011A4d00000058sv*sd*bc*sc*i*")

6.3.1.7 MODULE_ALIAS ("pci:v00001AA3d00000001sv*sd*bc*sc*i*")

6.3.1.8 MODULE_ALIAS ("pci:v00001AA3d00002002sv*sd*bc*sc*i*")

6.3.1.9 MODULE_INFO (vermagic , VERMAGIC_STRING)

6.3.1.10 MODULE_INFO (srcversion , "52136DEBA3F6C3A20C76FE8")

6.4 driver/OmniTek_linux.c File Reference

```
#include "OmniTek_linux.h"
```

Functions

- static u32 list_count (struct list_head *head)
- void OmniTekExtInit (struct pci_dev *Device, POMNITEK_INTERFACE_EXTENSION pExt, struct _OmniTekDriver *pDriver)

  Initialise the OmniTek device extension.

- void OmniTekExtShutdown (POMNITEK_INTERFACE_EXTENSION pExt)

  Cleanup for device extension.

- int GetRegValue (POMNITEK_INTERFACE_EXTENSION pExt, u32 nBar, u32 nReg, u32 *pVal)

  Get a cached register value for the device.

- int ReadRegValue (POMNITEK_INTERFACE_EXTENSION pExt, u32 nBar, u32 nReg, u32 *pVal)

  Read a register value from the hardware.

- int WriteRegValue (POMNITEK_INTERFACE_EXTENSION pExt, u32 nBar, u32 nReg, u32 nVal)

  Write a register value to the hardware.

- int OmniTekScanHw (POMNITEK_INTERFACE_EXTENSION pExt)

  Scan hardware for Resources.

- int OmniTekGetCapList (POMNITEK_INTERFACE_EXTENSION pExt, CapabilityList *pCapList)

  Interrupt service routine.

6.4.1 Function Documentation

6.4.1.1 int GetRegValue (POMNITEK_INTERFACE_EXTENSION *pExt*, u32 *nBar*, u32 *nReg*, u32 * *pVal*)

Get a cached register value for the device.

This will return the cached register value for the device (e.g. last written) - no hardware access

Parameters

`[in]` ***pExt*** Pointer to device extension to read from

`[in]` ***nBar*** PCIE Bar number

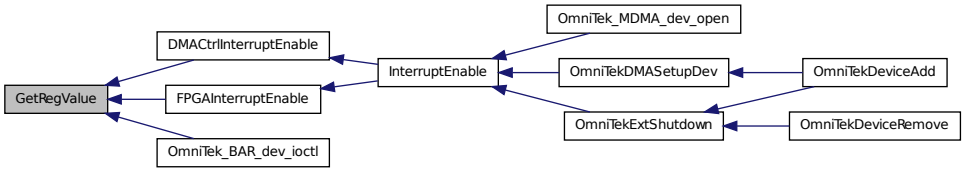
`[in]` ***nReg*** Register offset

`[in]` ***pVal*** Pointer to (32 bit) value to store result in

Returns

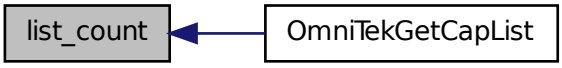
0 on success or error code

Here is the caller graph for this function:



6.4.1.2 static u32 list_count (struct list_head * *head*) [static]

Here is the caller graph for this function:



6.4.1.3 void OmniTekExtInit (struct pci_dev * *device*, POMNITEK_INTERFACE_EXTENSION *pExt*, struct _OmniTekDriver * *driver*)

Initialise the OmniTek device extension.

initial setup of the device extension data structure from the PCI device

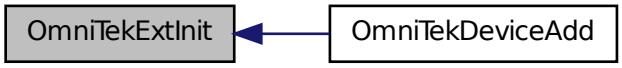
Parameters

[in] ***device*** The device that this extension is being initialised for

[in] ***pExt*** Pointer to OmniTek Extension struct to initialise

[in] ***driver*** Pointer to OmniTek Driver struct

Here is the caller graph for this function:



6.4.1.4 void OmniTekExtShutdown (POMNITEK_INTERFACE_EXTENSION *pExt*)

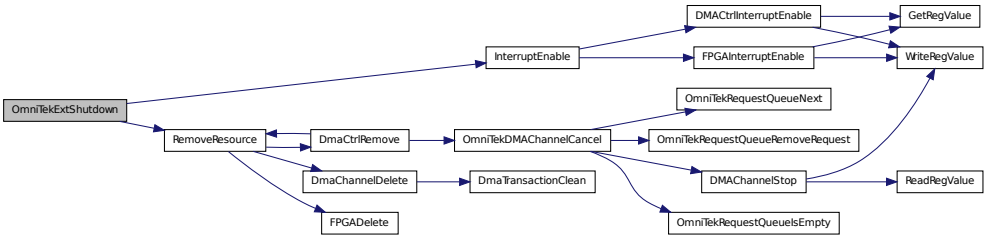
Cleanup for device extension.

Performs necessary cleanup for the device extension when it is shutdown

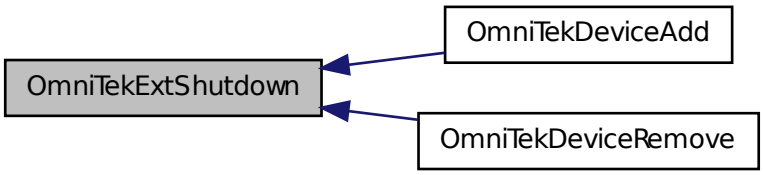
Parameters

[in] ***pExt*** Pointer to Omnitek Extension struct to shutdown

Here is the call graph for this function:



Here is the caller graph for this function:



6.4.1.5 int OmniTekGetCapList (POMNITEK_INTERFACE_EXTENSION *pExt*, CapabilityList * *pCapList*)

Interrupt service routine.

This routine determines whether an interrupt was generated by an Omnitek Device, if so the DPC routine will be called

Return a list of the available capabilities

This routine returns the available capabilities from the device

Parameters

[in] ***pExt*** Omnitek Device Extension to get capabilities from

[in] ***pCapList*** Pointer to capability list struct

Returns

0 on success or error code

Here is the call graph for this function:

6.4.1.6 int OmniTekScanHw (POMNITEK_INTERFACE_EXTENSION *pExt*)

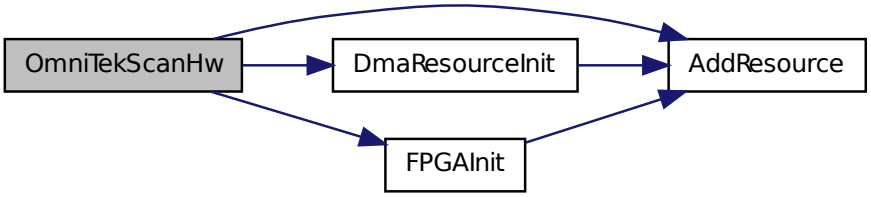
Scan hardware for Resources.

This performs a scan of the register space in each BAR. It uses the linked list structure to determine the maximum number of register in the BAR, and the resources present Resource types are initialised here (so virtual resources e.g. DMA channels are also elaborated here).

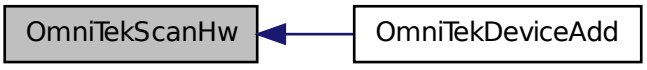
Returns

0 on success or negative error code

Here is the call graph for this function:



Here is the caller graph for this function:



6.4.1.7 int ReadRegValue (POMNITEK_INTERFACE_EXTENSION *pExt,* u32 *nBar,* u32 *nReg,* u32 * *pVal*)

Read a register value from the hardware.

This will perform a register read from the hardware. The cached register values are not updated.

Parameters

[in] ***pExt*** Pointer to device extension to read from

[in] ***nBar*** PCIE Bar number

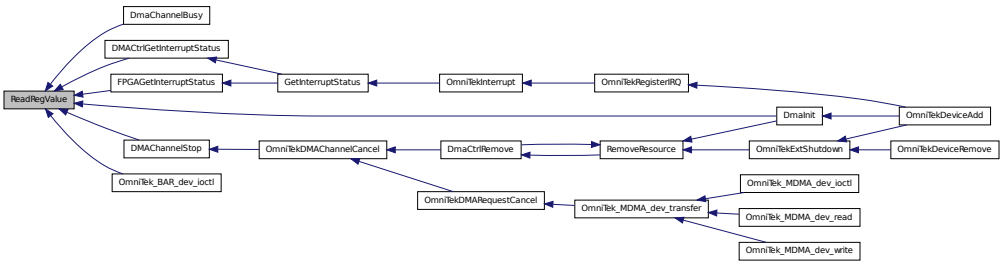
[in] ***nReg*** Register offset

[in] ***pVal*** Pointer to (32 bit) value to store result in

Returns

0 on success or error code

Here is the caller graph for this function:



6.4.1.8 int WriteRegValue (POMNITEK_INTERFACE_EXTENSION *pExt,* u32 *nBar,* u32 *nReg,* u32 *nVal*)

Write a register value to the hardware.

This performs a write to a hardware register. The cached register values are updated.

Parameters

[in] ***pExt*** Pointer to device extension to write to

[in] ***nBar*** PCIE Bar number

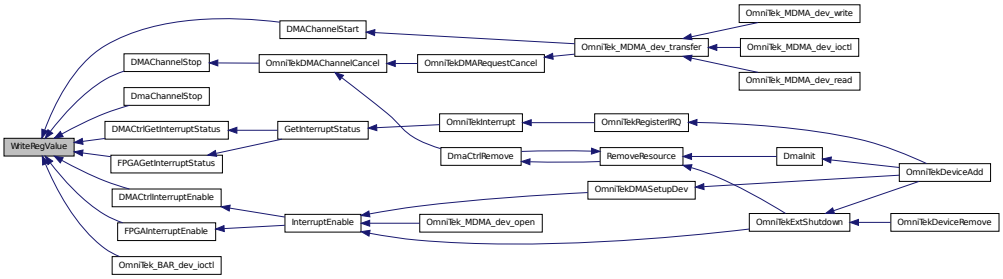
[in] ***nReg*** Register offset

[in] ***nVal*** value to write

Returns

0 on success or error code

Here is the caller graph for this function:

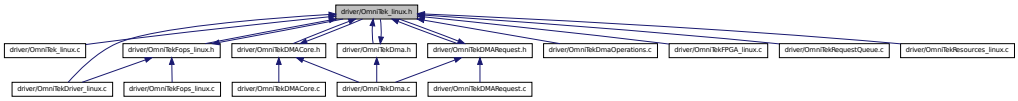


6.5 driver/OmniTek_linux.h File Reference

```
#include <linux/module.h>
#include <linux/init.h>
#include <linux/kernel.h>
#include <linux/list.h>
#include <linux/spinlock.h>
#include <linux/slab.h>
#include <linux/pci.h>
#include <linux/workqueue.h>
#include <linux/cdev.h>
#include <linux/fs.h>
#include <linux/types.h>
#include <linux/fcntl.h>
#include <linux/errno.h>
#include <linux/interrupt.h>
#include <linux/sched.h>
#include <linux/wait.h>
#include <asm/system.h>
#include <asm/uaccess.h>
#include "OmniTek_debug.h"
#include "OmniTekRequest_linux.h"
#include "../include/OmniTekTypes_linux.h"
#include "OmniTekResources_linux.h"
#include "OmniTekFPGA_linux.h"
#include "OmniTekFops_linux.h"
```

```
#include "OmniTekDma.h"
#include "OmniTekDMACore.h"
#include "OmniTekDMARequest.h"
```

This graph shows which files directly or indirectly include this file:



Data Structures

- struct _OmniTekDriver
- struct _PCI_BAR_INFO

  PCI BAR Space information.

- struct _OMNITEK_INTERFACE_EXTENSION
- struct _OMNITEK_INTERFACE_EXTENSION::_bar_registers

  details of the registers in each BAR

- struct _InterruptData

  data about interrupts

Defines

- #define STATUS_OMNITEK_MEMORY_ERROR 256
- #define STATUS_OMNITEK_RESOURCE_INVALID 257
- #define STATUS_OMNITEK_RESOURCE_LOCKED 258
- #define STATUS_OMNITEK_ILLEGAL_SESSION_ID 259
- #define STATUS_OMNITEK_RESOURCE_COMMAND_ERROR 260
- #define STATUS_INVALID_PARAMETER_1 261
- #define STATUS_INVALID_PARAMETER_2 262
- #define STATUS_INVALID_PARAMETER_3 263
- #define PCI_NUM_BARS 6
- #define MAX_NUM_MEM_BARS 4
- #define OMNITEK_INTERRUPT_MASK 0x3f0000
- #define OMNITEK_DMACTRL_INTERRUPT_MASK 0xff
- #define OMNITEK_DMA_INTERRUPT 0x2
- #define ReadHWValue(pExt, b, r) ioread32((pExt)->MemBar[(b)].pVa + (r))
- #define WriteHWValue(pExt, b, r, Val) iowrite32((Val),(pExt)->MemBar[(b)].pVa + (r))
- #define ReadHWValueByte(pExt, b, r) ioread8((u8*)((pExt)->MemBar[(b)].pVa) + (r))
- #define WriteHWValueByte(pExt, b, r, Val) iowrite8((Val) & 0xFF,((u8*)(pExt)->MemBar[(b)].pVa) + (r))

Typedefs

- typedef struct _OmniTekDriver OmniTekDriver
- typedef struct _PCI_BAR_INFO PCI_BAR_INFO
- typedef struct _OMNITEK_INTERFACE_EXTENSION OMNITEK_INTERFACE_EXTENSION
- typedef struct _OMNITEK_INTERFACE_EXTENSION * POMNITEK_INTERFACE_EXTENSION
- typedef struct _InterruptData InterruptData

Functions

- int DriverEntry (void)

  Driver entry function.

- int OmniTekEvtDeviceProbe (struct pci_dev *dev, const struct pci_device_id *id)

  Probe function.

- void OmniTekIoctl (struct inode *inode, struct file *filp, unsigned int cmd, unsigned long arg)

  IOCtl function.

- int OmniTekEvtDevicePrepareHardware (struct pci_dev *dev)

  Prepare hardware.

- int OmniTekEvtDeviceReleaseHardware (struct pci_dev *dev)

  Release hardware.

- int OmniTekEvtDeviceD0Entry (struct pci_dev *dev, int previous_state)

  Power Management - called when device enters D0 State.

- int OmniTekEvtDeviceD0EntryPostInterruptsEnabled (struct pci_dev *dev, int previous_state)

  Power Management - called after device enters D0 state and interrupts are enabled.

- int OmniTekEvtDeviceD0Exit (struct pci_dev *dev, int target_state)

  Power Management - called when the device leaves the D0 state.

- void OmniTekExtInit (struct pci_dev *device, POMNITEK_INTERFACE_EXTENSION pExt, struct _OmniTekDriver *driver)

  Initialise the OmniTek device extension.

- void OmniTekExtShutdown (POMNITEK_INTERFACE_EXTENSION pExt)

  Cleanup for device extension.

- int GetRegValue (POMNITEK_INTERFACE_EXTENSION pExt, u32 nBar, u32 nReg, u32 *pVal)

  Get a cached register value for the device.

- int ReadRegValue (POMNITEK_INTERFACE_EXTENSION pExt, u32 nBar, u32 nReg, u32 *pVal)

  Read a register value from the hardware.

- int WriteRegValue (POMNITEK_INTERFACE_EXTENSION pExt, u32 nBar, u32 nReg, u32 nVal)

  Write a register value to the hardware.

- int OmniTekScanHw (POMNITEK_INTERFACE_EXTENSION pExt)

  Scan hardware for Resources.

- int OmniTekGetCapList (POMNITEK_INTERFACE_EXTENSION pExt, CapabilityList *pCapList)

  Interrupt service routine.

- struct _OmniTekDriver * GetOmniTekDriver (struct pci_driver *driver)

  Get the OmniTekDriver struct from the pci_driver.

6.5.1 Define Documentation

6.5.1.1 #define MAX_NUM_MEM_BARS 4

6.5.1.2 #define OMNITEK_DMA_INTERRUPT 0x2

6.5.1.3 #define OMNITEK_DMACTRL_INTERRUPT_MASK 0xff

6.5.1.4 #define OMNITEK_INTERRUPT_MASK 0x3f0000

6.5.1.5 #define PCI_NUM_BARS 6

6.5.1.6 #define ReadHWValue(*pExt, b, r*) ioread32((pExt)->MemBar[(b)].pVa + (r))

6.5.1.7 #define ReadHWValueByte(*pExt, b, r*) ioread8((u8*)((pExt)->MemBar[(b)].pVa) + (r))

6.5.1.8 #define STATUS_INVALID_PARAMETER_1 261

6.5.1.9 #define STATUS_INVALID_PARAMETER_2 262

6.5.1.10 #define STATUS_INVALID_PARAMETER_3 263

6.5.1.11 #define STATUS_OMNITEK_ILLEGAL_SESSION_ID 259

6.5.1.12 #define STATUS_OMNITEK_MEMORY_ERROR 256

6.5.1.13 #define STATUS_OMNITEK_RESOURCE_COMMAND_ERROR 260

6.5.1.14 #define STATUS_OMNITEK_RESOURCE_INVALID 257

6.5.1.15 #define STATUS_OMNITEK_RESOURCE_LOCKED 258

6.5.1.16 #define WriteHWValue(*pExt, b, r, Val*) iowrite32((Val),(pExt)->MemBar[(b)].pVa + (r))

6.5.1.17 #define WriteHWValueByte(*pExt, b, r, Val*) iowrite8((Val) & 0xFF,((u8*)(pExt)->MemBar[(b)].pVa) + (r))

6.5.2 Typedef Documentation

6.5.2.1 typedef struct _InterruptData InterruptData

6.5.2.2 typedef struct _OMNITEK_INTERFACE_EXTENSION OMNITEK_INTERFACE_EXTENSION

6.5.2.3 typedef struct _OmniTekDriver OmniTekDriver

6.5.2.4 typedef struct _PCI_BAR_INFO PCI_BAR_INFO

6.5.2.5 typedef struct _OMNITEK_INTERFACE_EXTENSION * POMNITEK_INTERFACE_EXTENSION

6.5.3 Function Documentation

6.5.3.1 int DriverEntry (void)

Driver entry function.

Returns

0 on success, error code otherwise

6.5.3.2 struct _OmniTekDriver* GetOmniTekDriver (struct pci_driver * *driver*) [read]

Get the OmniTekDriver struct from the pci_driver.

Get the main driver data structure back from a pci_driver struct

Parameters

[in] ***driver*** Pointer to the pci_driver structure that is contained within the OmniTekDriver struct

Returns

A pointer to the OmniTekDriver struct

Get the OmniTekDriver struct from the pci_driver.

This returns a pointer to the OmniTek driver data structure given the pci_driver data structure

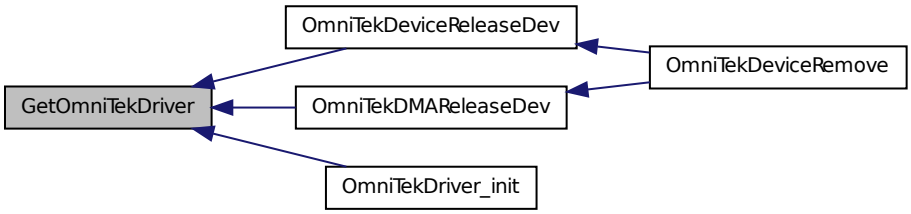
Parameters

[in] ***driver*** pointer to pci_driver struct

Returns

Pointer to OmniTek Driver data structure

Here is the caller graph for this function:



6.5.3.3 int GetRegValue (POMNITEK_INTERFACE_EXTENSION *pExt*, u32 *nBar*, u32 *nReg*, u32 * *pVal*)

Get a cached register value for the device.

This will return the cached register value for the device (e.g. last written) - no hardware access

Parameters

[in] ***pExt*** Pointer to device extension to read from

[in] ***nBar*** PCIE Bar number

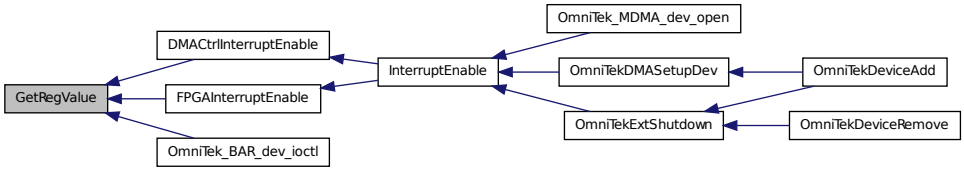
[in] ***nReg*** Register offset

[in] ***pVal*** Pointer to (32 bit) value to store result in

Returns

0 on success or error code

Here is the caller graph for this function:



6.5.3.4 int OmniTekEvtDeviceD0Entry (struct pci_dev * *dev*, int *previous_state*)

Power Management - called when device enters D0 State.

Called when the device enters the D0 power state. Doesn't do much currently

Parameters

[in] ***dev*** The device whose power state is changing

[in] ***previous_state*** Previous power state (so that the direction of the transition can be determined)

Returns

0 on success, error code otherwise

6.5.3.5 int OmniTekEvtDeviceD0EntryPostInterruptsEnabled (struct pci_dev * *dev*, int *previous_state*)

Power Management - called after device enters D0 state and interrupts are enabled.

Called when interrupts are enabled after the device enters the D0 power state. Doesn't do much currently...

Parameters

[in] ***dev*** The device whose power state is changing

[in] ***previous_state*** Previous power state (so that the direction of the transition can be determined)

Returns

0 on success, error code otherwise

6.5.3.6 int OmniTekEvtDeviceD0Exit (struct pci_dev * *dev*, int *target_state*)

Power Management - called when the device leaves the D0 state.

Called when the device transitions from D0 state

Parameters

[in] ***dev*** The device whose power state is changing

[in] ***target_state*** The device state that is being transitioned to

Returns

0 on success, error code otherwise

6.5.3.7 int OmniTekEvtDevicePrepareHardware (struct pci_dev * *dev*)

Prepare hardware.

This function is called to perform initial hardware setup. It first scans for and sets up the PCIE BAR data structures in the driver, it then scans the BARs for hardware resources, and initialises them. Finally it initialises the DMA portions of the driver and hardware.

Parameters

[in] ***dev*** Omnitek PCI Device to prepare

Returns

0 on success, error code otherwise

6.5.3.8 int OmniTekEvtDeviceProbe (struct pci_dev * *dev*, const struct pci_device_id * *id*)

Probe function.

This function is called for each PCI device with a matching ID

Returns

0 on success, error code otherwise

6.5.3.9 int OmniTekEvtDeviceReleaseHardware (struct pci_dev * *dev*)

Release hardware.

Shutdown and release hardware resources, clear up any memory mappings etc.

Parameters

[in] ***dev*** Omnitek PCI Device that is being released

Returns

0 on success, error code otherwise

6.5.3.10 void OmniTekExtInit (struct pci_dev * *device*, POMNITEK_INTERFACE_EXTENSION *pExt*, struct _OmniTekDriver * *driver*)

Initialise the OmniTek device extension.

initial setup of the device extension data structure from the PCI device

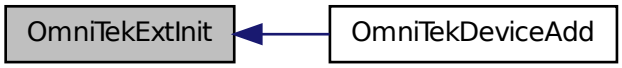
Parameters

[in] ***device*** The device that this extension is being initialised for

[in] ***pExt*** Pointer to OmniTek Extension struct to initialise

[in] ***driver*** Pointer to OmniTek Driver struct

Here is the caller graph for this function:



6.5.3.11 void OmniTekExtShutdown (POMNITEK_INTERFACE_EXTENSION *pExt*)

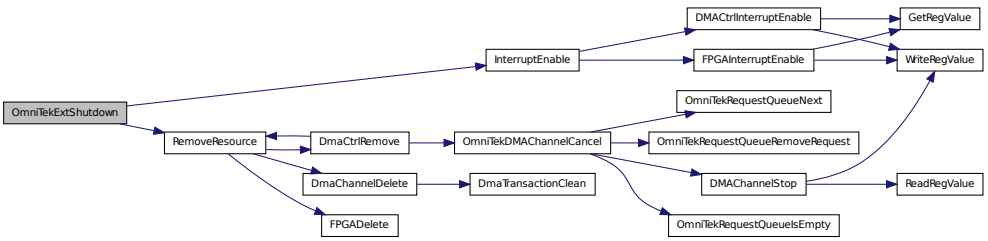
Cleanup for device extension.

Performs necessary cleanup for the device extension when it is shutdown

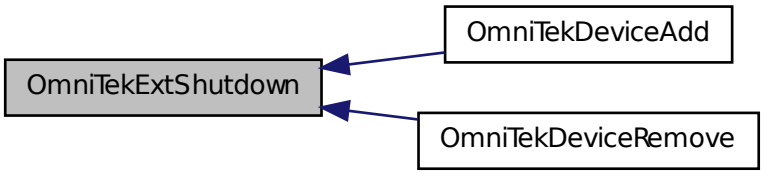
Parameters

[in] ***pExt*** Pointer to Omnitek Extension struct to shutdown

Here is the call graph for this function:



Here is the caller graph for this function:



6.5.3.12 int OmniTekGetCapList (POMNITEK_INTERFACE_EXTENSION *pExt,* CapabilityList * *pCapList*)

Interrupt service routine.

This routine determines whether an interrupt was generated by an Omnitek Device, if so the DPC routine will be called

Return a list of the available capabilities

This routine returns the available capabilities from the device

Parameters

[in] ***pExt*** Omnitek Device Extension to get capabilities from

[in] ***pCapList*** Pointer to capability list struct

Returns

0 on success or error code

Here is the call graph for this function:

6.5.3.13 void OmniTekIoctl (struct inode * *inode,* struct file * *filp,* unsigned int *cmd,* unsigned long *arg*)

IOCtl function.

This handles IOCtl calls to the driver. For all calls to the driver we presume that the arg value is NULL or a pointer to an OmnitekRequest struct. Copying data between user and kernel space will be performed automatically using the pointers in this struct and the call type.

6.5.3.14 int OmniTekScanHw (POMNITEK_INTERFACE_EXTENSION *pExt*)

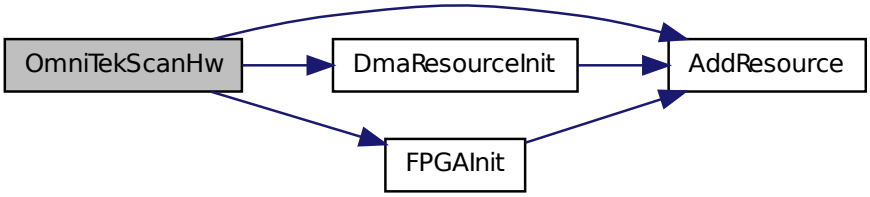
Scan hardware for Resources.

This performs a scan of the register space in each BAR. It uses the linked list structure to determine the maximum number of register in the BAR, and the resources present Resource types are initialised here (so virtual resources e.g. DMA channels are also elaborated here).

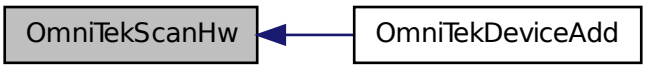
Returns

0 on success or negative error code

Here is the call graph for this function:



Here is the caller graph for this function:



6.5.3.15 int ReadRegValue (POMNITEK_INTERFACE_EXTENSION *pExt*, u32 *nBar*, u32 *nReg*, u32 * *pVal*)

Read a register value from the hardware.

This will perform a register read from the hardware. The cached register values are not updated.

Parameters

[in] ***pExt*** Pointer to device extension to read from

[in] ***nBar*** PCIE Bar number

[in] ***nReg*** Register offset

[in] ***pVal*** Pointer to (32 bit) value to store result in

Returns

0 on success or error code

6.5.3.16 int WriteRegValue (POMNITEK_INTERFACE_EXTENSION *pExt,* u32 *nBar,* u32 *nReg,* u32 *nVal*)

Write a register value to the hardware.

This performs a write to a hardware register. The cached register values are updated.

Parameters

[in] ***pExt*** Pointer to device extension to write to

[in] ***nBar*** PCIE Bar number

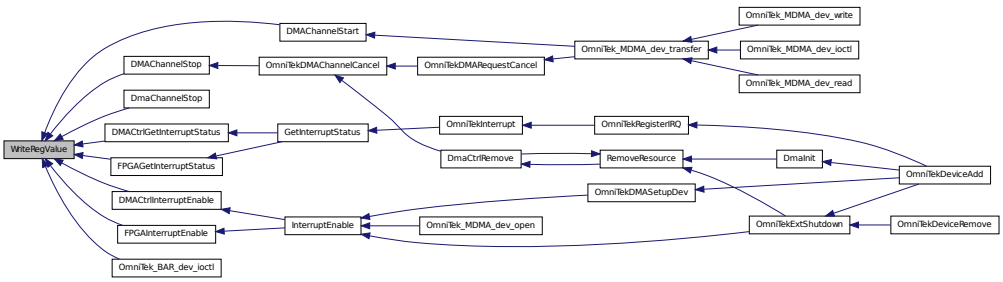
[in] ***nReg*** Register offset

[in] ***nVal*** value to write

Returns

0 on success or error code

Here is the caller graph for this function:



6.6 driver/OmniTek_MainPage.h File Reference

6.7 driver/OmniTekDma.c File Reference

```
#include "OmniTekDma.h"
#include "OmniTekDmaOperations.h"
```

```
#include "OmniTekRequestQueue.h"
#include "OmniTekDMACore.h"
#include "OmniTekDMARequest.h"
```

Defines

- #define TRANSACTION_WAIT_MSECS 2000

Functions

- void DmaChannelStop (PResource pChannel)

  Stop a DMA Channel (this is a hard abort).

- bool DmaChannelBusy (PResource pChannel)

  Has this channel got any outstanding transactions?

- static int DmaChannelInit (PResource pResource, u8 Index)
- void DmaResourceInit (Resource *pResource)

  Initialise DMA channel resource.

- int DmaInit (POMNITEK_INTERFACE_EXTENSION pExt)
- void DmaCtrlRemove (PResource pCtrl)

  Delete a DMA Controller resource.

- static void DmaTransactionClean (struct _OmniTekDmaTransactionContext *pContext)
- void DmaChannelDelete (PResource pChannel)

  Delete a DMA Channel resource.

- void OmniTekDMASetupDev (struct _OMNITEK_INTERFACE_EXTENSION *pExt)

  Setup DMA Device nodes.

- void OmniTekDMAReleaseDev (POMNITEK_INTERFACE_EXTENSION pExt)
- int OmniTek_MDMA_dev_ioctl (struct inode *inode, struct file *filp, unsigned int cmd, unsigned long arg)

  IOCtl call to MDMA Device.

- int OmniTek_MDMA_dev_open (struct inode *inode, struct file *filp)

  Device open call to MDMA Device.

- int OmniTek_MDMA_dev_release (struct inode *inode, struct file *filp)

  Device release to MDMA Device.

- static DECLARE_WAIT_QUEUE_HEAD (dma_wq)
- void OmniTek_MDMA_dev_complete (struct _OmniTekDmaTransactionContext *pTransaction, int status)
- ssize_t OmniTek_MDMA_dev_transfer (struct file *filp, const char __user *buf, size_t count, loff_t *f_pos, bool write)

  Perform MDMA Transfer.

- ssize_t OmniTek_MDMA_dev_read (struct file *filp, char __user *buf, size_t count, loff_t *f_pos)

  Read from MDMA Device.

- ssize_t OmniTek_MDMA_dev_write (struct file *filp, const char __user *buf, size_t count, loff_t *f_pos)

  Write to MDMA Device.

Variables

- static struct _OmniTek_dev OmniTekMDMADev
- struct file_operations OmniTek_MDMA_dev_fops

6.7.1 Define Documentation

6.7.1.1 #define TRANSACTION_WAIT_MSECS 2000

6.7.2 Function Documentation

6.7.2.1 static DECLARE_WAIT_QUEUE_HEAD (dma_wq) [static]

6.7.2.2 bool DmaChannelBusy (PResource *pChannel*)

Has this channel got any outstanding transactions?

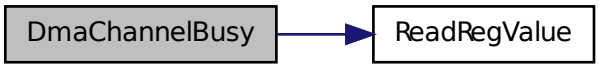
Parameters

`[in]` ***pChannel*** pointer to channel resource

Returns

True if there are transactions in the channel's active queue

Here is the call graph for this function:



6.7.2.3 void DmaChannelDelete (PResource *pChannel*)

Delete a DMA Channel resource.

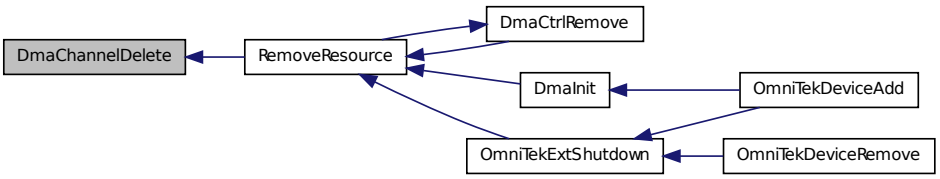
Parameters

[in] ***pChannel*** pointer to channel resource to delete

Will stop channel and cancel any outstanding requests

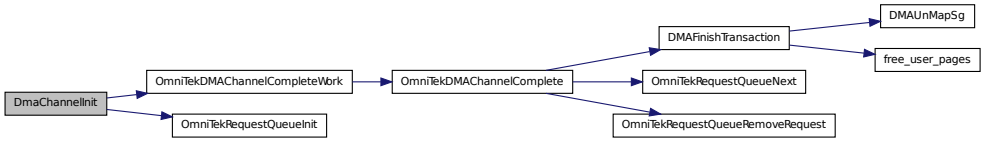
Here is the call graph for this function:

Here is the caller graph for this function:



6.7.2.4 static int DmaChannelInit (PResource *pResource*, u8 *Index*) [static]

Here is the call graph for this function:



Here is the caller graph for this function:

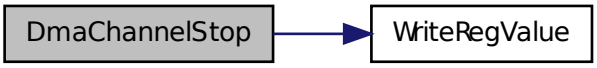
6.7.2.5 void DmaChannelStop (PResource *pChannel*)

Stop a DMA Channel (this is a hard abort).

Parameters

`[in]` ***pChannel*** pointer to channel resource to stop.

Here is the call graph for this function:



6.7.2.6 void DmaCtrlRemove (PResource *pCtrl*)

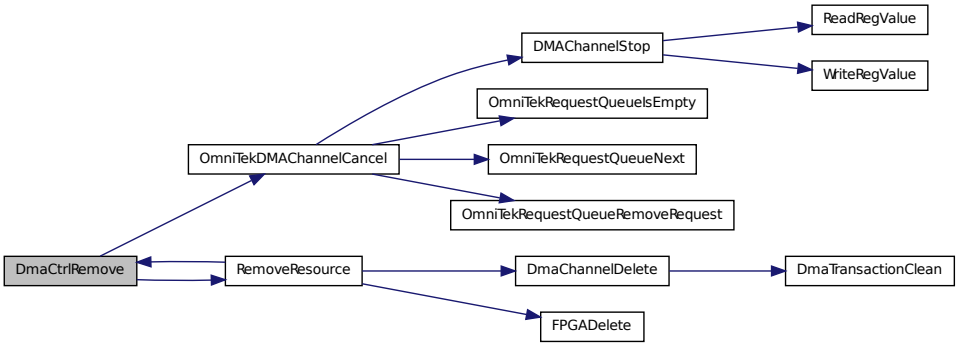
Delete a DMA Controller resource.

Parameters

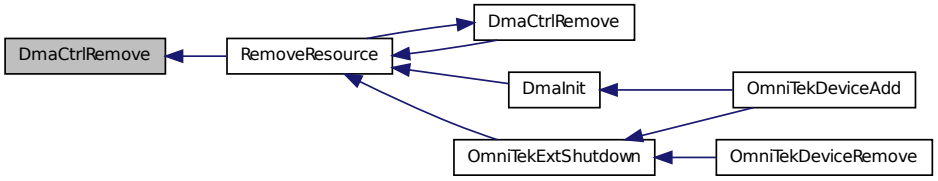
`[in]` ***pCtrl*** pointer to controller resource to remove

Will delete all channels associated with this controller

Here is the call graph for this function:

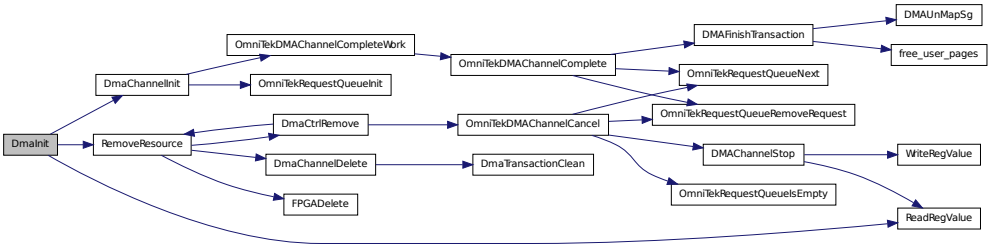


Here is the caller graph for this function:

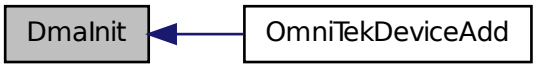


6.7.2.7 int DmaInit (POMNITEK_INTERFACE_EXTENSION *pExt*)

Here is the call graph for this function:



Here is the caller graph for this function:



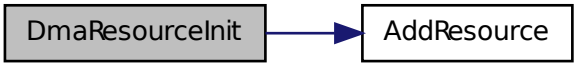
6.7.2.8 void DmaResourceInit (Resource * *pResource*)

Initialise DMA channel resource.

Parameters

`[in]` ***pResource*** pointer to DMA channel resource to initialize

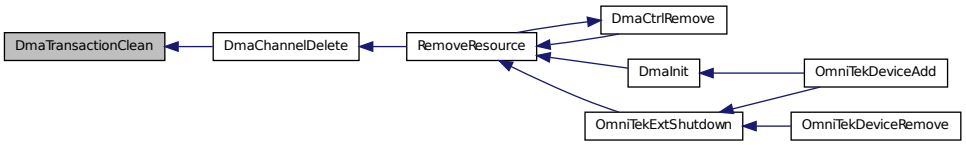
Here is the call graph for this function:



Here is the caller graph for this function:

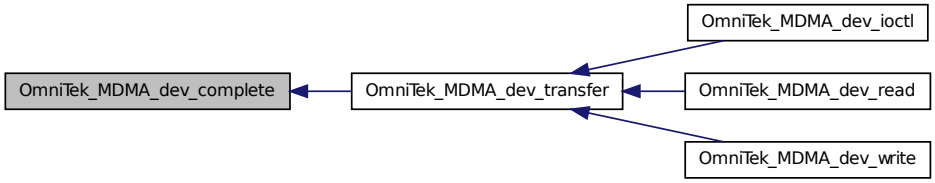
6.7.2.9 static void DmaTransactionClean (struct _OmniTekDmaTransactionContext * *pContext*) [static]

Here is the caller graph for this function:



6.7.2.10 void OmniTek_MDMA_dev_complete (struct _OmniTekDmaTransactionContext * *pTransaction,* int *status*)

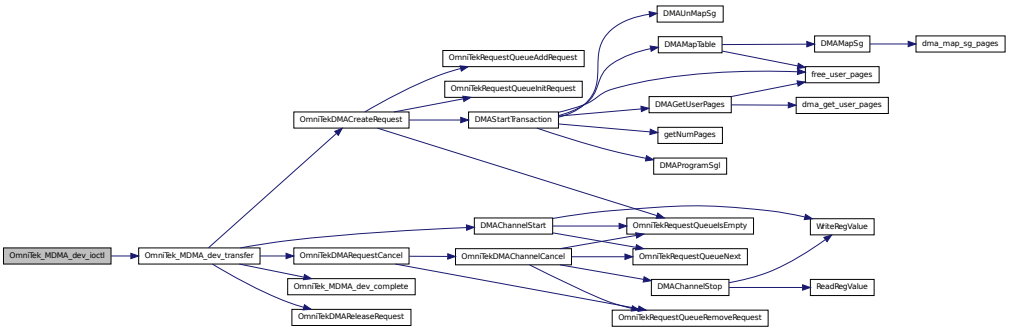
Here is the caller graph for this function:



6.7.2.11 int OmniTek_MDMA_dev_ioctl (struct inode * *inode,* struct file * *filp,* unsigned int *cmd,* unsigned long *arg*)

IOCtl call to MDMA Device.

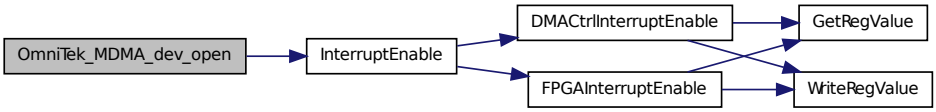
Here is the call graph for this function:



6.7.2.12 int OmniTek_MDMA_dev_open (struct inode * *inode*, struct file * *filp*)

Device open call to MDMA Device.

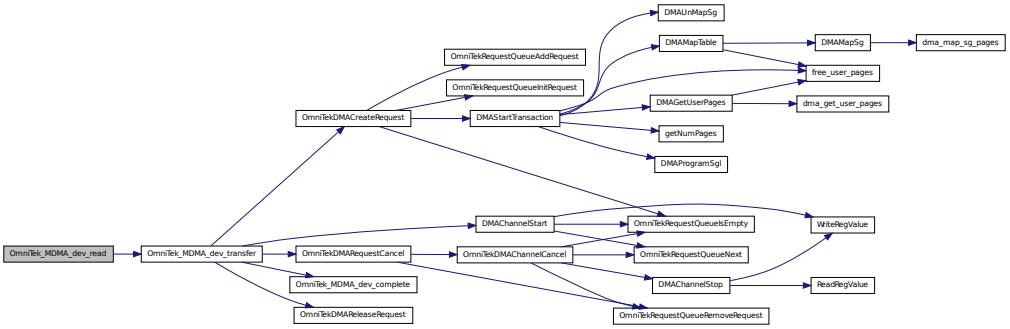
Here is the call graph for this function:



6.7.2.13 ssize_t OmniTek_MDMA_dev_read (struct file * *filp*, char __user * *buf*, size_t *count*, loff_t * *f_pos*)

Read from MDMA Device.

Here is the call graph for this function:



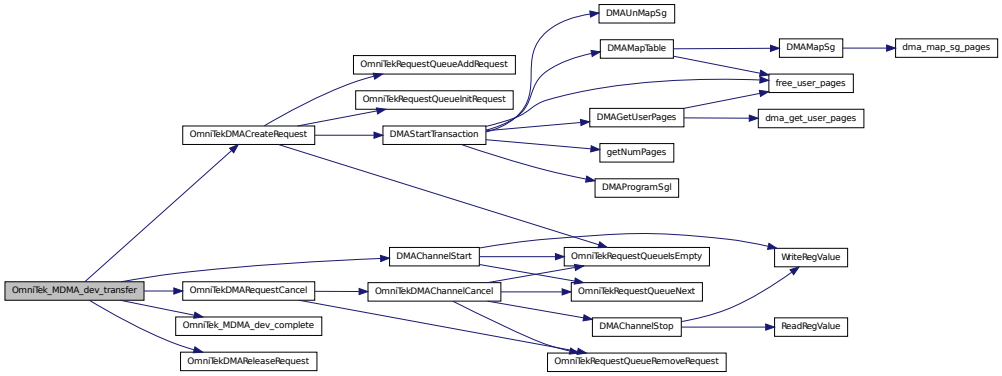
6.7.2.14 int OmniTek_MDMA_dev_release (struct inode * *inode*, struct file * *filp*)

Device release to MDMA Device.

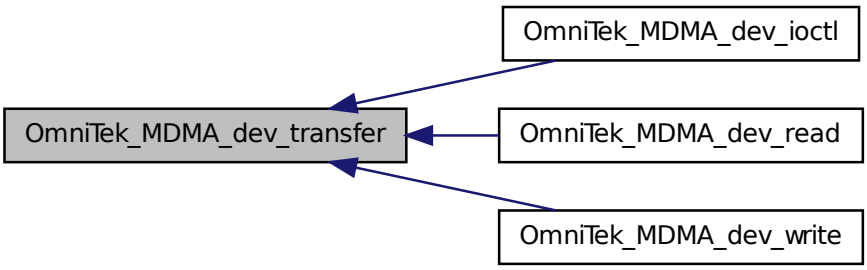
6.7.2.15 ssize_t OmniTek_MDMA_dev_transfer (struct file * *filp*, const char __user * *buf*, size_t *count*, loff_t * *f_pos*, bool *write*)

Perform MDMA Transfer.

Here is the call graph for this function:



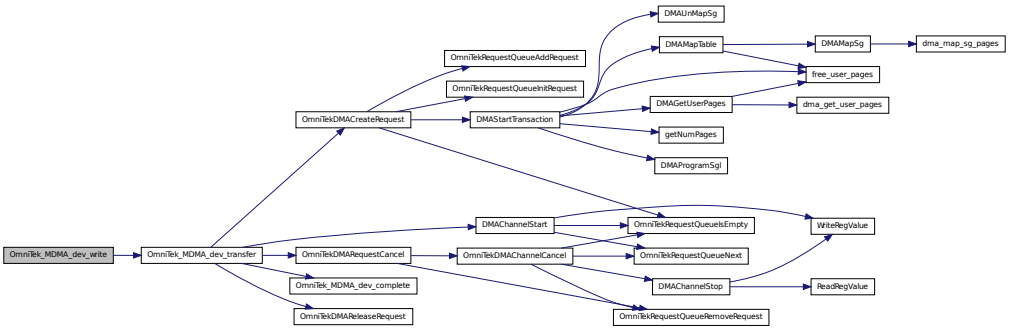
Here is the caller graph for this function:



6.7.2.16 ssize_t OmniTek_MDMA_dev_write (struct file * *filp*, const char __user * *buf*, size_t *count*, loff_t * *f_pos*)

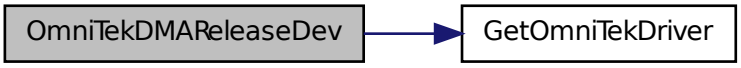
Write to MDMA Device.

Here is the call graph for this function:



6.7.2.17 void OmniTekDMAReleaseDev (POMNITEK_INTERFACE_EXTENSION *pExt*)

Here is the call graph for this function:



Here is the caller graph for this function:

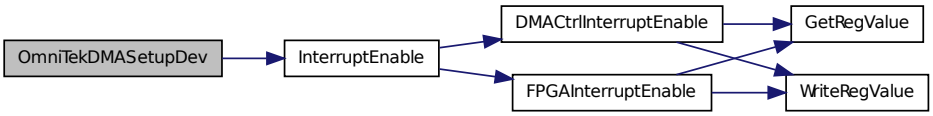
6.7.2.18 void OmniTekDMASetupDev (struct _OMNITEK_INTERFACE_EXTENSION * *pExt*)

Setup DMA Device nodes.

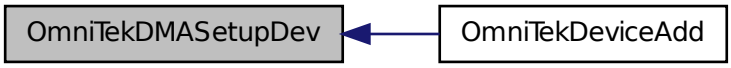
Parameters

`[in]` ***pExt*** Pointer to device extension to set up DMA devices for

Here is the call graph for this function:



Here is the caller graph for this function:



6.7.3 Variable Documentation

6.7.3.1 struct file_operations OmniTek_MDMA_dev_fops

Initial value:

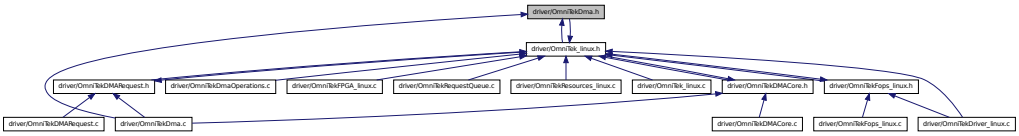
```
 {
  .owner = THIS_MODULE,
  .read = OmniTek_MDMA_dev_read,
  .write = OmniTek_MDMA_dev_write,
  .ioctl = OmniTek_MDMA_dev_ioctl,
  .open = OmniTek_MDMA_dev_open,
  .release = OmniTek_MDMA_dev_release
}
```

6.7.3.2 struct _OmniTek_dev OmniTekMDMADev [static]

6.8 driver/OmniTekDma.h File Reference

```
#include "OmniTek_linux.h"
#include "OmniTekRequestQueue.h"
```

This graph shows which files directly or indirectly include this file:



Data Structures

- struct _OmniTekDmaTransactionContext
- struct _OmniTekDmaTransactionContext::_XferInfo
- struct _OmniTekDmaTransactionContext::_SglInfo
- struct _OmniTekDmaTransactionContext::_DMACoreInfo

Defines

- #define DMA_CHANNEL_OFFSET 16
- #define DMA_CTRL_CAP_HEADER 0
- #define DMA_CTRL_CAP_REG 1
- #define DMA_CTRL_INTERRUPT_STATUS 2
- #define DMA_CHANNEL_PADR 0
- #define DMA_CHANNEL_PADR_HIGH 1
- #define DMA_CHANNEL_LADR 2
- #define DMA_CHANNEL_LADR_HIGH 3
- #define DMA_CHANNEL_DPR 4
- #define DMA_CHANNEL_DPR_HIGH 5
- #define DMA_CHANNEL_SIZE 6
- #define DMA_CHANNEL_SIZE_HIGH 7
- #define DMA_CHANNEL_CSR 8
- #define DMA_CHANNEL_BYTES_XFER 9
- #define NUM_REGS_PER_DMA_CHANNEL 10
- #define DMA_DPR_BIT_END_OF_CHAIN 0x02
- #define DMA_DPR_BIT_INTERRUPT 0x04
- #define DMA_DPR_BIT_DIRECTION_TO_PC 0x08
- #define SGL_ITEM_SIZE 32
- #define DMA_SGL_SIZE(Transfer) (((Transfer) / 0x1000) ∗ SGL_ITEM_SIZE)
- #define DMA_CHANNEL_FDMA 0x80000000
- #define DMA_CHANNEL_WRITE 0x40000000
- #define DMA_CHANNEL_READ 0x20000000
- #define DMA_FDMA_CHANNEL(type, w, x)
- #define DMA_MDMA_CHANNEL(x) ((x) & 0x7f)
- #define DMA_FDMA_TYPE(x) (((x) >> 8) & 0xffff)
- #define DMA_CHANNEL(x) ((x) & 0xff)

Typedefs

- typedef struct _OmniTekDmaTransactionContext OmniTekDmaTransactionContext
- typedef struct _OmniTekDmaTransactionContext ∗ POmniTekDmaTransactionContext

Functions

- void OmniTekDMASetupDev (struct _OMNITEK_INTERFACE_EXTENSION *pExt)

  Setup DMA Device nodes.

- void OmniTekDMAReleaseDev (struct _OMNITEK_INTERFACE_EXTENSION *pExt)

  Release DMA Device nodes.

- void DmaChannelStop (PResource pChannel)

  Stop a DMA Channel (this is a hard abort).

- bool DmaChannelBusy (PResource pChannel)

  Has this channel got any outstanding transactions?

- void DmaResourceInit (Resource *pResource)

  Initialise DMA channel resource.

- int DmaInit (struct _OMNITEK_INTERFACE_EXTENSION *pExt)

  Initialize DMA Controller.

- void DmaChannelDelete (PResource pChannel)

  Delete a DMA Channel resource.

- void DmaCtrlRemove (PResource pCtrl)

  Delete a DMA Controller resource.

- int OmniTek_MDMA_dev_ioctl (struct inode *, struct file *, unsigned int, unsigned long)

  IOCtl call to MDMA Device.

- int OmniTek_MDMA_dev_open (struct inode *, struct file *)

  Device open call to MDMA Device.

- int OmniTek_MDMA_dev_release (struct inode *, struct file *)

  Device release to MDMA Device.

- ssize_t OmniTek_MDMA_dev_read (struct file *filp, char __user *buf, size_t count, loff_t *f_pos)

  Read from MDMA Device.

- ssize_t OmniTek_MDMA_dev_write (struct file *filp, const char __user *buf, size_t count, loff_t *f_pos)

  Write to MDMA Device.

- ssize_t OmniTek_MDMA_dev_transfer (struct file *filp, const char __user *buf, size_t count, loff_t *f_pos, bool write)

  Perform MDMA Transfer.

6.8.1 Define Documentation

6.8.1.1 #define DMA_CHANNEL(*x*) ((x) & 0xff)

6.8.1.2 #define DMA_CHANNEL_BYTES_XFER 9

6.8.1.3 #define DMA_CHANNEL_CSR 8

6.8.1.4 #define DMA_CHANNEL_DPR 4

6.8.1.5 #define DMA_CHANNEL_DPR_HIGH 5

6.8.1.6 #define DMA_CHANNEL_FDMA 0x80000000

6.8.1.7 #define DMA_CHANNEL_LADR 2

6.8.1.8 #define DMA_CHANNEL_LADR_HIGH 3

6.8.1.9 #define DMA_CHANNEL_OFFSET 16

6.8.1.10 #define DMA_CHANNEL_PADR 0

6.8.1.11 #define DMA_CHANNEL_PADR_HIGH 1

6.8.1.12 #define DMA_CHANNEL_READ 0x20000000

6.8.1.13 #define DMA_CHANNEL_SIZE 6

6.8.1.14 #define DMA_CHANNEL_SIZE_HIGH 7

6.8.1.15 #define DMA_CHANNEL_WRITE 0x40000000

6.8.1.16 #define DMA_CTRL_CAP_HEADER 0

6.8.1.17 #define DMA_CTRL_CAP_REG 1

6.8.1.18 #define DMA_CTRL_INTERRUPT_STATUS 2

6.8.1.19 #define DMA_DPR_BIT_DIRECTION_TO_PC 0x08

6.8.1.20 #define DMA_DPR_BIT_END_OF_CHAIN 0x02

6.8.1.21 #define DMA_DPR_BIT_INTERRUPT 0x04

6.8.1.22 #define DMA_FDMA_CHANNEL(*type, w, x*)

Value:

```
(DMA_CHANNEL_FDMA | \

      (w) ? DMA_CHANNEL_WRITE : DMA_CHANNEL_READ) | \

      ((type) & 0xffff) << 8) | ((x) & 0xff))
```

6.8.1.23 #define DMA_FDMA_TYPE(*x*) (((x) >> 8) & 0xffff)

6.8.1.24 #define DMA_MDMA_CHANNEL(*x*) ((x) & 0x7f)

6.8.1.25 #define DMA_SGL_SIZE(*Transfer*) (((Transfer) / 0x1000) * SGL_ITEM_SIZE)

6.8.1.26 #define NUM_REGS_PER_DMA_CHANNEL 10

6.8.1.27 #define SGL_ITEM_SIZE 32

6.8.2 Typedef Documentation

6.8.2.1 typedef struct _OmniTekDmaTransactionContext OmniTekDmaTransactionContext

6.8.2.2 typedef struct _OmniTekDmaTransactionContext * POmniTekDmaTransactionContext

6.8.3 Function Documentation

6.8.3.1 bool DmaChannelBusy (PResource *pChannel*)

Has this channel got any outstanding transactions?

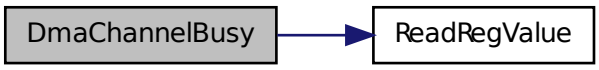
Parameters

`[in]` ***pChannel*** pointer to channel resource

Returns

True if there are transactions in the channel's active queue

Here is the call graph for this function:



6.8.3.2 void DmaChannelDelete (PResource *pChannel*)

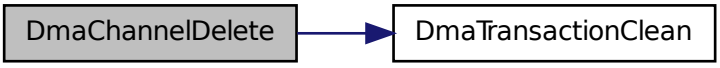
Delete a DMA Channel resource.

Parameters

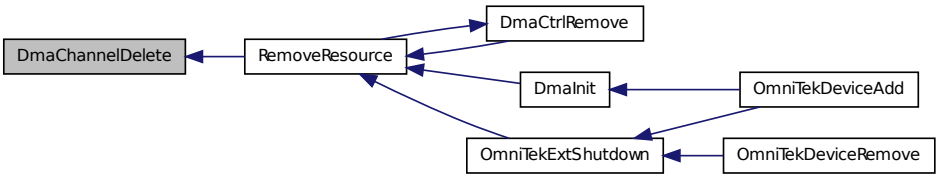
`[in]` ***pChannel*** pointer to channel resource to delete

Will stop channel and cancel any outstanding requests

Here is the call graph for this function:



Here is the caller graph for this function:



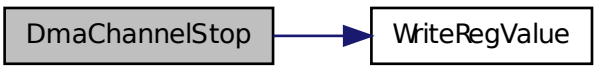
6.8.3.3 void DmaChannelStop (PResource *pChannel*)

Stop a DMA Channel (this is a hard abort).

Parameters

`[in]` ***pChannel*** pointer to channel resource to stop.

Here is the call graph for this function:



6.8.3.4 void DmaCtrlRemove (PResource *pCtrl*)

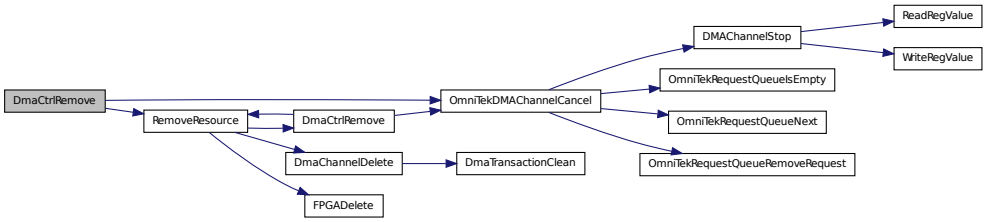
Delete a DMA Controller resource.

Parameters

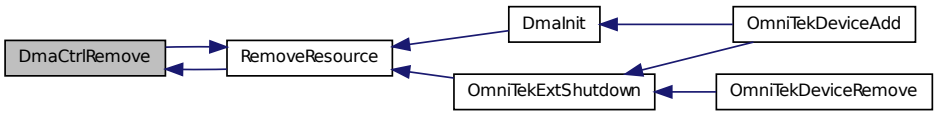
[in] ***pCtrl*** pointer to controller resource to remove

Will delete all channels associated with this controller

Here is the call graph for this function:



Here is the caller graph for this function:



6.8.3.5 int DmaInit (struct _OMNITEK_INTERFACE_EXTENSION * *pExt*)

Initialize DMA Controller.

Parameters

[in] ***pExt*** pointer to device extension to initialize

Returns

success if DMA Controller initialized

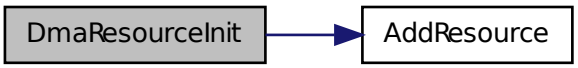
6.8.3.6 void DmaResourceInit (Resource * *pResource*)

Initialise DMA channel resource.

Parameters

[in] ***pResource*** pointer to DMA channel resource to initialize

Here is the call graph for this function:

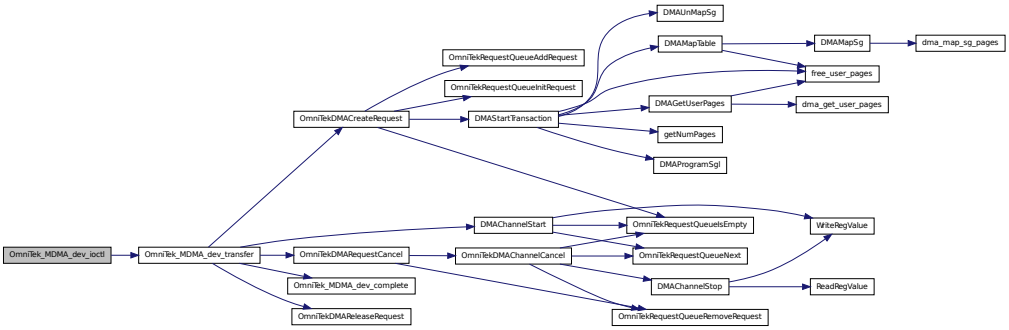


Here is the caller graph for this function:

6.8.3.7 int OmniTek_MDMA_dev_ioctl (struct inode *, struct file *, unsigned *int*, unsigned *long*)

IOCtl call to MDMA Device.

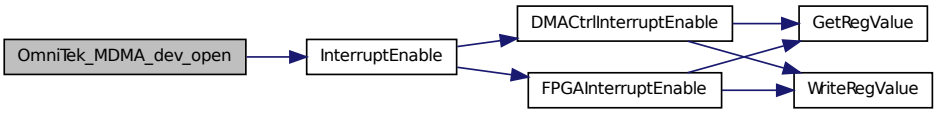
Here is the call graph for this function:



6.8.3.8 int OmniTek_MDMA_dev_open (struct inode *, struct file *)

Device open call to MDMA Device.

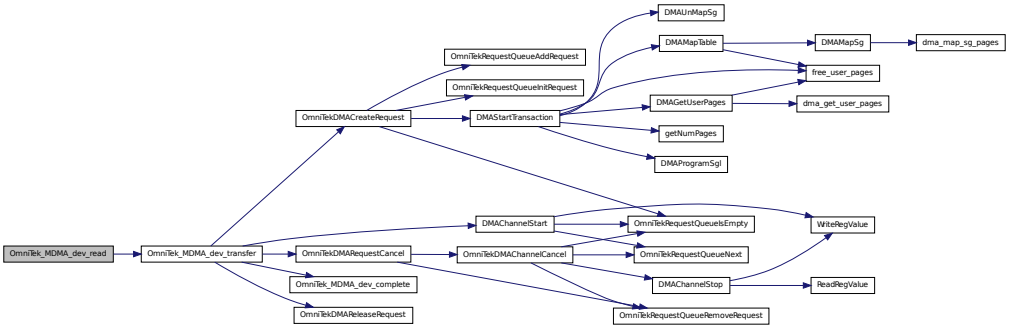
Here is the call graph for this function:



6.8.3.9 ssize_t OmniTek_MDMA_dev_read (struct file * *filp*, char __user * *buf*, size_t *count*, loff_t * *f_pos*)

Read from MDMA Device.

Here is the call graph for this function:



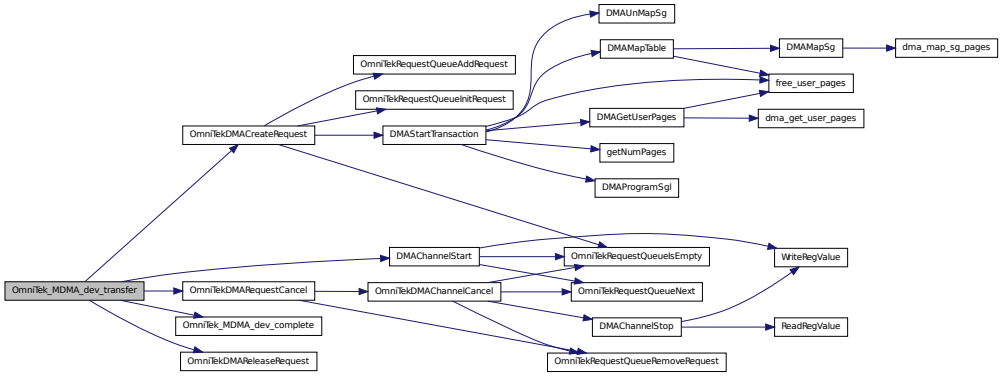
6.8.3.10 int OmniTek_MDMA_dev_release (struct inode * , struct file *)

Device release to MDMA Device.

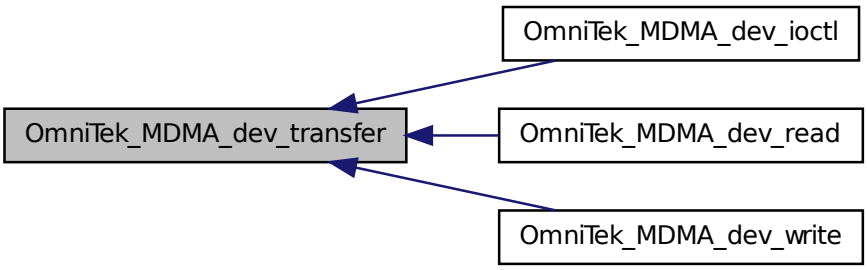
6.8.3.11 ssize_t OmniTek_MDMA_dev_transfer (struct file * *filp*, const char __user * *buf*, size_t *count*, loff_t * *f_pos*, bool *write*)

Perform MDMA Transfer.

Here is the call graph for this function:



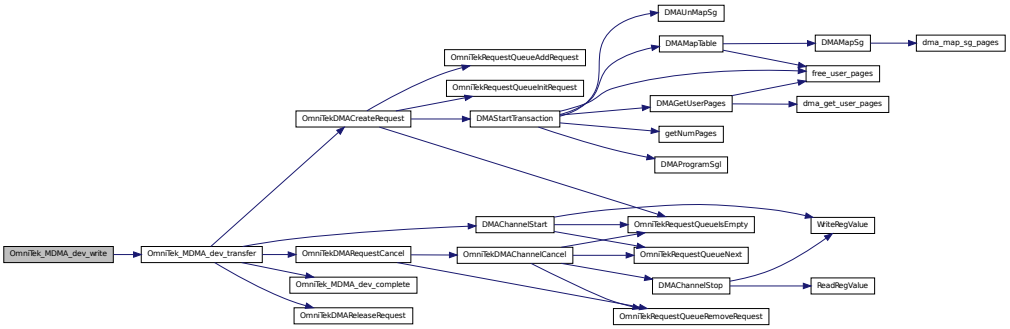
Here is the caller graph for this function:



6.8.3.12 ssize_t OmniTek_MDMA_dev_write (struct file * *filp*, const char __user * *buf*, size_t *count*, loff_t * *f_pos*)

Write to MDMA Device.

Here is the call graph for this function:



6.8.3.13 void OmniTekDMAReleaseDev (struct _OMNITEK_INTERFACE_EXTENSION * *pExt*)

Release DMA Device nodes.

Parameters

[in] ***pExt*** Pointer to device extension to release DMA devices for

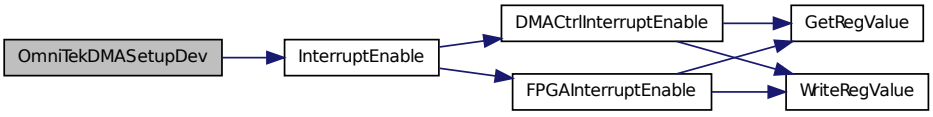
6.8.3.14 void OmniTekDMASetupDev (struct _OMNITEK_INTERFACE_EXTENSION * *pExt*)

Setup DMA Device nodes.

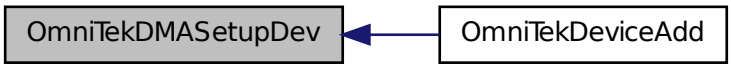
Parameters

[in] ***pExt*** Pointer to device extension to set up DMA devices for

Here is the call graph for this function:



Here is the caller graph for this function:



6.9 driver/OmniTekDMACore.c File Reference

```
#include "OmniTekDMACore.h"
#include "OmniTekDmaOperations.h"
#include <linux/pagemap.h>
#include <linux/dma-mapping.h>
```

Functions

- static void getNumPages (unsigned long buffer, size_t size, off_t *offset, unsigned long *num_pages, unsigned long *first, unsigned long *last)
- static void free_user_pages (struct _OmniTekDmaTransactionContext *pTransaction)
- static int DMAMapSg (struct device *dev, struct sg_table *sgt, struct page **pages, size_t size, int num_pages, off_t pageOffset, bool write)
- int DMAGetUserPages (struct _OmniTekDmaTransactionContext *pTransaction)
- int DMAMapTable (struct _OmniTekDmaTransactionContext *pTransaction)
- void DMAUnMapSg (struct _OmniTekDmaTransactionContext *pTransaction)
- int DMAProgramSgl (struct _OmniTekDmaTransactionContext *pTransaction)
- int DMAStartTransaction (struct _OmniTekDmaTransactionContext *pTransaction)

  Start this transaction.

- void DMAFinishTransaction (struct _OmniTekDmaTransactionContext *pTransaction)

  Finish this transaction.

- int DMAChannelStart (PResource pChannel)

  Start this DMA Channel.

- void DMAChannelStop (PResource pChannel)

  Stop this DMA Channel.

6.9.1 Function Documentation

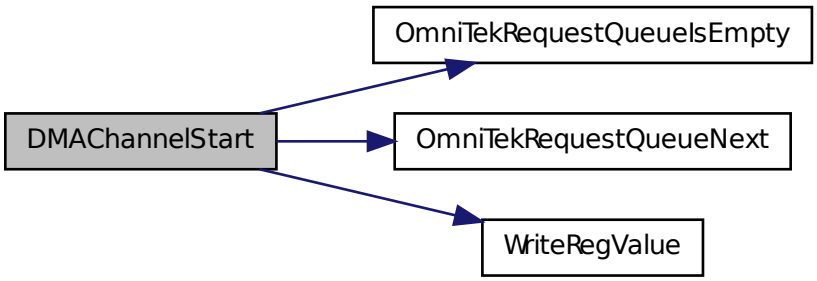
6.9.1.1 int DMAChannelStart (PResource *pChannel*)

Start this DMA Channel.

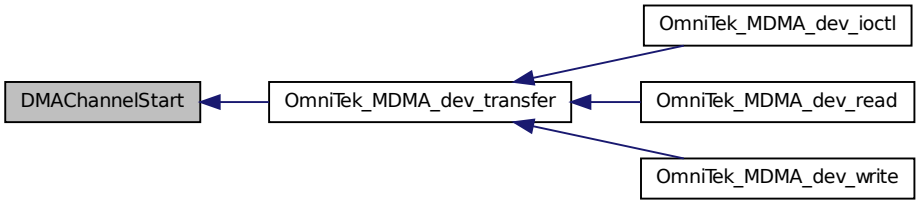
Parameters

[in] ***pChannel*** Pointer to DMA channel resource to start

Here is the call graph for this function:



Here is the caller graph for this function:



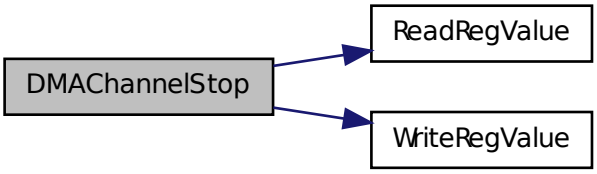
6.9.1.2 void DMAChannelStop (PResource *pChannel*)

Stop this DMA Channel.

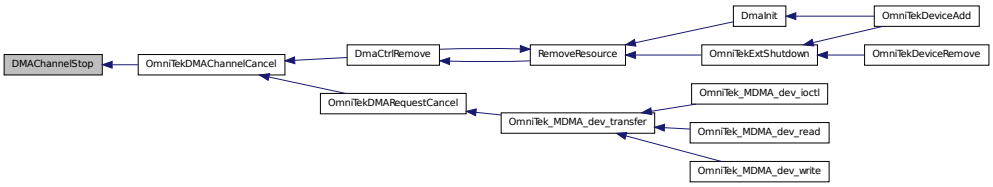
Parameters

[in] ***pChannel*** Pointer to DMA channel resource to stop

Here is the call graph for this function:



Here is the caller graph for this function:



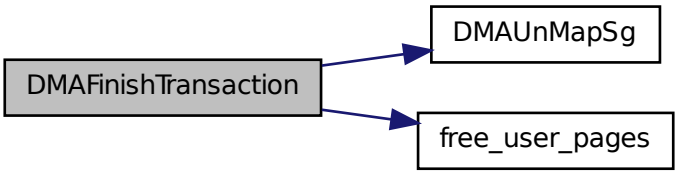
6.9.1.3 void DMAFinishTransaction (struct _OmniTekDmaTransactionContext * *pTransaction*)

Finish this transaction.

Parameters

`[in]` ***pTransaction*** Pointer to transaction to finish

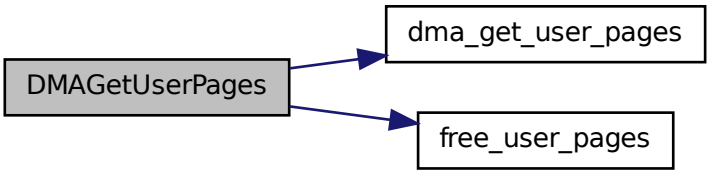
Here is the call graph for this function:



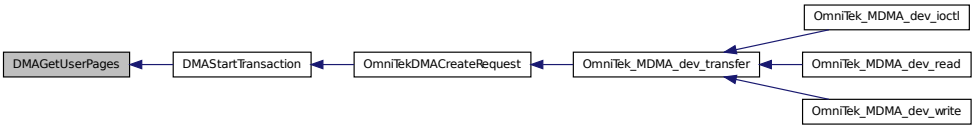
Here is the caller graph for this function:

6.9.1.4 int DMAGetUserPages (struct _OmniTekDmaTransactionContext * *pTransaction*)

Here is the call graph for this function:

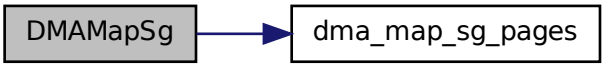


Here is the caller graph for this function:



6.9.1.5 static int DMAMapSg (struct device * *dev*, struct sg_table * *sgt*, struct page ** *pages*, size_t *size*, int *num_pages*, off_t *pageOffset*, bool *write*) [static]

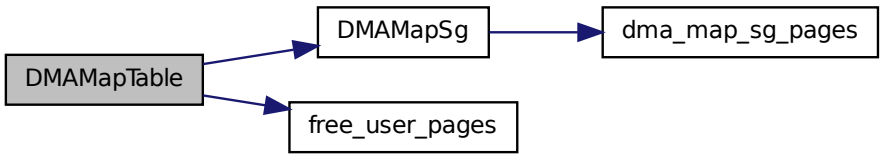
Here is the call graph for this function:



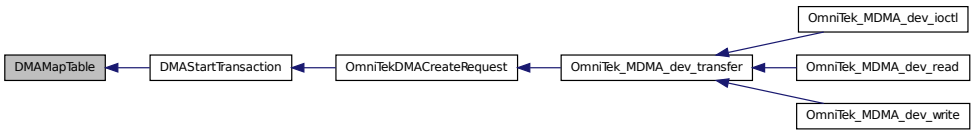
Here is the caller graph for this function:

6.9.1.6 int DMAMapTable (struct _OmniTekDmaTransactionContext * *pTransaction*)

Here is the call graph for this function:

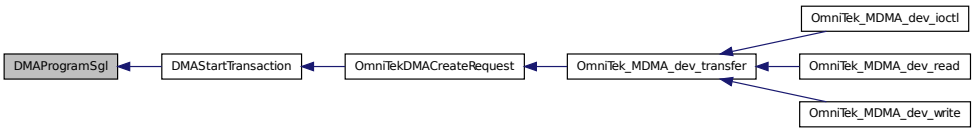


Here is the caller graph for this function:



6.9.1.7 int DMAProgramSgl (struct _OmniTekDmaTransactionContext * *pTransaction*)

Here is the caller graph for this function:



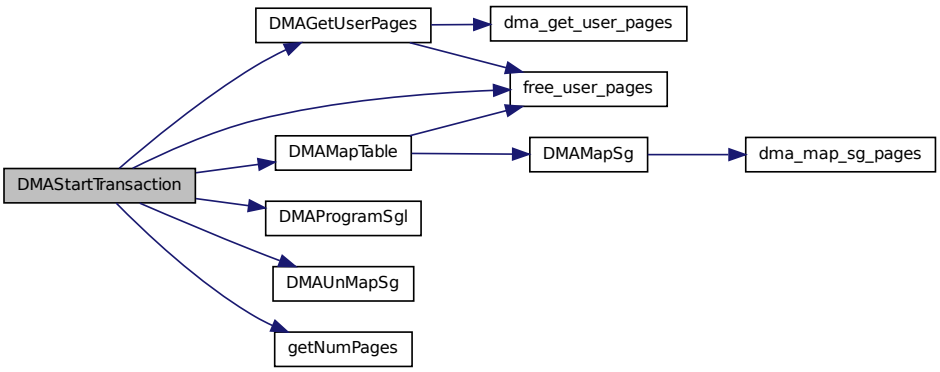
6.9.1.8 int DMAStartTransaction (struct _OmniTekDmaTransactionContext * *pTransaction*)

Start this transaction.

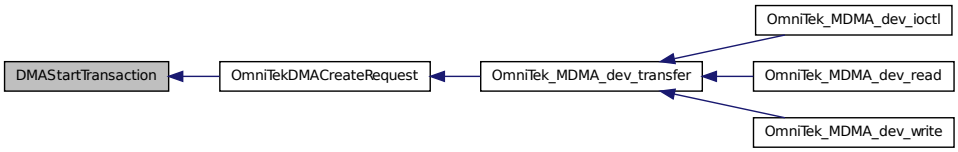
Parameters

[in] ***pTransaction*** Pointer to transaction to start

Here is the call graph for this function:

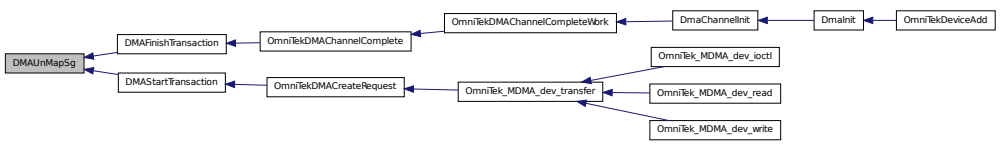


Here is the caller graph for this function:



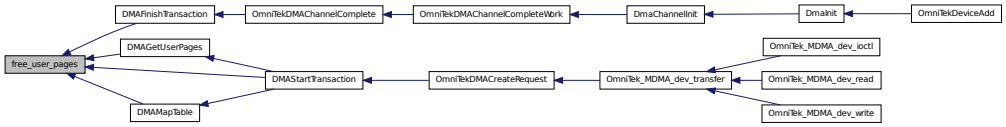
6.9.1.9 void DMAUnMapSg (struct _OmniTekDmaTransactionContext * *pTransaction*)

Here is the caller graph for this function:



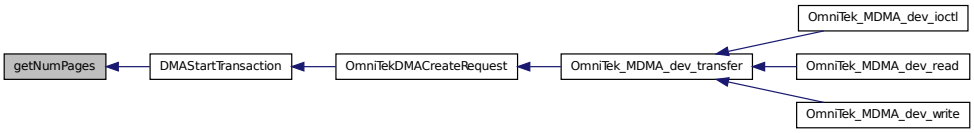
6.9.1.10 static void free_user_pages (struct _OmniTekDmaTransactionContext * *pTransaction*) [static]

Here is the caller graph for this function:



6.9.1.11 static void getNumPages (unsigned long *buffer*, size_t *size*, off_t * *offset*, unsigned long * *num_pages*, unsigned long * *first*, unsigned long * *last*) [static]

Here is the caller graph for this function:



6.10 driver/OmniTekDMACore.h File Reference

```
#include "OmniTek_linux.h"
```

This graph shows which files directly or indirectly include this file:

Functions

- int DMAChannelStart (PResource pChannel)

  Start this DMA Channel.

- void DMAChannelStop (PResource pChannel)

  Stop this DMA Channel.

- int DMAStartTransaction (struct _OmniTekDmaTransactionContext *pTransaction)

Start this transaction.

- void DMAFinishTransaction (struct _OmniTekDmaTransactionContext *pTransaction)

  Finish this transaction.

6.10.1 Function Documentation

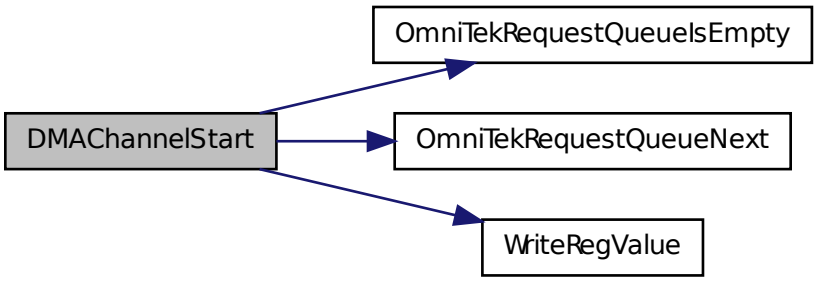
6.10.1.1 int DMAChannelStart (PResource *pChannel*)

Start this DMA Channel.

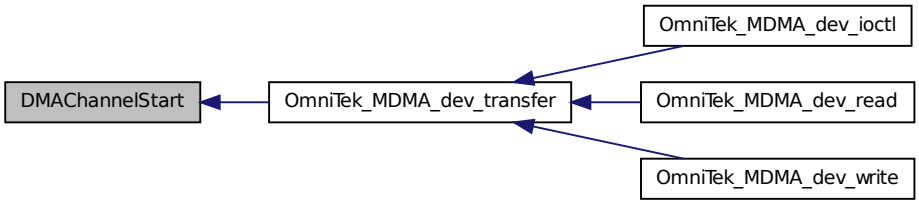
Parameters

`[in]` ***pChannel*** Pointer to DMA channel resource to start

Here is the call graph for this function:



Here is the caller graph for this function:



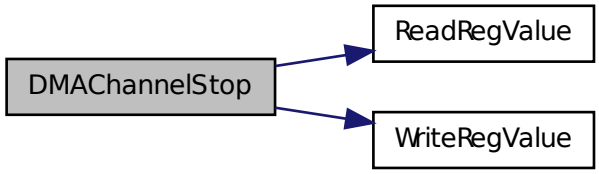
6.10.1.2 void DMAChannelStop (PResource *pChannel*)

Stop this DMA Channel.

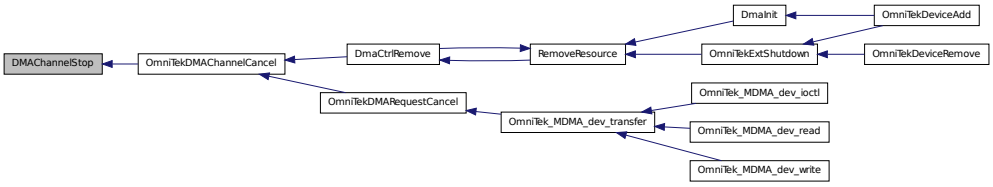
Parameters

[in] ***pChannel*** Pointer to DMA channel resource to stop

Here is the call graph for this function:



Here is the caller graph for this function:



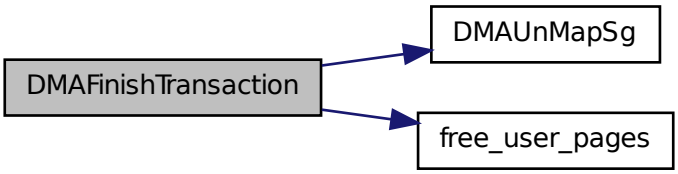
6.10.1.3 void DMAFinishTransaction (struct _OmniTekDmaTransactionContext * *pTransaction*)

Finish this transaction.

Parameters

[in] ***pTransaction*** Pointer to transaction to finish

Here is the call graph for this function:



Here is the caller graph for this function:

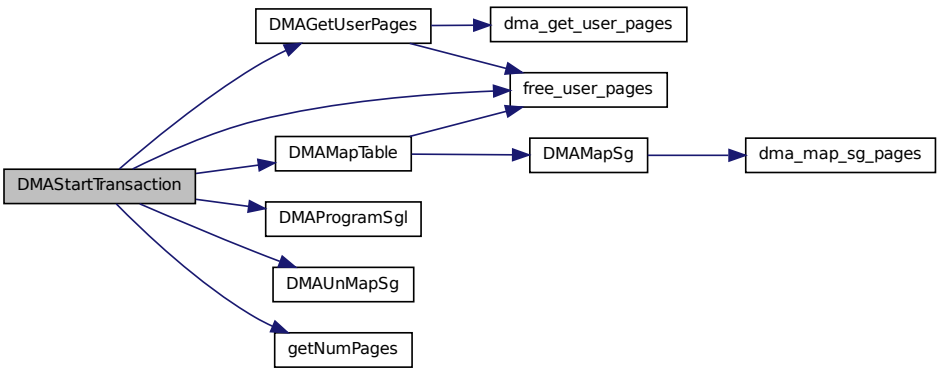
6.10.1.4 int DMAStartTransaction (struct _OmniTekDmaTransactionContext * *pTransaction*)

Start this transaction.

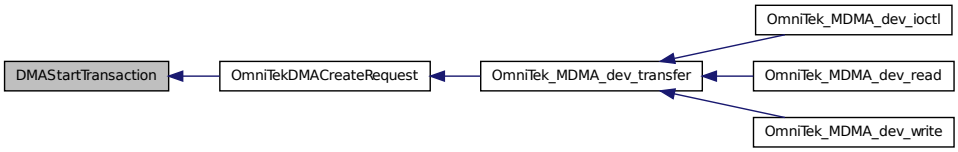
Parameters

`[in]` ***pTransaction*** Pointer to transaction to start

Here is the call graph for this function:



Here is the caller graph for this function:



6.11 driver/OmniTekDmaIsr_linux.c File Reference

Data Structures

- struct _dma_interrupt_info
- struct _dma_interrupt_info::_chan_int_counts

Defines

- #define DMA_CHANNEL_INT_BIT_EVENT 0x40
- #define DMA_CHANNEL_INT_BIT_SG 0x20

Functions

- int ReadRegValue (POMNITEK_INTERFACE_EXTENSION pExt, u32 nBar, u32 nReg, u32 *pVal)

  Read a register value from the hardware.

- irqreturn_t OmniTekDMAFastIsr (int irq, void *dev)
- irqreturn_t OmniTekDMASlowIsr (int irq, void *dev)

6.11.1 Define Documentation

6.11.1.1 #define DMA_CHANNEL_INT_BIT_EVENT 0x40

6.11.1.2 #define DMA_CHANNEL_INT_BIT_SG 0x20

6.11.2 Function Documentation

6.11.2.1 irqreturn_t OmniTekDMAFastIsr (int *irq*, void * *dev*)

6.11.2.2 irqreturn_t OmniTekDMASlowIsr (int *irq*, void * *dev*)

6.11.2.3 int ReadRegValue (POMNITEK_INTERFACE_EXTENSION *pExt*, u32 *nBar*, u32 *nReg*, u32 * *pVal*)

Read a register value from the hardware.

This will perform a register read from the hardware. The cached register values are not updated.

Parameters

[in] ***pExt*** Pointer to device extension to read from

[in] ***nBar*** PCIE Bar number

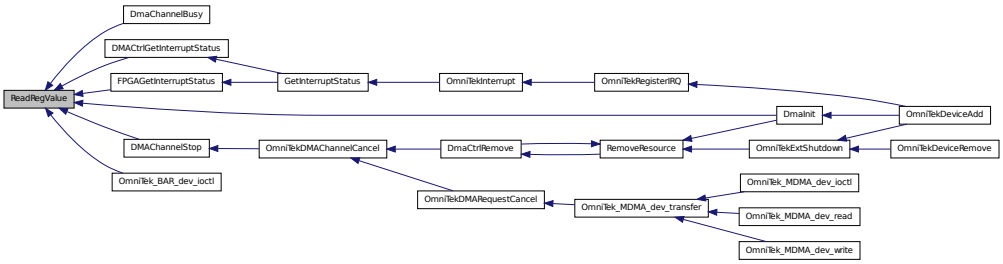
[in] ***nReg*** Register offset

[in] ***pVal*** Pointer to (32 bit) value to store result in

Returns

0 on success or error code

Here is the caller graph for this function:



6.12 driver/OmniTekDmaOperations.c File Reference

```
#include "OmniTek_linux.h"
#include "OmniTekDmaOperations.h"
```

Functions

- int dma_get_user_pages (unsigned long startPage, unsigned long numPages, struct page **pages, struct vm_area_struct **vmas)

  Fault in and lock pages for buffer.

- int dma_map_sg_pages (struct scatterlist **sg, struct page **pages, size_t nents, unsigned long len, loff_t offset)

  Map locked pages into scatterlist.

- int dma_map_sg_init_table_and_chain (struct sg_table *sgt, unsigned long num_pages)

  Initialise scatter gather table and chain entries.

- unsigned long dma_map_test_scatterlist (struct scatterlist **sg, int nents)

  Test a mapped scatter list structure.

6.12.1 Function Documentation

6.12.1.1 int dma_get_user_pages (unsigned long *startPage*, unsigned long *numPages*, struct page ** *pages*, struct vm_area_struct ** *vmas*)

Fault in and lock pages for buffer.

Parameters

[in] ***startPage*** first page to reserve

[in] ***numPages*** number of pages to reserve

[in] ***pages*** pointer to pages data structure to store details of reserved pages

[in] ***vmas*** pointer to vmas data structure - may be left null

Returns

number of pages locked or error code

Here is the caller graph for this function:

6.12.1.2 int dma_map_sg_init_table_and_chain (struct sg_table * *sgt*, unsigned long *num_pages*)

Initialise scatter gather table and chain entries.

Parameters

[in] ***sgt*** Scatter gather table structure to map entries into

[in] ***num_pages*** to map

Returns

number of pages mapped or error code

6.12.1.3 int dma_map_sg_pages (struct scatterlist ** *sg*, struct page ** *pages*, size_t *nents*, unsigned long *len*, loff_t *offset*)

Map locked pages into scatterlist.

Parameters

[in] ***sg*** scatterlist struct pointer

[in] ***pages*** array pointer to pages struct

[in] ***nents*** number of entries to map

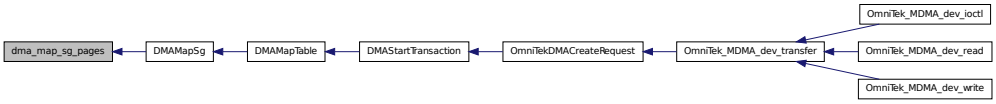
[in] ***len*** total length to allocate (size of transaction)

[in] ***offset*** into first page for beginning of transaction

Returns

total length of entries mapped in scatterlist

Here is the caller graph for this function:



6.12.1.4 unsigned long dma_map_test_scatterlist (struct scatterlist ** *sg*, int *nents*)

Test a mapped scatter list structure.

Parameters

[in] ***sg*** Scatterlist structure to test

[in] ***nents*** number of entries in scatter list

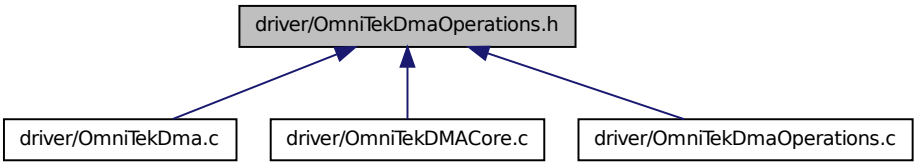
Returns

total size that would be transferred

6.13 driver/OmniTekDmaOperations.h File Reference

```
#include <linux/dma-mapping.h>
#include <linux/pagemap.h>
#include <linux/scatterlist.h>
#include <asm/current.h>
#include <asm/system.h>
#include <asm/uaccess.h>
#include <linux/pfn.h>
```

This graph shows which files directly or indirectly include this file:



Functions

- int dma_get_user_pages (unsigned long startPage, unsigned long numPages, struct page **pages, struct vm_area_struct **vmas)

  Fault in and lock pages for buffer.

- int dma_map_sg_pages (struct scatterlist **sg, struct page **pages, size_t nents, unsigned long len, loff_t offset)

  Map locked pages into scatterlist.

- int dma_map_sg_init_table_and_chain (struct sg_table *sgt, unsigned long num_pages)

  Initialise scatter gather table and chain entries.

- unsigned long dma_map_test_scatterlist (struct scatterlist **sg, int nents)

  Test a mapped scatter list structure.

6.13.1 Function Documentation

6.13.1.1 int dma_get_user_pages (unsigned long *startPage*, unsigned long *numPages*, struct page ** *pages*, struct vm_area_struct ** *vmas*)

Fault in and lock pages for buffer.

Parameters

[in] ***startPage*** first page to reserve

[in] ***numPages*** number of pages to reserve

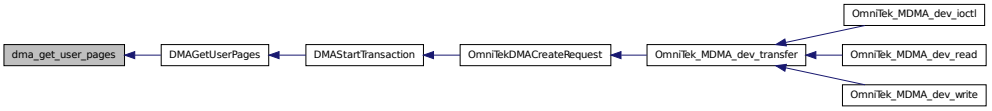
[in] ***pages*** pointer to pages data structure to store details of reserved pages

[in] ***vmas*** pointer to vmas data structure - may be left null

Returns

number of pages locked or error code

Here is the caller graph for this function:



6.13.1.2 int dma_map_sg_init_table_and_chain (struct sg_table * *sgt*, unsigned long *num_pages*)

Initialise scatter gather table and chain entries.

Parameters

[in] ***sgt*** Scatter gather table structure to map entries into

[in] ***num_pages*** to map

Returns

number of pages mapped or error code

6.13.1.3 int dma_map_sg_pages (struct scatterlist ** *sg*, struct page ** *pages*, size_t *nents*, unsigned long *len*, loff_t *offset*)

Map locked pages into scatterlist.

Parameters

[in] ***sg*** scatterlist struct pointer

[in] ***pages*** array pointer to pages struct

[in] ***nents*** number of entries to map

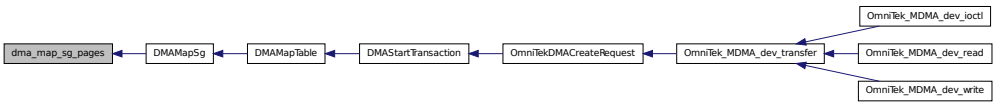
[in] ***len*** total length to allocate (size of transaction)

[in] ***offset*** into first page for beginning of transaction

Returns

total length of entries mapped in scatterlist

Here is the caller graph for this function:



6.13.1.4 unsigned long dma_map_test_scatterlist (struct scatterlist ** *sg*, int *nents*)

Test a mapped scatter list structure.

Parameters

[in] ***sg*** Scatterlist structure to test

[in] ***nents*** number of entries in scatter list

Returns

total size that would be transferred

6.14 driver/OmniTekDMARequest.c File Reference

```
#include "OmniTekDMARequest.h"
```

Functions

- int OmniTekDMACreateRequest (PResource pChannel, const char __user *pBuffer, size_t Size, loff_t LocalAddr, bool Write, OmniTekTransactionType type, void *requestData, OmniTekTransactionCompleteCb callback, struct _OmniTekDmaTransactionContext **ppTransaction)

  Create a dma request (transaction).

- int OmniTekDMAReleaseRequest (struct _OmniTekDmaTransactionContext *pTransaction)

  Release a completed request.

- void OmniTekDMAChannelCompleteWork (struct work_struct *work)

  Complete work function.

- void OmniTekDMAChannelComplete (PResource pChannel)

  Complete a request on this channel.

- int OmniTekDMARequestCancel (struct _OmniTekDmaTransactionContext *pTransaction)

  Cancel a dma transaction.

- int OmniTekDMAChannelCancel (PResource pChannel)

  Cancel all transactions on a DMA Channel.

6.14.1 Function Documentation

6.14.1.1 int OmniTekDMAChannelCancel (PResource *pChannel*)

Cancel all transactions on a DMA Channel.

Parameters

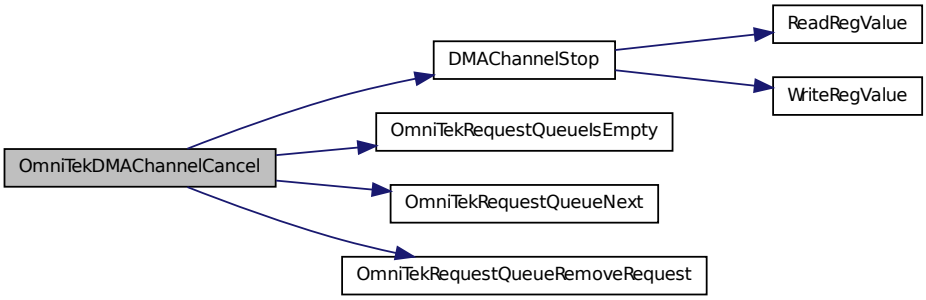
[in] ***pChannel*** resource to cancel.

Returns

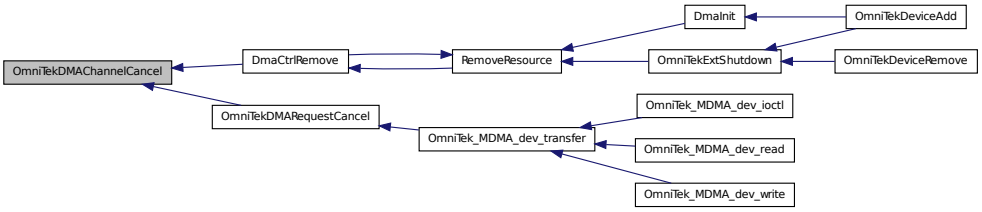
0 on success

All transactions on this channel will be cancelled (and completed with cancelled status)

Here is the call graph for this function:



Here is the caller graph for this function:



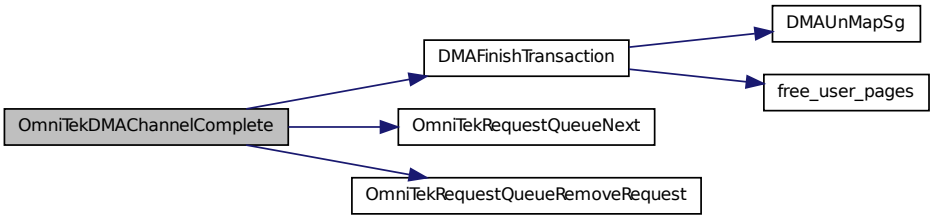
6.14.1.2 void OmniTekDMAChannelComplete (PResource *pChannel*)

Complete a request on this channel.

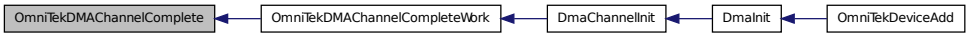
Parameters

`[in]` ***pChannel*** pointer to channel resource that has a completed request

Here is the call graph for this function:



Here is the caller graph for this function:



6.14.1.3 void OmniTekDMAChannelCompleteWork (struct work_struct * *work*)

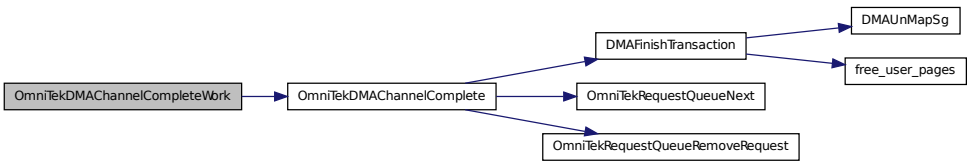
Complete work function.

Parameters

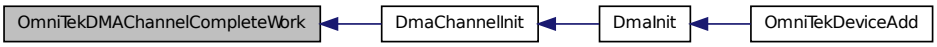
`[in]` ***work*** struct pointer

This function is scheduled when a transaction completes, it calls the channel complete function for the channel.

Here is the call graph for this function:



Here is the caller graph for this function:



6.14.1.4 int OmniTekDMACreateRequest (PResource *pChannel*, const char __user * *pBuffer*, size_t *Size*, loff_t *LocalAddr*, bool *Write*, OmniTekTransactionType *type*, void * *requestData*, OmniTekTransactionCompleteCb *callback*, struct _OmniTekDmaTransactionContext ** *ppTransaction*)

Create a dma request (transaction).

Parameters

[in] ***pChannel*** pointer to channel resource to create request for

[in] ***pBuffer*** pointer to user space buffer that is the source/destination for the request

[in] ***Size*** size of transfer

[in] ***LocalAddr*** device side address for transfer

[in] ***Write*** true if this is a write to the device

[in] ***type*** type of request

[in] ***requestData*** pointer to data to store in request

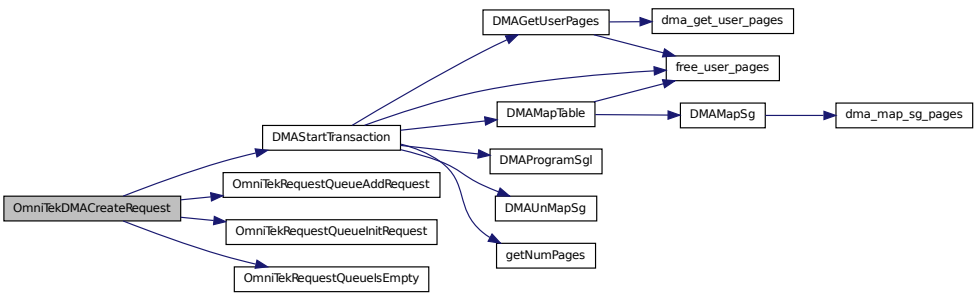
[in] ***callback*** completion callback function

[in] ***ppTransaction*** pointer to transaction struct pointer - transaction will be allocated by this function

Returns

0 on success, -EBUSY if the request was accepted into the pending queue, or error code if there is a problem creating the request

Here is the call graph for this function:



Here is the caller graph for this function:

6.14.1.5 int OmniTekDMAReleaseRequest (struct _OmniTekDmaTransactionContext * *pTransaction*)

Release a completed request.

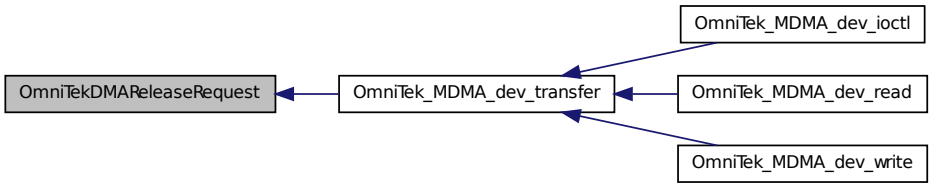
Parameters

`[in]` ***pTransaction*** pointer to completed transaction (will be freed, cannot be used after this call)

Returns

0 on success

Here is the caller graph for this function:



6.14.1.6 int OmniTekDMARequestCancel (struct _OmniTekDmaTransactionContext * *pTransaction*)

Cancel a dma transaction.

Parameters

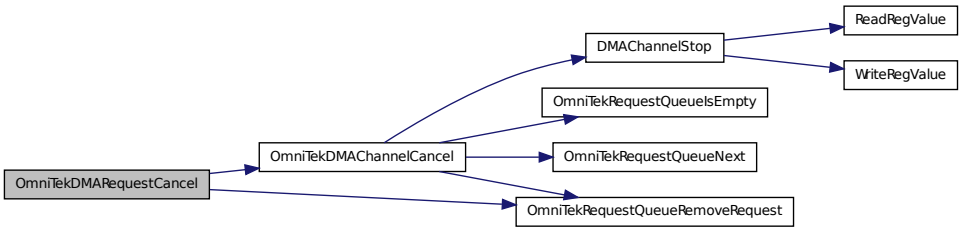
`[in]` ***pTransaction*** pointer to transaction to cancel

Returns

0 on success

If the request is pending then it is completed with cancelled status. If it is active then the channel is stopped and all transactions outstanding will be cancelled

Here is the call graph for this function:

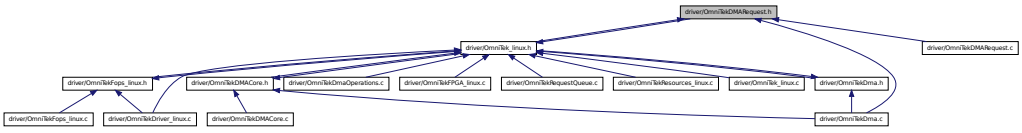


Here is the caller graph for this function:

6.15 driver/OmniTekDMARequest.h File Reference

```
#include "OmniTek_linux.h"
```

This graph shows which files directly or indirectly include this file:



Functions

- int OmniTekDMACreateRequest (PResource pChannel, const char __user *pBuffer, size_t Size, loff_t LocalAddr, bool Write, OmniTekTransactionType type, void *requestData, OmniTekTransactionCompleteCb callback, struct _OmniTekDmaTransactionContext **ppTransaction)

  Create a dma request (transaction).

- int OmniTekDMAReleaseRequest (struct _OmniTekDmaTransactionContext *pTransaction)

  Release a completed request.

- void OmniTekDMAChannelCompleteWork (struct work_struct *work)

  Complete work function.

- void OmniTekDMAChannelComplete (PResource pChannel)

  Complete a request on this channel.

- int OmniTekDMARequestCancel (struct _OmniTekDmaTransactionContext *pTransaction)

  Cancel a dma transaction.

- int OmniTekDMAChannelCancel (PResource pChannel)

  Cancel all transactions on a DMA Channel.

6.15.1 Function Documentation

6.15.1.1 int OmniTekDMAChannelCancel (PResource *pChannel*)

Cancel all transactions on a DMA Channel.

Parameters

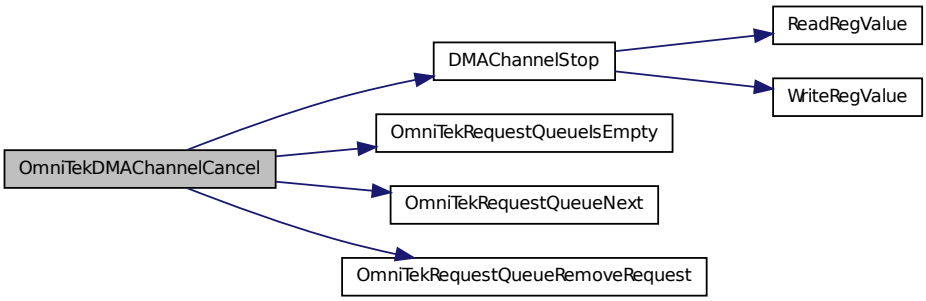
`[in]` ***pChannel*** resource to cancel.

Returns

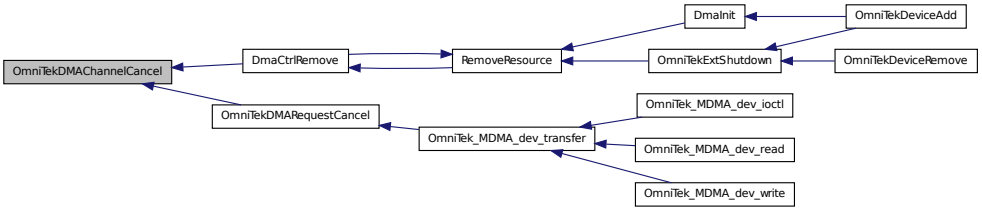
0 on success

All transactions on this channel will be cancelled (and completed with cancelled status)

Here is the call graph for this function:



Here is the caller graph for this function:



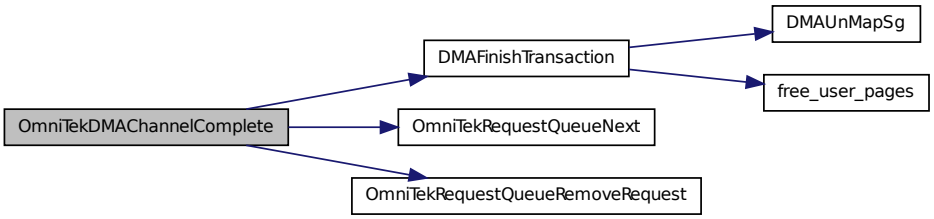
6.15.1.2 void OmniTekDMAChannelComplete (PResource *pChannel*)

Complete a request on this channel.

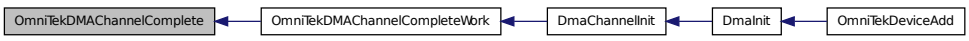
Parameters

`[in]` ***pChannel*** pointer to channel resource that has a completed request

Here is the call graph for this function:



Here is the caller graph for this function:



6.15.1.3 void OmniTekDMAChannelCompleteWork (struct work_struct * *work*)

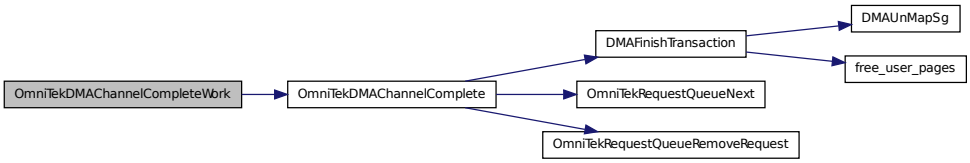
Complete work function.

Parameters

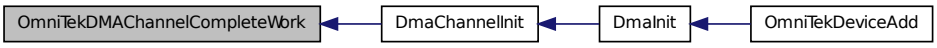
`[in]` ***work*** struct pointer

This function is scheduled when a transaction completes, it calls the channel complete function for the channel.

Here is the call graph for this function:



Here is the caller graph for this function:



6.15.1.4 int OmniTekDMACreateRequest (PResource *pChannel*, const char __user * *pBuffer*, size_t *Size*, loff_t *LocalAddr*, bool *Write*, OmniTekTransactionType *type*, void * *requestData*, OmniTekTransactionCompleteCb *callback*, struct _OmniTekDmaTransactionContext ** *ppTransaction*)

Create a dma request (transaction).

Parameters

[in] ***pChannel*** pointer to channel resource to create request for

[in] ***pBuffer*** pointer to user space buffer that is the source/destination for the request

[in] ***Size*** size of transfer

[in] ***LocalAddr*** device side address for transfer

[in] ***Write*** true if this is a write to the device

[in] ***type*** type of request

[in] ***requestData*** pointer to data to store in request

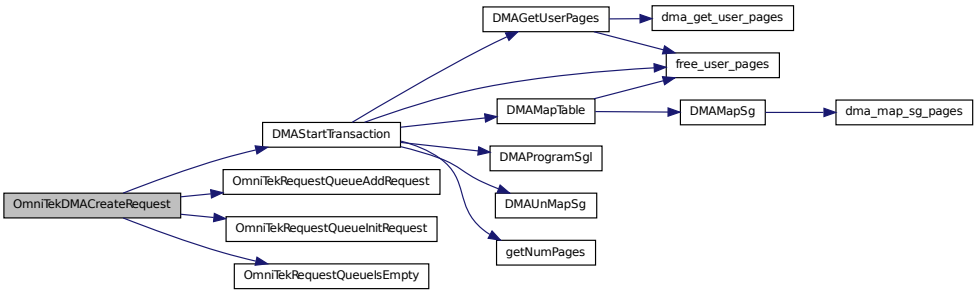
[in] ***callback*** completion callback function

[in] ***ppTransaction*** pointer to transaction struct pointer - transaction will be allocated by this function

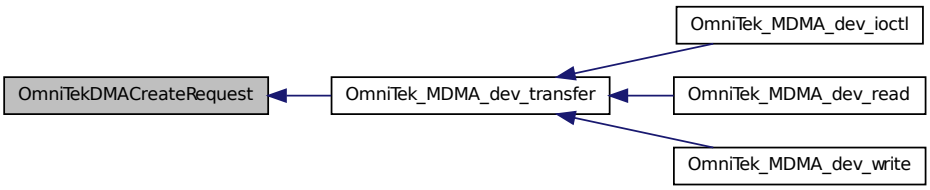
Returns

0 on success, -EBUSY if the request was accepted into the pending queue, or error code if there is a problem creating the request

Here is the call graph for this function:



Here is the caller graph for this function:



6.15.1.5 int OmniTekDMAReleaseRequest (struct _OmniTekDmaTransactionContext * *pTransaction*)

Release a completed request.

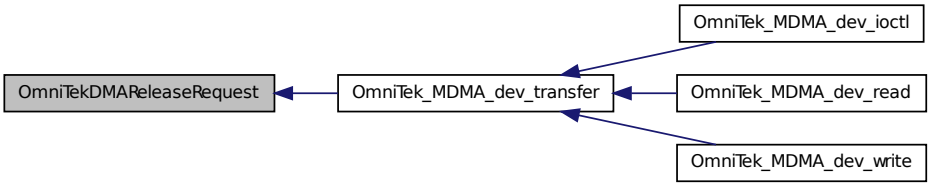
Parameters

`[in]` ***pTransaction*** pointer to completed transaction (will be freed, cannot be used after this call)

Returns

0 on success

Here is the caller graph for this function:



6.15.1.6 int OmniTekDMARequestCancel (struct _OmniTekDmaTransactionContext * *pTransaction*)

Cancel a dma transaction.

Parameters

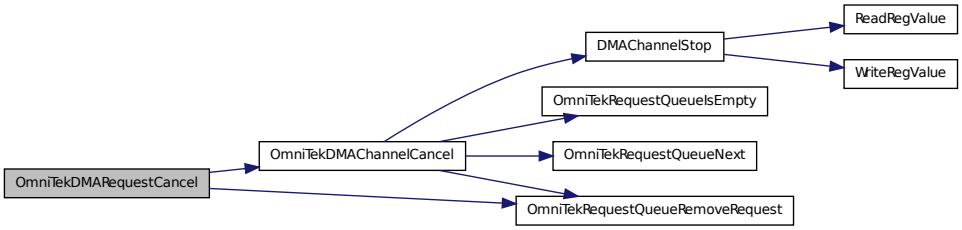
`[in]` ***pTransaction*** pointer to transaction to cancel

Returns

0 on success

If the request is pending then it is completed with cancelled status. If it is active then the channel is stopped and all transactions outstanding will be cancelled

Here is the call graph for this function:



Here is the caller graph for this function:

6.16 driver/OmniTekDriver_linux.c File Reference

```
#include "OmniTek_linux.h"
#include "OmniTekFops_linux.h"
#include "OmniTekDriver_linux.h"
#include <asm/atomic.h>
```

Functions

- MODULE_DEVICE_TABLE (pci, ids)
- static u16 OmniTekGetDeviceId (struct pci_dev *dev)

  Get the device ID.

- static u8 GetNumPciLanes (struct pci_dev *dev)

  Get the number of PCIE lanes enabled for the device.

- static irqreturn_t OmniTekInterrupt (int irq, void *dev_id)

  IRQ Handler.

- void DmaChannelISR (PResource pChannel)

  Interrupt service routine for DMA Channels.

- u32 DmaStatus (PResource pCtrl)

  Get interrupt status for DMA.

- void DmaISR (POMNITEK_INTERFACE_EXTENSION pExt)

  Process DMA Interrupts.

- static irqreturn_t OmniTekInterruptHandler (int irq, void *dev_id)

  Bottom half of interrupt handler.

- void OmniTekRegisterIRQ (POMNITEK_INTERFACE_EXTENSION pExt)

  Register device extension for an interrupt.

- void OmniTekUnRegisterIRQ (POMNITEK_INTERFACE_EXTENSION pExt)
  Unregister device extension for an interrupt.
- int OmniTekDeviceAdd (struct pci_dev *dev, const struct pci_device_id *id)
  PCI Device Probe function.
- static void OmniTekDeviceRemove (struct pci_dev *dev)
  Called when a device is removed from the system.
- static int __init OmniTekDriver_init (void)
  Driver init function.
- static void __exit OmniTekDriver_exit (void)
  Driver exit function.
- struct _OmniTekDriver * GetOmniTekDriver (struct pci_driver *driver)
  Get Driver pointer.
- MODULE_LICENSE ("GPL")
- module_init (OmniTekDriver_init)
- module_exit (OmniTekDriver_exit)

Variables

- static OmniTekDriver omnitek_driver
- static u32 nInterruptStatus
- atomic_t irqPend = ATOMIC_INIT(0)
- atomic_t irqTotal = ATOMIC_INIT(0)
- atomic_t handlerCount = ATOMIC_INIT(0)
- atomic_t handlerHandled = ATOMIC_INIT(0)

6.16.1 Function Documentation

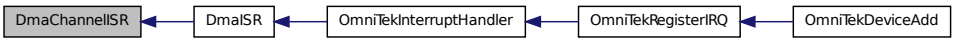
6.16.1.1 void DmaChannelISR (PResource *pChannel*)

Interrupt service routine for DMA Channels.

Parameters

`[in]` ***pChannel*** Pointer to channel that has interrupted

Here is the caller graph for this function:



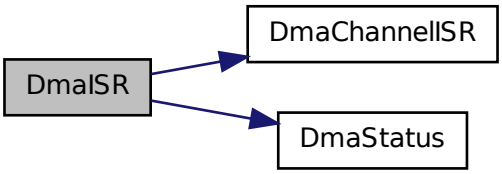
6.16.1.2 void DmaISR (POMNITEK_INTERFACE_EXTENSION *pExt*)

Process DMA Interrupts.

Parameters

[in] ***pExt*** pointer to Driver extension that has interrupted

Here is the call graph for this function:



Here is the caller graph for this function:

6.16.1.3 u32 DmaStatus (PResource *pCtrl*)

Get interrupt status for DMA.

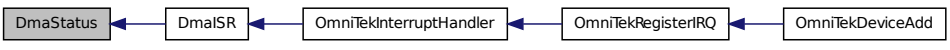
Parameters

[in] ***pCtrl*** pointer to DMA controller resource

Returns

Status bits for DMA channels

Here is the caller graph for this function:



6.16.1.4 static u8 GetNumPciLanes (struct pci_dev * *dev*) [static]

Get the number of PCIE lanes enabled for the device.

Reads the PCI Configuration space to determine the number of lanes available to the device

Parameters

[in] ***dev*** PCI Device struct to check

Returns

Number of active lanes or zero on error.

Here is the caller graph for this function:

6.16.1.5 struct _OmniTekDriver* GetOmniTekDriver (struct pci_driver * *driver*) [read]

Get Driver pointer.

Get the OmniTekDriver struct from the pci_driver.

This returns a pointer to the OmniTek driver data structure given the pci_driver data structure

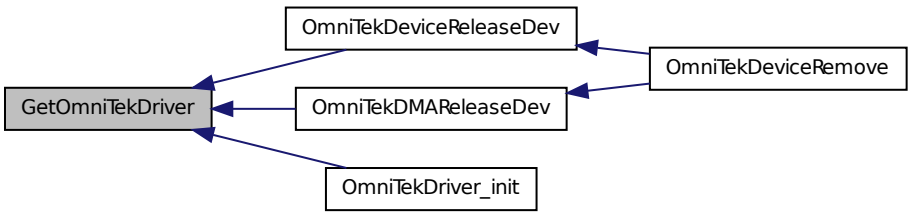
Parameters

[in] ***driver*** pointer to pci_driver struct

Returns

Pointer to OmniTek Driver data structure

Here is the caller graph for this function:



6.16.1.6 MODULE_DEVICE_TABLE (pci , ids)

6.16.1.7 module_exit (OmniTekDriver_exit)

6.16.1.8 module_init (OmniTekDriver_init)

6.16.1.9 MODULE_LICENSE ("GPL")

6.16.1.10 int OmniTekDeviceAdd (struct pci_dev * *dev*, const struct pci_device_id * *id*)

PCI Device Probe function.

Device add function (pci probe function).

Parameters

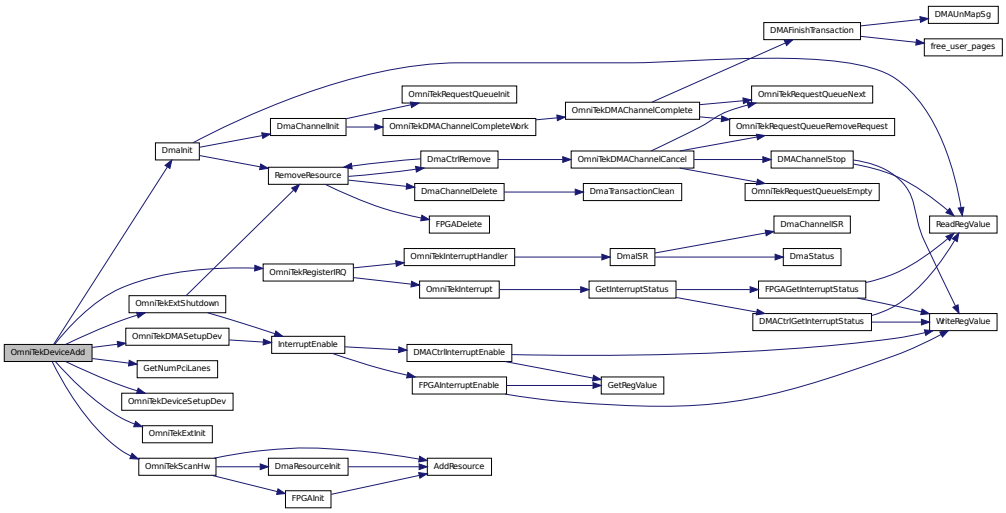
[in] ***dev*** Pointer to pci device that matches our vendor/device IDs

[in] ***id*** vendor/device ID that matches one of the ids we specify

Returns

success if we can set up our driver for the device or an error code

Here is the call graph for this function:



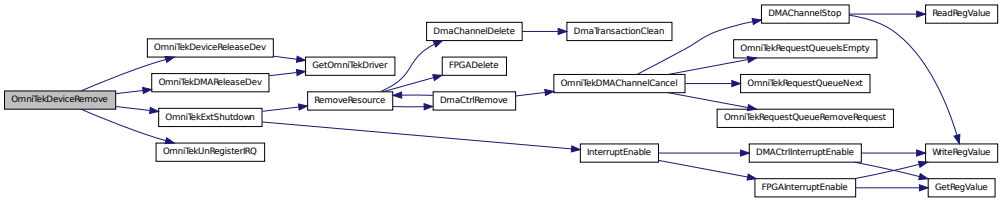
6.16.1.11 static void OmniTekDeviceRemove (struct pci_dev * *dev*) [static]

Called when a device is removed from the system.

Parameters

[in] ***dev*** pointer to pci device struct that is being removed

Here is the call graph for this function:



6.16.1.12 static void __exit OmniTekDriver_exit (void) [static]

Driver exit function.

This shuts down and cleans up any device extensions that are being managed by this driver

6.16.1.13 static int __init OmniTekDriver_init (void) [static]

Driver init function.

This sets up the initial driver data structures so we can keep track of devices we manage

Returns

Sucess or error code

Here is the call graph for this function:

6.16.1.14 static u16 OmniTekGetDeviceId (struct pci_dev * *dev*) [static]

Get the device ID.

Parameters

[in] ***dev*** PCI Device struct to check

Returns

Device id or 0 on error.

6.16.1.15 static irqreturn_t OmniTekInterrupt (int *irq*, void * *dev_id*) [static]

IRQ Handler.

Parameters

[in] ***irq*** IRQ Number that fired

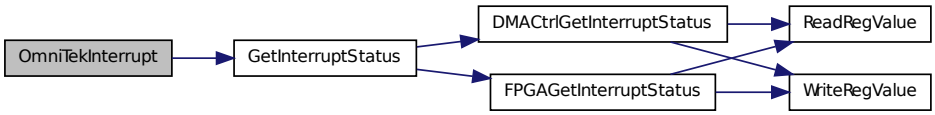
[in] ***dev_id*** Device ID That interrupted

Returns

Value indicating whether an interrupt was handled or not

This is the 'fast' half of the ISR,

Here is the call graph for this function:



Here is the caller graph for this function:

6.16.1.16 static irqreturn_t OmniTekInterruptHandler (int *irq*, void * *dev_id*) [static]

Bottom half of interrupt handler.

Parameters

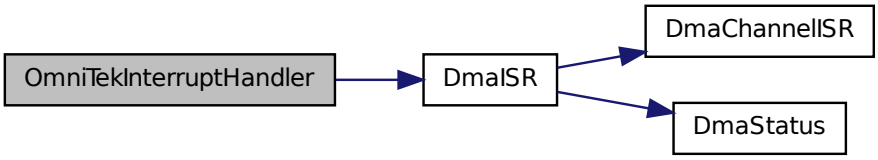
[in] ***irq*** IRQ number that has been triggered

[in] ***dev_id*** Pointer to device extension that has interrupted

Returns

Result depending on whether interrupt has been handled

Here is the call graph for this function:



Here is the caller graph for this function:

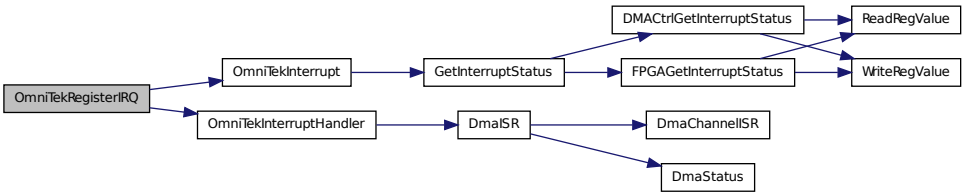
6.16.1.17 void OmniTekRegisterIRQ (POMNITEK_INTERFACE_EXTENSION *pExt*)

Register device extension for an interrupt.

Parameters

`[in]` ***pExt*** Pointer to device extension to register interrupt for

Here is the call graph for this function:



Here is the caller graph for this function:

6.16.1.18 void OmniTekUnRegisterIRQ (POMNITEK_INTERFACE_EXTENSION *pExt*)

Unregister device extension for an interrupt.

Parameters

`[in]` ***pExt*** Pointer to device extension to unregister interrupt for

Here is the caller graph for this function:

6.16.2 Variable Documentation

6.16.2.1 atomic_t handlerCount = ATOMIC_INIT(0)

6.16.2.2 atomic_t handlerHandled = ATOMIC_INIT(0)

6.16.2.3 atomic_t irqPend = ATOMIC_INIT(0)

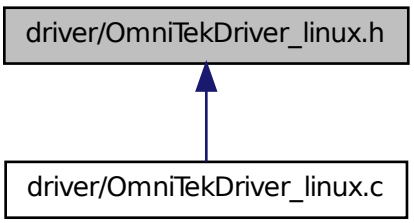
6.16.2.4 atomic_t irqTotal = ATOMIC_INIT(0)

6.16.2.5 u32 nInterruptStatus [static]

6.16.2.6 OmniTekDriver omnitek_driver [static]

6.17 driver/OmniTekDriver_linux.h File Reference

This graph shows which files directly or indirectly include this file:



Defines

- #define USE_IRQ_THREAD
- #define IORESOURCE_MEM_64 0x00100000

Functions

- static u16 OmniTekGetDeviceId (struct pci_dev *dev)
  Get Device ID from pcie endpoint.
- static u8 GetNumPciLanes (struct pci_dev *dev)
  Get Device ID from pcie endpoint.
- static int OmniTekDeviceAdd (struct pci_dev *dev, const struct pci_device_id *id)
  Device add function (pci probe function).
- static void OmniTekDeviceRemove (struct pci_dev *dev)

Device remove function (pci remove function).

Variables

- static struct pci_device_id ids [ ]

6.17.1 Define Documentation

6.17.1.1 #define IORESOURCE_MEM_64 0x00100000

6.17.1.2 #define USE_IRQ_THREAD

6.17.2 Function Documentation

6.17.2.1 static u8 GetNumPciLanes (struct pci_dev * *dev*) [static]

Get Device ID from pcie endpoint.

Parameters

[in] ***dev*** pointer to pci_dev struct to request ID from

Returns

device ID

6.17.2.2 static int OmniTekDeviceAdd (struct pci_dev * *dev*, const struct pci_device_id * *id*) [static]

Device add function (pci probe function).

Parameters

[in] ***dev*** pci device struct that is being probed

[in] ***id*** device id that we are being called for

Returns

0 if probing successful or error code

Device add function (pci probe function).

Parameters

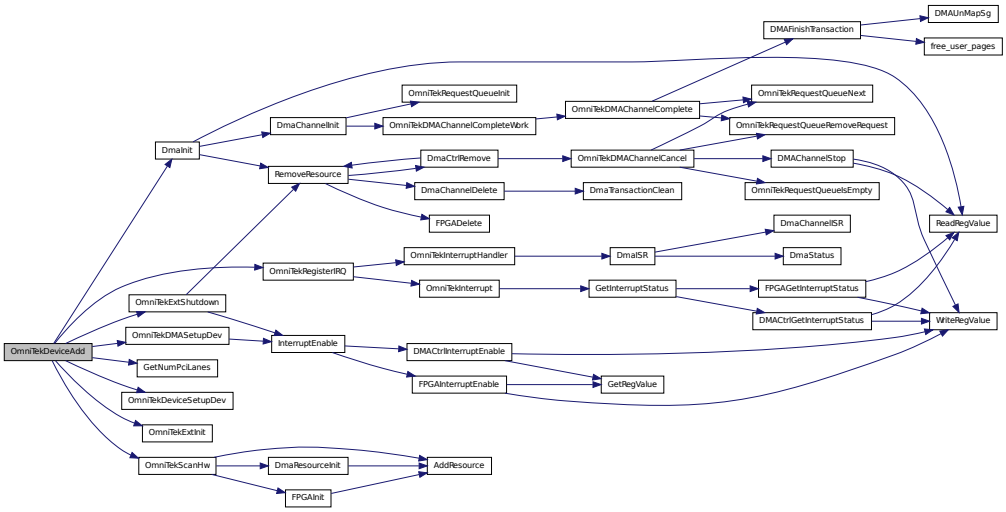
[in] ***dev*** Pointer to pci device that matches our vendor/device IDs

[in] ***id*** vendor/device ID that matches one of the ids we specify

Returns

success if we can set up our driver for the device or an error code

Here is the call graph for this function:



6.17.2.3 static void OmniTekDeviceRemove (struct pci_dev * *dev*) [static]

Device remove function (pci remove function).

Parameters

[in] ***dev*** pci device struct that is being removed

6.17.2.4 static u16 OmniTekGetDeviceId (struct pci_dev * *dev*) [static]

Get Device ID from pcie endpoint.

Parameters

[in] ***dev*** pointer to pci_dev struct to request ID from

Returns

device ID

6.17.3 Variable Documentation

6.17.3.1 struct pci_device_id ids[ ] [static]

Initial value:

```
{
        { PCI_DEVICE(0x1AA3, 0x0001), },
        { PCI_DEVICE(0x1AA3, 0x2002), },
        { PCI_DEVICE(0x1AA3, 0x0010), },
```

```
  { PCI_DEVICE(0x11A4, 0x0057), },
  { PCI_DEVICE(0x11A4, 0x0058), },
          { 0, }
}
```

6.18 driver/OmniTekFops_linux.c File Reference

```
#include "OmniTekFops_linux.h"
#include "asm/uaccess.h"
```

Functions

- void OmniTekDeviceSetupDev (struct _OMNITEK_INTERFACE_EXTENSION *pExt)

  Create system /dev entries for our device.

- void OmniTekDeviceReleaseDev (POMNITEK_INTERFACE_EXTENSION pExt)
- int OmniTek_BAR_dev_ioctl (struct inode *inode, struct file *filp, unsigned int cmd, unsigned long arg)

  IOCtl call to MDMA Device.

- int OmniTek_BAR_dev_open (struct inode *inode, struct file *filp)

  BAR Device open.

- int OmniTek_BAR_dev_release (struct inode *inode, struct file *filp)

  BAR Device close/release.

Variables

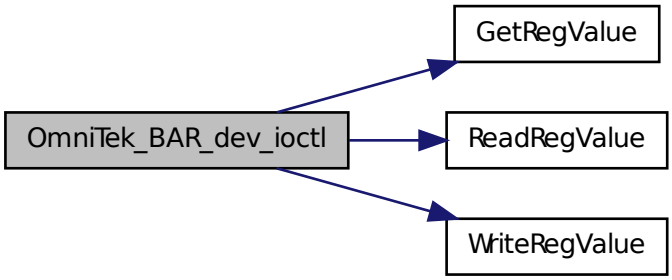
- static struct _OmniTek_dev OmniTekBoardev
- struct file_operations OmniTek_BAR_dev_fops

6.18.1 Function Documentation

6.18.1.1 int OmniTek_BAR_dev_ioctl (struct inode * *inode*, struct file * *filp*, unsigned int *cmd*, unsigned long *arg*)

IOCtl call to MDMA Device.

Here is the call graph for this function:



6.18.1.2 int OmniTek_BAR_dev_open (struct inode * *inode*, struct file * *filp*)

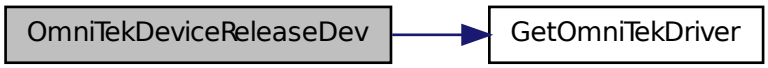
BAR Device open.

6.18.1.3 int OmniTek_BAR_dev_release (struct inode * *inode*, struct file * *filp*)

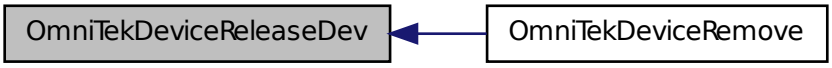
BAR Device close/release.

6.18.1.4 void OmniTekDeviceReleaseDev (POMNITEK_INTERFACE_EXTENSION *pExt*)

Here is the call graph for this function:



Here is the caller graph for this function:



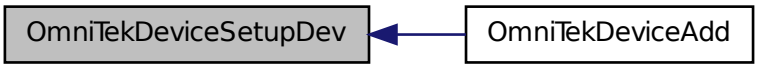
6.18.1.5 void OmniTekDeviceSetupDev (struct _OMNITEK_INTERFACE_EXTENSION * *pExt*)

Create system /dev entries for our device.

Setup BAR Device nodes.

Once the hardware is fully elaborated we create the system devices for our device.

Here is the caller graph for this function:



6.18.2 Variable Documentation

6.18.2.1 struct file_operations OmniTek_BAR_dev_fops

Initial value:

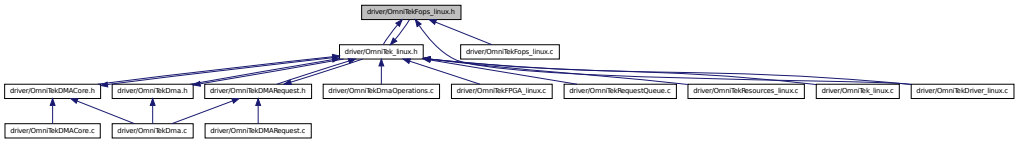
```
 {
  .owner = THIS_MODULE,
  .ioctl = OmniTek_BAR_dev_ioctl,
  .open = OmniTek_BAR_dev_open,
  .release = OmniTek_BAR_dev_release
}
```

6.18.2.2 struct _OmniTek_dev OmniTekBoardev [static]

6.19 driver/OmniTekFops_linux.h File Reference

```
#include "OmniTek_linux.h"
```

This graph shows which files directly or indirectly include this file:



Data Structures

- struct _OmniTek_dev

Typedefs

- typedef enum _OMNITEK_DEVTYPES OmniTek_DevTypes
- typedef struct _OmniTek_dev OmniTek_dev

Enumerations

- enum _OMNITEK_DEVTYPES { OMNITEK_DEV_BAR, OMNITEK_DEV_RESOURCE, OMNITEK_DEV_MDMA }

  This enumeration lists the types of device that are used in the driver.

Functions

- int OmniTek_BAR_dev_ioctl (struct inode *, struct file *, unsigned int, unsigned long)

  IOCtl call to MDMA Device.

- int OmniTek_BAR_dev_open (struct inode *, struct file *)

  BAR Device open.

- int OmniTek_BAR_dev_release (struct inode *, struct file *)

  BAR Device close/release.

- void OmniTekDeviceSetupDev (struct _OMNITEK_INTERFACE_EXTENSION *pExt)

  Setup BAR Device nodes.

- void OmniTekDeviceReleaseDev (struct _OMNITEK_INTERFACE_EXTENSION *pExt)

  Release BAR Device nodes.

6.19.1 Typedef Documentation

6.19.1.1 typedef struct _OmniTek_dev OmniTek_dev

6.19.1.2 typedef enum _OMNITEK_DEVTYPES OmniTek_DevTypes

6.19.2 Enumeration Type Documentation

6.19.2.1 enum _OMNITEK_DEVTYPES

This enumeration lists the types of device that are used in the driver.

Enumerator:

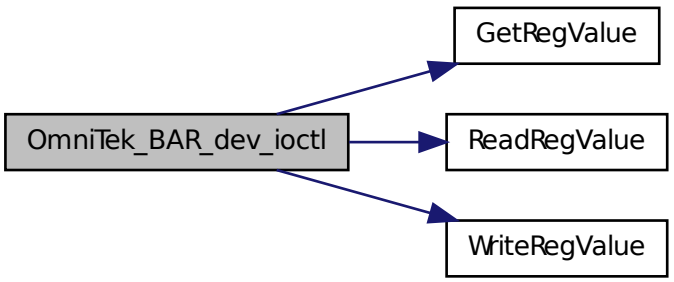
OMNITEK_DEV_BAR BAR type device
OMNITEK_DEV_RESOURCE Resource type device
OMNITEK_DEV_MDMA MDMA Type Device

6.19.3 Function Documentation

6.19.3.1 int OmniTek_BAR_dev_ioctl (struct inode *, struct file *, unsigned *int*, unsigned *long*)

IOCtl call to MDMA Device.

Here is the call graph for this function:



6.19.3.2 int OmniTek_BAR_dev_open (struct inode *, struct file *)

BAR Device open.

6.19.3.3 int OmniTek_BAR_dev_release (struct inode *, struct file *)

BAR Device close/release.

6.19.3.4 void OmniTekDeviceReleaseDev (struct _OMNITEK_INTERFACE_EXTENSION * *pExt*)

Release BAR Device nodes.

Parameters

[in] ***pExt*** Pointer to device extension to release BAR devices for

6.19.3.5 void OmniTekDeviceSetupDev (struct _OMNITEK_INTERFACE_EXTENSION * *pExt*)

Setup BAR Device nodes.

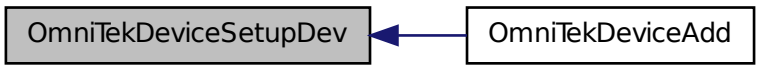
Parameters

[in] ***pExt*** Pointer to device extension to set up BAR devices for

Setup BAR Device nodes.

Once the hardware is fully elaborated we create the system devices for our device.

Here is the caller graph for this function:



6.20 driver/OmniTekFPGA_linux.c File Reference

```
#include "OmniTek_linux.h"
#include "../include/OmniTekIoctl_linux.h"
```

Functions

- void FPGAInit (Resource *pResource)

  Initialise the FPGA resource.

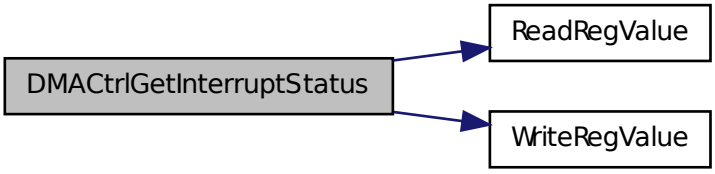
- void FPGADelete (PResource pFPGA)
- void FPGAInterruptEnable (PResource pFPGA, u32 Interrupt, bool Enable)
- void DMACtrlInterruptEnable (PResource pDmaCtrl, u32 Interrupt, bool Enable)
- void InterruptEnable (POMNITEK_INTERFACE_EXTENSION pExt, u32 Interrupt, bool Enable)
- u32 FPGAGetInterruptStatus (PResource pFPGA)
- u32 DMACtrlGetInterruptStatus (PResource pDmaCtrl)

- u32 GetInterruptStatus (POMNITEK_INTERFACE_EXTENSION pExt)
- u64 FPGAReadTime (PResource pFPGA)
- void FPGAGetTime (PResource pFPGA, u64 *Count)

6.20.1 Function Documentation

6.20.1.1 u32 DMACtrlGetInterruptStatus (PResource *pDmaCtrl*)

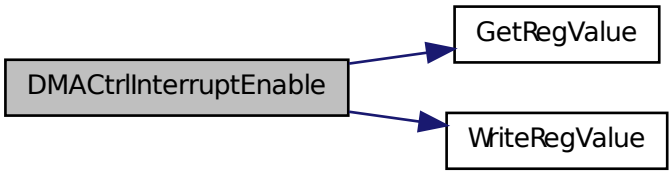
Here is the call graph for this function:



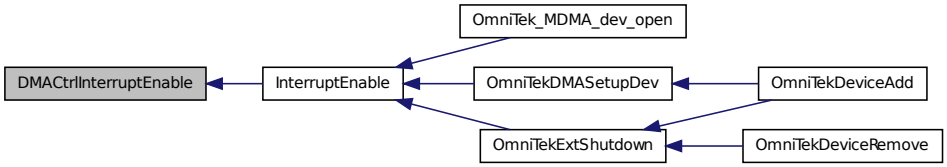
Here is the caller graph for this function:

6.20.1.2 void DMACtrlInterruptEnable (PResource *pDmaCtrl*, u32 *Interrupt*, bool *Enable*)

Here is the call graph for this function:

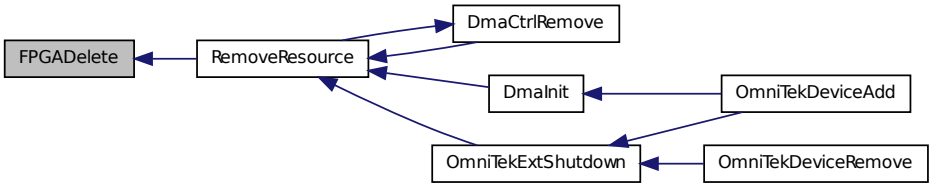


Here is the caller graph for this function:



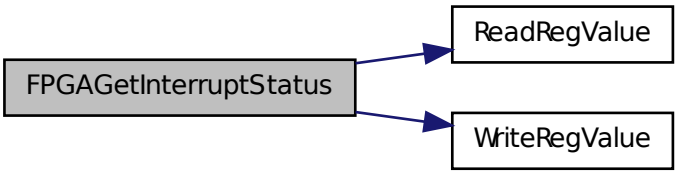
6.20.1.3 void FPGADelete (PResource *pFPGA*)

Here is the caller graph for this function:



6.20.1.4 u32 FPGAGetInterruptStatus (PResource *pFPGA*)

Here is the call graph for this function:



Here is the caller graph for this function:

6.20.1.5 void FPGAGetTime (PResource *pFPGA*, u64 * *Count*)

Here is the call graph for this function:

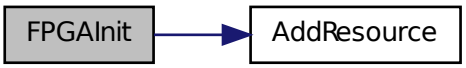
6.20.1.6 void FPGAInit (Resource * *pResource*)

Initialise the FPGA resource.

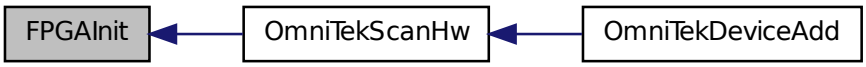
Parameters

pResource Pointer to (FPGA) resource to initialise

Here is the call graph for this function:

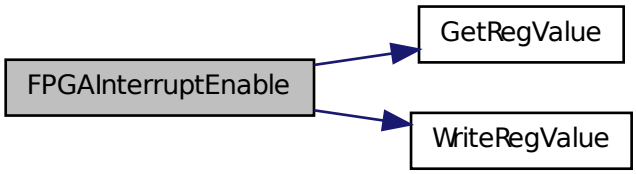


Here is the caller graph for this function:

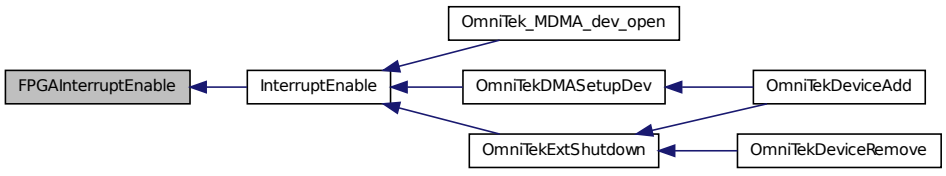


6.20.1.7 void FPGAInterruptEnable (PResource *pFPGA*, u32 *Interrupt*, bool *Enable*)

Here is the call graph for this function:



Here is the caller graph for this function:

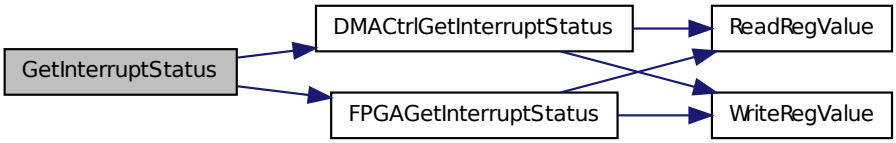


6.20.1.8 u64 FPGAReadTime (PResource *pFPGA*)

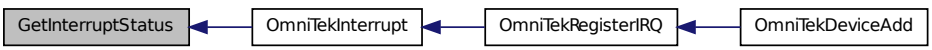
Here is the caller graph for this function:

6.20.1.9 u32 GetInterruptStatus (POMNITEK_INTERFACE_EXTENSION *pExt*)

Here is the call graph for this function:

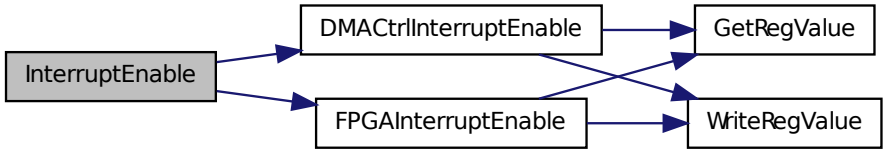


Here is the caller graph for this function:

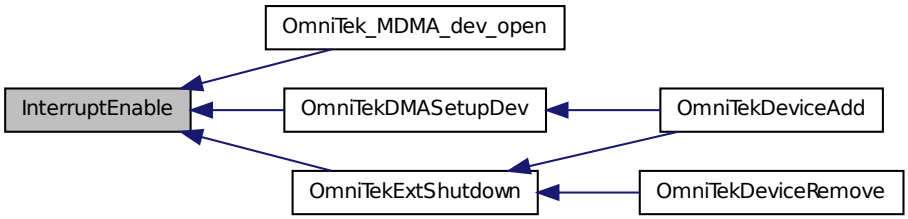


6.20.1.10 void InterruptEnable (POMNITEK_INTERFACE_EXTENSION *pExt*, u32 *Interrupt*, bool *Enable*)

Here is the call graph for this function:

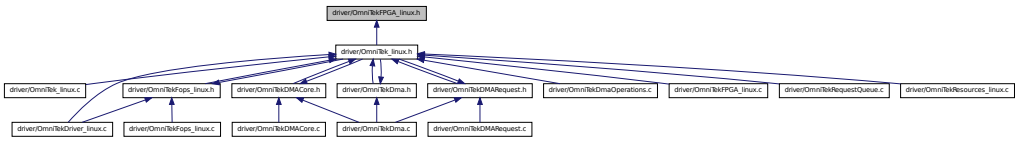


Here is the caller graph for this function:



6.21 driver/OmniTekFPGA_linux.h File Reference

This graph shows which files directly or indirectly include this file:



Functions

- void FPGAInit (Resource *pResource)

  Initialise the FPGA resource.

- void FPGADelete (Resource *pResource)

delete the FPGA resource

- void InterruptEnable (struct _OMNITEK_INTERFACE_EXTENSION *pExt, u32 Interrupt, bool Enable)

  Enable interrupts for the device.

- u32 GetInterruptStatus (struct _OMNITEK_INTERFACE_EXTENSION *pExt)

  Get Interrupt status.

- u32 FPGAGetStandard (struct _Resource *pResource)

  Get FPGA video standard.

- int FPGAControl (struct _Resource *pFPGA, struct _ResourceInfo *pResourceInfo)

  deal with control messages for the FPGA resource

- u64 FPGAReadTime (struct _Resource *pFPGA)

  Read time from FGPA.

- void FPGAGetTime (struct _Resource *pFPGA, u64 *Count)

  Get time from FGPA.

6.21.1 Function Documentation

6.21.1.1 int FPGAControl (struct _Resource * *pFPGA*, struct _ResourceInfo * *pResourceInfo*)

deal with control messages for the FPGA resource

Returns

0 on success or error code

Parameters

pFPGA Pointer to the FPGA resource

pResourceInfo Additional information for command

6.21.1.2 void FPGADelete (Resource * *pResource*)

delete the FPGA resource

Parameters

pResource Pointer to (FPGA) resource to delete

6.21.1.3 u32 FPGAGetStandard (struct _Resource * *pResource*)

Get FPGA video standard.

Returns

the current video standard (output)

Parameters

pResource Pointer to the FPGA resource

6.21.1.4 void FPGAGetTime (struct _Resource * *pFPGA*, u64 * *Count*)

Get time from FGPA.

Parameters

pFPGA Pointer to the FPGA resource

Count Pointer to value to store time in

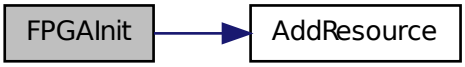
6.21.1.5 void FPGAInit (Resource * *pResource*)

Initialise the FPGA resource.

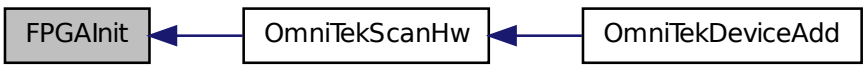
Parameters

pResource Pointer to (FPGA) resource to initialise

Here is the call graph for this function:



Here is the caller graph for this function:



6.21.1.6 u64 FPGAReadTime (struct _Resource * *pFPGA*)

Read time from FGPA.

Returns

64 bit time value from FPGA

Parameters

pFPGA Pointer to the FPGA resource

6.21.1.7 u32 GetInterruptStatus (struct _OMNITEK_INTERFACE_EXTENSION * *pExt*)

Get Interrupt status.

Parameters

pExt Pointer to the device extension

Returns

the enabled interrupt bitmask

6.21.1.8 void InterruptEnable (struct _OMNITEK_INTERFACE_EXTENSION * *pExt,* u32 *Interrupt,* bool *Enable*)

Enable interrupts for the device.

Parameters

pExt Pointer to the device extension

Interrupt Bitmask for interrupts to enable

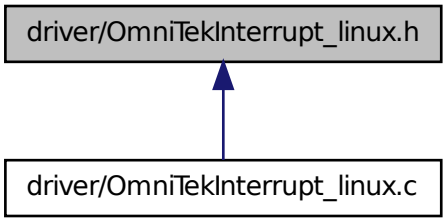
Enable Enable or disable interrupts

6.22 driver/OmniTekInterrupt_linux.c File Reference

```
#include <linux/interrupt.h>
#include "OmniTekInterrupt_linux.h"
```

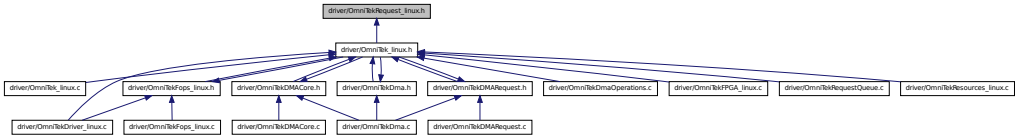
6.23 driver/OmniTekInterrupt_linux.h File Reference

This graph shows which files directly or indirectly include this file:



6.24 driver/OmniTekRequest_linux.h File Reference

This graph shows which files directly or indirectly include this file:



Data Structures

- struct _OmniTekKernelRequest
- struct _OmniTekUserRequest

Typedefs

- typedef enum _RequestStatus RequestStatus
- typedef struct _OmniTekKernelRequest OmniTekKernelRequest
- typedef struct _OmniTekKernelRequest * POmniTekKernelRequest
- typedef struct _OmniTekUserRequest OmniTekUserRequest
- typedef struct _OmniTekUserRequest * POmniTekUserRequest

Enumerations

- enum _RequestStatus {

  OMNITEK_REQUEST_UNINITIALIZED = 0, OMNITEK_REQUEST_INITIALIZED, OMNITEK_REQUEST_USER_KERNEL_COPIED, OMNITEK_REQUEST_PENDING,

OMNITEK_REQUEST_PROCESSED, OMNITEK_REQUEST_KERNEL_USER_COPIED, OMNITEK_REQUEST_COMPLETE, OMNITEK_REQUEST_CANCELLED = -1 }

This describes the current status of a request.

6.24.1 Typedef Documentation

6.24.1.1 typedef struct _OmniTekKernelRequest OmniTekKernelRequest

6.24.1.2 typedef struct _OmniTekUserRequest OmniTekUserRequest

6.24.1.3 typedef struct _OmniTekKernelRequest * POmniTekKernelRequest

6.24.1.4 typedef struct _OmniTekUserRequest * POmniTekUserRequest

6.24.1.5 typedef enum _RequestStatus RequestStatus

6.24.2 Enumeration Type Documentation

6.24.2.1 enum _RequestStatus

This describes the current status of a request.

Enumerator:

OMNITEK_REQUEST_UNINITIALIZED Request is uninitialized

OMNITEK_REQUEST_INITIALIZED Request has been initialized - user space pointers are set

OMNITEK_REQUEST_USER_KERNEL_COPIED Request has been received by kernel and user data copied to kernel space

OMNITEK_REQUEST_PENDING Request is pending (e.g. DMA transfer)

OMNITEK_REQUEST_PROCESSED Request has been processed (e.g. DMA complete or function call complete)

OMNITEK_REQUEST_KERNEL_USER_COPIED Data has been copied from kernel output buffer to user output buffer

OMNITEK_REQUEST_COMPLETE Request has been completed (once returned to user it is no longer valid)

OMNITEK_REQUEST_CANCELLED Request has been cancelled (once returned to user it is no longer valid)

6.25 driver/OmniTekRequestQueue.c File Reference

```
#include "OmniTek_linux.h"
#include "OmniTekRequestQueue.h"
```

Functions

- void OmniTekRequestQueueInit (struct _OmniTekRequestQueue *pQueue, char *Name)

Initialize a request queue structure.

- int OmniTekRequestQueueAddRequest (struct _OmniTekRequestQueue *pQueue, struct _OmniTekRequestQueueObject *pRequest)

  Add Request to a queue.

- int OmniTekRequestQueueRemoveRequest (struct _OmniTekRequestQueue *pQueue, struct _OmniTekRequestQueueObject *pRequest)

  Remove Request to a queue.

- int OmniTekRequestQueueMoveRequest (struct _OmniTekRequestQueue *pCurrentQueue, struct _OmniTekRequestQueue *pNewQueue, struct _OmniTekRequestQueueObject *pRequest)

  Move Request between queues.

- int OmniTekRequestQueueContains (struct _OmniTekRequestQueue *pQueue, struct _OmniTekRequestQueueObject *pRequest)

  Does this queue contain a specified request.

- int OmniTekRequestQueueIsEmpty (struct _OmniTekRequestQueue *pQueue)

  Is this request queue empty.

- int OmniTekRequestQueueSize (struct _OmniTekRequestQueue *pQueue)

  How many entries in this queue.

- int OmniTekRequestQueueNext (struct _OmniTekRequestQueue *pQueue, struct _OmniTekRequestQueueObject **ppRequest)

  Get the next request from this queue.

- int OmniTekRequestQueueInitRequest (struct _OmniTekRequestQueueObject *pRequest)

  Initialize a new request.

6.25.1 Function Documentation

6.25.1.1 int OmniTekRequestQueueAddRequest (struct _OmniTekRequestQueue * *pQueue*, struct _OmniTekRequestQueueObject * *pRequest*)

Add Request to a queue.

Parameters

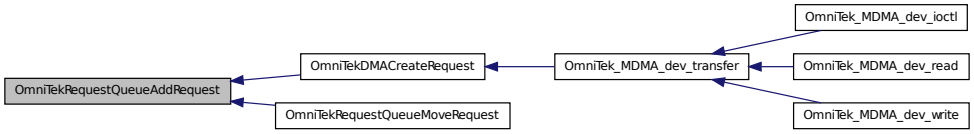
[in] ***pQueue*** Request Queue to add to

[in] ***pRequest*** Request to add to queue

Returns

0 on success or error code

Here is the caller graph for this function:



6.25.1.2 int OmniTekRequestQueueContains (struct _OmniTekRequestQueue * *pQueue*, struct _OmniTekRequestQueueObject * *pRequest*)

Does this queue contain a specified request.

Parameters

[in] ***pQueue*** Request Queue to look in

[in] ***pRequest*** to look for

Returns

0 if request is in queue or error code

6.25.1.3 void OmniTekRequestQueueInit (struct _OmniTekRequestQueue * *pQueue*, char * *Name*)

Initialize a request queue structure.

Parameters

[in] ***pQueue*** pointer to queue struct to initialize

[in] ***Name*** for queue (will default if left NULL)

Here is the caller graph for this function:

6.25.1.4 int OmniTekRequestQueueInitRequest (struct _OmniTekRequestQueueObject * *pRequest*)

Initialize a new request.

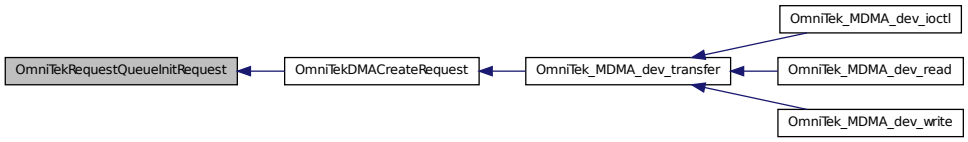
Parameters

[in] ***pRequest*** pointer to request struct to initialize

Returns

0 on success

Here is the caller graph for this function:



6.25.1.5 int OmniTekRequestQueueIsEmpty (struct _OmniTekRequestQueue * *pQueue*)

Is this request queue empty.

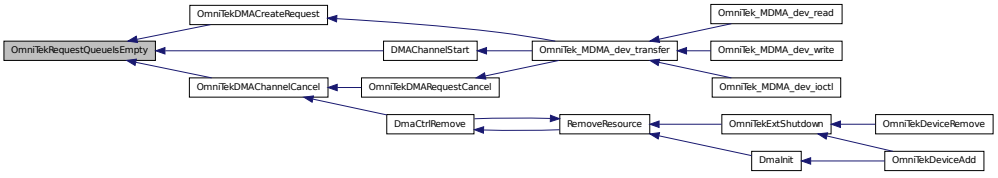
Parameters

[in] ***pQueue*** Request Queue to check

Returns

0 if empty, positive integer otherwise

Here is the caller graph for this function:



6.25.1.6 int OmniTekRequestQueueMoveRequest (struct _OmniTekRequestQueue * *pCurrentQueue*, struct _OmniTekRequestQueue * *pNewQueue*, struct _OmniTekRequestQueueObject * *pRequest*)

Move Request between queues.

Parameters

[in] ***pCurrentQueue*** Request Queue to remove from

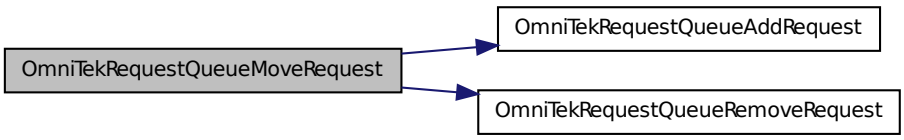
[in] ***pNewQueue*** Request Queue to add to

[in] ***pRequest*** Request to move

Returns

0 on success or error code

Here is the call graph for this function:



6.25.1.7 int OmniTekRequestQueueNext (struct _OmniTekRequestQueue * *pQueue*, struct _OmniTekRequestQueueObject ** *ppRequest*)

Get the next request from this queue.

Parameters

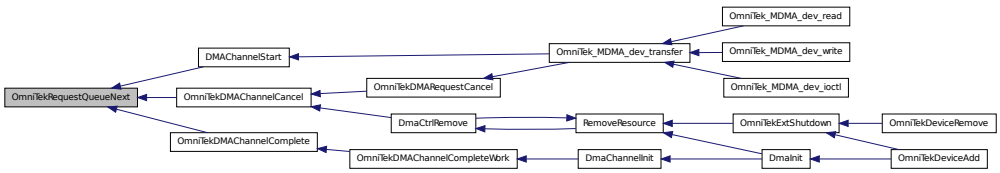
[in] ***pQueue*** Request Queue to retrieve request from

[in] ***ppRequest*** pointer to request struct pointer that will be set to the address of the next request

Returns

0 if succesful or error code if queue is empty

Here is the caller graph for this function:



6.25.1.8 int OmniTekRequestQueueRemoveRequest (struct _OmniTekRequestQueue * *pQueue*, struct _OmniTekRequestQueueObject * *pRequest*)

Remove Request to a queue.

Parameters

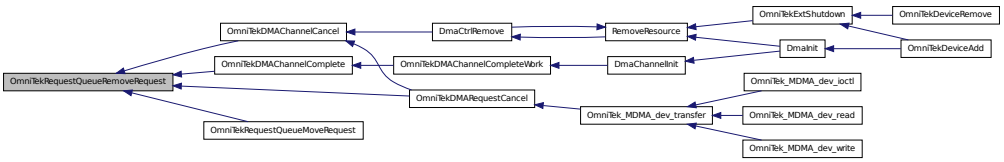
[in] ***pQueue*** Request Queue to remove from

[in] ***pRequest*** Request to remove from queue

Returns

0 on success or error code

Here is the caller graph for this function:



6.25.1.9 int OmniTekRequestQueueSize (struct _OmniTekRequestQueue * *pQueue*)

How many entries in this queue.

Parameters

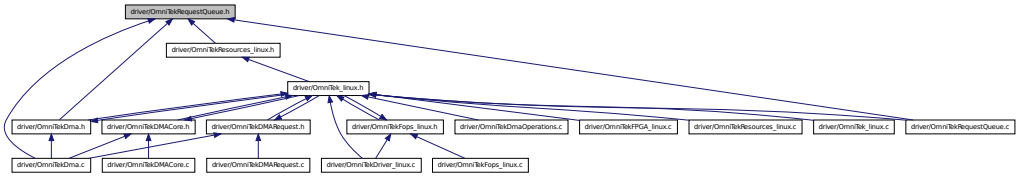
[in] ***pQueue*** Request Queue to check

Returns

number of entries in queue

6.26 driver/OmniTekRequestQueue.h File Reference

This graph shows which files directly or indirectly include this file:



Data Structures

- struct _OmniTekRequestQueue
- struct _OmniTekRequestQueueObject

Typedefs

- typedef struct _OmniTekRequestQueue OmniTekRequestQueue
- typedef struct _OmniTekRequestQueue * POmniTekRequestQueue
- typedef struct _OmniTekRequestQueueObject OmniTekRequestQueueObject
- typedef struct _OmniTekRequestQueueObject * POmniTekRequestQueueObject

Functions

- void OmniTekRequestQueueInit (struct _OmniTekRequestQueue *pQueue, char *Name)

  Initialize a request queue structure.

- int OmniTekRequestQueueAddRequest (struct _OmniTekRequestQueue *pQueue, struct _OmniTekRequestQueueObject *pRequest)

  Add Request to a queue.

- int OmniTekRequestQueueRemoveRequest (struct _OmniTekRequestQueue *pQueue, struct _OmniTekRequestQueueObject *pRequest)

  Remove Request to a queue.

- int OmniTekRequestQueueMoveRequest (struct _OmniTekRequestQueue *pCurrentQueue, struct _OmniTekRequestQueue *pNewQueue, struct _OmniTekRequestQueueObject *pRequest)

  Move Request between queues.

- int OmniTekRequestQueueContains (struct _OmniTekRequestQueue *pQueue, struct _OmniTekRequestQueueObject *pRequest)

  Does this queue contain a specified request.

- int OmniTekRequestQueueIsEmpty (struct _OmniTekRequestQueue *pQueue)

  Is this request queue empty.

- int OmniTekRequestQueueSize (struct _OmniTekRequestQueue *pQueue)

  How many entries in this queue.

- int OmniTekRequestQueueNext (struct _OmniTekRequestQueue *pQueue, struct _OmniTekRequestQueueObject **ppRequest)

  Get the next request from this queue.

- int OmniTekRequestQueueInitRequest (struct _OmniTekRequestQueueObject *pRequest)

  Initialize a new request.

6.26.1 Typedef Documentation

6.26.1.1 typedef struct _OmniTekRequestQueue OmniTekRequestQueue

6.26.1.2 typedef struct _OmniTekRequestQueueObject OmniTekRequestQueueObject

6.26.1.3 typedef struct _OmniTekRequestQueue * POmniTekRequestQueue

6.26.1.4 typedef struct _OmniTekRequestQueueObject * POmniTekRequestQueueObject

6.26.2 Function Documentation

6.26.2.1 int OmniTekRequestQueueAddRequest (struct _OmniTekRequestQueue * *pQueue*, struct _OmniTekRequestQueueObject * *pRequest*)

Add Request to a queue.

Parameters

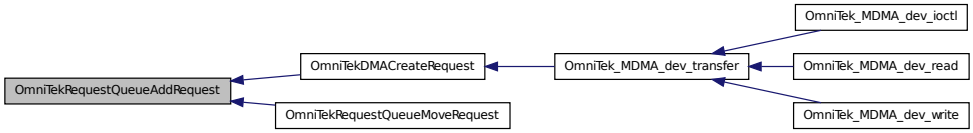
[in] ***pQueue*** Request Queue to add to

[in] ***pRequest*** Request to add to queue

Returns

0 on success or error code

Here is the caller graph for this function:



6.26.2.2 int OmniTekRequestQueueContains (struct _OmniTekRequestQueue * *pQueue*, struct _OmniTekRequestQueueObject * *pRequest*)

Does this queue contain a specified request.

Parameters

[in] ***pQueue*** Request Queue to look in

[in] ***pRequest*** to look for

Returns

0 if request is in queue or error code

6.26.2.3 void OmniTekRequestQueueInit (struct _OmniTekRequestQueue * *pQueue*, char * *Name*)

Initialize a request queue structure.

Parameters

[in] ***pQueue*** pointer to queue struct to initialize

[in] ***Name*** for queue (will default if left NULL)

Here is the caller graph for this function:

6.26.2.4 int OmniTekRequestQueueInitRequest (struct _OmniTekRequestQueueObject * *pRequest*)

Initialize a new request.

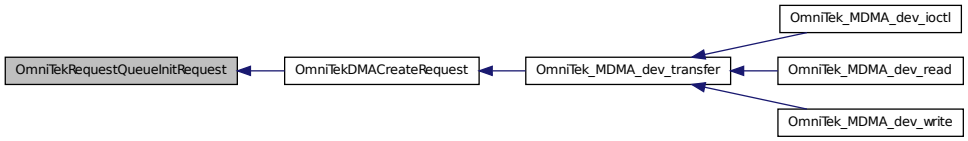
Parameters

[in] ***pRequest*** pointer to request struct to initialize

Returns

0 on success

Here is the caller graph for this function:



6.26.2.5 int OmniTekRequestQueueIsEmpty (struct _OmniTekRequestQueue * *pQueue*)

Is this request queue empty.

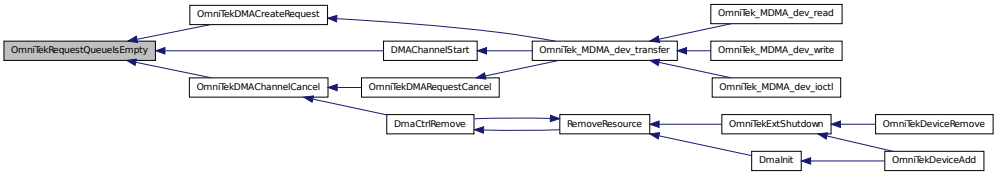
Parameters

[in] ***pQueue*** Request Queue to check

Returns

0 if empty, positive integer otherwise

Here is the caller graph for this function:



6.26.2.6 int OmniTekRequestQueueMoveRequest (struct _OmniTekRequestQueue * *pCurrentQueue,* struct _OmniTekRequestQueue * *pNewQueue,* struct _OmniTekRequestQueueObject * *pRequest*)

Move Request between queues.

Parameters

[in] ***pCurrentQueue*** Request Queue to remove from

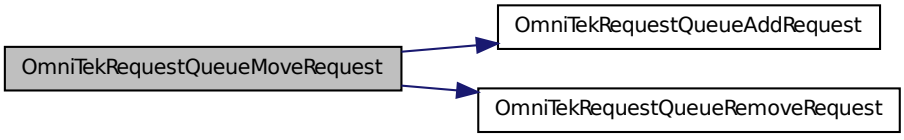
[in] ***pNewQueue*** Request Queue to add to

[in] ***pRequest*** Request to move

Returns

0 on success or error code

Here is the call graph for this function:



6.26.2.7 int OmniTekRequestQueueNext (struct _OmniTekRequestQueue * *pQueue,* struct _OmniTekRequestQueueObject ** *ppRequest*)

Get the next request from this queue.

Parameters

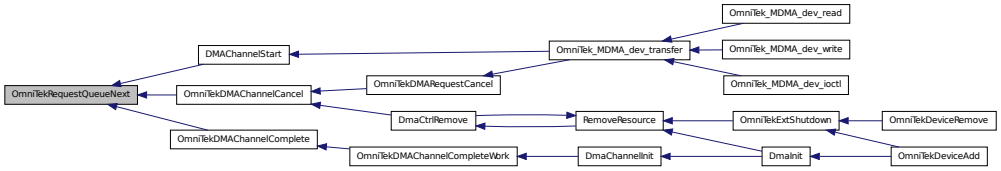
[in] ***pQueue*** Request Queue to retrieve request from

[in] ***ppRequest*** pointer to request struct pointer that will be set to the address of the next request

Returns

0 if succesful or error code if queue is empty

Here is the caller graph for this function:



6.26.2.8 int OmniTekRequestQueueRemoveRequest (struct _OmniTekRequestQueue * *pQueue*, struct _OmniTekRequestQueueObject * *pRequest*)

Remove Request to a queue.

Parameters

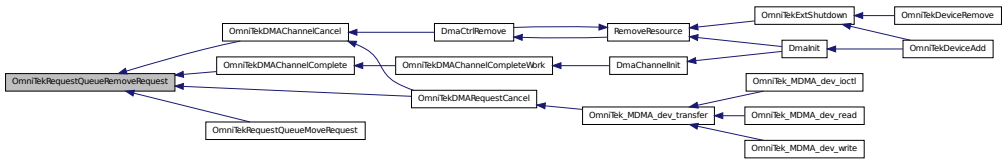
[in] ***pQueue*** Request Queue to remove from

[in] ***pRequest*** Request to remove from queue

Returns

0 on success or error code

Here is the caller graph for this function:



6.26.2.9 int OmniTekRequestQueueSize (struct _OmniTekRequestQueue * *pQueue*)

How many entries in this queue.

Parameters

[in] ***pQueue*** Request Queue to check

Returns

number of entries in queue

6.27 driver/OmniTekResources_linux.c File Reference

```
#include "OmniTek_linux.h"
#include "../include/OmniTekIoctl_linux.h"
```

Functions

- Resource * AddResource (struct _OMNITEK_INTERFACE_EXTENSION *pExt, ResourceType Type, u8 Bar, u32 RegOffset)

  Register a resource.

- int RemoveResource (PResource pResource)

  Remove a resource.

- static int ResourceItemRelease (u32 SessionId, PResource pResource)
- int ReleaseResource (struct _OMNITEK_INTERFACE_EXTENSION *pExt, u32 SessionId, PResource pResource)

  Release a reference to a resource.

- static PResource ResourceFind (struct _OMNITEK_INTERFACE_EXTENSION *pExt, ResourceType Type, u32 Identifier)
- bool ResourceCheck (struct _OMNITEK_INTERFACE_EXTENSION *pExt, PResource pResource)

  Check a resource.

- PResource DmaChannelFind (struct _OMNITEK_INTERFACE_EXTENSION *pExt, u32 Identifier)

  Find a DMA Channel by number.

6.27.1 Function Documentation

6.27.1.1 Resource* AddResource (struct _OMNITEK_INTERFACE_EXTENSION * *pExt*, ResourceType *Type*, u8 *Bar*, u32 *RegOffset*)

Register a resource.

Called to register a resource during hardware scanning

Parameters

pExt Pointer to extension to add resource to

Type Type of resource to add

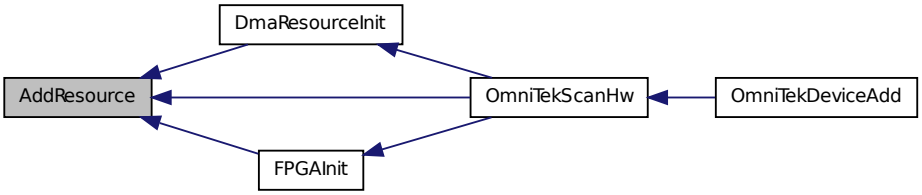
Bar Resource BAR number

RegOffset Offset of first register for resource

Returns

pointer to the resource struct

Here is the caller graph for this function:



6.27.1.2 PResource DmaChannelFind (struct _OMNITEK_INTERFACE_EXTENSION * *pExt,* u32 *Identifier*)

Find a DMA Channel by number.

Parameters

[in] ***pExt*** Device extension to find channel in

[in] ***Identifier*** Number of channel to find

Returns

Pointer to DMA Channel resource

6.27.1.3 int ReleaseResource (struct _OMNITEK_INTERFACE_EXTENSION * *pExt,* u32 *SessionId,* PResource *pResource*)

Release a reference to a resource.

The specified resource reference is released and the reference count is decremented

Parameters

pExt Pointer to extension to lock frame for

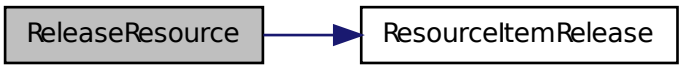
SessionId ID for session

pResource Resource to release

Returns

0 on success or error code

Here is the call graph for this function:



6.27.1.4 int RemoveResource (PResource *pResource*)

Remove a resource.

Called to remove a resource during driver shutdown

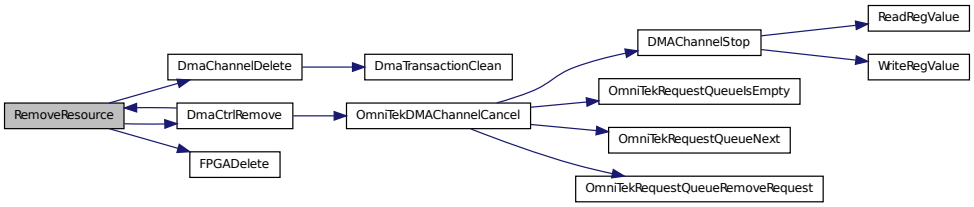
Parameters

pResource Pointer to resource to be removed

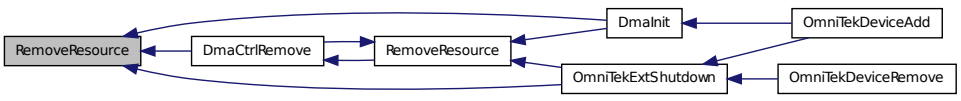
Returns

0 on success or error code

Here is the call graph for this function:



Here is the caller graph for this function:



6.27.1.5 bool ResourceCheck (struct _OMNITEK_INTERFACE_EXTENSION * *pExt*, PResource *pResource*)

Check a resource.

Confirms that this resource is available on the specified extension

Parameters

pExt Pointer to extension with resource to check

pResource Resource to check

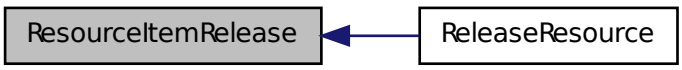
Returns

true if resource belongs to the supplied extension

6.27.1.6 static PResource ResourceFind (struct _OMNITEK_INTERFACE_EXTENSION * *pExt*, ResourceType *Type*, u32 *Identifier*) [static]

6.27.1.7 static int ResourceItemRelease (u32 *SessionId*, PResource *pResource*) [static]

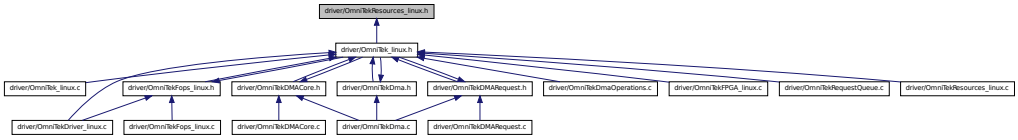
Here is the caller graph for this function:



6.28 driver/OmniTekResources_linux.h File Reference

```
#include <linux/workqueue.h>
#include "../include/OmniTekIoctl_linux.h"
#include "OmniTekRequestQueue.h"
```

This graph shows which files directly or indirectly include this file:



Data Structures

- struct _GeneralCtrl

  Data structure for PCIE BAR.

- struct _ResourceVersion

  Version details of a device resource.

- struct _FPGACtrl

  FPGA Control Resource.

- struct _DmaSglBuffer

  DMA Scatter Gather List buffer.

- struct _DmaSglBuffer::_Allocated
- struct _DmaChannel

  Data structure containing details of a DMA Channel Resource.

- struct _DmaCtrl

  DMA Control Resource.

- struct _Resource

  Generic resource Details.

- union _Resource::_ResourceExtension

Typedefs

- typedef struct _GeneralCtrl GeneralCtrl
- typedef struct _ResourceVersion ResourceVersion
- typedef struct _FPGACtrl FPGACtrl
- typedef struct _DmaSglBuffer DmaSglBuffer
- typedef void(OmnitekDmaInterruptComplete)(void *, u32)

  Called when a DMA Interrupt completes.

- typedef struct _DmaChannel DmaChannel
- typedef struct _DmaChannel * PDmaChannel
- typedef struct _DmaCtrl DmaCtrl
- typedef struct _DmaCtrl * PDmaCtrl
- typedef struct _Resource Resource
- typedef struct _Resource * PResource

Functions

- PResource AddResource (struct _OMNITEK_INTERFACE_EXTENSION *pExt, ResourceType Type, u8 Bar, u32 RegOffset)

  Register a resource.

- int RemoveResource (PResource pResource)

Remove a resource.

- void DmaResourceInit (PResource pResource)

  Initialise DMA Resource.

- int ResourceRegisterWatchdog (struct _OMNITEK_INTERFACE_EXTENSION *pExt, u32 Id, POmniTekKernelRequest pRequest)

  Register watchdog for resource.

- int ResourceControl (struct _OMNITEK_INTERFACE_EXTENSION *pExt, struct _OmnitekKernelRequest *pRequest)

  Send control command to the resource.

- bool ResourceCheck (struct _OMNITEK_INTERFACE_EXTENSION *pExt, PResource pResource)

  Check a resource.

- int ReleaseResource (struct _OMNITEK_INTERFACE_EXTENSION *pExt, u32 SessionId, PResource pResource)

  Release a reference to a resource.

- PResource DmaChannelFind (struct _OMNITEK_INTERFACE_EXTENSION *pExt, u32 Identifier)

  Find a DMA Channel by number.

6.28.1 Typedef Documentation

6.28.1.1 typedef struct _DmaChannel DmaChannel

6.28.1.2 typedef struct _DmaCtrl DmaCtrl

6.28.1.3 typedef struct _DmaSglBuffer DmaSglBuffer

6.28.1.4 typedef struct _FPGACtrl FPGACtrl

6.28.1.5 typedef struct _GeneralCtrl GeneralCtrl

6.28.1.6 typedef void(OmnitekDmaInterruptComplete)(void *, u32)

Called when a DMA Interrupt completes.

6.28.1.7 typedef struct _DmaChannel * PDmaChannel

6.28.1.8 typedef struct _DmaCtrl * PDmaCtrl

6.28.1.9 typedef struct _Resource * PResource

6.28.1.10 typedef struct _Resource Resource

6.28.1.11 typedef struct _ResourceVersion ResourceVersion

6.28.2 Function Documentation

6.28.2.1 PResource AddResource (struct _OMNITEK_INTERFACE_EXTENSION * *pExt*, ResourceType *Type*, u8 *Bar*, u32 *RegOffset*)

Register a resource.

Called to register a resource during hardware scanning

Parameters

pExt Pointer to extension to add resource to

Type Type of resource to add

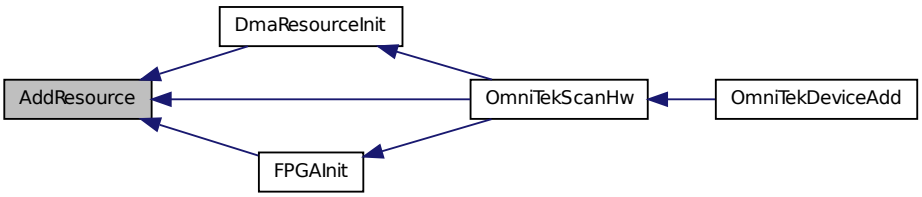
Bar Resource BAR number

RegOffset Offset of first register for resource

Returns

pointer to the resource struct

Here is the caller graph for this function:



6.28.2.2 PResource DmaChannelFind (struct _OMNITEK_INTERFACE_EXTENSION * *pExt*, u32 *Identifier*)

Find a DMA Channel by number.

Parameters

`[in]` ***pExt*** Device extension to find channel in

[in] ***Identifier*** Number of channel to find

Returns

Pointer to DMA Channel resource

6.28.2.3 void DmaResourceInit (PResource *pResource*)

Initialise DMA Resource.

Parameters

pResource DMA) resource to initialise

Called to initialise the DMA resource and set up individual channels

6.28.2.4 int ReleaseResource (struct _OMNITEK_INTERFACE_EXTENSION * *pExt*, u32 *SessionId*, PResource *pResource*)

Release a reference to a resource.

The specified resource reference is released and the reference count is decremented

Parameters

pExt Pointer to extension to lock frame for

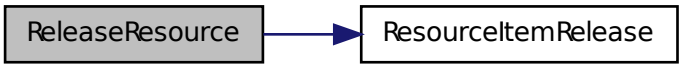
SessionId ID for session

pResource Resource to release

Returns

0 on success or error code

Here is the call graph for this function:



6.28.2.5 int RemoveResource (PResource *pResource*)

Remove a resource.

Called to remove a resource during driver shutdown

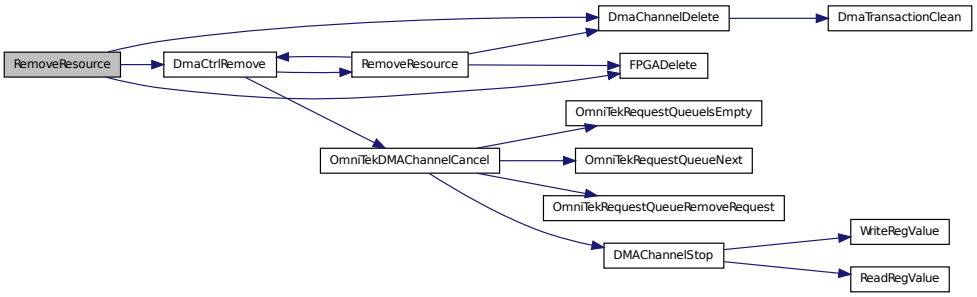
Parameters

pResource Pointer to resource to be removed

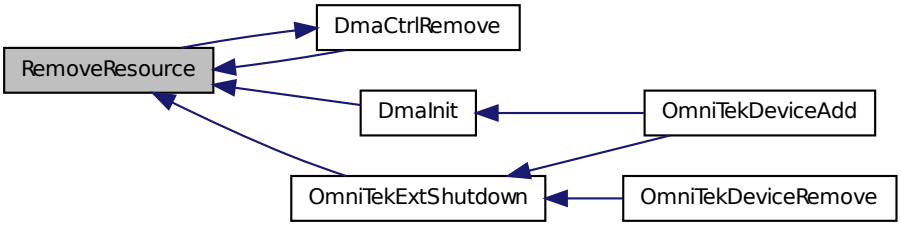
Returns

0 on success or error code

Here is the call graph for this function:



Here is the caller graph for this function:



6.28.2.6 bool ResourceCheck (struct _OMNITEK_INTERFACE_EXTENSION * *pExt*, PResource *pResource*)

Check a resource.

Confirms that this resource is available on the specified extension

Parameters

pExt Pointer to extension with resource to check
pResource Resource to check

Returns

true if resource belongs to the supplied extension

6.28.2.7 int ResourceControl (struct _OMNITEK_INTERFACE_EXTENSION * *pExt,* struct _OmnitekKernelRequest * *pRequest*)

Send control command to the resource.

Parameters

pExt Pointer to extension with resource to control

pRequest Details for command

Returns

0 on success or error code

6.28.2.8 int ResourceRegisterWatchdog (struct _OMNITEK_INTERFACE_EXTENSION * *pExt,* u32 *Id,* POmniTekKernelRequest *pRequest*)

Register watchdog for resource.

Parameters

pExt Pointer to extension to register watchdog for

Id ID Of session to watchdog

pRequest Details for command

Returns

0 on success or error code

Index