

# OmniTek NetViz Driver

Generated by Doxygen 1.7.1

Fri Oct 29 2010 16:07:11



# Contents

<b>1</b>	<b>OmniTek NetViz DMA Driver</b>	<b>1</b>
1.1	Introduction . . . . .	1
1.1.1	Copyright . . . . .	1
1.2	Guides . . . . .	1
<b>2</b>	<b>NetViz Driver DMA API Documentation</b>	<b>3</b>
2.1	Introduction . . . . .	3
2.2	Implementation . . . . .	3
2.2.1	Request Creation . . . . .	3
2.2.2	Completion . . . . .	4
2.2.3	Cancellation . . . . .	4
<b>3</b>	<b>Data Structure Index</b>	<b>5</b>
3.1	Data Structures . . . . .	5
<b>4</b>	<b>File Index</b>	<b>7</b>
4.1	File List . . . . .	7
<b>5</b>	<b>Data Structure Documentation</b>	<b>9</b>
5.1	_DmaSglBuffer::_Allocated Struct Reference . . . . .	9
5.1.1	Detailed Description . . . . .	9
5.1.2	Field Documentation . . . . .	9
5.1.2.1	Free . . . . .	9
5.1.2.2	Memory . . . . .	9
5.1.2.3	Size . . . . .	9
5.2	_OMNITEK_INTERFACE_EXTENSION::_bar_registers Struct Reference . . . . .	10
5.2.1	Detailed Description . . . . .	10
5.2.2	Field Documentation . . . . .	10
5.2.2.1	num_regs . . . . .	10
5.2.2.2	regs . . . . .	10

5.3	<a href="#">_dma_interrupt_info::chan_int_counts Struct Reference</a>	10
5.3.1	Field Documentation	10
5.3.1.1	<a href="#">n_event_ints</a>	10
5.3.1.2	<a href="#">n_sg_ints</a>	10
5.4	<a href="#">_dma_interrupt_info Struct Reference</a>	11
5.4.1	Field Documentation	12
5.4.1.1	<a href="#">chan_done</a>	12
5.4.1.2	<a href="#">chan_int_counts</a>	12
5.4.1.3	<a href="#">chan_interrupts</a>	12
5.4.1.4	<a href="#">sem</a>	12
5.5	<a href="#">_DmaChannel Struct Reference</a>	12
5.5.1	Detailed Description	14
5.5.2	Field Documentation	14
5.5.2.1	<a href="#">Active</a>	14
5.5.2.2	<a href="#">DMA_64BIT_LADR</a>	14
5.5.2.3	<a href="#">DMA_64BIT_PADR</a>	14
5.5.2.4	<a href="#">FDMA_Enabled</a>	14
5.5.2.5	<a href="#">FDMA_Read</a>	14
5.5.2.6	<a href="#">FDMA_Write</a>	14
5.5.2.7	<a href="#">Index</a>	15
5.5.2.8	<a href="#">ISRWork</a>	15
5.5.2.9	<a href="#">Label</a>	15
5.5.2.10	<a href="#">Object</a>	15
5.5.2.11	<a href="#">Pending</a>	15
5.5.2.12	<a href="#">Running</a>	15
5.5.2.13	<a href="#">SglBuffer</a>	15
5.5.2.14	<a href="#">SpinLock</a>	15
5.6	<a href="#">_OmniTekDmaTransactionContext::_DMACoreInfo Struct Reference</a>	15
5.6.1	Field Documentation	16
5.6.1.1	<a href="#">first_page</a>	16
5.6.1.2	<a href="#">last_page</a>	16
5.6.1.3	<a href="#">num_pages</a>	16
5.6.1.4	<a href="#">offset</a>	16
5.6.1.5	<a href="#">pages</a>	16
5.6.1.6	<a href="#">sgt</a>	16
5.7	<a href="#">_DmaCtrl Struct Reference</a>	16

5.7.1	Detailed Description	17
5.7.2	Field Documentation	17
5.7.2.1	Channels	17
5.7.2.2	DMA_64BIT_LADR	17
5.7.2.3	DMA_64BIT_PADR	17
5.7.2.4	DMA_Wait_Queue	17
5.7.2.5	DMA_Work_Queue	17
5.7.2.6	DmaInterrupts	17
5.7.2.7	nChannels	17
5.7.2.8	nFDMABoth	17
5.7.2.9	nFDMARead	17
5.7.2.10	nFDMAWrite	18
5.7.2.11	nMDMA	18
5.8	_DmaSglBuffer Struct Reference	18
5.8.1	Detailed Description	19
5.8.2	Field Documentation	19
5.8.2.1	Allocated	19
5.8.2.2	CommonBuffer	19
5.8.2.3	DMA_Handle	19
5.8.2.4	SpinLock	19
5.9	_FPGACtrl Struct Reference	19
5.9.1	Detailed Description	20
5.9.2	Field Documentation	20
5.9.2.1	FPGAType	20
5.9.2.2	Version	20
5.10	_GeneralCtrl Struct Reference	21
5.10.1	Detailed Description	21
5.10.2	Field Documentation	21
5.10.2.1	Bar	21
5.10.2.2	Initialised	21
5.10.2.3	RegisterOffset	21
5.11	_InterruptData Struct Reference	21
5.11.1	Detailed Description	23
5.11.2	Field Documentation	23
5.11.2.1	nInterruptStatus	23
5.11.2.2	pExt	23

5.12	<a href="#">_OmniTek_dev Struct Reference</a>	23
5.12.1	<a href="#">Field Documentation</a>	25
5.12.1.1	<a href="#">cDev</a>	25
5.12.1.2	<a href="#">firstMinor</a>	25
5.12.1.3	<a href="#">major</a>	25
5.12.1.4	<a href="#">nMinors</a>	25
5.12.1.5	<a href="#">Object</a>	25
5.12.1.6	<a href="#">pExt</a>	25
5.12.1.7	<a href="#">Type</a>	25
5.13	<a href="#">_OMNITEK_INTERFACE_EXTENSION Struct Reference</a>	26
5.13.1	<a href="#">Field Documentation</a>	28
5.13.1.1	<a href="#">Device</a>	28
5.13.1.2	<a href="#">irq</a>	28
5.13.1.3	<a href="#">IrqLock</a>	28
5.13.1.4	<a href="#">MemBar</a>	28
5.13.1.5	<a href="#">nBars</a>	28
5.13.1.6	<a href="#">Object</a>	28
5.13.1.7	<a href="#">pDmaCtrl</a>	29
5.13.1.8	<a href="#">pDriver</a>	29
5.13.1.9	<a href="#">pFlashProgrammer</a>	29
5.13.1.10	<a href="#">pFPGA</a>	29
5.13.1.11	<a href="#">RegValue</a>	29
5.13.1.12	<a href="#">Resources</a>	29
5.13.1.13	<a href="#">SpinLock</a>	29
5.14	<a href="#">_OmniTekDmaTransactionContext Struct Reference</a>	29
5.14.1	<a href="#">Field Documentation</a>	31
5.14.1.1	<a href="#">CompleteFunc</a>	31
5.14.1.2	<a href="#">DMACoreInfo</a>	31
5.14.1.3	<a href="#">generic</a>	31
5.14.1.4	<a href="#">kiocb</a>	31
5.14.1.5	<a href="#">pChannel</a>	31
5.14.1.6	<a href="#">QueueObject</a>	31
5.14.1.7	<a href="#">request</a>	31
5.14.1.8	<a href="#">Sgl</a>	31
5.14.1.9	<a href="#">state</a>	31
5.14.1.10	<a href="#">type</a>	32

5.14.1.11 typeInfo . . . . .	32
5.14.1.12 Xfer . . . . .	32
5.15 _OmniTekDriver Struct Reference . . . . .	32
5.15.1 Field Documentation . . . . .	32
5.15.1.1 dev . . . . .	32
5.15.1.2 Devices . . . . .	32
5.15.1.3 Extensions . . . . .	32
5.15.1.4 NextMinor . . . . .	32
5.15.1.5 pci_driver . . . . .	32
5.15.1.6 ResourcePool . . . . .	32
5.16 _OmniTekKernelRequest Struct Reference . . . . .	32
5.16.1 Field Documentation . . . . .	34
5.16.1.1 inBufferSize . . . . .	34
5.16.1.2 outBufferSize . . . . .	34
5.16.1.3 pExt . . . . .	34
5.16.1.4 pInBuffer . . . . .	34
5.16.1.5 pOutBuffer . . . . .	34
5.16.1.6 userRequest . . . . .	34
5.17 _OmniTekRequestQueue Struct Reference . . . . .	34
5.17.1 Field Documentation . . . . .	35
5.17.1.1 Entries . . . . .	35
5.17.1.2 Name . . . . .	35
5.17.1.3 SpinLock . . . . .	35
5.18 _OmniTekRequestQueueObject Struct Reference . . . . .	35
5.18.1 Field Documentation . . . . .	36
5.18.1.1 Object . . . . .	36
5.18.1.2 pCurrentQueue . . . . .	36
5.19 _OmniTekUserRequest Struct Reference . . . . .	36
5.19.1 Field Documentation . . . . .	38
5.19.1.1 inBufferSize . . . . .	38
5.19.1.2 kernelRequest . . . . .	38
5.19.1.3 outBufferSize . . . . .	38
5.19.1.4 pid . . . . .	38
5.19.1.5 pInBuffer . . . . .	38
5.19.1.6 pOutBuffer . . . . .	38
5.20 _PCI_BAR_INFO Struct Reference . . . . .	38

5.20.1 Detailed Description . . . . .	38
5.20.2 Field Documentation . . . . .	39
5.20.2.1 IsIoMapped . . . . .	39
5.20.2.2 Physical . . . . .	39
5.20.2.3 pVa . . . . .	39
5.20.2.4 Size . . . . .	39
5.21 _Resource Struct Reference . . . . .	39
5.21.1 Detailed Description . . . . .	41
5.21.2 Field Documentation . . . . .	41
5.21.2.1 CapVersion . . . . .	41
5.21.2.2 General . . . . .	41
5.21.2.3 LockedBy . . . . .	41
5.21.2.4 NumRegisters . . . . .	41
5.21.2.5 Object . . . . .	41
5.21.2.6 pExt . . . . .	41
5.21.2.7 ReferenceCount . . . . .	41
5.21.2.8 SpinLock . . . . .	41
5.21.2.9 Type . . . . .	41
5.21.2.10 u . . . . .	41
5.22 _Resource::_ResourceExtension Union Reference . . . . .	42
5.22.1 Detailed Description . . . . .	43
5.22.2 Field Documentation . . . . .	43
5.22.2.1 DmaChannel . . . . .	43
5.22.2.2 DmaCtrl . . . . .	43
5.22.2.3 FPGACtrl . . . . .	43
5.23 _ResourceVersion Struct Reference . . . . .	43
5.23.1 Detailed Description . . . . .	43
5.23.2 Field Documentation . . . . .	43
5.23.2.1 Major . . . . .	43
5.23.2.2 Minor . . . . .	44
5.24 _OmniTekDmaTransactionContext::_SglInfo Struct Reference . . . . .	44
5.24.1 Field Documentation . . . . .	44
5.24.1.1 CoherentMem . . . . .	44
5.24.1.2 DmaMem . . . . .	44
5.25 _OmniTekDmaTransactionContext::_XferInfo Struct Reference . . . . .	44
5.25.1 Field Documentation . . . . .	44



5.25.1.1	Buffer	44
5.25.1.2	LocalAddr	45
5.25.1.3	Size	45
5.25.1.4	Write	45
5.26	OmniTek_dev Struct Reference	45
5.26.1	Detailed Description	45
5.27	OmniTekDriver Struct Reference	45
5.27.1	Detailed Description	45
5.28	OmniTekKernelRequest Struct Reference	45
5.29	OmniTekUserRequest Struct Reference	46
<b>6</b>	<b>File Documentation</b>	<b>47</b>
6.1	driver/OmniTekResource_linux.o.d File Reference	47
6.2	driver/OmniTek_debug.h File Reference	47
6.2.1	Define Documentation	48
6.2.1.1	DMA	48
6.2.1.2	DMA_CORE	48
6.2.1.3	DMA_OPS	48
6.2.1.4	DMA_PAGES	48
6.2.1.5	DMA_REQUEST	48
6.2.1.6	FOPS	48
6.2.1.7	GENERAL	48
6.2.1.8	IRQ	48
6.2.1.9	OMNITEK_DEBUG_CATEGORIES	48
6.2.1.10	OmniTekDebug	48
6.2.1.11	REQUEST_QUEUE	48
6.2.1.12	RESOURCES	48
6.3	driver/OmniTek_Driver.mod.c File Reference	48
6.3.1	Function Documentation	49
6.3.1.1	__attribute__	49
6.3.1.2	__attribute__	49
6.3.1.3	__attribute__	49
6.3.1.4	MODULE_ALIAS	49
6.3.1.5	MODULE_ALIAS	49
6.3.1.6	MODULE_ALIAS	49
6.3.1.7	MODULE_ALIAS	49
6.3.1.8	MODULE_ALIAS	49

6.3.1.9	MODULE_INFO	49
6.3.1.10	MODULE_INFO	49
6.4	driver/OmniTek_linux.c File Reference	49
6.4.1	Function Documentation	50
6.4.1.1	GetRegValue	50
6.4.1.2	list_count	51
6.4.1.3	OmniTekExtInit	51
6.4.1.4	OmniTekExtShutdown	51
6.4.1.5	OmniTekGetCapList	52
6.4.1.6	OmniTekScanHw	53
6.4.1.7	ReadRegValue	54
6.4.1.8	WriteRegValue	54
6.5	driver/OmniTek_linux.h File Reference	55
6.5.1	Define Documentation	60
6.5.1.1	MAX_NUM_MEM_BARS	60
6.5.1.2	OMNITEK_DMA_INTERRUPT	60
6.5.1.3	OMNITEK_DMACTRL_INTERRUPT_MASK	60
6.5.1.4	OMNITEK_INTERRUPT_MASK	60
6.5.1.5	PCI_NUM_BARS	60
6.5.1.6	ReadHWValue	60
6.5.1.7	ReadHWValueByte	60
6.5.1.8	STATUS_INVALID_PARAMETER_1	60
6.5.1.9	STATUS_INVALID_PARAMETER_2	60
6.5.1.10	STATUS_INVALID_PARAMETER_3	60
6.5.1.11	STATUS_OMNITEK_ILLEGAL_SESSION_ID	60
6.5.1.12	STATUS_OMNITEK_MEMORY_ERROR	60
6.5.1.13	STATUS_OMNITEK_RESOURCE_COMMAND_ERROR	60
6.5.1.14	STATUS_OMNITEK_RESOURCE_INVALID	60
6.5.1.15	STATUS_OMNITEK_RESOURCE_LOCKED	60
6.5.1.16	WriteHWValue	60
6.5.1.17	WriteHWValueByte	60
6.5.2	Typedef Documentation	60
6.5.2.1	InterruptData	60
6.5.2.2	OMNITEK_INTERFACE_EXTENSION	60
6.5.2.3	OmniTekDriver	60
6.5.2.4	PCI_BAR_INFO	60

6.5.2.5	POMNITEK_INTERFACE_EXTENSION . . . . .	60
6.5.3	Function Documentation . . . . .	60
6.5.3.1	DriverEntry . . . . .	60
6.5.3.2	GetOmniTekDriver . . . . .	61
6.5.3.3	GetRegValue . . . . .	61
6.5.3.4	OmniTekEvtDeviceD0Entry . . . . .	62
6.5.3.5	OmniTekEvtDeviceD0EntryPostInterruptsEnabled . . . . .	62
6.5.3.6	OmniTekEvtDeviceD0Exit . . . . .	63
6.5.3.7	OmniTekEvtDevicePrepareHardware . . . . .	63
6.5.3.8	OmniTekEvtDeviceProbe . . . . .	63
6.5.3.9	OmniTekEvtDeviceReleaseHardware . . . . .	63
6.5.3.10	OmniTekExtInit . . . . .	64
6.5.3.11	OmniTekExtShutdown . . . . .	64
6.5.3.12	OmniTekGetCapList . . . . .	65
6.5.3.13	OmniTekIoctl . . . . .	65
6.5.3.14	OmniTekScanHw . . . . .	66
6.5.3.15	ReadRegValue . . . . .	66
6.5.3.16	WriteRegValue . . . . .	67
6.6	driver/OmniTek_MainPage.h File Reference . . . . .	67
6.7	driver/OmniTekDma.c File Reference . . . . .	67
6.7.1	Define Documentation . . . . .	69
6.7.1.1	TRANSACTION_WAIT_MSECS . . . . .	69
6.7.2	Function Documentation . . . . .	69
6.7.2.1	DECLARE_WAIT_QUEUE_HEAD . . . . .	69
6.7.2.2	DmaChannelBusy . . . . .	69
6.7.2.3	DmaChannelDelete . . . . .	69
6.7.2.4	DmaChannelInit . . . . .	70
6.7.2.5	DmaChannelStop . . . . .	71
6.7.2.6	DmaCtrlRemove . . . . .	71
6.7.2.7	DmaInit . . . . .	72
6.7.2.8	DmaResourceInit . . . . .	73
6.7.2.9	DmaTransactionClean . . . . .	74
6.7.2.10	OmniTek_MDMA_dev_complete . . . . .	74
6.7.2.11	OmniTek_MDMA_dev_ioctl . . . . .	74
6.7.2.12	OmniTek_MDMA_dev_open . . . . .	75
6.7.2.13	OmniTek_MDMA_dev_read . . . . .	75

6.7.2.14	OmniTek_MDMA_dev_release . . . . .	75
6.7.2.15	OmniTek_MDMA_dev_transfer . . . . .	75
6.7.2.16	OmniTek_MDMA_dev_write . . . . .	76
6.7.2.17	OmniTekDMAReleaseDev . . . . .	77
6.7.2.18	OmniTekDMASetupDev . . . . .	77
6.7.3	Variable Documentation . . . . .	78
6.7.3.1	OmniTek_MDMA_dev_fops . . . . .	78
6.7.3.2	OmniTekMDMADev . . . . .	78
6.8	driver/OmniTekDma.h File Reference . . . . .	78
6.8.1	Define Documentation . . . . .	81
6.8.1.1	DMA_CHANNEL . . . . .	81
6.8.1.2	DMA_CHANNEL_BYTES_XFER . . . . .	81
6.8.1.3	DMA_CHANNEL_CSR . . . . .	81
6.8.1.4	DMA_CHANNEL_DPR . . . . .	81
6.8.1.5	DMA_CHANNEL_DPR_HIGH . . . . .	81
6.8.1.6	DMA_CHANNEL_FDMA . . . . .	81
6.8.1.7	DMA_CHANNEL_LADR . . . . .	81
6.8.1.8	DMA_CHANNEL_LADR_HIGH . . . . .	81
6.8.1.9	DMA_CHANNEL_OFFSET . . . . .	81
6.8.1.10	DMA_CHANNEL_PADR . . . . .	81
6.8.1.11	DMA_CHANNEL_PADR_HIGH . . . . .	81
6.8.1.12	DMA_CHANNEL_READ . . . . .	81
6.8.1.13	DMA_CHANNEL_SIZE . . . . .	81
6.8.1.14	DMA_CHANNEL_SIZE_HIGH . . . . .	81
6.8.1.15	DMA_CHANNEL_WRITE . . . . .	81
6.8.1.16	DMA_CTRL_CAP_HEADER . . . . .	81
6.8.1.17	DMA_CTRL_CAP_REG . . . . .	81
6.8.1.18	DMA_CTRL_INTERRUPT_STATUS . . . . .	81
6.8.1.19	DMA_DPR_BIT_DIRECTION_TO_PC . . . . .	81
6.8.1.20	DMA_DPR_BIT_END_OF_CHAIN . . . . .	81
6.8.1.21	DMA_DPR_BIT_INTERRUPT . . . . .	81
6.8.1.22	DMA_FDMA_CHANNEL . . . . .	81
6.8.1.23	DMA_FDMA_TYPE . . . . .	82
6.8.1.24	DMA_MDMA_CHANNEL . . . . .	82
6.8.1.25	DMA_SGL_SIZE . . . . .	82
6.8.1.26	NUM_REGS_PER_DMA_CHANNEL . . . . .	82

6.8.1.27	SGL_ITEM_SIZE . . . . .	82
6.8.2	Typedef Documentation . . . . .	82
6.8.2.1	OmniTekDmaTransactionContext . . . . .	82
6.8.2.2	POmniTekDmaTransactionContext . . . . .	82
6.8.3	Function Documentation . . . . .	82
6.8.3.1	DmaChannelBusy . . . . .	82
6.8.3.2	DmaChannelDelete . . . . .	82
6.8.3.3	DmaChannelStop . . . . .	83
6.8.3.4	DmaCtrlRemove . . . . .	83
6.8.3.5	DmaInit . . . . .	84
6.8.3.6	DmaResourceInit . . . . .	84
6.8.3.7	OmniTek_MDMA_dev_ioctl . . . . .	85
6.8.3.8	OmniTek_MDMA_dev_open . . . . .	85
6.8.3.9	OmniTek_MDMA_dev_read . . . . .	86
6.8.3.10	OmniTek_MDMA_dev_release . . . . .	86
6.8.3.11	OmniTek_MDMA_dev_transfer . . . . .	86
6.8.3.12	OmniTek_MDMA_dev_write . . . . .	87
6.8.3.13	OmniTekDMAReleaseDev . . . . .	88
6.8.3.14	OmniTekDMASetupDev . . . . .	88
6.9	driver/OmniTekDMACore.c File Reference . . . . .	89
6.9.1	Function Documentation . . . . .	89
6.9.1.1	DMACHannelStart . . . . .	89
6.9.1.2	DMACHannelStop . . . . .	90
6.9.1.3	DMAFinishTransaction . . . . .	91
6.9.1.4	DMAGetUserPages . . . . .	92
6.9.1.5	DMAMapSg . . . . .	92
6.9.1.6	DMAMapTable . . . . .	93
6.9.1.7	DMAProgramSgl . . . . .	93
6.9.1.8	DMAStartTransaction . . . . .	94
6.9.1.9	DMAUnMapSg . . . . .	94
6.9.1.10	free_user_pages . . . . .	95
6.9.1.11	getNumPages . . . . .	95
6.10	driver/OmniTekDMACore.h File Reference . . . . .	95
6.10.1	Function Documentation . . . . .	96
6.10.1.1	DMACHannelStart . . . . .	96
6.10.1.2	DMACHannelStop . . . . .	97

6.10.1.3	DMAFinishTransaction	97
6.10.1.4	DMAStartTransaction	98
6.11	driver/OmniTekDmaIsr_linux.c File Reference	99
6.11.1	Define Documentation	99
6.11.1.1	DMA_CHANNEL_INT_BIT_EVENT	99
6.11.1.2	DMA_CHANNEL_INT_BIT_SG	99
6.11.2	Function Documentation	99
6.11.2.1	OmniTekDMAFastIsr	99
6.11.2.2	OmniTekDMASlowIsr	99
6.11.2.3	ReadRegValue	99
6.12	driver/OmniTekDmaOperations.c File Reference	100
6.12.1	Function Documentation	101
6.12.1.1	dma_get_user_pages	101
6.12.1.2	dma_map_sg_init_table_and_chain	101
6.12.1.3	dma_map_sg_pages	101
6.12.1.4	dma_map_test_scatterlist	102
6.13	driver/OmniTekDmaOperations.h File Reference	102
6.13.1	Function Documentation	103
6.13.1.1	dma_get_user_pages	103
6.13.1.2	dma_map_sg_init_table_and_chain	104
6.13.1.3	dma_map_sg_pages	104
6.13.1.4	dma_map_test_scatterlist	105
6.14	driver/OmniTekDMARequest.c File Reference	105
6.14.1	Function Documentation	105
6.14.1.1	OmniTekDMAChannelCancel	105
6.14.1.2	OmniTekDMAChannelComplete	106
6.14.1.3	OmniTekDMAChannelCompleteWork	107
6.14.1.4	OmniTekDMACreateRequest	108
6.14.1.5	OmniTekDMAReleaseRequest	109
6.14.1.6	OmniTekDMARequestCancel	109
6.15	driver/OmniTekDMARequest.h File Reference	110
6.15.1	Function Documentation	111
6.15.1.1	OmniTekDMAChannelCancel	111
6.15.1.2	OmniTekDMAChannelComplete	112
6.15.1.3	OmniTekDMAChannelCompleteWork	112
6.15.1.4	OmniTekDMACreateRequest	113

6.15.1.5	OmniTekDMAReleaseRequest	114
6.15.1.6	OmniTekDMARequestCancel	115
6.16	driver/OmniTekDriver_linux.c File Reference	116
6.16.1	Function Documentation	117
6.16.1.1	DmaChannelISR	117
6.16.1.2	DmaISR	118
6.16.1.3	DmaStatus	118
6.16.1.4	GetNumPciLanes	119
6.16.1.5	GetOmniTekDriver	119
6.16.1.6	MODULE_DEVICE_TABLE	120
6.16.1.7	module_exit	120
6.16.1.8	module_init	120
6.16.1.9	MODULE_LICENSE	120
6.16.1.10	OmniTekDeviceAdd	120
6.16.1.11	OmniTekDeviceRemove	120
6.16.1.12	OmniTekDriver_exit	121
6.16.1.13	OmniTekDriver_init	121
6.16.1.14	OmniTekGetDeviceId	121
6.16.1.15	OmniTekInterrupt	122
6.16.1.16	OmniTekInterruptHandler	122
6.16.1.17	OmniTekRegisterIRQ	123
6.16.1.18	OmniTekUnRegisterIRQ	124
6.16.2	Variable Documentation	125
6.16.2.1	handlerCount	125
6.16.2.2	handlerHandled	125
6.16.2.3	irqPend	125
6.16.2.4	irqTotal	125
6.16.2.5	nInterruptStatus	125
6.16.2.6	omnitek_driver	125
6.17	driver/OmniTekDriver_linux.h File Reference	125
6.17.1	Define Documentation	126
6.17.1.1	IORESOURCE_MEM_64	126
6.17.1.2	USE_IRQ_THREAD	126
6.17.2	Function Documentation	126
6.17.2.1	GetNumPciLanes	126
6.17.2.2	OmniTekDeviceAdd	126

6.17.2.3	OmniTekDeviceRemove . . . . .	127
6.17.2.4	OmniTekGetDeviceId . . . . .	127
6.17.3	Variable Documentation . . . . .	127
6.17.3.1	ids . . . . .	127
6.18	driver/OmniTekFops_linux.c File Reference . . . . .	128
6.18.1	Function Documentation . . . . .	128
6.18.1.1	OmniTek_BAR_dev_ioctl . . . . .	128
6.18.1.2	OmniTek_BAR_dev_open . . . . .	129
6.18.1.3	OmniTek_BAR_dev_release . . . . .	129
6.18.1.4	OmniTekDeviceReleaseDev . . . . .	129
6.18.1.5	OmniTekDeviceSetupDev . . . . .	130
6.18.2	Variable Documentation . . . . .	130
6.18.2.1	OmniTek_BAR_dev_fops . . . . .	130
6.18.2.2	OmniTekBoardev . . . . .	130
6.19	driver/OmniTekFops_linux.h File Reference . . . . .	130
6.19.1	Typedef Documentation . . . . .	132
6.19.1.1	OmniTek_dev . . . . .	132
6.19.1.2	OmniTek_DevTypes . . . . .	132
6.19.2	Enumeration Type Documentation . . . . .	132
6.19.2.1	_OMNITEK_DEVTYPES . . . . .	132
6.19.3	Function Documentation . . . . .	132
6.19.3.1	OmniTek_BAR_dev_ioctl . . . . .	132
6.19.3.2	OmniTek_BAR_dev_open . . . . .	132
6.19.3.3	OmniTek_BAR_dev_release . . . . .	132
6.19.3.4	OmniTekDeviceReleaseDev . . . . .	133
6.19.3.5	OmniTekDeviceSetupDev . . . . .	133
6.20	driver/OmniTekFPGA_linux.c File Reference . . . . .	133
6.20.1	Function Documentation . . . . .	134
6.20.1.1	DMACtrlGetInterruptStatus . . . . .	134
6.20.1.2	DMACtrlInterruptEnable . . . . .	134
6.20.1.3	FPGADelete . . . . .	135
6.20.1.4	FPGAGetInterruptStatus . . . . .	135
6.20.1.5	FPGAGetTime . . . . .	136
6.20.1.6	FPGAInit . . . . .	136
6.20.1.7	FPGAInterruptEnable . . . . .	137
6.20.1.8	FPGAReadTime . . . . .	138



6.20.1.9	GetInterruptStatus	138
6.20.1.10	InterruptEnable	139
6.21	driver/OmniTekFPGA_linux.h File Reference	139
6.21.1	Function Documentation	140
6.21.1.1	FPGAControl	140
6.21.1.2	FPGADelete	140
6.21.1.3	FPGAGetStandard	140
6.21.1.4	FPGAGetTime	141
6.21.1.5	FPGAInit	141
6.21.1.6	FPGAReadTime	142
6.21.1.7	GetInterruptStatus	142
6.21.1.8	InterruptEnable	142
6.22	driver/OmniTekInterrupt_linux.c File Reference	142
6.23	driver/OmniTekInterrupt_linux.h File Reference	143
6.24	driver/OmniTekRequest_linux.h File Reference	143
6.24.1	Typedef Documentation	144
6.24.1.1	OmniTekKernelRequest	144
6.24.1.2	OmniTekUserRequest	144
6.24.1.3	POmniTekKernelRequest	144
6.24.1.4	POmniTekUserRequest	144
6.24.1.5	RequestStatus	144
6.24.2	Enumeration Type Documentation	144
6.24.2.1	_RequestStatus	144
6.25	driver/OmniTekRequestQueue.c File Reference	144
6.25.1	Function Documentation	145
6.25.1.1	OmniTekRequestQueueAddRequest	145
6.25.1.2	OmniTekRequestQueueContains	146
6.25.1.3	OmniTekRequestQueueInit	146
6.25.1.4	OmniTekRequestQueueInitRequest	146
6.25.1.5	OmniTekRequestQueueIsEmpty	147
6.25.1.6	OmniTekRequestQueueMoveRequest	147
6.25.1.7	OmniTekRequestQueueNext	148
6.25.1.8	OmniTekRequestQueueRemoveRequest	148
6.25.1.9	OmniTekRequestQueueSize	149
6.26	driver/OmniTekRequestQueue.h File Reference	149
6.26.1	Typedef Documentation	151

6.26.1.1	OmniTekRequestQueue	151
6.26.1.2	OmniTekRequestQueueObject	151
6.26.1.3	POmniTekRequestQueue	151
6.26.1.4	POmniTekRequestQueueObject	151
6.26.2	Function Documentation	151
6.26.2.1	OmniTekRequestQueueAddRequest	151
6.26.2.2	OmniTekRequestQueueContains	151
6.26.2.3	OmniTekRequestQueueInit	152
6.26.2.4	OmniTekRequestQueueInitRequest	152
6.26.2.5	OmniTekRequestQueueIsEmpty	152
6.26.2.6	OmniTekRequestQueueMoveRequest	153
6.26.2.7	OmniTekRequestQueueNext	153
6.26.2.8	OmniTekRequestQueueRemoveRequest	154
6.26.2.9	OmniTekRequestQueueSize	154
6.27	driver/OmniTekResources_linux.c File Reference	155
6.27.1	Function Documentation	155
6.27.1.1	AddResource	155
6.27.1.2	DmaChannelFind	156
6.27.1.3	ReleaseResource	156
6.27.1.4	RemoveResource	157
6.27.1.5	ResourceCheck	158
6.27.1.6	ResourceFind	158
6.27.1.7	ResourceItemRelease	158
6.28	driver/OmniTekResources_linux.h File Reference	158
6.28.1	Typedef Documentation	160
6.28.1.1	DmaChannel	160
6.28.1.2	DmaCtrl	160
6.28.1.3	DmaSglBuffer	160
6.28.1.4	FPGACtrl	160
6.28.1.5	GeneralCtrl	160
6.28.1.6	OmnitekDmaInterruptComplete	160
6.28.1.7	PDmaChannel	161
6.28.1.8	PDmaCtrl	161
6.28.1.9	PResource	161
6.28.1.10	Resource	161
6.28.1.11	ResourceVersion	161

---

6.28.2	Function Documentation . . . . .	161
6.28.2.1	AddResource . . . . .	161
6.28.2.2	DmaChannelFind . . . . .	161
6.28.2.3	DmaResourceInit . . . . .	162
6.28.2.4	ReleaseResource . . . . .	162
6.28.2.5	RemoveResource . . . . .	162
6.28.2.6	ResourceCheck . . . . .	163
6.28.2.7	ResourceControl . . . . .	164
6.28.2.8	ResourceRegisterWatchdog . . . . .	164



# 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

Copyright (c) 2007 Image Processing Techniques, Ltd. [www.imageproc.com](http://www.imageproc.com).

Image Processing Techniques Ltd. licenses this software under specific terms and conditions. Use of any of the software or derivatives thereof in any product without a Image Processing Techniques is strictly prohibited.

This file is provided without any express or implied warranties, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. It is not intended for use in life support appliances, devices, or systems. Use in such applications is expressly prohibited.

Image Processing Techniques makes no guarantee or representations regarding the use of, or the results of the use of, the software and documentation in terms of correctness, accuracy, reliability, currentness, or otherwise; and you rely on the software, documentation and results solely at your own risk.

IN NO EVENT SHALL IMAGE PROCESSING TECHNIQUES LTD. BE LIABLE FOR ANY LOSS OF USE, LOSS OF BUSINESS, LOSS OF PROFITS, INDIRECT, INCIDENTAL, SPECIAL OR CONSEQUENTIAL DAMAGES OF ANY KIND.

### 1.2 Guides

[Driver DMA API Guide](#)



## 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 successfully, 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:

<a href="#">_DmaSglBuffer::_Allocated</a> . . . . .	9
<a href="#">_OMNITEK_INTERFACE_EXTENSION::_bar_registers</a> (Details of the registers in each BAR ) . . . . .	10
<a href="#">_dma_interrupt_info::_chan_int_counts</a> . . . . .	10
<a href="#">_dma_interrupt_info</a> . . . . .	11
<a href="#">_DmaChannel</a> (Data structure containing details of a DMA Channel Resource ) . . . . .	12
<a href="#">_OmniTekDmaTransactionContext::_DMACoreInfo</a> . . . . .	15
<a href="#">_DmaCtrl</a> (DMA Control Resource ) . . . . .	16
<a href="#">_DmaSglBuffer</a> (DMA Scatter Gather List buffer ) . . . . .	18
<a href="#">_FPGACtrl</a> (FPGA Control Resource ) . . . . .	19
<a href="#">_GeneralCtrl</a> (Data structure for PCIE BAR ) . . . . .	21
<a href="#">_InterruptData</a> (Data about interrupts ) . . . . .	21
<a href="#">_OmniTek_dev</a> . . . . .	23
<a href="#">_OMNITEK_INTERFACE_EXTENSION</a> . . . . .	26
<a href="#">_OmniTekDmaTransactionContext</a> . . . . .	29
<a href="#">_OmniTekDriver</a> . . . . .	32
<a href="#">_OmniTekKernelRequest</a> . . . . .	32
<a href="#">_OmniTekRequestQueue</a> . . . . .	34
<a href="#">_OmniTekRequestQueueObject</a> . . . . .	35
<a href="#">_OmniTekUserRequest</a> . . . . .	36
<a href="#">_PCI_BAR_INFO</a> (PCI BAR Space information ) . . . . .	38
<a href="#">_Resource</a> (Generic resource Details ) . . . . .	39
<a href="#">_Resource::_ResourceExtension</a> . . . . .	42
<a href="#">_ResourceVersion</a> (Version details of a device resource ) . . . . .	43
<a href="#">_OmniTekDmaTransactionContext::_SglInfo</a> . . . . .	44
<a href="#">_OmniTekDmaTransactionContext::_XferInfo</a> . . . . .	44
<a href="#">OmniTek_dev</a> (BAR Device data structure ) . . . . .	45
<a href="#">OmniTekDriver</a> (Used to keep track of the Extensions created by the driver ) . . . . .	45
<a href="#">OmniTekKernelRequest</a> . . . . .	45
<a href="#">OmniTekUserRequest</a> . . . . .	46



# Chapter 4

## File Index

### 4.1 File List

Here is a list of all files with brief descriptions:

driver/ <a href="#">OmniTekResource_linux.o.d</a>	47
driver/ <a href="#">OmniTek_debug.h</a>	47
driver/ <a href="#">OmniTek_Driver.mod.c</a>	48
driver/ <a href="#">OmniTek_linux.c</a>	49
driver/ <a href="#">OmniTek_linux.h</a>	55
driver/ <a href="#">OmniTek_MainPage.h</a>	67
driver/ <a href="#">OmniTekDma.c</a>	67
driver/ <a href="#">OmniTekDma.h</a>	78
driver/ <a href="#">OmniTekDMACore.c</a>	89
driver/ <a href="#">OmniTekDMACore.h</a>	95
driver/ <a href="#">OmniTekDmaIsr_linux.c</a>	99
driver/ <a href="#">OmniTekDmaOperations.c</a>	100
driver/ <a href="#">OmniTekDmaOperations.h</a>	102
driver/ <a href="#">OmniTekDMARequest.c</a>	105
driver/ <a href="#">OmniTekDMARequest.h</a>	110
driver/ <a href="#">OmniTekDriver_linux.c</a>	116
driver/ <a href="#">OmniTekDriver_linux.h</a>	125
driver/ <a href="#">OmniTekFops_linux.c</a>	128
driver/ <a href="#">OmniTekFops_linux.h</a>	130
driver/ <a href="#">OmniTekFPGA_linux.c</a>	133
driver/ <a href="#">OmniTekFPGA_linux.h</a>	139
driver/ <a href="#">OmniTekInterrupt_linux.c</a>	142
driver/ <a href="#">OmniTekInterrupt_linux.h</a>	143
driver/ <a href="#">OmniTekRequest_linux.h</a>	143
driver/ <a href="#">OmniTekRequestQueue.c</a>	144
driver/ <a href="#">OmniTekRequestQueue.h</a>	149
driver/ <a href="#">OmniTekResources_linux.c</a>	155
driver/ <a href="#">OmniTekResources_linux.h</a>	158



## 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`

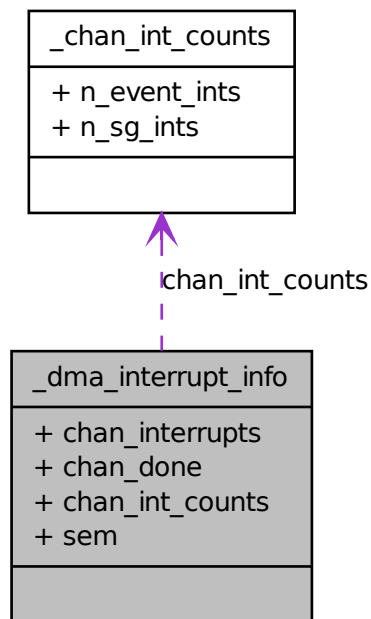
##### 5.3.1.2 `atomic_t n_sg_ints`

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](#)

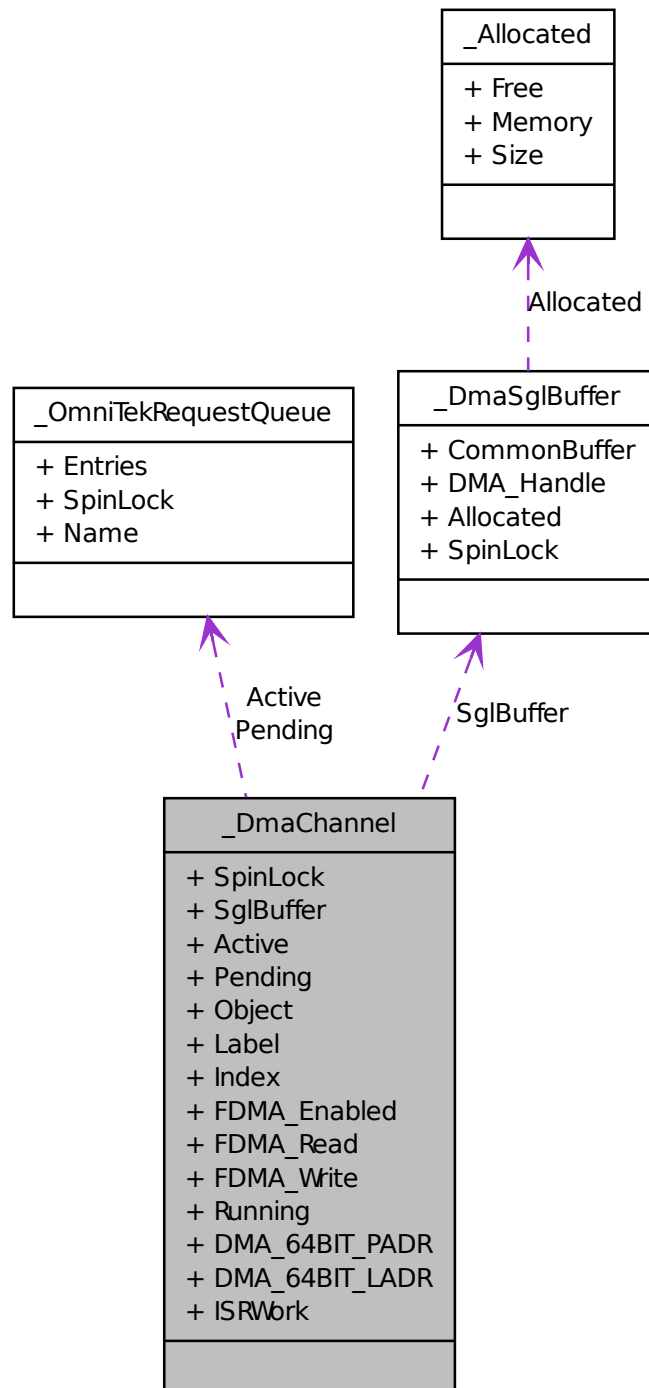
## 5.5 \_DmaChannel Struct Reference

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:

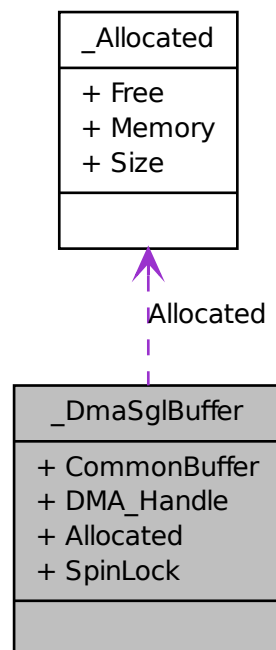
- driver/[OmniTekResources\\_linux.h](#)

## 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:

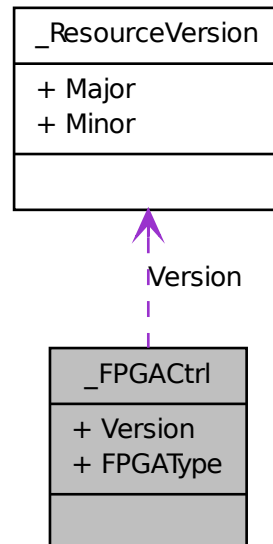
- driver/[OmniTekResources\\_linux.h](#)

## 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:

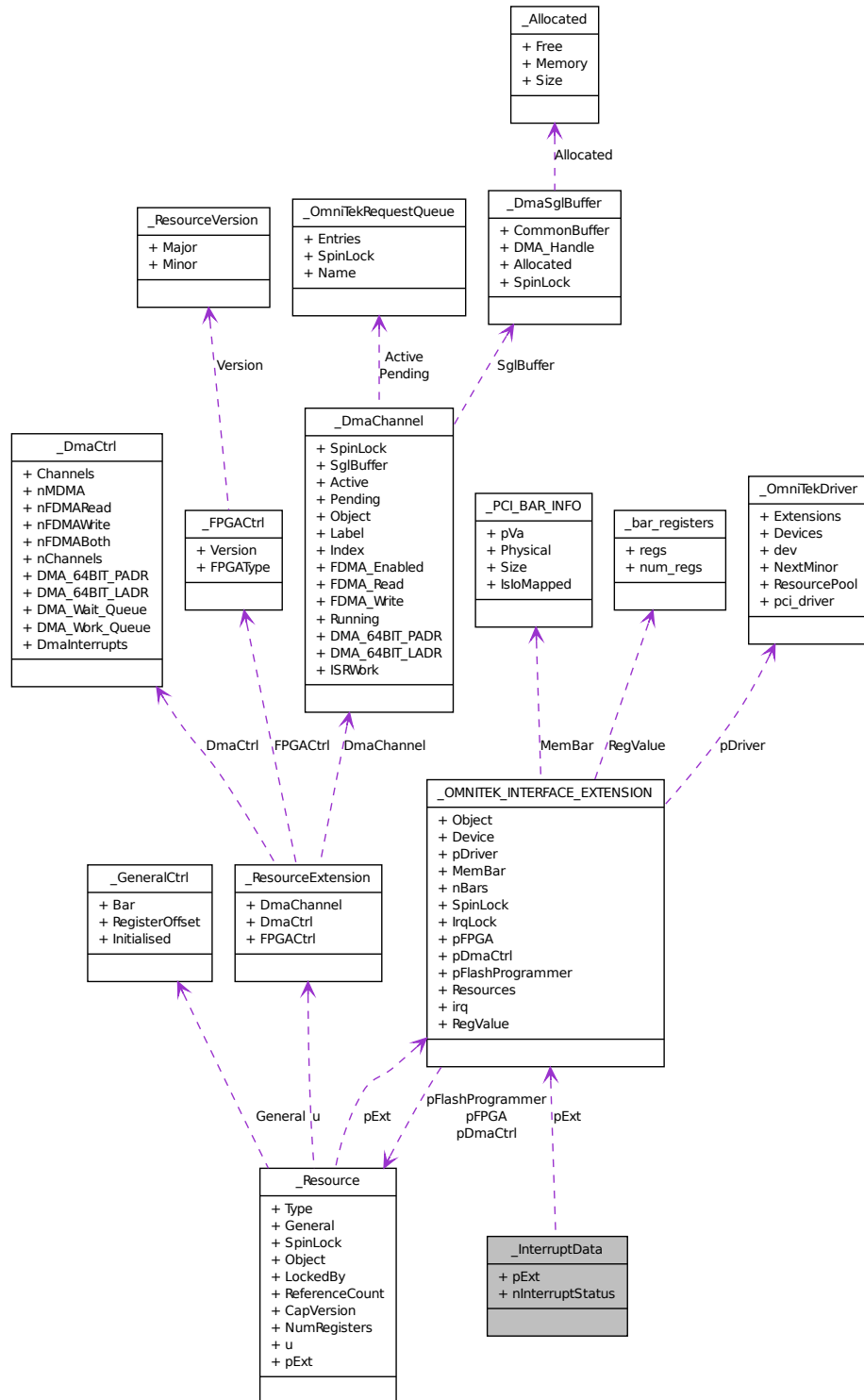
- driver/[OmniTekResources\\_linux.h](#)

## 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

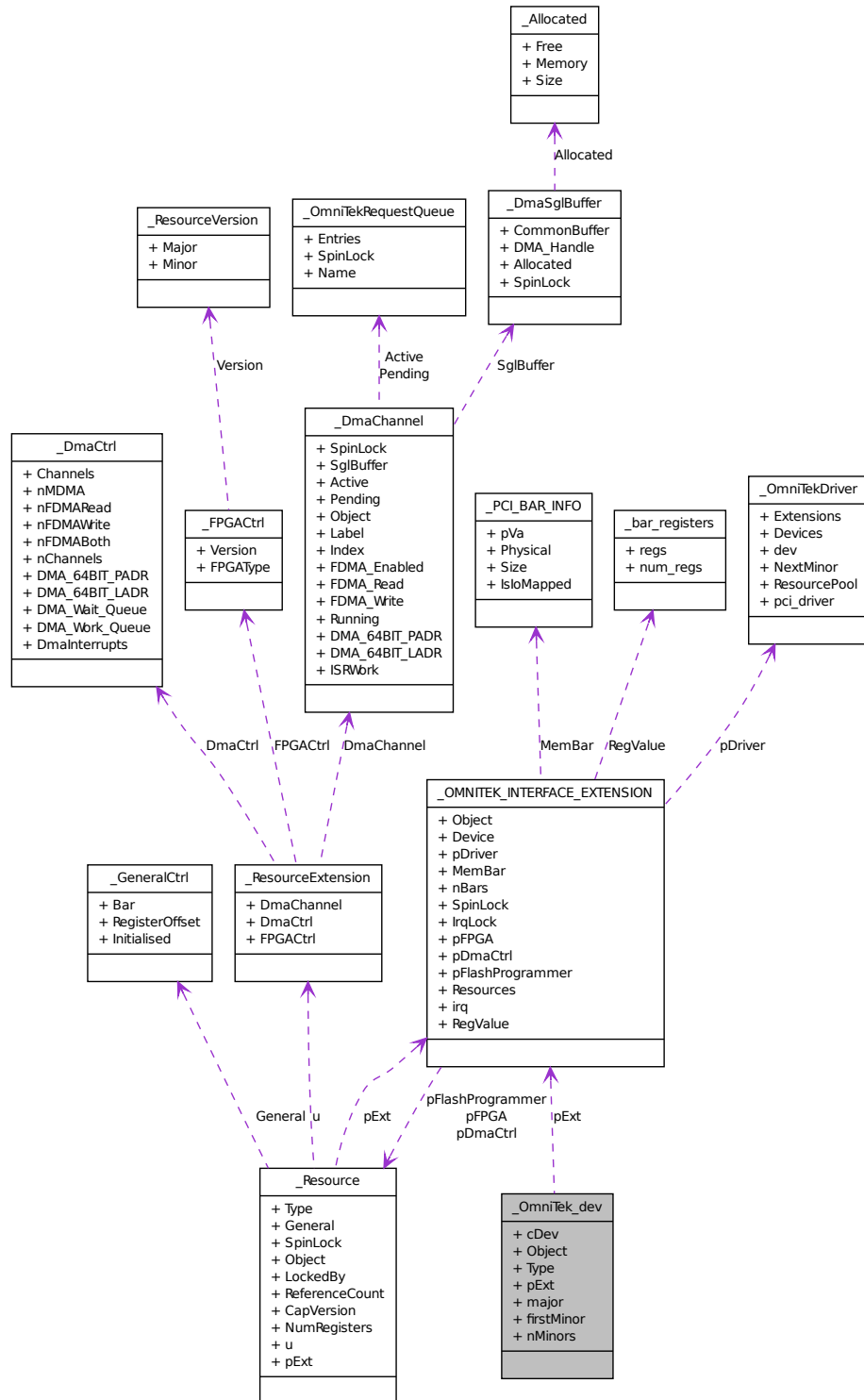
The documentation for this struct was generated from the following file:

- [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

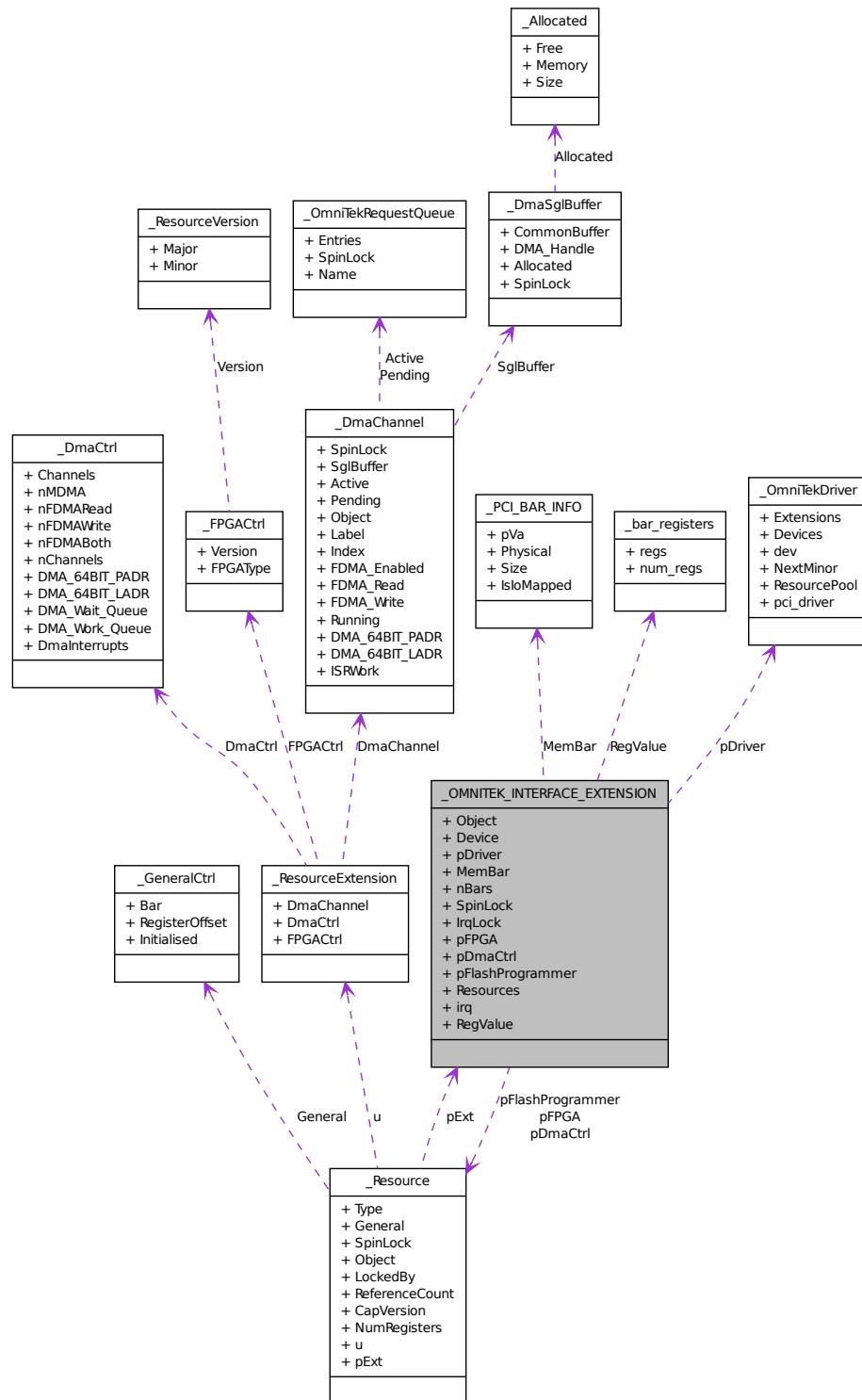
The documentation for this struct was generated from the following file:

- 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

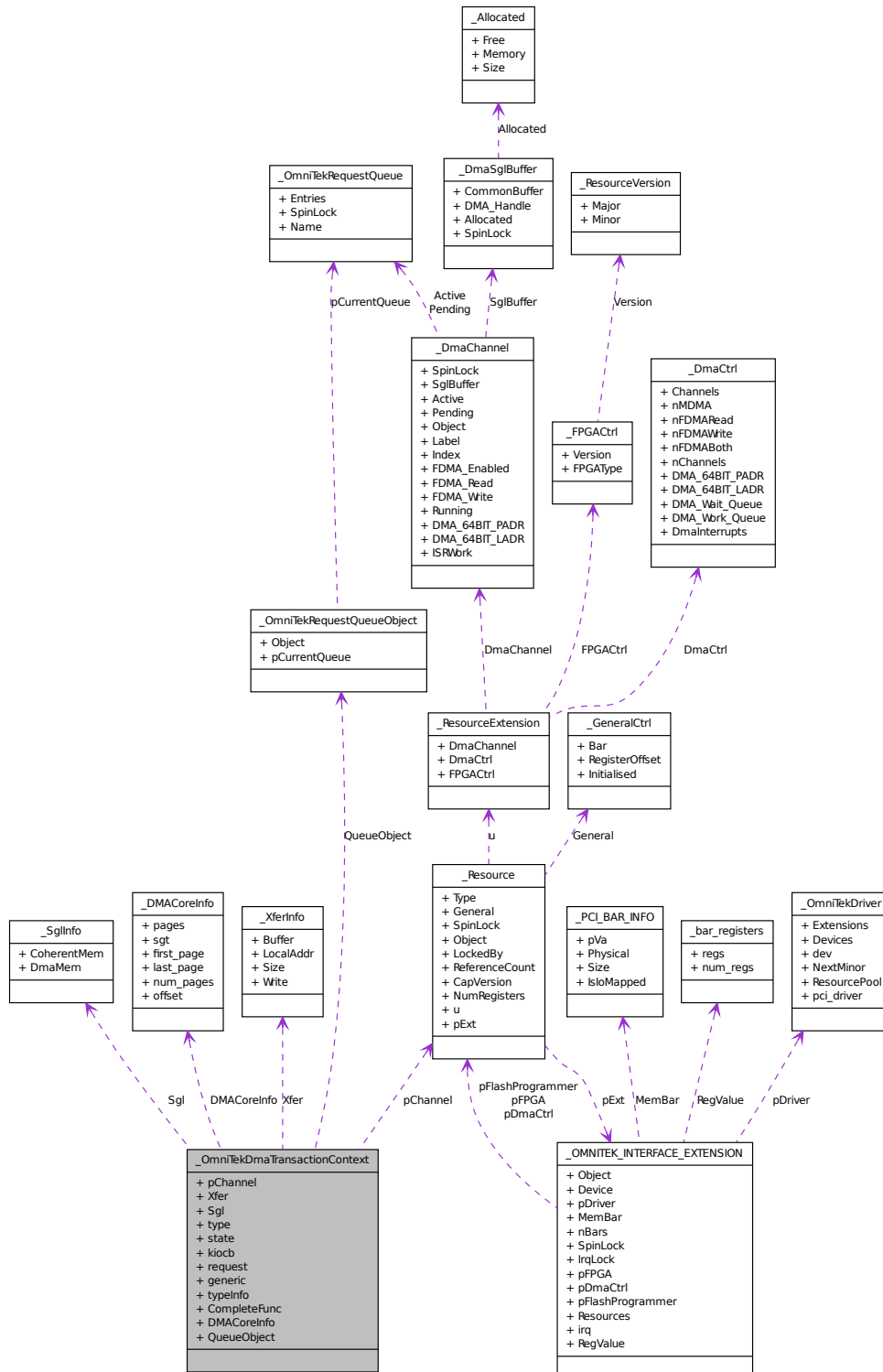
The documentation for this struct was generated from the following file:

- 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

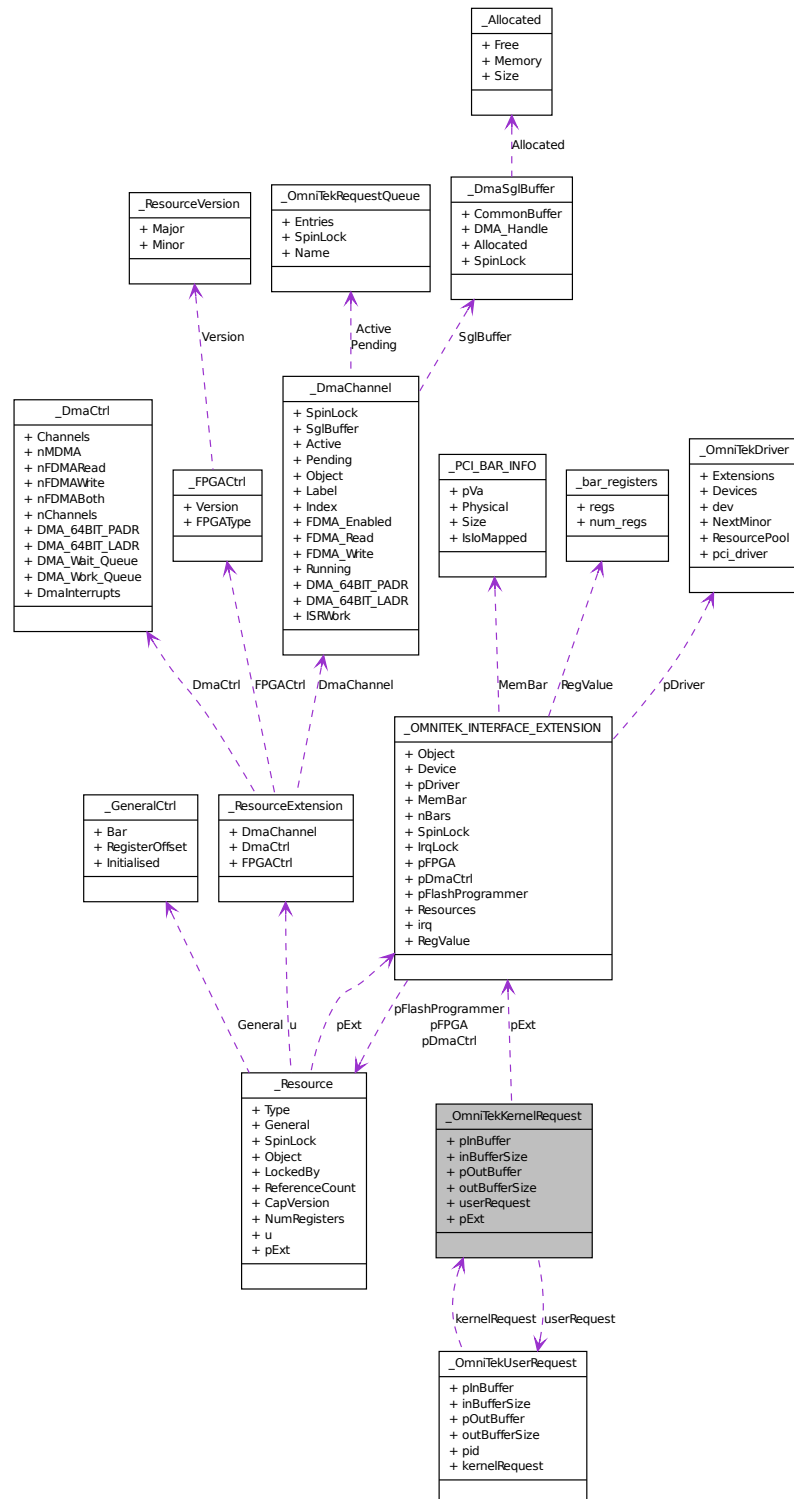
The documentation for this struct was generated from the following file:

- 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

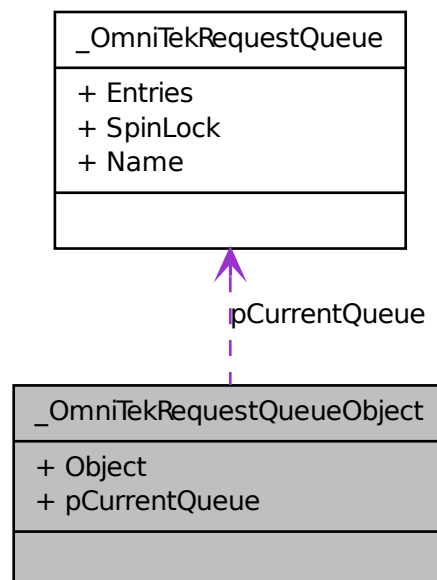
The documentation for this struct was generated from the following file:

- 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:

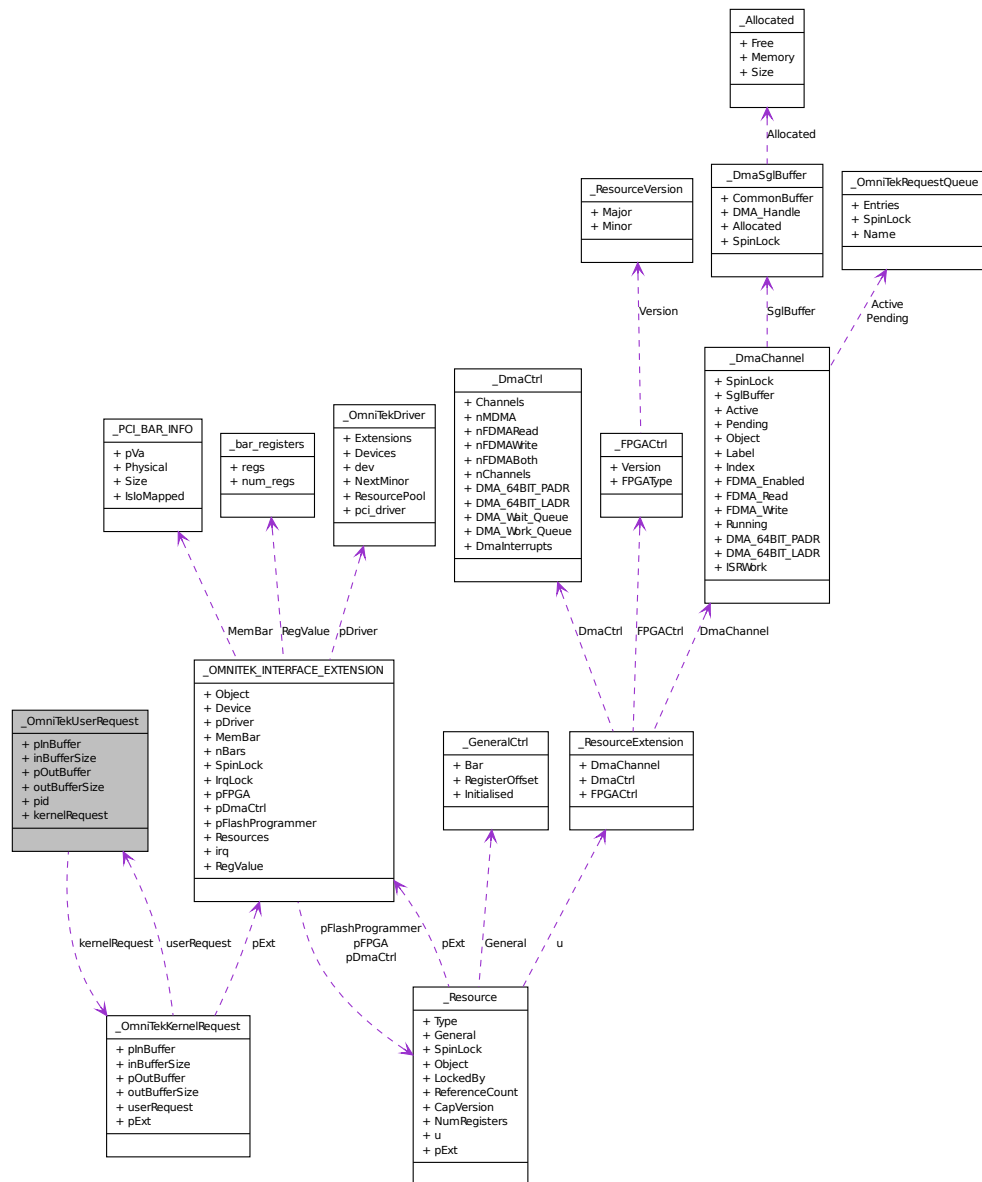
- driver/[OmniTekRequestQueue.h](#)

## 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:

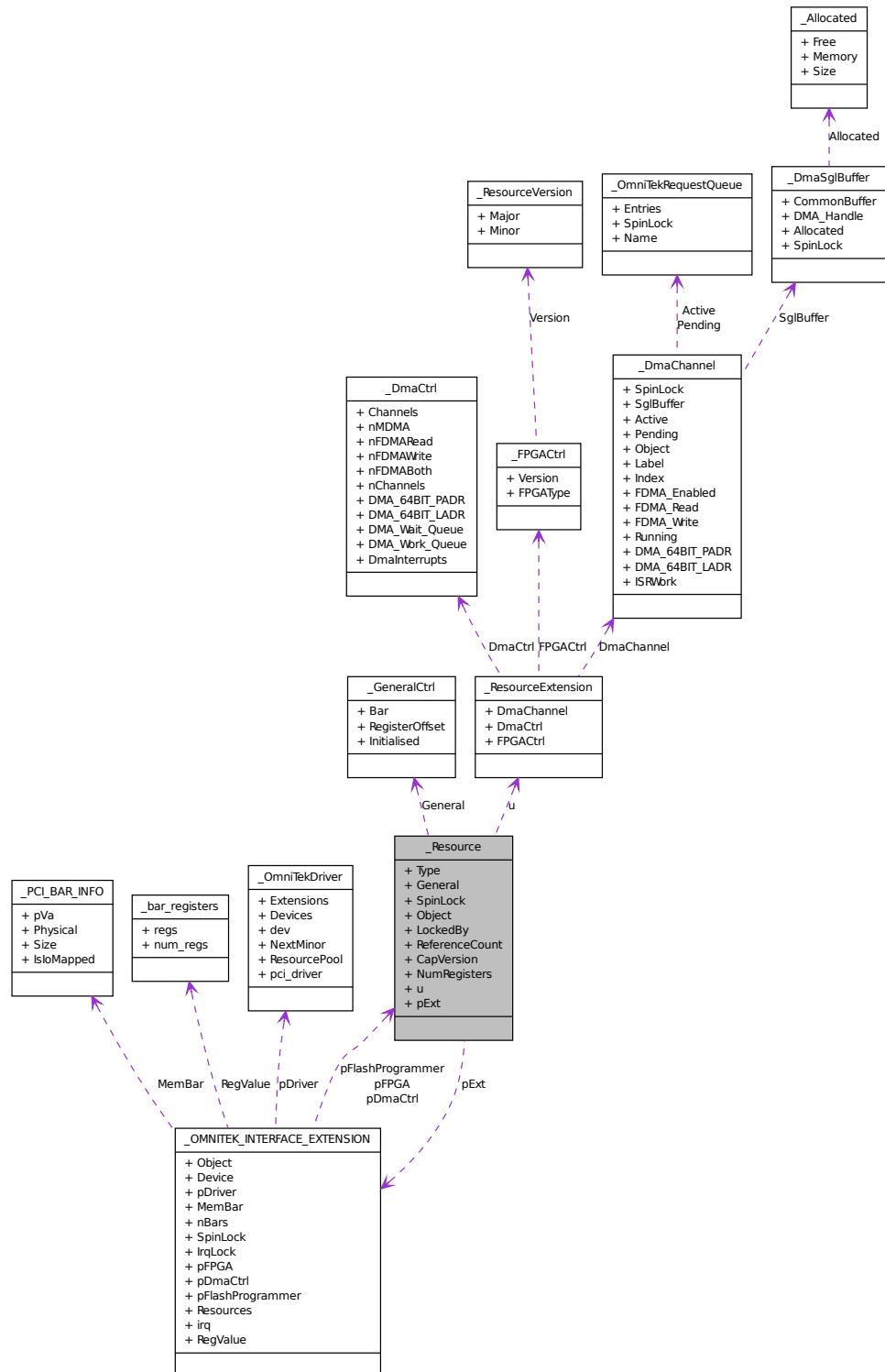
- [driver/OmniTek\\_linux.h](#)

## 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

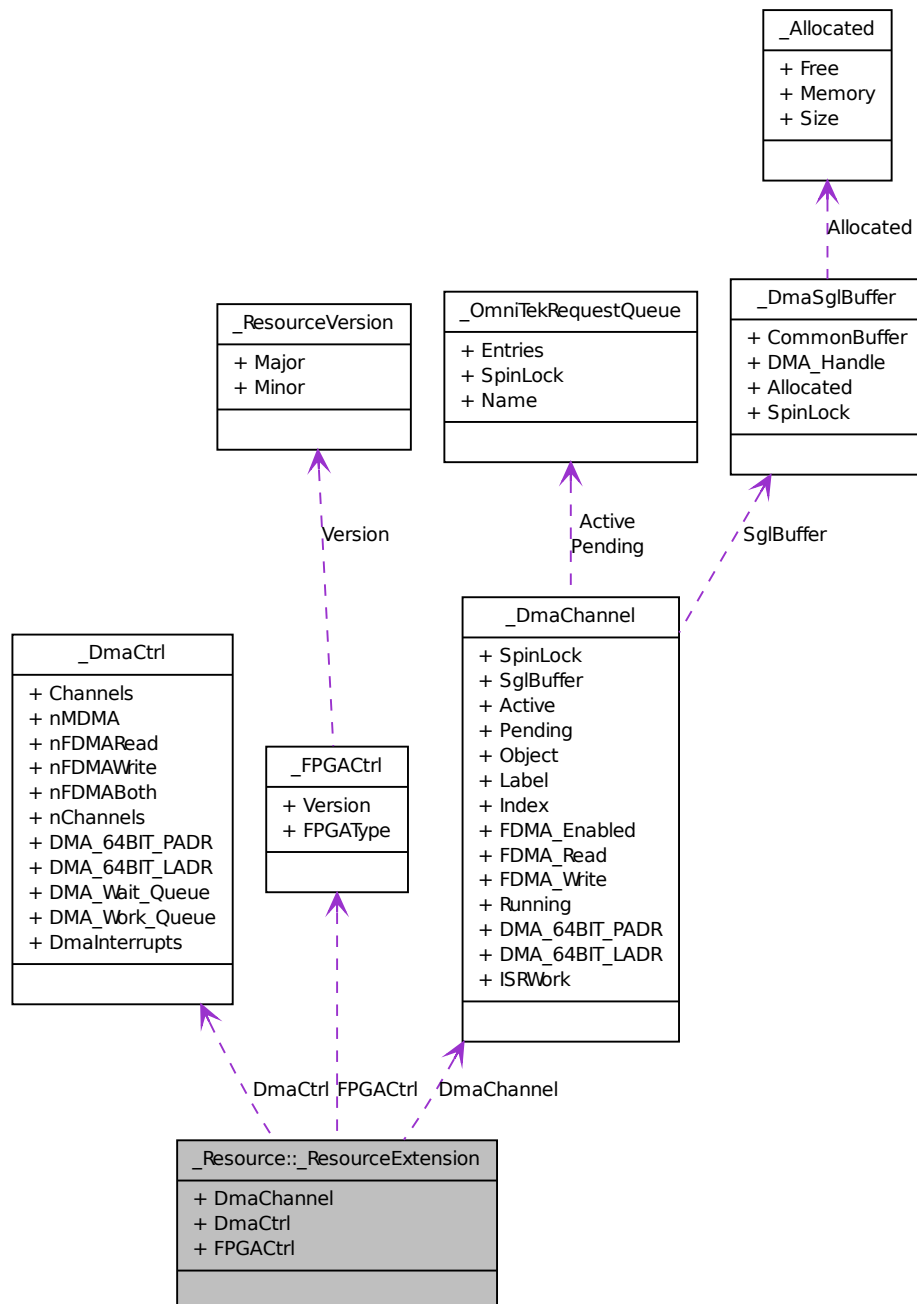
The documentation for this struct was generated from the following file:

- 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](#)

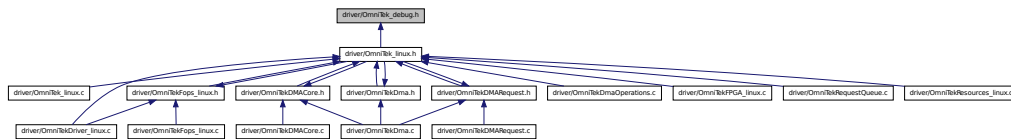
# Chapter 6

## 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

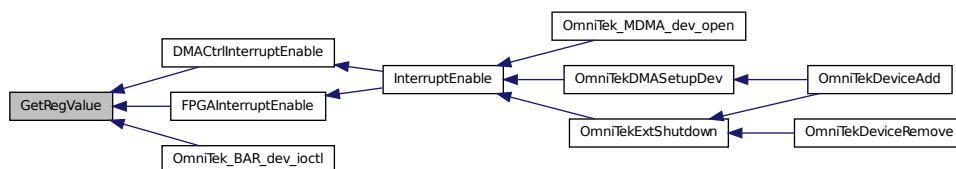
[in] **nReg** Register offset

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

#### Returns

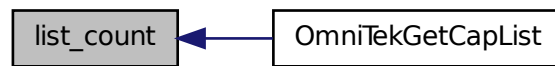
0 on success or error code

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* )

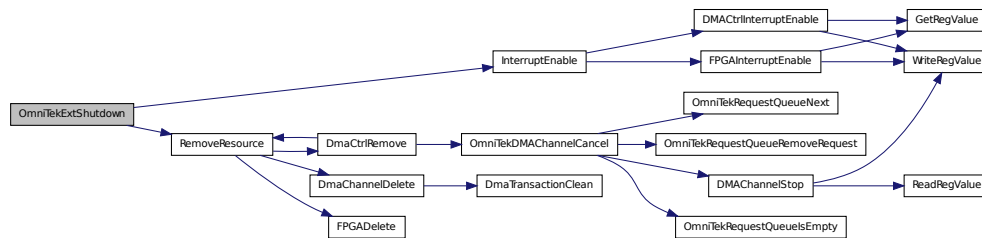
Cleanup for device extension.

Performs necessary cleanup for the device extension when it is shutdown

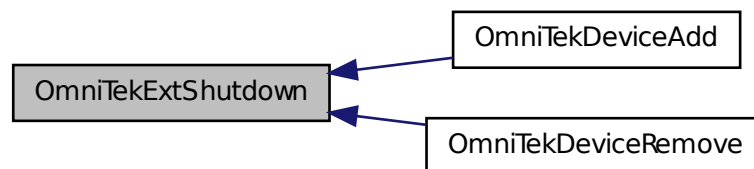
##### Parameters

- [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* )

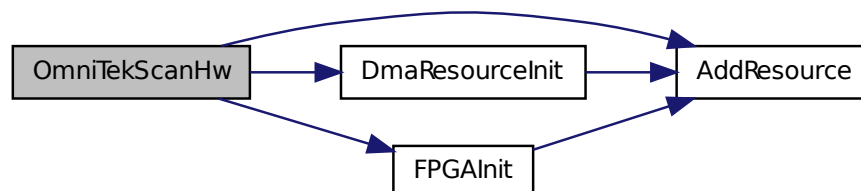
Scan hardware for Resources.

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.

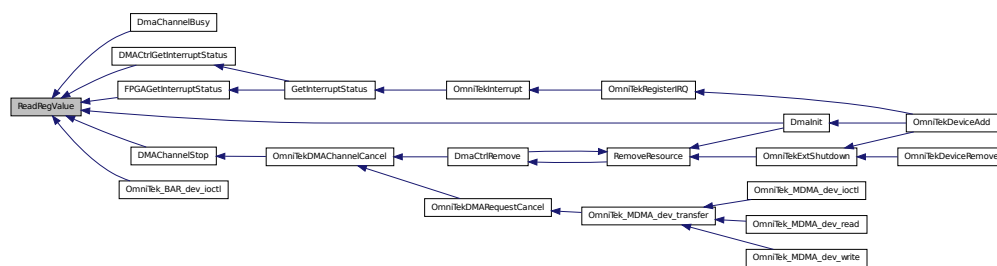
##### Parameters

- [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.

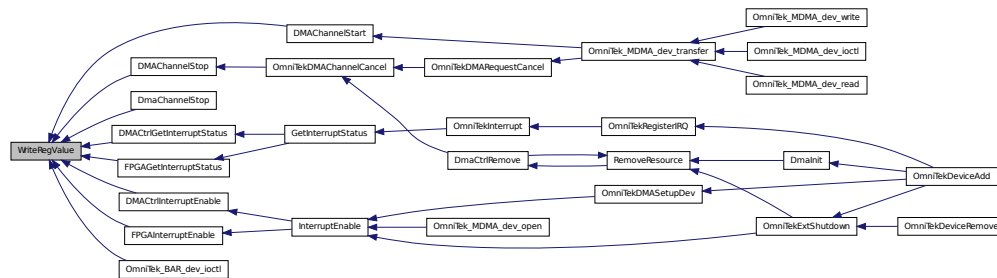
##### Parameters

- [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

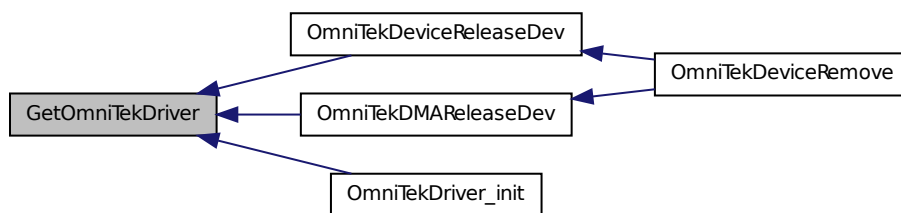
**Parameters**

[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**

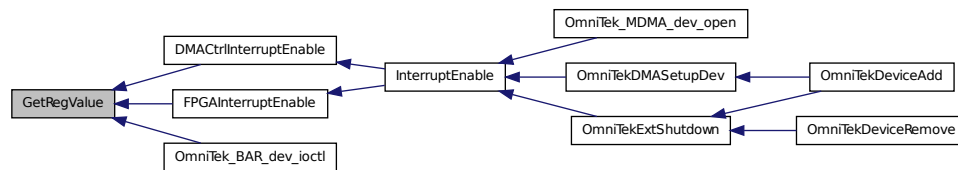
[in] **pExt** Pointer to device extension to read from

[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* )

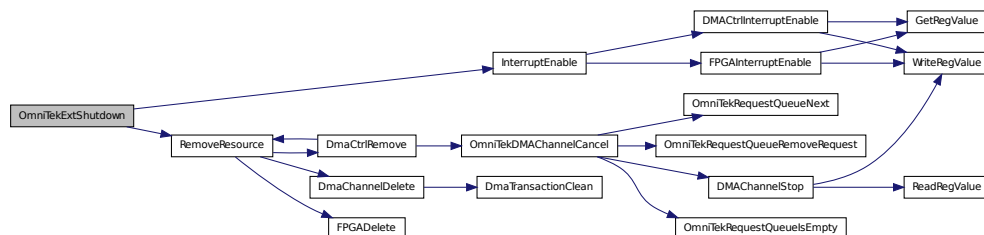
Cleanup for device extension.

Performs necessary cleanup for the device extension when it is shutdown

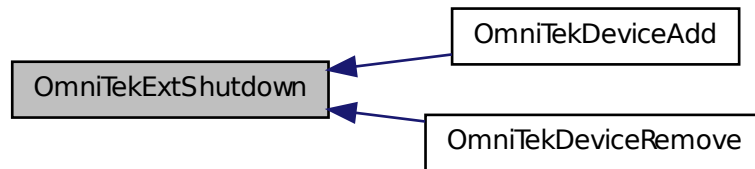
#### Parameters

[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* )

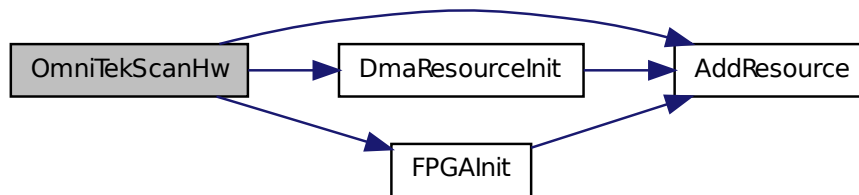
Scan hardware for Resources.

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.



```
#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**

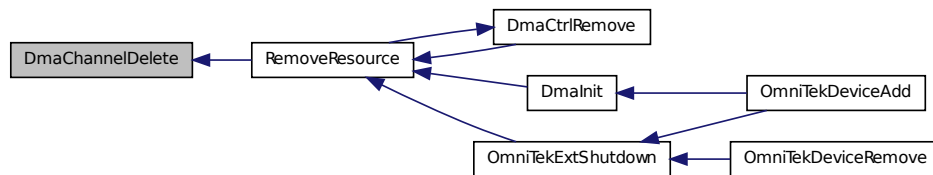
[in] *pChannel* pointer to channel resource to delete

Will stop channel and cancel any outstanding requests

Here is the call graph for this function:



Here is the caller graph for this function:

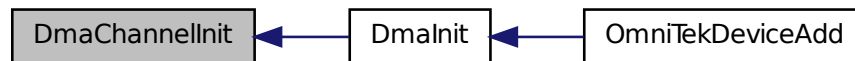


#### 6.7.2.4 static int DmaChannelInit ( PResource *pResource*, u8 *Index* ) [static]

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* )

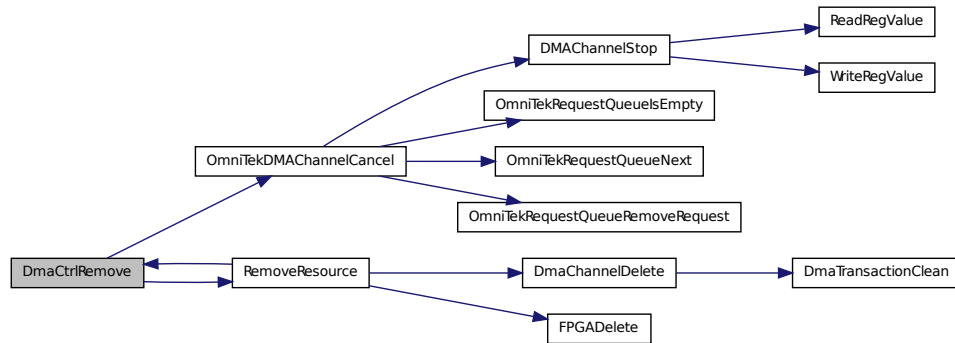
Delete a DMA Controller resource.

##### Parameters

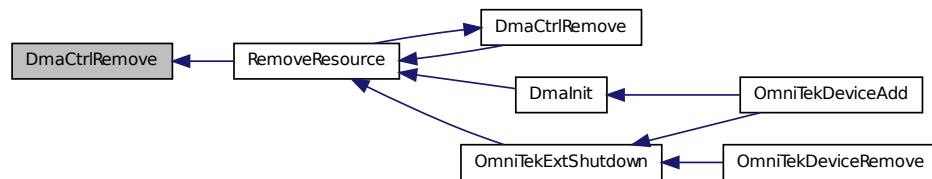
[in] *pCtrl* pointer to controller resource to remove

Will delete all channels associated with this controller

Here is the call graph for this function:

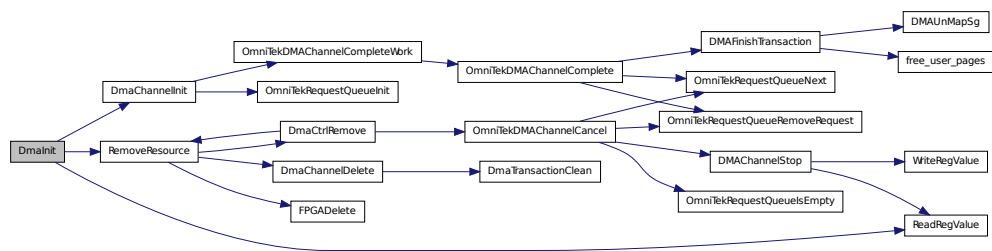


Here is the caller graph for this function:

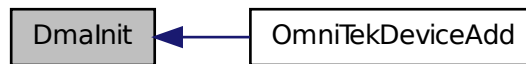


### 6.7.2.7 int DmaInit ( POMNITEK\_INTERFACE\_EXTENSION *pExt* )

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:

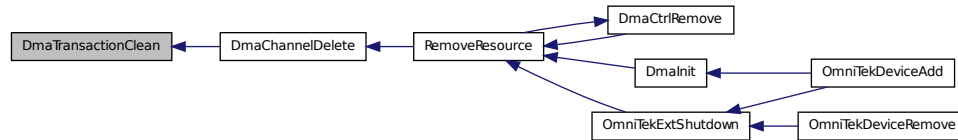


Here is the caller graph for this function:



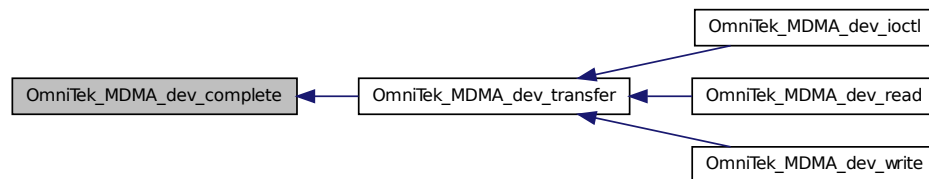
### 6.7.2.9 static void DmaTransactionClean ( struct \_OmniTekDmaTransactionContext \* *pContext* ) [static]

Here is the caller graph for this function:



### 6.7.2.10 void OmniTek\_MDMA\_dev\_complete ( struct \_OmniTekDmaTransactionContext \* *pTransaction*, int *status* )

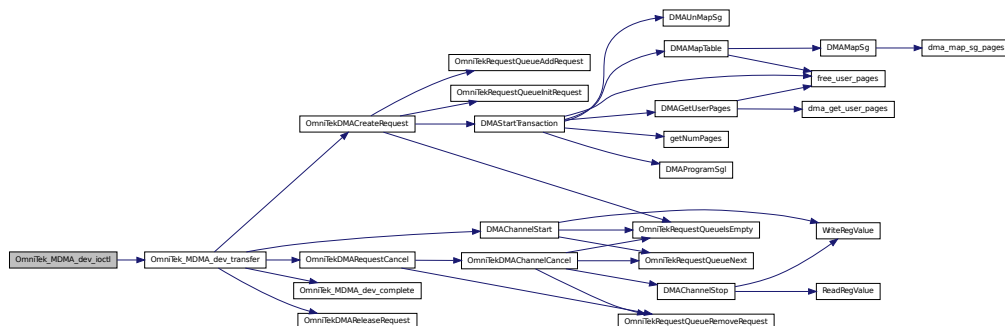
Here is the caller graph for this function:



### 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.

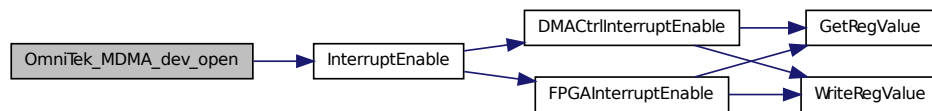
Here is the call graph for this function:



### 6.7.2.12 int OmniTek\_MDMA\_dev\_open ( struct inode \* *inode*, struct file \* *filp* )

Device open call to MDMA Device.

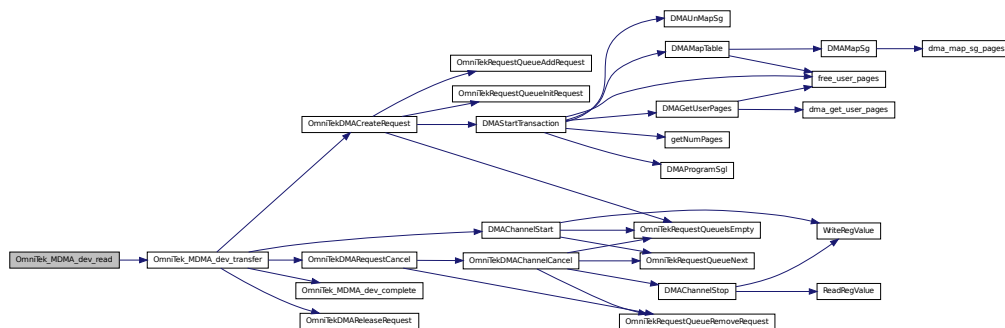
Here is the call graph for this function:



### 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:



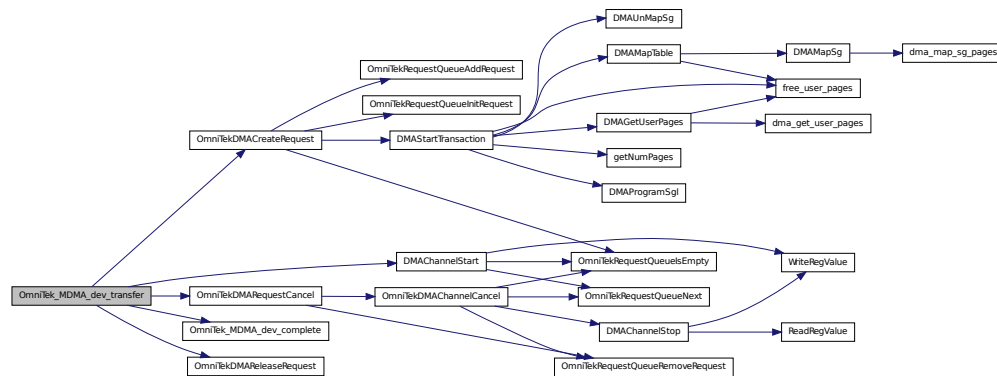
### 6.7.2.14 int OmniTek\_MDMA\_dev\_release ( struct inode \* *inode*, struct file \* *filp* )

Device release to MDMA Device.

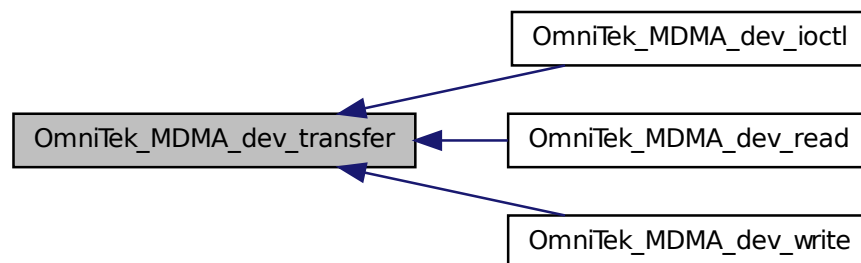
### 6.7.2.15 ssize\_t OmniTek\_MDMA\_dev\_transfer ( struct file \* *filp*, const char \_\_user \* *buf*, size\_t *count*, loff\_t \* *f\_pos*, bool *write* )

Perform MDMA Transfer.

Here is the call graph for this function:



Here is the caller graph for this function:

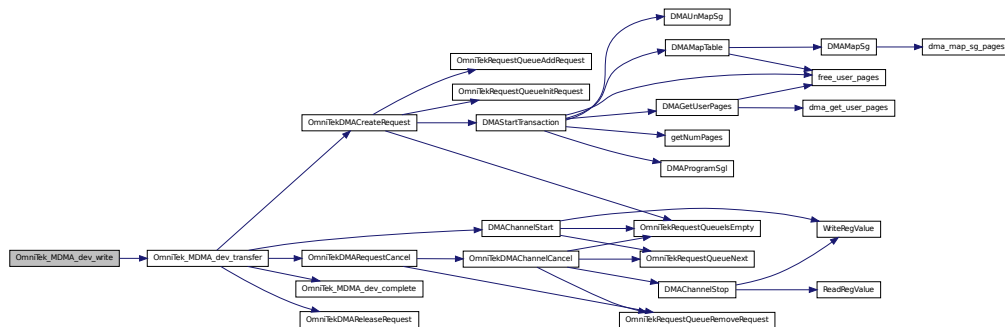


**6.7.2.16** `ssize_t OmniTek_MDMA_dev_write ( struct file * filp, const char __user * buf, size_t count, loff_t * f_pos )`

Write to MDMA Device.

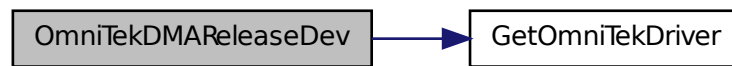


Here is the call graph for this function:



#### 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:



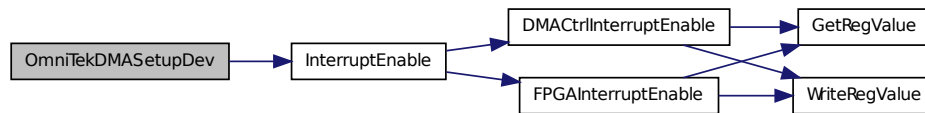
#### 6.7.2.18 void OmniTekDMASetupDev ( struct \_OMNITEK\_INTERFACE\_EXTENSION \* *pExt* )

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]

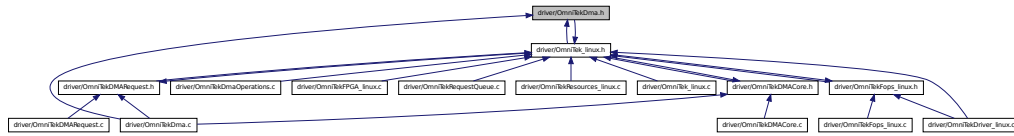
## 6.8 driver/OmniTekDma.h File Reference

```

#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

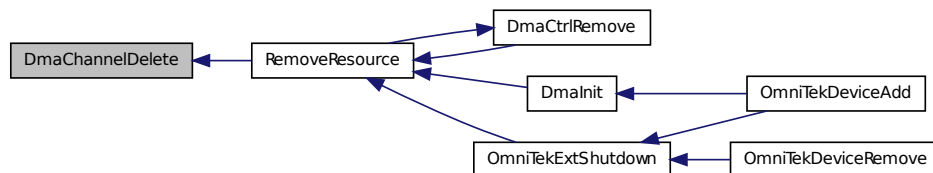
[in] *pChannel* pointer to channel resource to delete

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 )

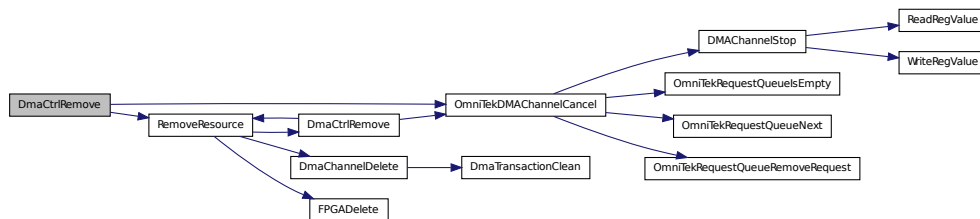
Delete a DMA Controller resource.

### Parameters

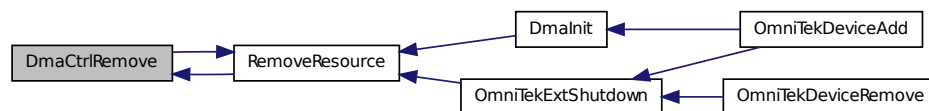
[in] *pCtrl* pointer to controller resource to remove

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:



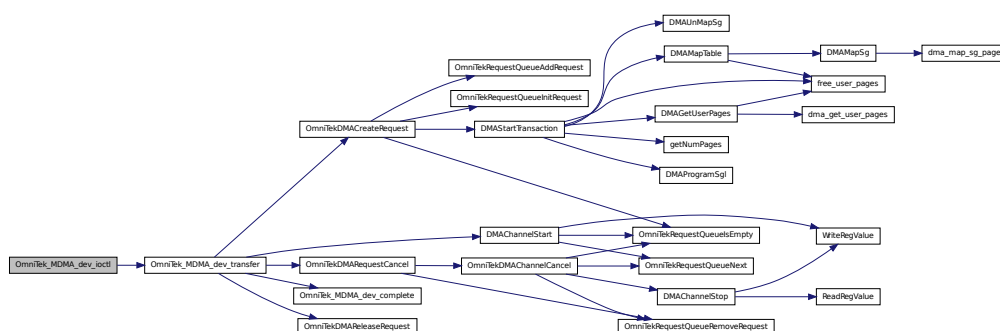
Here is the caller graph for this function:



### 6.8.3.7 int OmniTek\_MDMA\_dev\_ioctl ( struct inode \*, struct file \*, unsigned int, unsigned long )

IOctl call to MDMA Device.

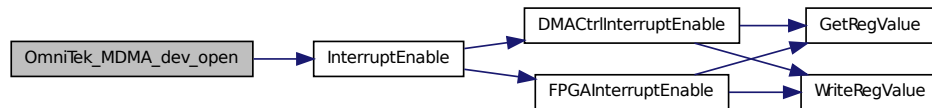
Here is the call graph for this function:



### 6.8.3.8 int OmniTek\_MDMA\_dev\_open ( struct inode \*, struct file \* )

Device open call to MDMA Device.

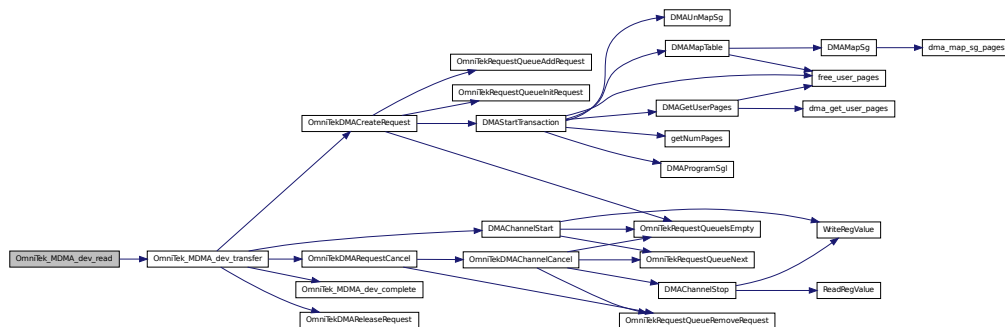
Here is the call graph for this function:



### 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:



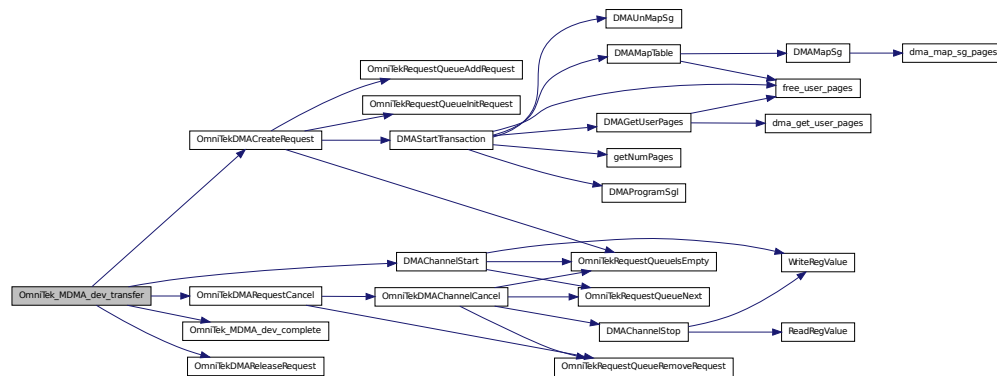
### 6.8.3.10 int OmniTek\_MDMA\_dev\_release ( struct inode \*, struct file \* )

Device release to MDMA Device.

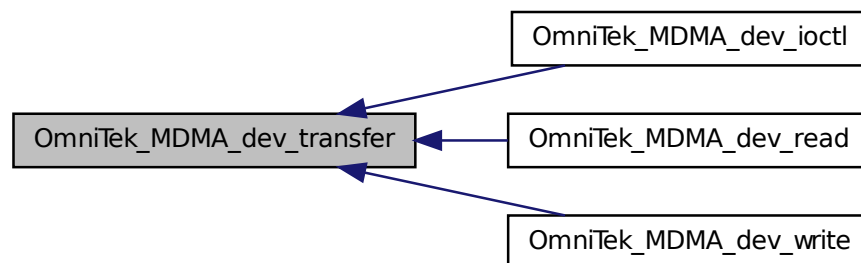
### 6.8.3.11 ssize\_t OmniTek\_MDMA\_dev\_transfer ( struct file \* *filp*, const char \_\_user \* *buf*, size\_t *count*, loff\_t \* *f\_pos*, bool *write* )

Perform MDMA Transfer.

Here is the call graph for this function:



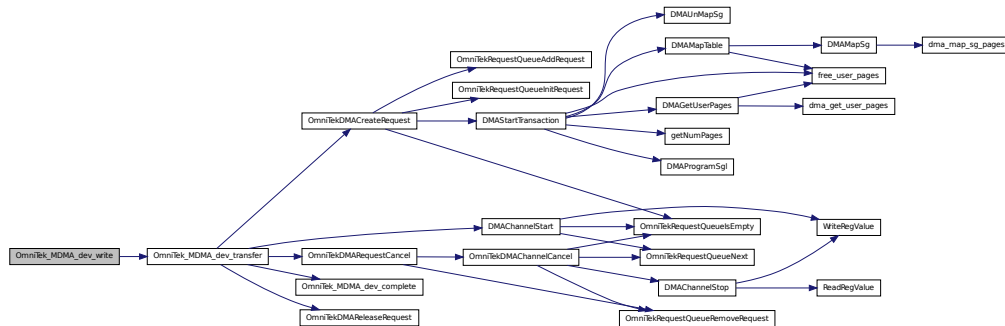
Here is the caller graph for this function:



**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

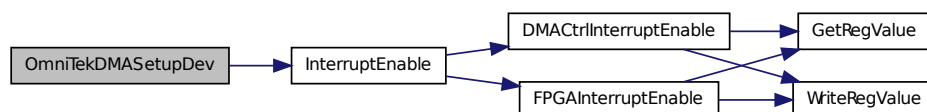
### 6.8.3.14 void OmniTekDMASetupDev ( struct \_OMNITEK\_INTERFACE\_EXTENSION \* *pExt* )

Setup DMA Device nodes.

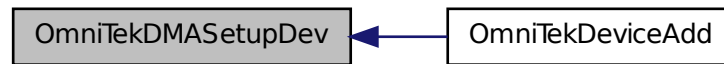
#### Parameters

[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

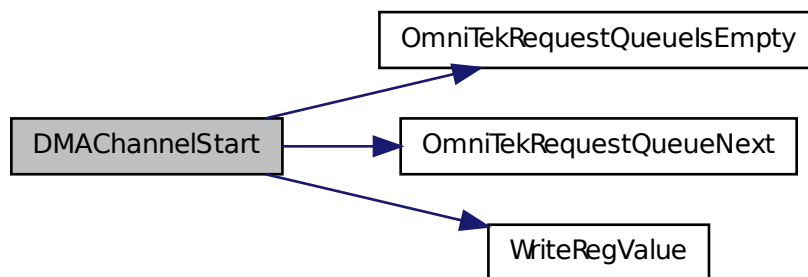
#### 6.9.1.1 int DMAChannelStart ( [PResource](#) pChannel )

Start this DMA Channel.

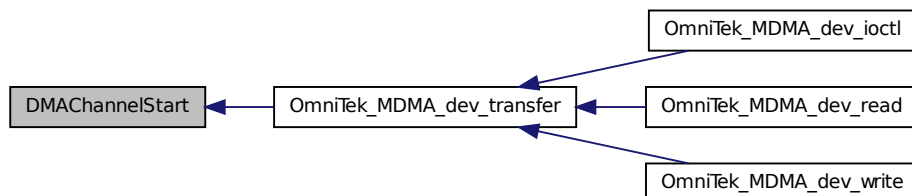
**Parameters**

[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:

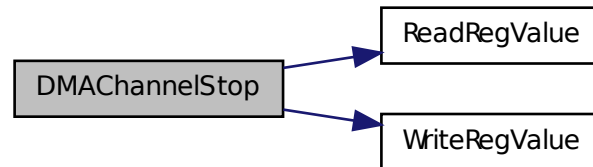
**6.9.1.2 void DMACHannelStop ( PResource *pChannel* )**

Stop this DMA Channel.

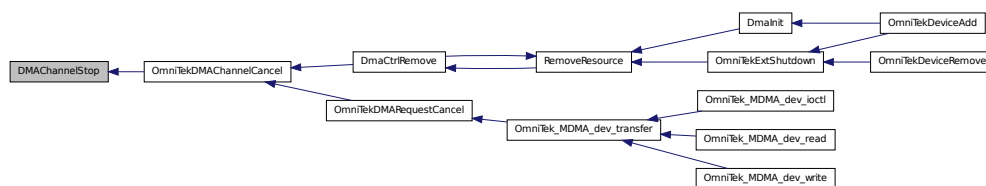
**Parameters**

[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:



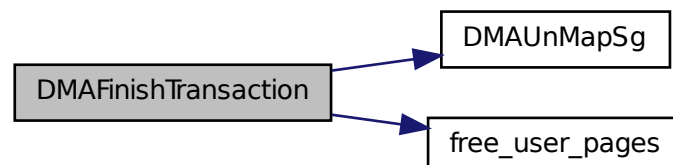
### 6.9.1.3 void DMAFinishTransaction ( struct \_OmniTekDmaTransactionContext \* *pTransaction* )

Finish this transaction.

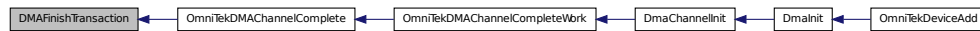
#### Parameters

[in] *pTransaction* Pointer to transaction to finish

Here is the call graph for this function:

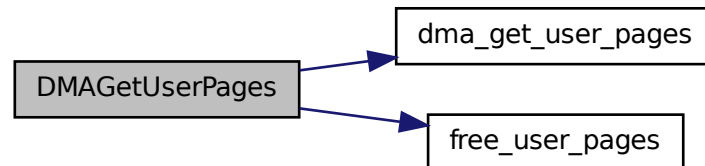


Here is the caller graph for this function:

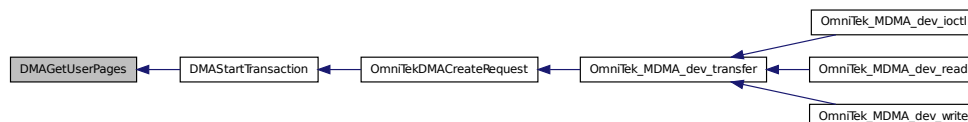


#### 6.9.1.4 int DMAGetUserPages ( struct \_OmniTekDmaTransactionContext \* *pTransaction* )

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:



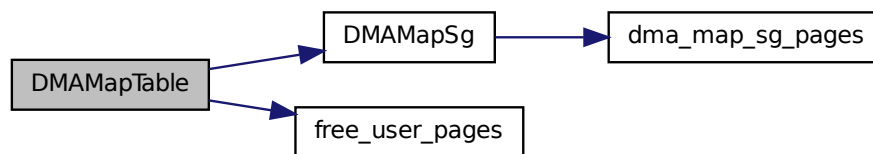


Here is the caller graph for this function:



#### 6.9.1.6 int DMAMapTable ( struct \_OmniTekDmaTransactionContext \* pTransaction )

Here is the call graph for this function:

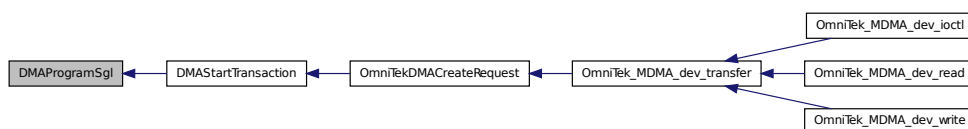


Here is the caller graph for this function:



#### 6.9.1.7 int DMAProgramSgl ( struct \_OmniTekDmaTransactionContext \* pTransaction )

Here is the caller graph for this function:



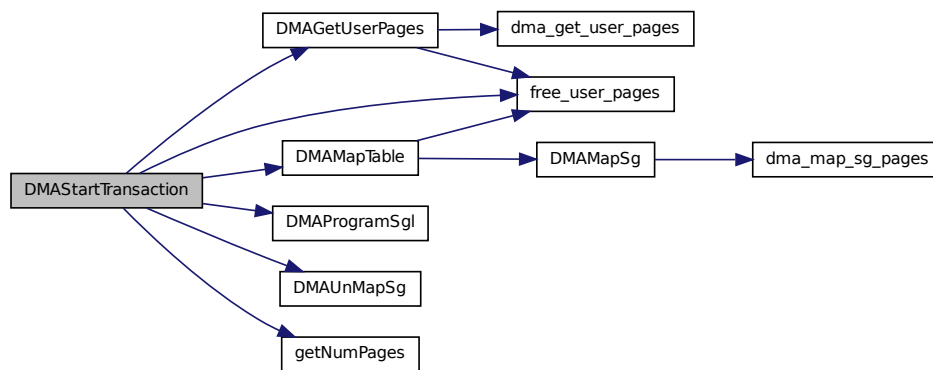
### 6.9.1.8 int DMAStartTransaction ( struct \_OmniTekDmaTransactionContext \* *pTransaction* )

Start this transaction.

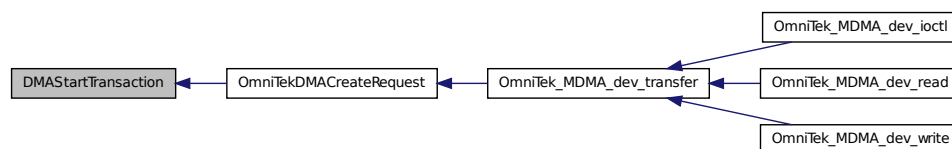
#### Parameters

[in] *pTransaction* Pointer to transaction to start

Here is the call graph for this function:

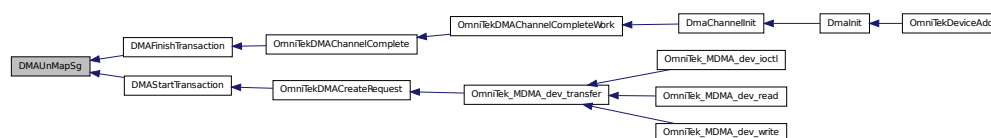


Here is the caller graph for this function:



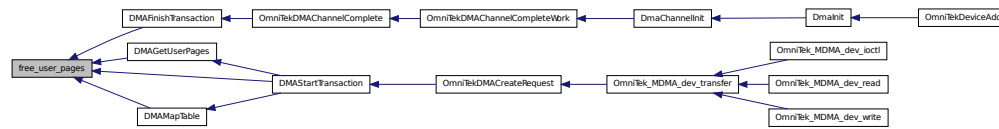
### 6.9.1.9 void DMAUnMapSg ( struct \_OmniTekDmaTransactionContext \* *pTransaction* )

Here is the caller graph for this function:



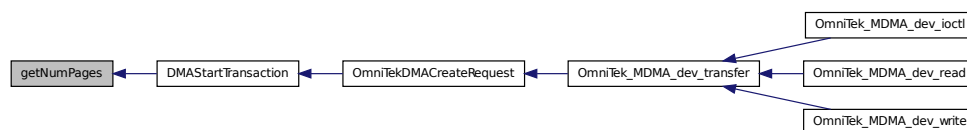
```
6.9.1.10 static void free_user_pages ( struct _OmniTekDmaTransactionContext * pTransaction )
[static]
```

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]
```

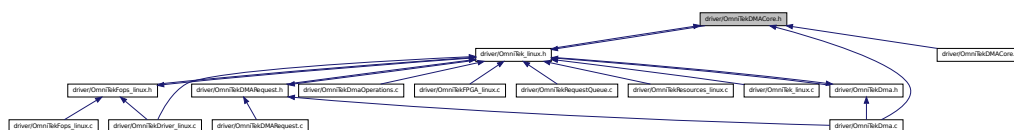
Here is the caller graph for this function:



## 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

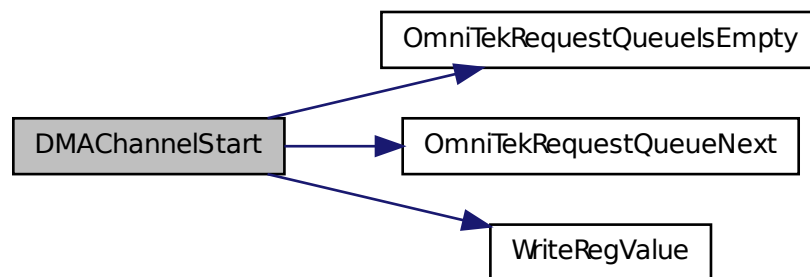
### 6.10.1.1 int DMAChannelStart ( PResource pChannel )

Start this DMA Channel.

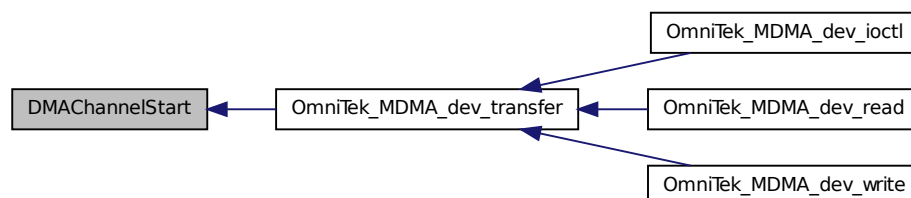
#### Parameters

[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:



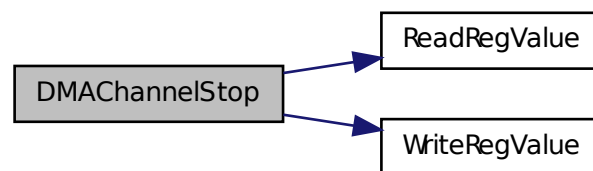
### 6.10.1.2 void DMACHannelStop ( PResource *pChannel* )

Stop this DMA Channel.

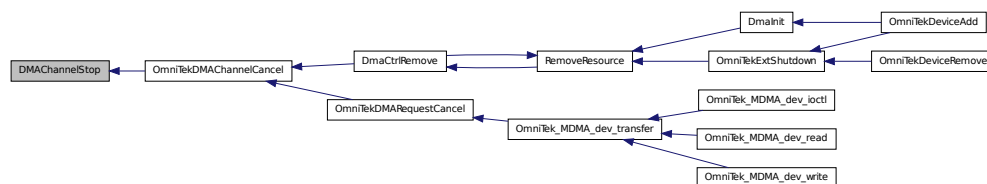
#### Parameters

[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:



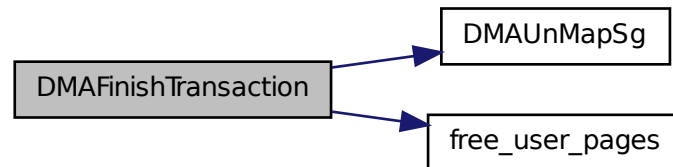
### 6.10.1.3 void DMAFinishTransaction ( struct \_OmniTekDmaTransactionContext \* *pTransaction* )

Finish this transaction.

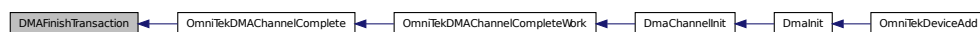
#### Parameters

[in] *pTransaction* Pointer to transaction to finish

Here is the call graph for this function:



Here is the caller graph for this function:



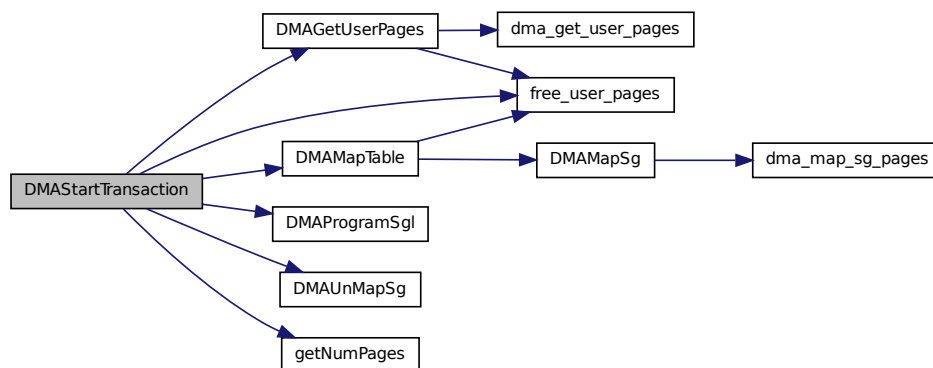
#### 6.10.1.4 int DMAStartTransaction ( struct \_OmniTekDmaTransactionContext \* *pTransaction* )

Start this transaction.

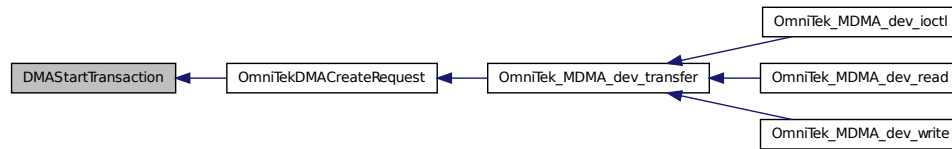
##### Parameters

[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.

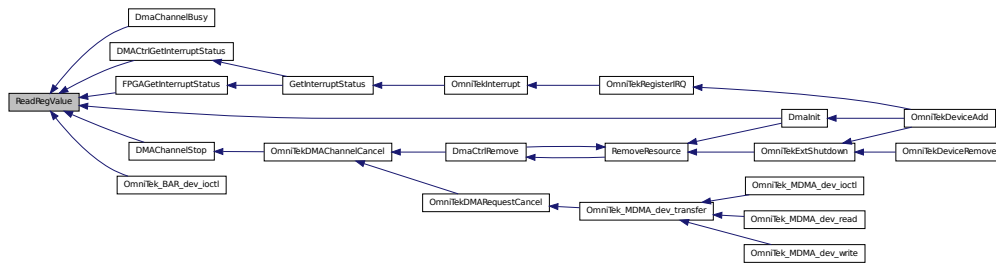
### Parameters

- [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

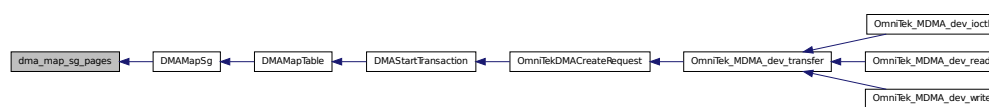
- [in] *sg* scatterlist struct pointer
- [in] *pages* array pointer to pages struct

- [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

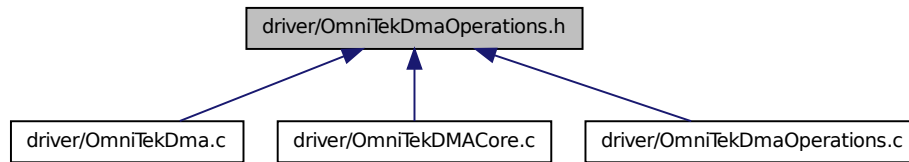
## 6.13 driver/OmniTekDmaOperations.h File Reference

```

#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.

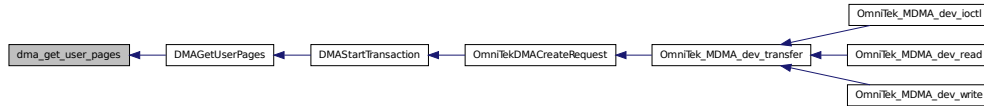
#### Parameters

- [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 vmass 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.

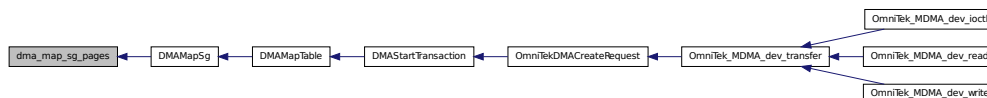
#### Parameters

- [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**

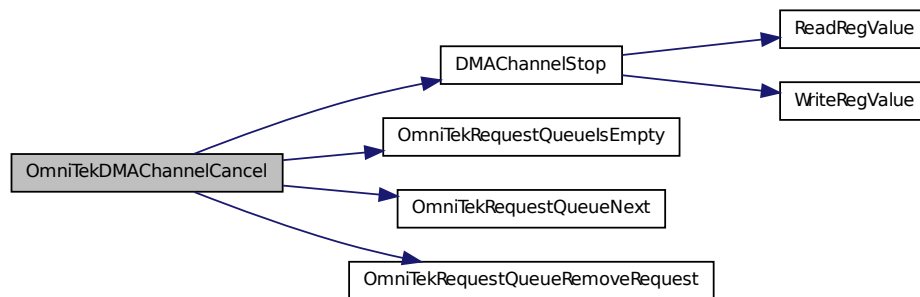
- [in] *pChannel* resource to cancel.

**Returns**

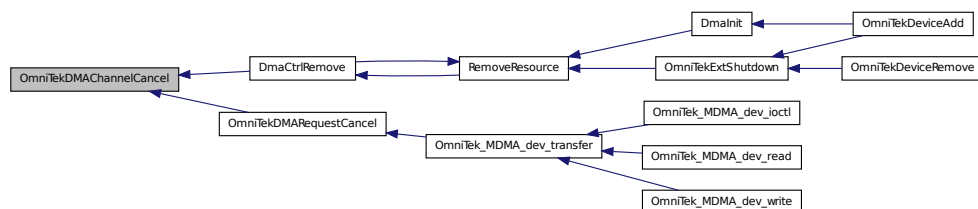
0 on success

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:

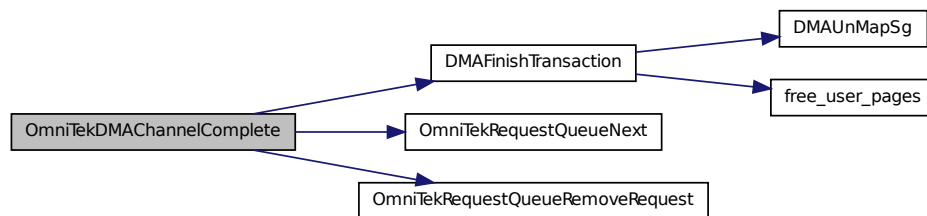
**6.14.1.2 void OmniTekDMAChannelComplete ( PResource *pChannel* )**

Complete a request on this channel.

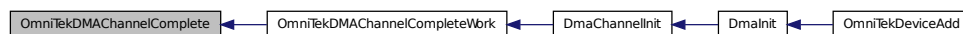
**Parameters**

[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 )

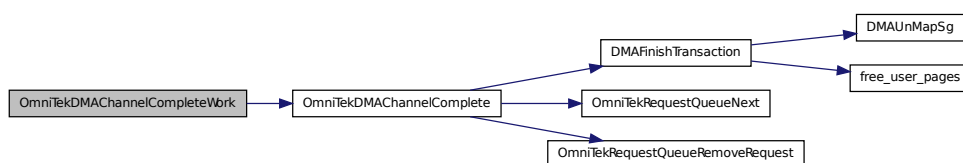
Complete work function.

#### Parameters

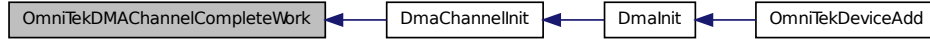
[in] *work* struct pointer

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).

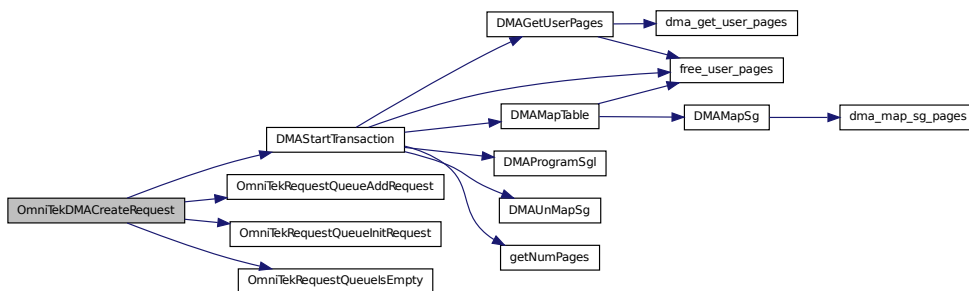
#### Parameters

- [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

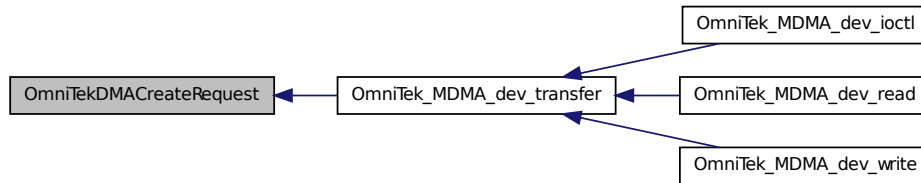
0 on success, -EBUSY if the request was accepted into the pending queue, or error code if there is a problem creating the request

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.

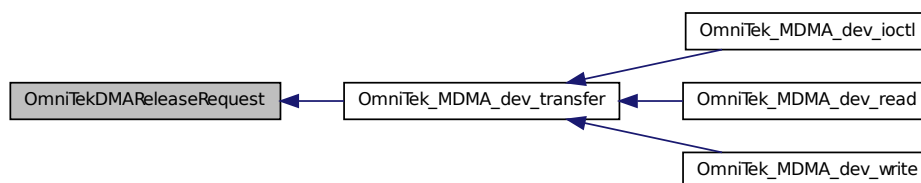
##### Parameters

[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



- 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

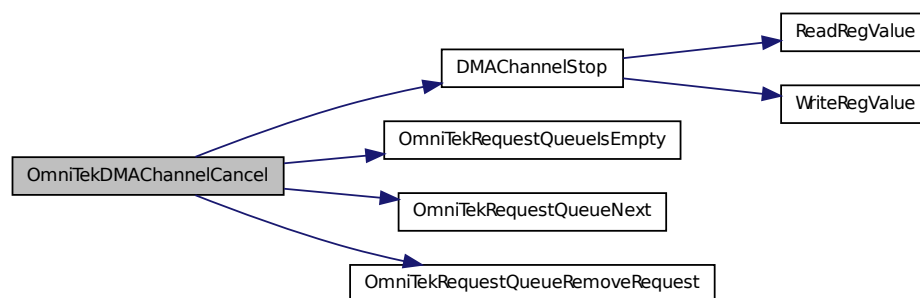
[in] *pChannel* resource to cancel.

#### Returns

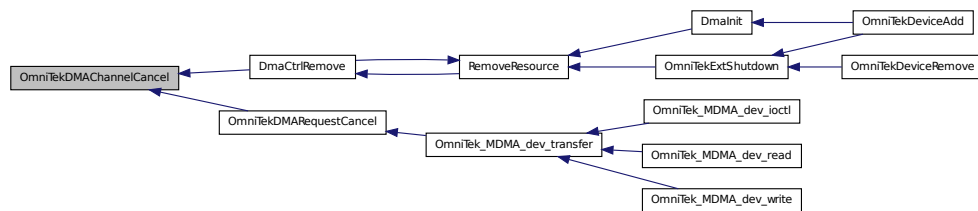
0 on success

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:



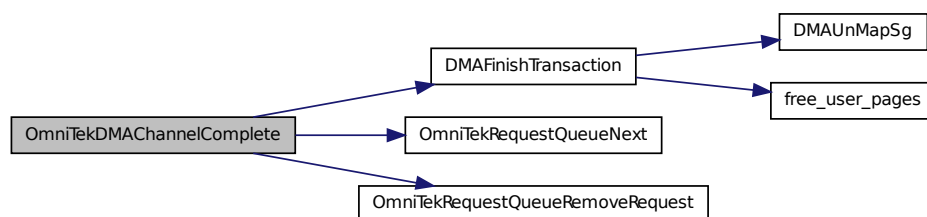
#### 6.15.1.2 void OmniTekDMAChannelComplete ( PResource *pChannel* )

Complete a request on this channel.

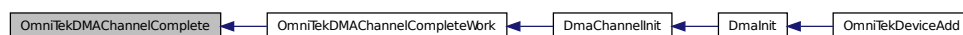
##### Parameters

[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* )

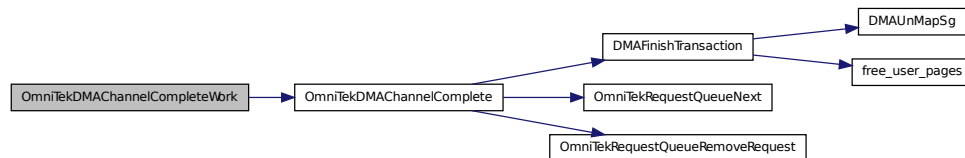
Complete work function.

##### Parameters

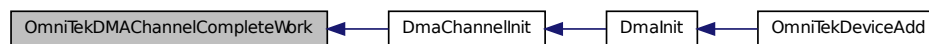
[in] *work* struct pointer

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).

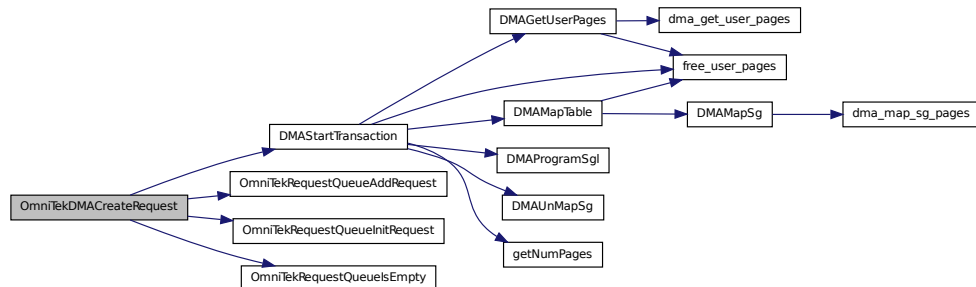
#### Parameters

- [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

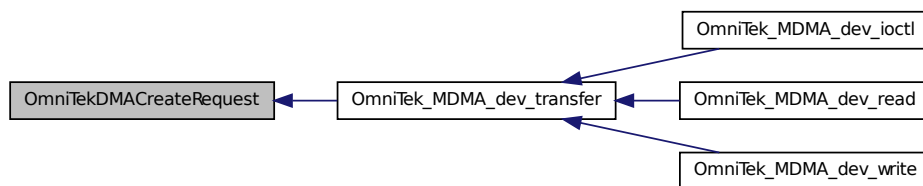
#### Returns

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.

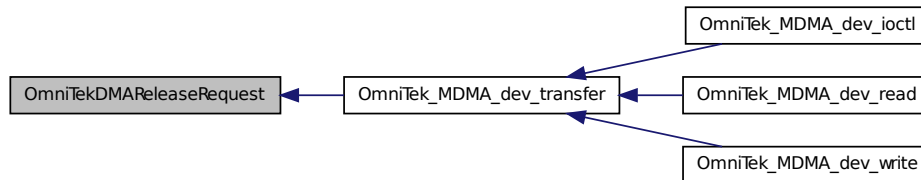
##### Parameters

[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

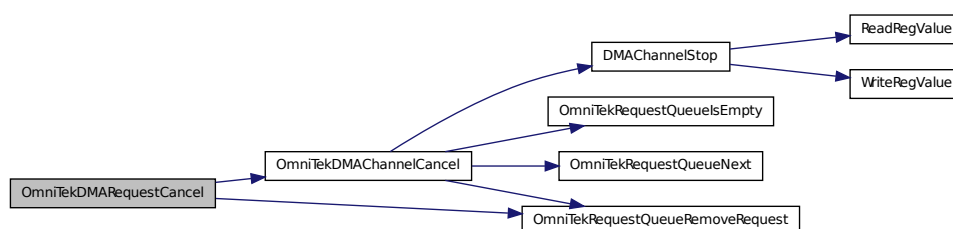
[in] *pTransaction* pointer to transaction to cancel

##### Returns

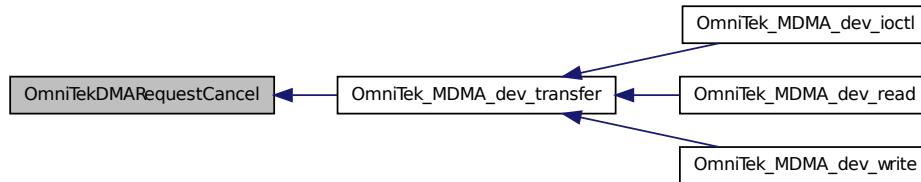
0 on success

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

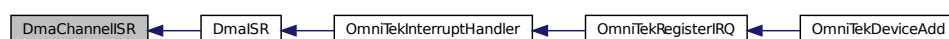
### 6.16.1.1 void DmaChannelISR ( PResource pChannel )

Interrupt service routine for DMA Channels.

#### Parameters

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

Here is the caller graph for this function:



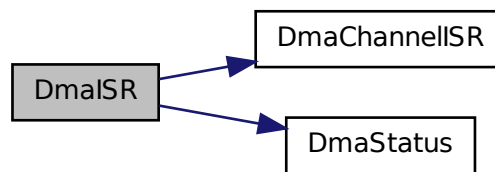
### 6.16.1.2 void DmaISR ( POMNITEK\_INTERFACE\_EXTENSION *pExt* )

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.

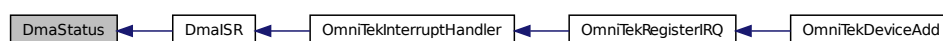
#### Parameters

[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

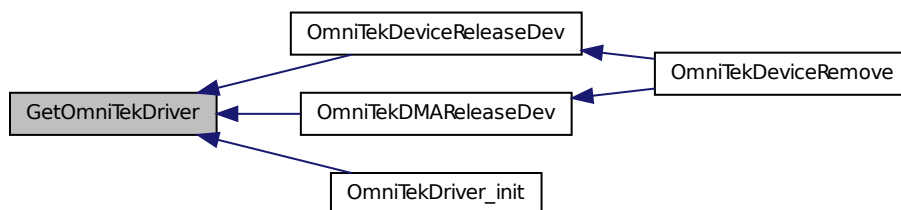
##### Parameters

[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

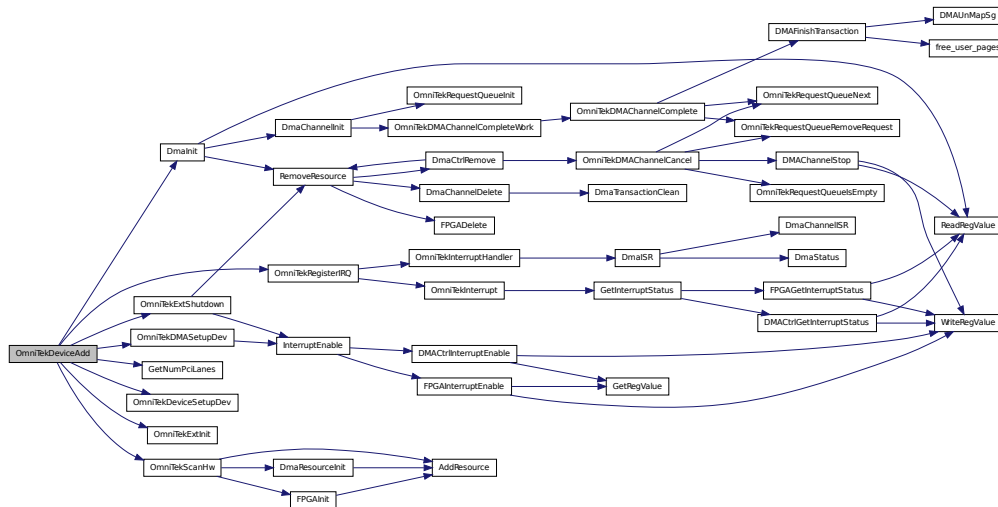
[in] *dev* Pointer to pci device that matches our vendor/device IDs

[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:



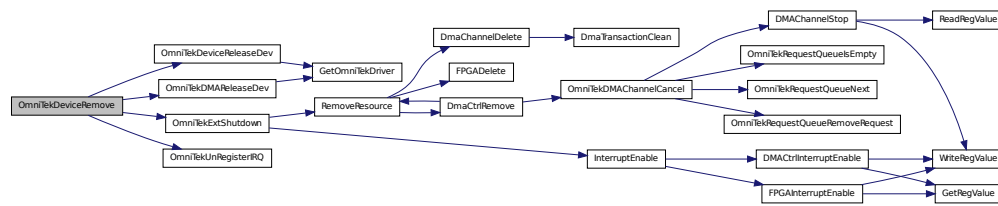
**6.16.1.11** `static void OmniTekDeviceRemove ( struct pci_dev * dev ) [static]`

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

Success 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

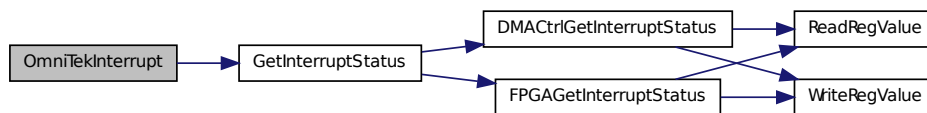
- [in] *irq* IRQ Number that fired
- [in] *dev\_id* Device ID That interrupted

##### 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.

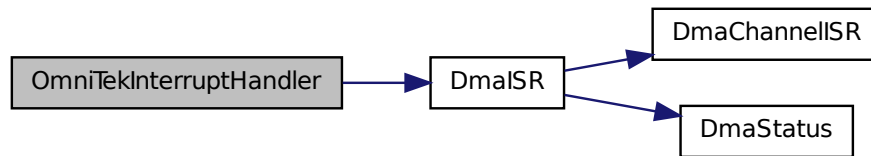
##### Parameters

- [in] *irq* IRQ number that has been triggered
- [in] *dev\_id* Pointer to device extension that has interrupted

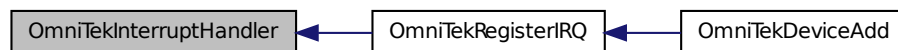
##### Returns

Result depending on whether interrupt has been handled

Here is the call graph for this function:



Here is the caller graph for this function:



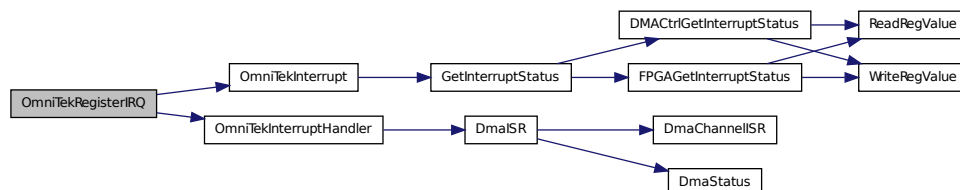
#### 6.16.1.17 void OmniTekRegisterIRQ ( POMNITEK\_INTERFACE\_EXTENSION *pExt* )

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)`

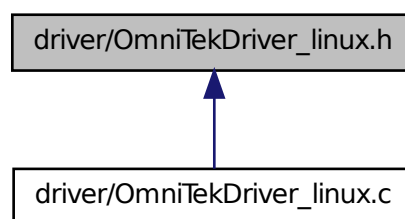
6.16.2.4 `atomic_t irqTotal = ATOMIC_INIT(0)`

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

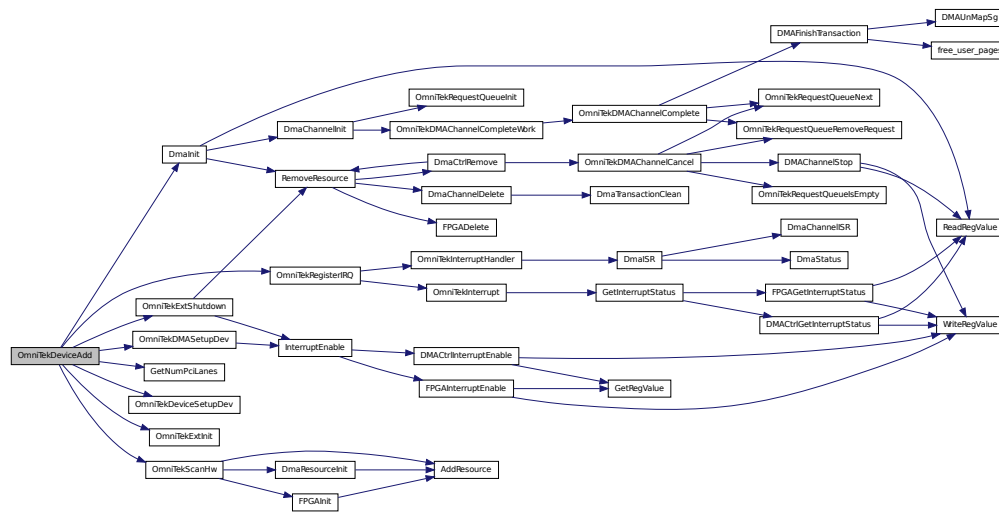
[in] *dev* Pointer to pci device that matches our vendor/device IDs

[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

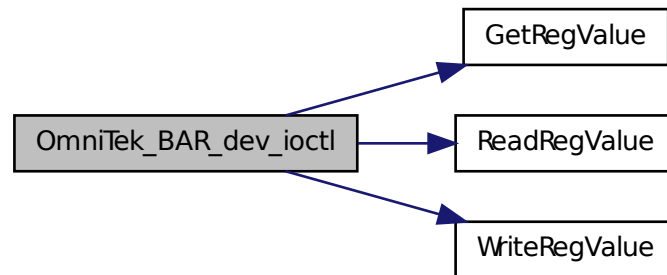
- static struct [\\_OmniTek\\_dev](#) [OmniTekBoardev](#)
- struct file\_operations [OmniTek\\_BAR\\_dev\\_fops](#)

#### 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 )`

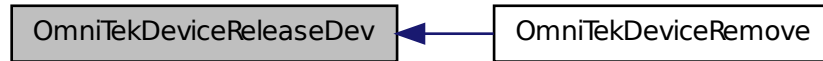
BAR Device close/release.

#### 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:**

```

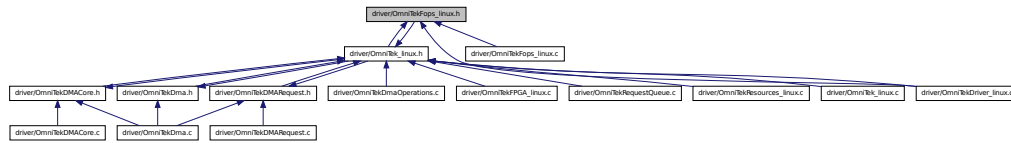
{
    .owner = THIS_MODULE,
    .ioctl = OmniTek_BAR_dev_ioctl,
    .open = OmniTek_BAR_dev_open,
    .release = OmniTek_BAR_dev_release
}
  
```

### 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:

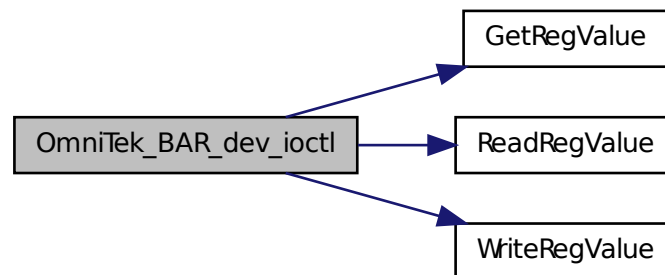
*OMNITEK\_DEV\_BAR* BAR type device  
*OMNITEK\_DEV\_RESOURCE* Resource type device  
*OMNITEK\_DEV\_MDMA* MDMA Type Device

### 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

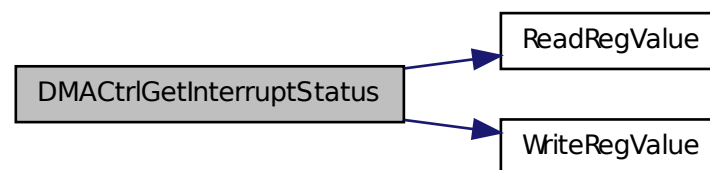
- void [FPGAInit](#) ([Resource](#) \*pResource)  
*Initialise the FPGA resource.*
- void [FPGADelete](#) ([PResource](#) pFPGA)
- void [FPGAInterruptEnable](#) ([PResource](#) pFPGA, u32 Interrupt, bool Enable)
- void [DMACtrlInterruptEnable](#) ([PResource](#) pDmaCtrl, u32 Interrupt, bool Enable)
- 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)

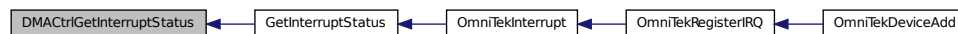
## 6.20.1 Function Documentation

### 6.20.1.1 u32 DMACtrlGetInterruptStatus ( PResource pDmaCtrl )

Here is the call graph for this function:

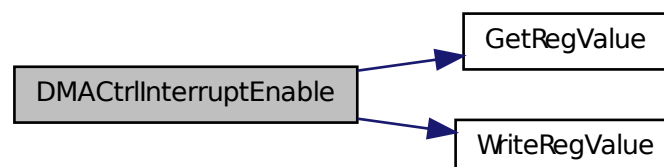


Here is the caller graph for this function:

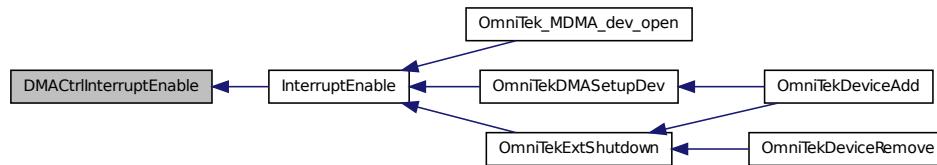


### 6.20.1.2 void DMACtrlInterruptEnable ( PResource pDmaCtrl, u32 Interrupt, bool Enable )

Here is the call graph for this function:

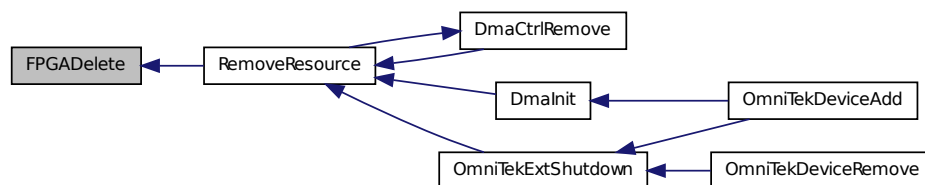


Here is the caller graph for this function:



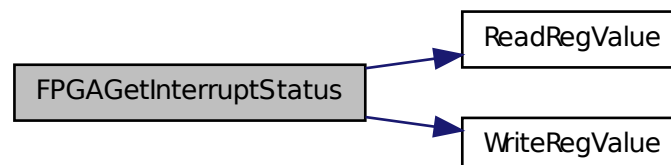
### 6.20.1.3 void FPGADelete ( PResource pFPGA )

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.

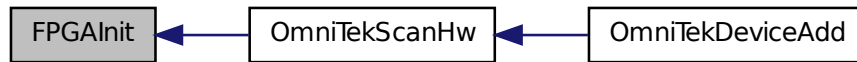
##### Parameters

*pResource* Pointer to (FPGA) resource to initialise

Here is the call graph for this function:

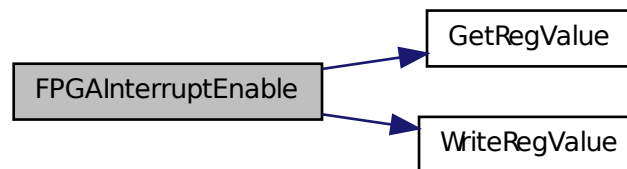


Here is the caller graph for this function:

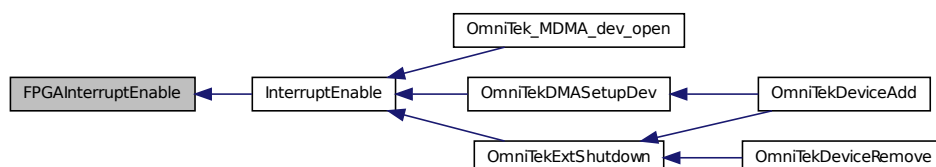


#### 6.20.1.7 void FPGAInterruptEnable ( PResource *pFPGA*, u32 *Interrupt*, bool *Enable* )

Here is the call graph for this function:



Here is the caller graph for this function:



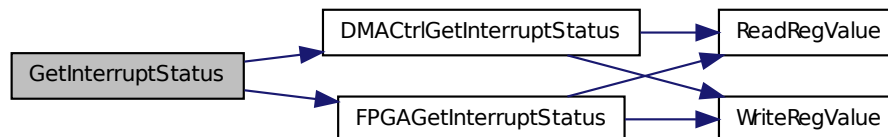
### 6.20.1.8 u64 FPGAReadTime ( PResource *pFPGA* )

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:

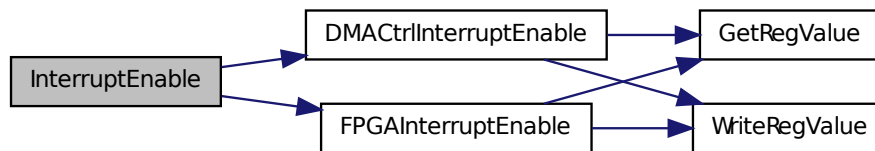


Here is the caller graph for this function:

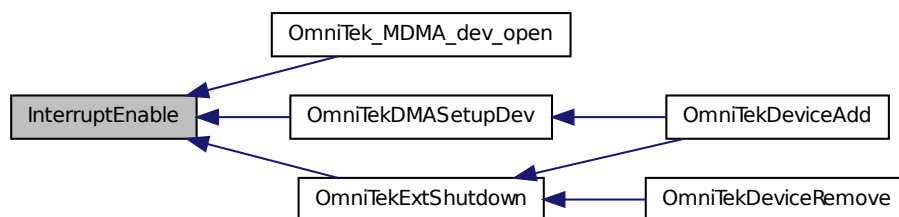


### 6.20.1.10 void InterruptEnable ( POMNITEK\_INTERFACE\_EXTENSION *pExt*, u32 *Interrupt*, bool *Enable* )

Here is the call graph for this function:

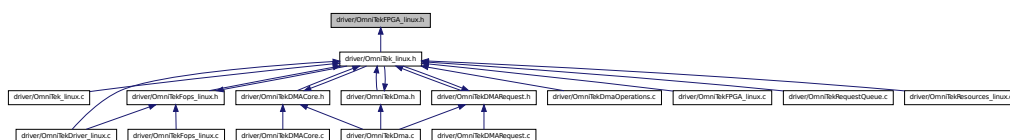


Here is the caller graph for this function:



## 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

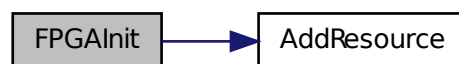
**6.21.1.5 void FPGAInit ( Resource \* pResource )**

Initialise the FPGA resource.

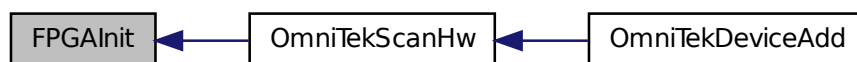
**Parameters**

*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

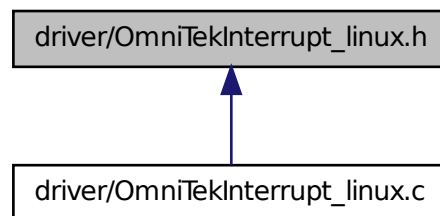
*Enable* Enable or disable interrupts

## 6.22 driver/OmniTekInterrupt\_linux.c File Reference

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

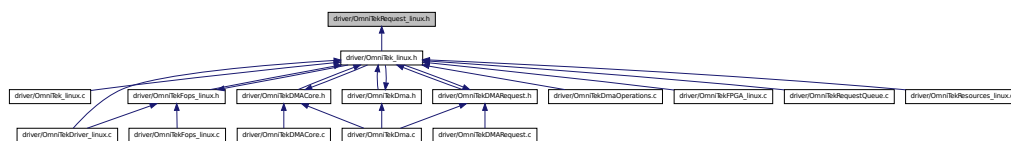
## 6.23 driver/OmniTekInterrupt\_linux.h File Reference

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

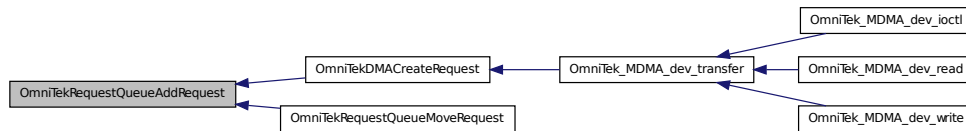
[in] *pQueue* Request Queue to add to

[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

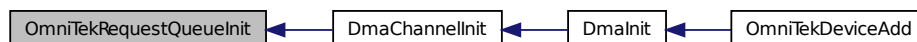
#### 6.25.1.3 void OmniTekRequestQueueInit ( struct \_OmniTekRequestQueue \* *pQueue*, char \* *Name* )

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.

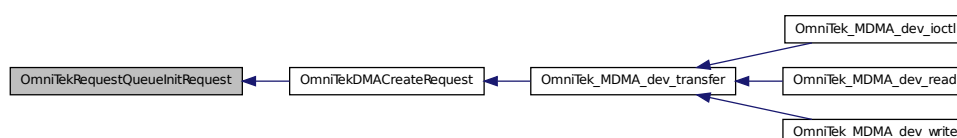
**Parameters**

[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.

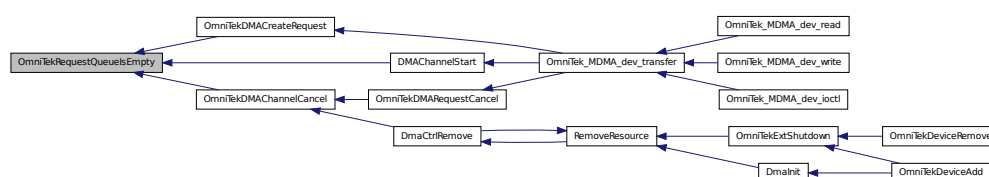
**Parameters**

[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

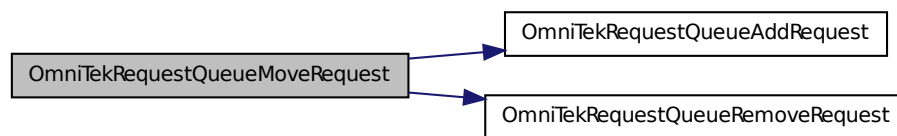
[in] *pNewQueue* Request Queue to add to

[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

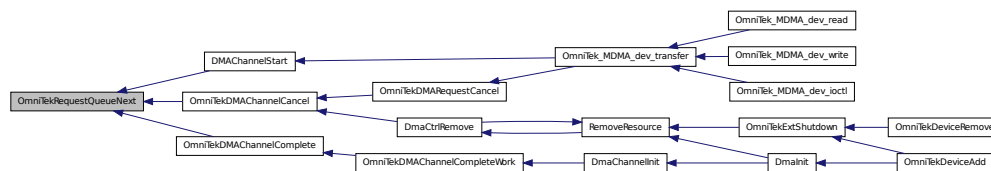
[in] *pQueue* Request Queue to retrieve request from

[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.



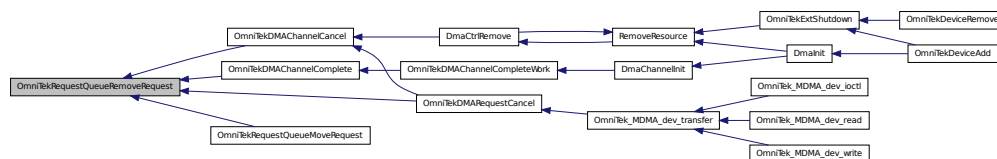
**Parameters**

- [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**

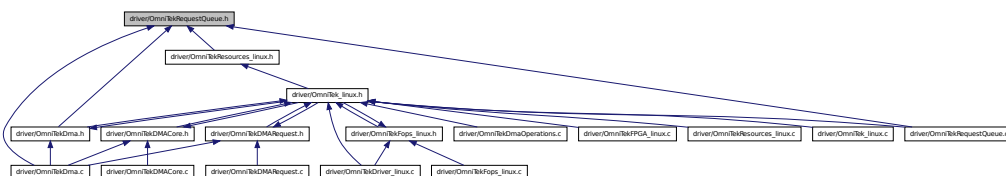
- [in] *pQueue* Request Queue to check

**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

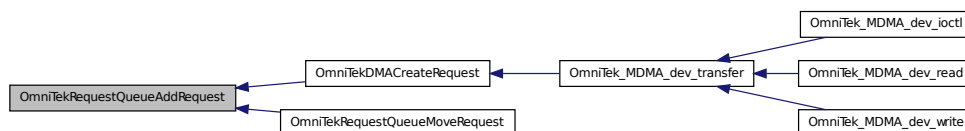
[in] *pQueue* Request Queue to add to

[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.

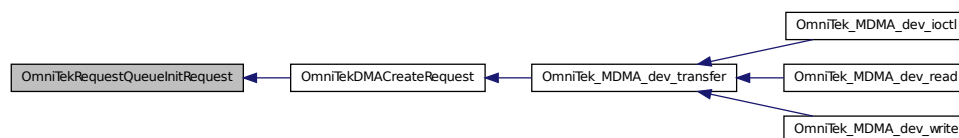
#### Parameters

- [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.

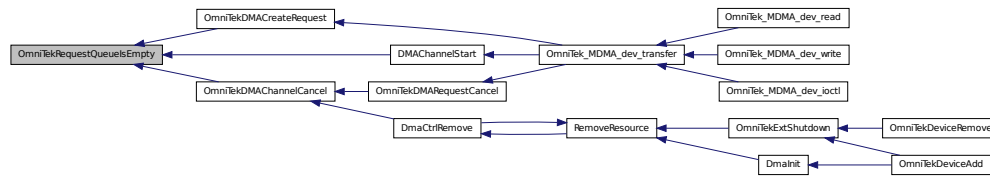
#### Parameters

- [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.

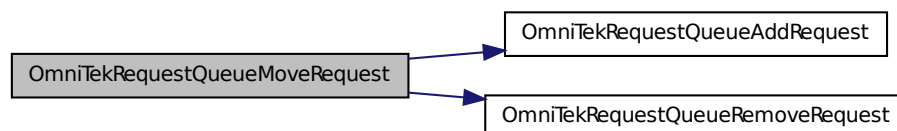
**Parameters**

- [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.

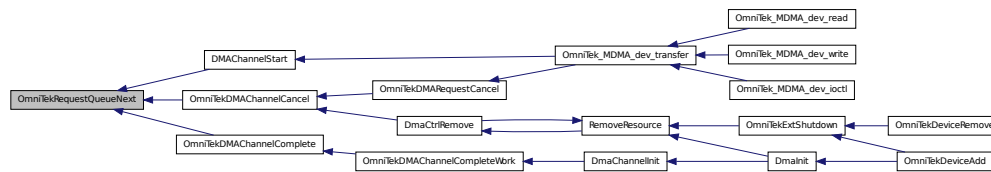
**Parameters**

- [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 \_OmniTekRequestQueueObj \* *pRequest* )

Remove Request to a queue.

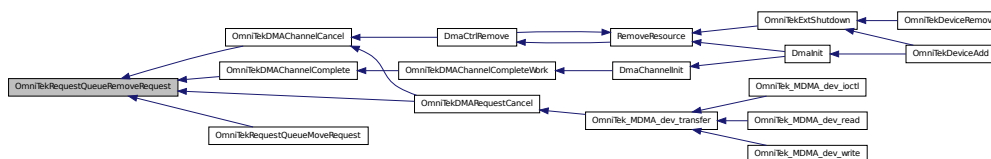
**Parameters**

- [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

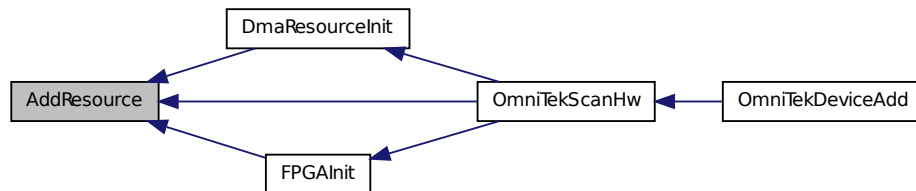
**Bar** Resource BAR number

**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

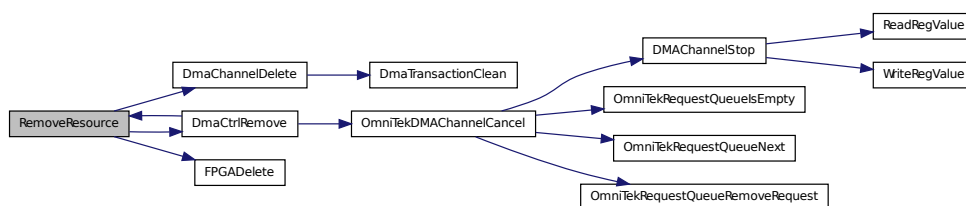
##### Parameters

*pResource* Pointer to resource to be removed

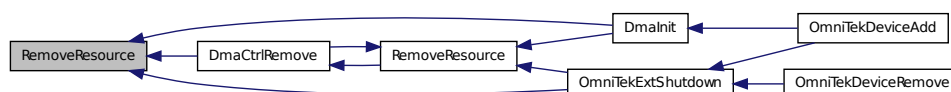
##### Returns

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]`

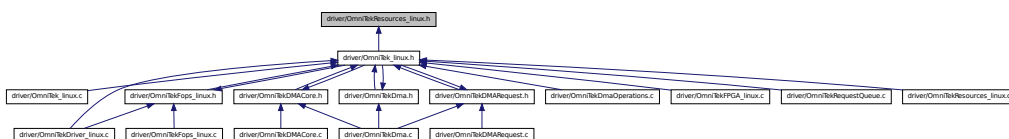
Here is the caller graph for this function:



## 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

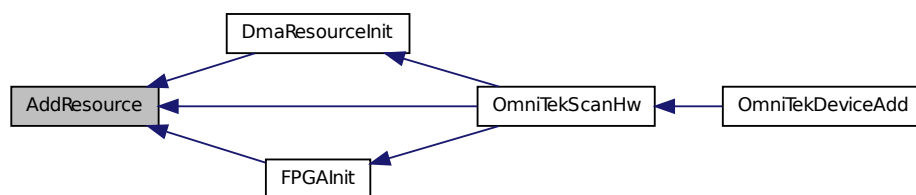
*Bar* Resource BAR number

*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

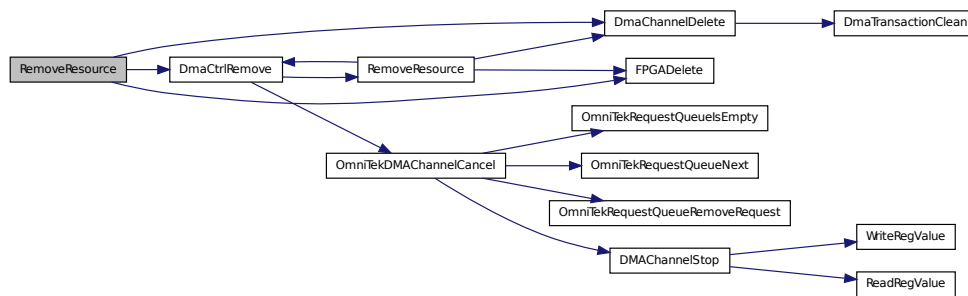
**Parameters**

*pResource* Pointer to resource to be removed

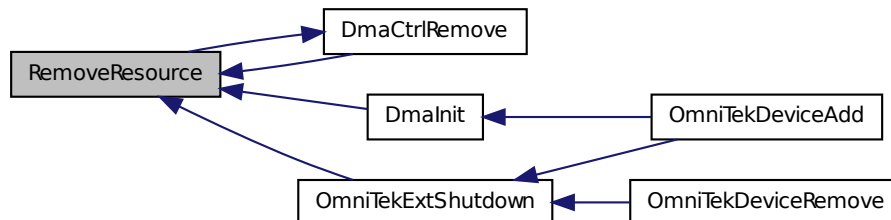
**Returns**

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

- [\\_DmaChannel, 12](#)
  - [Active, 14](#)
  - [DMA\\_64BIT\\_LADR, 14](#)
  - [DMA\\_64BIT\\_PADR, 14](#)
  - [FDMA\\_Enabled, 14](#)
  - [FDMA\\_Read, 14](#)
  - [FDMA\\_Write, 14](#)
  - [Index, 14](#)
  - [ISRWork, 15](#)
  - [Label, 15](#)
  - [Object, 15](#)
  - [Pending, 15](#)
  - [Running, 15](#)
  - [SglBuffer, 15](#)
  - [SpinLock, 15](#)
- [\\_DmaCtrl, 16](#)
  - [Channels, 17](#)
  - [DMA\\_64BIT\\_LADR, 17](#)
  - [DMA\\_64BIT\\_PADR, 17](#)
  - [DMA\\_Wait\\_Queue, 17](#)
  - [DMA\\_Work\\_Queue, 17](#)
  - [DmaInterrupts, 17](#)
  - [nChannels, 17](#)
  - [nFDMABoth, 17](#)
  - [nFDMARead, 17](#)
  - [nFDMAWrite, 17](#)
  - [nMDMA, 18](#)
- [\\_DmaSglBuffer, 18](#)
  - [Allocated, 19](#)
  - [CommonBuffer, 19](#)
  - [DMA\\_Handle, 19](#)
  - [SpinLock, 19](#)
- [\\_DmaSglBuffer::\\_Allocated, 9](#)
  - [Free, 9](#)
  - [Memory, 9](#)
  - [Size, 9](#)
- [\\_FPGACtrl, 19](#)
  - [FPGAType, 20](#)
  - [Version, 20](#)
- [\\_GeneralCtrl, 21](#)
  - [Bar, 21](#)
  - [Initialised, 21](#)
  - [RegisterOffset, 21](#)
- [\\_InterruptData, 21](#)
  - [nInterruptStatus, 23](#)
- [pExt, 23](#)
- [\\_OMNITEK\\_DEVTYPES](#)
  - [OmniTekFops\\_linux.h, 132](#)
- [\\_OMNITEK\\_INTERFACE\\_EXTENSION, 26](#)
  - [Device, 28](#)
  - [irq, 28](#)
  - [IrqLock, 28](#)
  - [MemBar, 28](#)
  - [nBars, 28](#)
  - [Object, 28](#)
  - [pDmaCtrl, 28](#)
  - [pDriver, 29](#)
  - [pFlashProgrammer, 29](#)
  - [pFPGA, 29](#)
  - [RegValue, 29](#)
  - [Resources, 29](#)
  - [SpinLock, 29](#)
- [\\_OMNITEK\\_INTERFACE\\_EXTENSION::\\_bar\\_-  
    registers, 10](#)
  - [num\\_regs, 10](#)
  - [regs, 10](#)
- [\\_OmniTekDmaTransactionContext, 29](#)
  - [CompleteFunc, 31](#)
  - [DMACoreInfo, 31](#)
  - [generic, 31](#)
  - [kiocb, 31](#)
  - [pChannel, 31](#)
  - [QueueObject, 31](#)
  - [request, 31](#)
  - [Sgl, 31](#)
  - [state, 31](#)
  - [type, 31](#)
  - [typeInfo, 32](#)
  - [Xfer, 32](#)
- [\\_OmniTekDmaTransactionContext::\\_-  
    DMACoreInfo, 15](#)
  - [first\\_page, 16](#)
  - [last\\_page, 16](#)
  - [num\\_pages, 16](#)
  - [offset, 16](#)
  - [pages, 16](#)
  - [sgt, 16](#)
- [\\_OmniTekDmaTransactionContext::\\_SglInfo, 44](#)
  - [CoherentMem, 44](#)
  - [DmaMem, 44](#)

- `_OmniTekDmaTransactionContext::_XferInfo`, 44
  - Buffer, 44
  - LocalAddr, 44
  - Size, 45
  - Write, 45
- `_OmniTekDriver`, 32
  - dev, 32
  - Devices, 32
  - Extensions, 32
  - NextMinor, 32
  - pci\_driver, 32
  - ResourcePool, 32
- `_OmniTekKernelRequest`, 32
  - inBufferSize, 34
  - outBufferSize, 34
  - pExt, 34
  - pInBuffer, 34
  - pOutBuffer, 34
  - userRequest, 34
- `_OmniTekRequestQueue`, 34
  - Entries, 35
  - Name, 35
  - SpinLock, 35
- `_OmniTekRequestQueueObject`, 35
  - Object, 36
  - pCurrentQueue, 36
- `_OmniTekUserRequest`, 36
  - inBufferSize, 38
  - kernelRequest, 38
  - outBufferSize, 38
  - pid, 38
  - pInBuffer, 38
  - pOutBuffer, 38
- `_OmniTek_dev`, 23
  - cDev, 25
  - firstMinor, 25
  - major, 25
  - nMinors, 25
  - Object, 25
  - pExt, 25
  - Type, 25
- `_PCI_BAR_INFO`, 38
  - IsIoMapped, 39
  - Physical, 39
  - pVa, 39
  - Size, 39
- `_RequestStatus`
  - OmniTekRequest\_linux.h, 144
- `_Resource`, 39
  - CapVersion, 41
  - General, 41
  - LockedBy, 41
  - NumRegisters, 41
  - Object, 41
  - pExt, 41
  - ReferenceCount, 41
  - SpinLock, 41
  - Type, 41
  - u, 41
- `_Resource::_ResourceExtension`, 42
  - DmaChannel, 43
  - DmaCtrl, 43
  - FPGACtrl, 43
- `_ResourceVersion`, 43
  - Major, 43
  - Minor, 43
- `__attribute__`
  - OmniTek\_Driver.mod.c, 49
- `_dma_interrupt_info`, 11
  - chan\_done, 12
  - chan\_int\_counts, 12
  - chan\_interrupts, 12
  - sem, 12
- `_dma_interrupt_info::chan_int_counts`, 10
  - n\_event\_ints, 10
  - n\_sg\_ints, 10
- Active
  - `_DmaChannel`, 14
- AddResource
  - OmniTekResources\_linux.c, 155
  - OmniTekResources\_linux.h, 161
- Allocated
  - `_DmaSglBuffer`, 19
- Bar
  - `_GeneralCtrl`, 21
- Buffer
  - `_OmniTekDmaTransactionContext::_XferInfo`, 44
- CapVersion
  - `_Resource`, 41
- cDev
  - `_OmniTek_dev`, 25
- chan\_done
  - `_dma_interrupt_info`, 12
- chan\_int\_counts
  - `_dma_interrupt_info`, 12
- chan\_interrupts
  - `_dma_interrupt_info`, 12
- Channels
  - `_DmaCtrl`, 17
- CoherentMem
  - `_OmniTekDmaTransactionContext::_SglInfo`, 44
- CommonBuffer
  - `_DmaSglBuffer`, 19

- CompleteFunc
  - \_OmniTekDmaTransactionContext, [31](#)
- DECLARE\_WAIT\_QUEUE\_HEAD
  - OmniTekDma.c, [69](#)
- dev
  - \_OmniTekDriver, [32](#)
- Device
  - \_OMNITEK\_INTERFACE\_EXTENSION, [28](#)
- Devices
  - \_OmniTekDriver, [32](#)
- DMA
  - OmniTek\_debug.h, [48](#)
- DMA\_64BIT\_LADR
  - \_DmaChannel, [14](#)
  - \_DmaCtrl, [17](#)
- DMA\_64BIT\_PADR
  - \_DmaChannel, [14](#)
  - \_DmaCtrl, [17](#)
- DMA\_CHANNEL
  - OmniTekDma.h, [81](#)
- DMA\_CHANNEL\_BYTES\_XFER
  - OmniTekDma.h, [81](#)
- DMA\_CHANNEL\_CSR
  - OmniTekDma.h, [81](#)
- DMA\_CHANNEL\_DPR
  - OmniTekDma.h, [81](#)
- DMA\_CHANNEL\_DPR\_HIGH
  - OmniTekDma.h, [81](#)
- DMA\_CHANNEL\_FDMA
  - OmniTekDma.h, [81](#)
- DMA\_CHANNEL\_INT\_BIT\_EVENT
  - OmniTekDmaIsr\_linux.c, [99](#)
- DMA\_CHANNEL\_INT\_BIT\_SG
  - OmniTekDmaIsr\_linux.c, [99](#)
- DMA\_CHANNEL\_LADR
  - OmniTekDma.h, [81](#)
- DMA\_CHANNEL\_LADR\_HIGH
  - OmniTekDma.h, [81](#)
- DMA\_CHANNEL\_OFFSET
  - OmniTekDma.h, [81](#)
- DMA\_CHANNEL\_PADR
  - OmniTekDma.h, [81](#)
- DMA\_CHANNEL\_PADR\_HIGH
  - OmniTekDma.h, [81](#)
- DMA\_CHANNEL\_READ
  - OmniTekDma.h, [81](#)
- DMA\_CHANNEL\_SIZE
  - OmniTekDma.h, [81](#)
- DMA\_CHANNEL\_SIZE\_HIGH
  - OmniTekDma.h, [81](#)
- DMA\_CHANNEL\_WRITE
  - OmniTekDma.h, [81](#)
- DMA\_CORE
  - OmniTek\_debug.h, [48](#)
- DMA\_CTRL\_CAP\_HEADER
  - OmniTekDma.h, [81](#)
- DMA\_CTRL\_CAP\_REG
  - OmniTekDma.h, [81](#)
- DMA\_CTRL\_INTERRUPT\_STATUS
  - OmniTekDma.h, [81](#)
- DMA\_DPR\_BIT\_DIRECTION\_TO\_PC
  - OmniTekDma.h, [81](#)
- DMA\_DPR\_BIT\_END\_OF\_CHAIN
  - OmniTekDma.h, [81](#)
- DMA\_DPR\_BIT\_INTERRUPT
  - OmniTekDma.h, [81](#)
- DMA\_FDMA\_CHANNEL
  - OmniTekDma.h, [81](#)
- DMA\_FDMA\_TYPE
  - OmniTekDma.h, [81](#)
- dma\_get\_user\_pages
  - OmniTekDmaOperations.c, [101](#)
  - OmniTekDmaOperations.h, [103](#)
- DMA\_Handle
  - \_DmaSglBuffer, [19](#)
- dma\_map\_sg\_init\_table\_and\_chain
  - OmniTekDmaOperations.c, [101](#)
  - OmniTekDmaOperations.h, [104](#)
- dma\_map\_sg\_pages
  - OmniTekDmaOperations.c, [101](#)
  - OmniTekDmaOperations.h, [104](#)
- dma\_map\_test\_scatterlist
  - OmniTekDmaOperations.c, [102](#)
  - OmniTekDmaOperations.h, [104](#)
- DMA\_MDMA\_CHANNEL
  - OmniTekDma.h, [82](#)
- DMA\_OPS
  - OmniTek\_debug.h, [48](#)
- DMA\_PAGES
  - OmniTek\_debug.h, [48](#)
- DMA\_REQUEST
  - OmniTek\_debug.h, [48](#)
- DMA\_SGL\_SIZE
  - OmniTekDma.h, [82](#)
- DMA\_Wait\_Queue
  - \_DmaCtrl, [17](#)
- DMA\_Work\_Queue
  - \_DmaCtrl, [17](#)
- DmaChannel
  - \_Resource::\_ResourceExtension, [43](#)
  - OmniTekResources\_linux.h, [160](#)
- DmaChannelBusy
  - OmniTekDma.c, [69](#)
  - OmniTekDma.h, [82](#)
- DmaChannelDelete
  - OmniTekDma.c, [69](#)
  - OmniTekDma.h, [82](#)

- DmaChannelFind
  - OmniTekResources\_linux.c, [156](#)
  - OmniTekResources\_linux.h, [161](#)
- DmaChannelInit
  - OmniTekDma.c, [70](#)
- DmaChannelISR
  - OmniTekDriver\_linux.c, [117](#)
- DMACHannelStart
  - OmniTekDMACore.c, [89](#)
  - OmniTekDMACore.h, [96](#)
- DMACHannelStop
  - OmniTekDMACore.c, [90](#)
  - OmniTekDMACore.h, [96](#)
- DmaChannelStop
  - OmniTekDma.c, [71](#)
  - OmniTekDma.h, [83](#)
- DMACoreInfo
  - \_OmniTekDmaTransactionContext, [31](#)
- DmaCtrl
  - \_Resource::\_ResourceExtension, [43](#)
  - OmniTekResources\_linux.h, [160](#)
- DMACtrlGetInterruptStatus
  - OmniTekFPGA\_linux.c, [134](#)
- DMACtrlInterruptEnable
  - OmniTekFPGA\_linux.c, [134](#)
- DmaCtrlRemove
  - OmniTekDma.c, [71](#)
  - OmniTekDma.h, [83](#)
- DMAFinishTransaction
  - OmniTekDMACore.c, [91](#)
  - OmniTekDMACore.h, [97](#)
- DMAGetUserPages
  - OmniTekDMACore.c, [92](#)
- DmaInit
  - OmniTekDma.c, [72](#)
  - OmniTekDma.h, [84](#)
- DmaInterrupts
  - \_DmaCtrl, [17](#)
- DmaISR
  - OmniTekDriver\_linux.c, [117](#)
- DMAMapSg
  - OmniTekDMACore.c, [92](#)
- DMAMapTable
  - OmniTekDMACore.c, [93](#)
- DmaMem
  - \_OmniTekDmaTransactionContext::\_SglInfo, [44](#)
- DMAProgramSgl
  - OmniTekDMACore.c, [93](#)
- DmaResourceInit
  - OmniTekDma.c, [73](#)
  - OmniTekDma.h, [84](#)
  - OmniTekResources\_linux.h, [162](#)
- DmaSglBuffer
  - OmniTekResources\_linux.h, [160](#)
- DMAStartTransaction
  - OmniTekDMACore.c, [93](#)
  - OmniTekDMACore.h, [98](#)
- DmaStatus
  - OmniTekDriver\_linux.c, [118](#)
- DmaTransactionClean
  - OmniTekDma.c, [73](#)
- DMAUnMapSg
  - OmniTekDMACore.c, [94](#)
- driver/OmniTekResource\_linux.o.d, [47](#)
- driver/OmniTek\_debug.h, [47](#)
- driver/OmniTek\_Driver.mod.c, [48](#)
- driver/OmniTek\_linux.c, [49](#)
- driver/OmniTek\_linux.h, [55](#)
- driver/OmniTek\_MainPage.h, [67](#)
- driver/OmniTekDma.c, [67](#)
- driver/OmniTekDma.h, [78](#)
- driver/OmniTekDMACore.c, [89](#)
- driver/OmniTekDMACore.h, [95](#)
- driver/OmniTekDmaIsr\_linux.c, [99](#)
- driver/OmniTekDmaOperations.c, [100](#)
- driver/OmniTekDmaOperations.h, [102](#)
- driver/OmniTekDMARequest.c, [105](#)
- driver/OmniTekDMARequest.h, [110](#)
- driver/OmniTekDriver\_linux.c, [116](#)
- driver/OmniTekDriver\_linux.h, [125](#)
- driver/OmniTekFops\_linux.c, [128](#)
- driver/OmniTekFops\_linux.h, [130](#)
- driver/OmniTekFPGA\_linux.c, [133](#)
- driver/OmniTekFPGA\_linux.h, [139](#)
- driver/OmniTekInterrupt\_linux.c, [142](#)
- driver/OmniTekInterrupt\_linux.h, [143](#)
- driver/OmniTekRequest\_linux.h, [143](#)
- driver/OmniTekRequestQueue.c, [144](#)
- driver/OmniTekRequestQueue.h, [149](#)
- driver/OmniTekResources\_linux.c, [155](#)
- driver/OmniTekResources\_linux.h, [158](#)
- DriverEntry
  - OmniTek\_linux.h, [60](#)
- Entries
  - \_OmniTekRequestQueue, [35](#)
- Extensions
  - \_OmniTekDriver, [32](#)
- FDMA\_Enabled
  - \_DmaChannel, [14](#)
- FDMA\_Read
  - \_DmaChannel, [14](#)
- FDMA\_Write
  - \_DmaChannel, [14](#)
- first\_page

- \_OmniTekDmaTransactionContext::\_-
  - DMACoreInfo, [16](#)
- firstMinor
  - \_OmniTek\_dev, [25](#)
- FOPS
  - OmniTek\_debug.h, [48](#)
- FPGAControl
  - OmniTekFPGA\_linux.h, [140](#)
- FPGACtrl
  - \_Resource::\_ResourceExtension, [43](#)
  - OmniTekResources\_linux.h, [160](#)
- FPGADelete
  - OmniTekFPGA\_linux.c, [135](#)
  - OmniTekFPGA\_linux.h, [140](#)
- FPGAGetInterruptStatus
  - OmniTekFPGA\_linux.c, [135](#)
- FPGAGetStandard
  - OmniTekFPGA\_linux.h, [140](#)
- FPGAGetTime
  - OmniTekFPGA\_linux.c, [136](#)
  - OmniTekFPGA\_linux.h, [141](#)
- FPGAInit
  - OmniTekFPGA\_linux.c, [136](#)
  - OmniTekFPGA\_linux.h, [141](#)
- FPGAInterruptEnable
  - OmniTekFPGA\_linux.c, [137](#)
- FPGAReadTime
  - OmniTekFPGA\_linux.c, [137](#)
  - OmniTekFPGA\_linux.h, [141](#)
- FPGAType
  - \_FPGACtrl, [20](#)
- Free
  - \_DmaSglBuffer::\_Allocated, [9](#)
- free\_user\_pages
  - OmniTekDMACore.c, [94](#)
- GENERAL
  - OmniTek\_debug.h, [48](#)
- General
  - \_Resource, [41](#)
- GeneralCtrl
  - OmniTekResources\_linux.h, [160](#)
- generic
  - \_OmniTekDmaTransactionContext, [31](#)
- GetInterruptStatus
  - OmniTekFPGA\_linux.c, [138](#)
  - OmniTekFPGA\_linux.h, [142](#)
- getNumPages
  - OmniTekDMACore.c, [95](#)
- GetNumPciLanes
  - OmniTekDriver\_linux.c, [118](#)
  - OmniTekDriver\_linux.h, [126](#)
- GetOmniTekDriver
  - OmniTek\_linux.h, [61](#)
- OmniTekDriver\_linux.c, [119](#)
- GetRegValue
  - OmniTek\_linux.c, [50](#)
  - OmniTek\_linux.h, [61](#)
- handlerCount
  - OmniTekDriver\_linux.c, [125](#)
- handlerHandled
  - OmniTekDriver\_linux.c, [125](#)
- ids
  - OmniTekDriver\_linux.h, [127](#)
- inBufferSize
  - \_OmniTekKernelRequest, [34](#)
  - \_OmniTekUserRequest, [38](#)
- Index
  - \_DmaChannel, [14](#)
- Initialised
  - \_GeneralCtrl, [21](#)
- InterruptData
  - OmniTek\_linux.h, [60](#)
- InterruptEnable
  - OmniTekFPGA\_linux.c, [138](#)
  - OmniTekFPGA\_linux.h, [142](#)
- IORESOURCE\_MEM\_64
  - OmniTekDriver\_linux.h, [126](#)
- IRQ
  - OmniTek\_debug.h, [48](#)
- irq
  - \_OMNITEK\_INTERFACE\_EXTENSION, [28](#)
- IrqLock
  - \_OMNITEK\_INTERFACE\_EXTENSION, [28](#)
- irqPend
  - OmniTekDriver\_linux.c, [125](#)
- irqTotal
  - OmniTekDriver\_linux.c, [125](#)
- IsIoMapped
  - \_PCI\_BAR\_INFO, [39](#)
- ISRWork
  - \_DmaChannel, [15](#)
- kernelRequest
  - \_OmniTekUserRequest, [38](#)
- kiocb
  - \_OmniTekDmaTransactionContext, [31](#)
- Label
  - \_DmaChannel, [15](#)
- last\_page
  - \_OmniTekDmaTransactionContext::\_-
  - DMACoreInfo, [16](#)
- list\_count
  - OmniTek\_linux.c, [50](#)
- LocalAddr

- [\\_OmniTekDmaTransactionContext::\\_XferInfo](#), 44
- [LockedBy](#)
  - [\\_Resource](#), 41
- [Major](#)
  - [\\_ResourceVersion](#), 43
- [major](#)
  - [\\_OmniTek\\_dev](#), 25
- [MAX\\_NUM\\_MEM\\_BARS](#)
  - [OmniTek\\_linux.h](#), 60
- [MemBar](#)
  - [\\_OMNITEK\\_INTERFACE\\_EXTENSION](#), 28
- [Memory](#)
  - [\\_DmaSglBuffer::\\_Allocated](#), 9
- [Minor](#)
  - [\\_ResourceVersion](#), 43
- [MODULE\\_ALIAS](#)
  - [OmniTek\\_Driver.mod.c](#), 49
- [MODULE\\_DEVICE\\_TABLE](#)
  - [OmniTekDriver\\_linux.c](#), 120
- [module\\_exit](#)
  - [OmniTekDriver\\_linux.c](#), 120
- [MODULE\\_INFO](#)
  - [OmniTek\\_Driver.mod.c](#), 49
- [module\\_init](#)
  - [OmniTekDriver\\_linux.c](#), 120
- [MODULE\\_LICENSE](#)
  - [OmniTekDriver\\_linux.c](#), 120
- [n\\_event\\_ints](#)
  - [\\_dma\\_interrupt\\_info::chan\\_int\\_counts](#), 10
- [n\\_sg\\_ints](#)
  - [\\_dma\\_interrupt\\_info::chan\\_int\\_counts](#), 10
- [Name](#)
  - [\\_OmniTekRequestQueue](#), 35
- [nBars](#)
  - [\\_OMNITEK\\_INTERFACE\\_EXTENSION](#), 28
- [nChannels](#)
  - [\\_DmaCtrl](#), 17
- [NextMinor](#)
  - [\\_OmniTekDriver](#), 32
- [nFDMABoth](#)
  - [\\_DmaCtrl](#), 17
- [nFDMARead](#)
  - [\\_DmaCtrl](#), 17
- [nFDMAWrite](#)
  - [\\_DmaCtrl](#), 17
- [nInterruptStatus](#)
  - [\\_InterruptData](#), 23
  - [OmniTekDriver\\_linux.c](#), 125
- [nMDMA](#)
  - [\\_DmaCtrl](#), 18
- [nMinors](#)
  - [\\_OmniTek\\_dev](#), 25
- [num\\_pages](#)
  - [\\_OmniTekDmaTransactionContext::\\_DMACoreInfo](#), 16
- [num\\_regs](#)
  - [\\_OMNITEK\\_INTERFACE\\_EXTENSION::\\_bar\\_registers](#), 10
- [NUM\\_REGS\\_PER\\_DMA\\_CHANNEL](#)
  - [OmniTekDma.h](#), 82
- [NumRegisters](#)
  - [\\_Resource](#), 41
- [Object](#)
  - [\\_DmaChannel](#), 15
  - [\\_OMNITEK\\_INTERFACE\\_EXTENSION](#), 28
  - [\\_OmniTekRequestQueueObject](#), 36
  - [\\_OmniTek\\_dev](#), 25
  - [\\_Resource](#), 41
- [offset](#)
  - [\\_OmniTekDmaTransactionContext::\\_DMACoreInfo](#), 16
- [OMNITEK\\_DEV\\_BAR](#)
  - [OmniTekFops\\_linux.h](#), 132
- [OMNITEK\\_DEV\\_MDMA](#)
  - [OmniTekFops\\_linux.h](#), 132
- [OMNITEK\\_DEV\\_RESOURCE](#)
  - [OmniTekFops\\_linux.h](#), 132
- [OMNITEK\\_REQUEST\\_CANCELLED](#)
  - [OmniTekRequest\\_linux.h](#), 144
- [OMNITEK\\_REQUEST\\_COMPLETE](#)
  - [OmniTekRequest\\_linux.h](#), 144
- [OMNITEK\\_REQUEST\\_INITIALIZED](#)
  - [OmniTekRequest\\_linux.h](#), 144
- [OMNITEK\\_REQUEST\\_KERNEL\\_USER\\_COPIED](#)
  - [OmniTekRequest\\_linux.h](#), 144
- [OMNITEK\\_REQUEST\\_PENDING](#)
  - [OmniTekRequest\\_linux.h](#), 144
- [OMNITEK\\_REQUEST\\_PROCESSED](#)
  - [OmniTekRequest\\_linux.h](#), 144
- [OMNITEK\\_REQUEST\\_UNINITIALIZED](#)
  - [OmniTekRequest\\_linux.h](#), 144
- [OMNITEK\\_REQUEST\\_USER\\_KERNEL\\_COPIED](#)
  - [OmniTekRequest\\_linux.h](#), 144
- [OmniTek\\_BAR\\_dev\\_fops](#)
  - [OmniTekFops\\_linux.c](#), 130
- [OmniTek\\_BAR\\_dev\\_ioctl](#)
  - [OmniTekFops\\_linux.c](#), 128
  - [OmniTekFops\\_linux.h](#), 132
- [OmniTek\\_BAR\\_dev\\_open](#)
  - [OmniTekFops\\_linux.c](#), 129
  - [OmniTekFops\\_linux.h](#), 132
- [OmniTek\\_BAR\\_dev\\_release](#)

- OmniTekFops\_linux.c, 129
- OmniTekFops\_linux.h, 132
- OmniTek\_debug.h
  - DMA, 48
  - DMA\_CORE, 48
  - DMA\_OPS, 48
  - DMA\_PAGES, 48
  - DMA\_REQUEST, 48
  - FOPS, 48
  - GENERAL, 48
  - IRQ, 48
  - OMNITEK\_DEBUG\_CATEGORIES, 48
  - OmniTekDebug, 48
  - REQUEST\_QUEUE, 48
  - RESOURCES, 48
- OMNITEK\_DEBUG\_CATEGORIES
  - OmniTek\_debug.h, 48
- OmniTek\_dev, 45
  - OmniTekFops\_linux.h, 132
- OmniTek\_DevTypes
  - OmniTekFops\_linux.h, 132
- OMNITEK\_DMA\_INTERRUPT
  - OmniTek\_linux.h, 60
- OMNITEK\_DMACTRL\_INTERRUPT\_MASK
  - OmniTek\_linux.h, 60
- omnitek\_driver
  - OmniTekDriver\_linux.c, 125
- OmniTek\_Driver.mod.c
  - \_\_attribute\_\_, 49
  - MODULE\_ALIAS, 49
  - MODULE\_INFO, 49
- OMNITEK\_INTERFACE\_EXTENSION
  - OmniTek\_linux.h, 60
- OMNITEK\_INTERRUPT\_MASK
  - OmniTek\_linux.h, 60
- OmniTek\_linux.c
  - GetRegValue, 50
  - list\_count, 50
  - OmniTekExtInit, 51
  - OmniTekExtShutdown, 51
  - OmniTekGetCapList, 52
  - OmniTekScanHw, 53
  - ReadRegValue, 53
  - WriteRegValue, 54
- OmniTek\_linux.h
  - DriverEntry, 60
  - GetOmniTekDriver, 61
  - GetRegValue, 61
  - InterruptData, 60
  - MAX\_NUM\_MEM\_BARS, 60
  - OMNITEK\_DMA\_INTERRUPT, 60
  - OMNITEK\_DMACTRL\_INTERRUPT\_-  
MASK, 60
  - OMNITEK\_INTERFACE\_EXTENSION, 60
  - OMNITEK\_INTERRUPT\_MASK, 60
  - OmniTekDriver, 60
  - OmniTekEvtDeviceD0Entry, 62
  - OmniTekEvtDe-  
viceD0EntryPostInterruptsEnabled,  
62
  - OmniTekEvtDeviceD0Exit, 62
  - OmniTekEvtDevicePrepareHardware, 63
  - OmniTekEvtDeviceProbe, 63
  - OmniTekEvtDeviceReleaseHardware, 63
  - OmniTekExtInit, 63
  - OmniTekExtShutdown, 64
  - OmniTekGetCapList, 65
  - OmniTekIoctl, 65
  - OmniTekScanHw, 66
  - PCI\_BAR\_INFO, 60
  - PCI\_NUM\_BARS, 60
  - POMNITEK\_INTERFACE\_EXTENSION, 60
  - ReadHWValue, 60
  - ReadHWValueByte, 60
  - ReadRegValue, 66
  - STATUS\_INVALID\_PARAMETER\_1, 60
  - STATUS\_INVALID\_PARAMETER\_2, 60
  - STATUS\_INVALID\_PARAMETER\_3, 60
  - STATUS\_OMNITEK\_ILLEGAL\_-  
SESSION\_ID, 60
  - STATUS\_OMNITEK\_MEMORY\_ERROR,  
60
  - STATUS\_OMNITEK\_RESOURCE\_-  
COMMAND\_ERROR, 60
  - STATUS\_OMNITEK\_RESOURCE\_-  
INVALID, 60
  - STATUS\_OMNITEK\_RESOURCE\_-  
LOCKED, 60
  - WriteHWValue, 60
  - WriteHWValueByte, 60
  - WriteRegValue, 67
- OmniTek\_MDMA\_dev\_complete
  - OmniTekDma.c, 74
- OmniTek\_MDMA\_dev\_fops
  - OmniTekDma.c, 78
- OmniTek\_MDMA\_dev\_ioctl
  - OmniTekDma.c, 74
  - OmniTekDma.h, 85
- OmniTek\_MDMA\_dev\_open
  - OmniTekDma.c, 75
  - OmniTekDma.h, 85
- OmniTek\_MDMA\_dev\_read
  - OmniTekDma.c, 75
  - OmniTekDma.h, 86
- OmniTek\_MDMA\_dev\_release
  - OmniTekDma.c, 75
  - OmniTekDma.h, 86
- OmniTek\_MDMA\_dev\_transfer



- OmniTekDma.c, [75](#)
- OmniTekDma.h, [86](#)
- OmniTek\_MDMA\_dev\_write
  - OmniTekDma.c, [76](#)
  - OmniTekDma.h, [87](#)
- OmniTekBoarddev
  - OmniTekFops\_linux.c, [130](#)
- OmniTekDebug
  - OmniTek\_debug.h, [48](#)
- OmniTekDeviceAdd
  - OmniTekDriver\_linux.c, [120](#)
  - OmniTekDriver\_linux.h, [126](#)
- OmniTekDeviceReleaseDev
  - OmniTekFops\_linux.c, [129](#)
  - OmniTekFops\_linux.h, [132](#)
- OmniTekDeviceRemove
  - OmniTekDriver\_linux.c, [120](#)
  - OmniTekDriver\_linux.h, [127](#)
- OmniTekDeviceSetupDev
  - OmniTekFops\_linux.c, [130](#)
  - OmniTekFops\_linux.h, [133](#)
- OmniTekDma.c
  - DECLARE\_WAIT\_QUEUE\_HEAD, [69](#)
  - DmaChannelBusy, [69](#)
  - DmaChannelDelete, [69](#)
  - DmaChannelInit, [70](#)
  - DmaChannelStop, [71](#)
  - DmaCtrlRemove, [71](#)
  - DmaInit, [72](#)
  - DmaResourceInit, [73](#)
  - DmaTransactionClean, [73](#)
  - OmniTek\_MDMA\_dev\_complete, [74](#)
  - OmniTek\_MDMA\_dev\_fops, [78](#)
  - OmniTek\_MDMA\_dev\_ioctl, [74](#)
  - OmniTek\_MDMA\_dev\_open, [75](#)
  - OmniTek\_MDMA\_dev\_read, [75](#)
  - OmniTek\_MDMA\_dev\_release, [75](#)
  - OmniTek\_MDMA\_dev\_transfer, [75](#)
  - OmniTek\_MDMA\_dev\_write, [76](#)
  - OmniTekDMAReleaseDev, [77](#)
  - OmniTekDMASetupDev, [77](#)
  - OmniTekMDMADev, [78](#)
  - TRANSACTION\_WAIT\_MSECS, [69](#)
- OmniTekDma.h
  - DMA\_CHANNEL, [81](#)
  - DMA\_CHANNEL\_BYTES\_XFER, [81](#)
  - DMA\_CHANNEL\_CSR, [81](#)
  - DMA\_CHANNEL\_DPR, [81](#)
  - DMA\_CHANNEL\_DPR\_HIGH, [81](#)
  - DMA\_CHANNEL\_FDMA, [81](#)
  - DMA\_CHANNEL\_LADR, [81](#)
  - DMA\_CHANNEL\_LADR\_HIGH, [81](#)
  - DMA\_CHANNEL\_OFFSET, [81](#)
  - DMA\_CHANNEL\_PADR, [81](#)
  - DMA\_CHANNEL\_PADR\_HIGH, [81](#)
  - DMA\_CHANNEL\_READ, [81](#)
  - DMA\_CHANNEL\_SIZE, [81](#)
  - DMA\_CHANNEL\_SIZE\_HIGH, [81](#)
  - DMA\_CHANNEL\_WRITE, [81](#)
  - DMA\_CTRL\_CAP\_HEADER, [81](#)
  - DMA\_CTRL\_CAP\_REG, [81](#)
  - DMA\_CTRL\_INTERRUPT\_STATUS, [81](#)
  - DMA\_DPR\_BIT\_DIRECTION\_TO\_PC, [81](#)
  - DMA\_DPR\_BIT\_END\_OF\_CHAIN, [81](#)
  - DMA\_DPR\_BIT\_INTERRUPT, [81](#)
  - DMA\_FDMA\_CHANNEL, [81](#)
  - DMA\_FDMA\_TYPE, [81](#)
  - DMA\_MDMA\_CHANNEL, [82](#)
  - DMA\_SGL\_SIZE, [82](#)
  - DmaChannelBusy, [82](#)
  - DmaChannelDelete, [82](#)
  - DmaChannelStop, [83](#)
  - DmaCtrlRemove, [83](#)
  - DmaInit, [84](#)
  - DmaResourceInit, [84](#)
  - NUM\_REGS\_PER\_DMA\_CHANNEL, [82](#)
  - OmniTek\_MDMA\_dev\_ioctl, [85](#)
  - OmniTek\_MDMA\_dev\_open, [85](#)
  - OmniTek\_MDMA\_dev\_read, [86](#)
  - OmniTek\_MDMA\_dev\_release, [86](#)
  - OmniTek\_MDMA\_dev\_transfer, [86](#)
  - OmniTek\_MDMA\_dev\_write, [87](#)
  - OmniTekDMAReleaseDev, [88](#)
  - OmniTekDMASetupDev, [88](#)
  - OmniTekDmaTransactionContext, [82](#)
  - POmniTekDmaTransactionContext, [82](#)
  - SGL\_ITEM\_SIZE, [82](#)
- OmniTekDMACHannelCancel
  - OmniTekDMARquest.c, [105](#)
  - OmniTekDMARquest.h, [111](#)
- OmniTekDMACHannelComplete
  - OmniTekDMARquest.c, [106](#)
  - OmniTekDMARquest.h, [112](#)
- OmniTekDMACHannelCompleteWork
  - OmniTekDMARquest.c, [107](#)
  - OmniTekDMARquest.h, [112](#)
- OmniTekDMACore.c
  - DMACHannelStart, [89](#)
  - DMACHannelStop, [90](#)
  - DMAFinishTransaction, [91](#)
  - DMAGetUserPages, [92](#)
  - DMAMapSg, [92](#)
  - DMAMapTable, [93](#)
  - DMAProgramSgl, [93](#)
  - DMAStartTransaction, [93](#)
  - DMAUnMapSg, [94](#)
  - free\_user\_pages, [94](#)
  - getNumPages, [95](#)



- OmniTekDMACore.h
  - DMAChannelStart, [96](#)
  - DMAChannelStop, [96](#)
  - DMAFinishTransaction, [97](#)
  - DMAStartTransaction, [98](#)
- OmniTekDMACreateRequest
  - OmniTekDMARequest.c, [108](#)
  - OmniTekDMARequest.h, [113](#)
- OmniTekDMAFastIsr
  - OmniTekDmaIsr\_linux.c, [99](#)
- OmniTekDmaInterruptComplete
  - OmniTekResources\_linux.h, [160](#)
- OmniTekDmaIsr\_linux.c
  - DMA\_CHANNEL\_INT\_BIT\_EVENT, [99](#)
  - DMA\_CHANNEL\_INT\_BIT\_SG, [99](#)
  - OmniTekDMAFastIsr, [99](#)
  - OmniTekDMASlowIsr, [99](#)
  - ReadRegValue, [99](#)
- OmniTekDmaOperations.c
  - dma\_get\_user\_pages, [101](#)
  - dma\_map\_sg\_init\_table\_and\_chain, [101](#)
  - dma\_map\_sg\_pages, [101](#)
  - dma\_map\_test\_scatterlist, [102](#)
- OmniTekDmaOperations.h
  - dma\_get\_user\_pages, [103](#)
  - dma\_map\_sg\_init\_table\_and\_chain, [104](#)
  - dma\_map\_sg\_pages, [104](#)
  - dma\_map\_test\_scatterlist, [104](#)
- OmniTekDMAReleaseDev
  - OmniTekDma.c, [77](#)
  - OmniTekDma.h, [88](#)
- OmniTekDMAReleaseRequest
  - OmniTekDMARequest.c, [109](#)
  - OmniTekDMARequest.h, [114](#)
- OmniTekDMARequest.c
  - OmniTekDMAChannelCancel, [105](#)
  - OmniTekDMAChannelComplete, [106](#)
  - OmniTekDMAChannelCompleteWork, [107](#)
  - OmniTekDMACreateRequest, [108](#)
  - OmniTekDMAReleaseRequest, [109](#)
  - OmniTekDMARequestCancel, [109](#)
- OmniTekDMARequest.h
  - OmniTekDMAChannelCancel, [111](#)
  - OmniTekDMAChannelComplete, [112](#)
  - OmniTekDMAChannelCompleteWork, [112](#)
  - OmniTekDMACreateRequest, [113](#)
  - OmniTekDMAReleaseRequest, [114](#)
  - OmniTekDMARequestCancel, [115](#)
- OmniTekDMARequestCancel
  - OmniTekDMARequest.c, [109](#)
  - OmniTekDMARequest.h, [115](#)
- OmniTekDMASetupDev
  - OmniTekDma.c, [77](#)
  - OmniTekDma.h, [88](#)
- OmniTekDMASlowIsr
  - OmniTekDmaIsr\_linux.c, [99](#)
- OmniTekDmaTransactionContext
  - OmniTekDma.h, [82](#)
- OmniTekDriver, [45](#)
  - OmniTek\_linux.h, [60](#)
- OmniTekDriver\_exit
  - OmniTekDriver\_linux.c, [121](#)
- OmniTekDriver\_init
  - OmniTekDriver\_linux.c, [121](#)
- OmniTekDriver\_linux.c
  - DmaChannelISR, [117](#)
  - DmaISR, [117](#)
  - DmaStatus, [118](#)
  - GetNumPciLanes, [118](#)
  - GetOmniTekDriver, [119](#)
  - handlerCount, [125](#)
  - handlerHandled, [125](#)
  - irqPend, [125](#)
  - irqTotal, [125](#)
  - MODULE\_DEVICE\_TABLE, [120](#)
  - module\_exit, [120](#)
  - module\_init, [120](#)
  - MODULE\_LICENSE, [120](#)
  - nInterruptStatus, [125](#)
  - omnitek\_driver, [125](#)
  - OmniTekDeviceAdd, [120](#)
  - OmniTekDeviceRemove, [120](#)
  - OmniTekDriver\_exit, [121](#)
  - OmniTekDriver\_init, [121](#)
  - OmniTekGetDeviceId, [121](#)
  - OmniTekInterrupt, [121](#)
  - OmniTekInterruptHandler, [122](#)
  - OmniTekRegisterIRQ, [123](#)
  - OmniTekUnRegisterIRQ, [124](#)
- OmniTekDriver\_linux.h
  - GetNumPciLanes, [126](#)
  - ids, [127](#)
  - IORESOURCE\_MEM\_64, [126](#)
  - OmniTekDeviceAdd, [126](#)
  - OmniTekDeviceRemove, [127](#)
  - OmniTekGetDeviceId, [127](#)
  - USE\_IRQ\_THREAD, [126](#)
- OmniTekEvtDeviceD0Entry
  - OmniTek\_linux.h, [62](#)
- OmniTekEvtDeviceD0EntryPostInterruptsEnabled
  - OmniTek\_linux.h, [62](#)
- OmniTekEvtDeviceD0Exit
  - OmniTek\_linux.h, [62](#)
- OmniTekEvtDevicePrepareHardware
  - OmniTek\_linux.h, [63](#)
- OmniTekEvtDeviceProbe
  - OmniTek\_linux.h, [63](#)
- OmniTekEvtDeviceReleaseHardware

- OmniTek\_linux.h, 63
- OmniTekExtInit
  - OmniTek\_linux.c, 51
  - OmniTek\_linux.h, 63
- OmniTekExtShutdown
  - OmniTek\_linux.c, 51
  - OmniTek\_linux.h, 64
- OmniTekFops\_linux.h
  - OMNITEK\_DEV\_BAR, 132
  - OMNITEK\_DEV\_MDMA, 132
  - OMNITEK\_DEV\_RESOURCE, 132
- OmniTekFops\_linux.c
  - OmniTek\_BAR\_dev\_fops, 130
  - OmniTek\_BAR\_dev\_ioctl, 128
  - OmniTek\_BAR\_dev\_open, 129
  - OmniTek\_BAR\_dev\_release, 129
  - OmniTekBoarddev, 130
  - OmniTekDeviceReleaseDev, 129
  - OmniTekDeviceSetupDev, 130
- OmniTekFops\_linux.h
  - \_OMNITEK\_DEVTYPES, 132
  - OmniTek\_BAR\_dev\_ioctl, 132
  - OmniTek\_BAR\_dev\_open, 132
  - OmniTek\_BAR\_dev\_release, 132
  - OmniTek\_dev, 132
  - OmniTek\_DevTypes, 132
  - OmniTekDeviceReleaseDev, 132
  - OmniTekDeviceSetupDev, 133
- OmniTekFPGA\_linux.c
  - DMACtrlGetInterruptStatus, 134
  - DMACtrlInterruptEnable, 134
  - FPGADelete, 135
  - FPGAGetInterruptStatus, 135
  - FPGAGetTime, 136
  - FPGAInit, 136
  - FPGAInterruptEnable, 137
  - FPGAReadTime, 137
  - GetInterruptStatus, 138
  - InterruptEnable, 138
- OmniTekFPGA\_linux.h
  - FPGAControl, 140
  - FPGADelete, 140
  - FPGAGetStandard, 140
  - FPGAGetTime, 141
  - FPGAInit, 141
  - FPGAReadTime, 141
  - GetInterruptStatus, 142
  - InterruptEnable, 142
- OmniTekGetCapList
  - OmniTek\_linux.c, 52
  - OmniTek\_linux.h, 65
- OmniTekGetDeviceId
  - OmniTekDriver\_linux.c, 121
  - OmniTekDriver\_linux.h, 127
- OmniTekInterrupt
  - OmniTekDriver\_linux.c, 121
- OmniTekInterruptHandler
  - OmniTekDriver\_linux.c, 122
- OmniTekIoctl
  - OmniTek\_linux.h, 65
- OmniTekKernelRequest, 45
  - OmniTekRequest\_linux.h, 144
- OmniTekMDMADev
  - OmniTekDma.c, 78
- OmniTekRegisterIRQ
  - OmniTekDriver\_linux.c, 123
- OmniTekRequest\_linux.h
  - OMNITEK\_REQUEST\_CANCELLED, 144
  - OMNITEK\_REQUEST\_COMPLETE, 144
  - OMNITEK\_REQUEST\_INITIALIZED, 144
  - OMNITEK\_REQUEST\_KERNEL\_USER\_-COPIED, 144
  - OMNITEK\_REQUEST\_PENDING, 144
  - OMNITEK\_REQUEST\_PROCESSED, 144
  - OMNITEK\_REQUEST\_UNINITIALIZED, 144
  - OMNITEK\_REQUEST\_USER\_KERNEL\_-COPIED, 144
- OmniTekRequest\_linux.h
  - \_RequestStatus, 144
  - OmniTekKernelRequest, 144
  - OmniTekUserRequest, 144
  - POmniTekKernelRequest, 144
  - POmniTekUserRequest, 144
  - RequestStatus, 144
- OmniTekRequestQueue
  - OmniTekRequestQueue.h, 151
- OmniTekRequestQueue.c
  - OmniTekRequestQueueAddRequest, 145
  - OmniTekRequestQueueContains, 146
  - OmniTekRequestQueueInit, 146
  - OmniTekRequestQueueInitRequest, 146
  - OmniTekRequestQueueIsEmpty, 147
  - OmniTekRequestQueueMoveRequest, 147
  - OmniTekRequestQueueNext, 148
  - OmniTekRequestQueueRemoveRequest, 148
  - OmniTekRequestQueueSize, 149
- OmniTekRequestQueue.h
  - OmniTekRequestQueue, 151
  - OmniTekRequestQueueAddRequest, 151
  - OmniTekRequestQueueContains, 151
  - OmniTekRequestQueueInit, 151
  - OmniTekRequestQueueInitRequest, 152
  - OmniTekRequestQueueIsEmpty, 152
  - OmniTekRequestQueueMoveRequest, 153
  - OmniTekRequestQueueNext, 153
  - OmniTekRequestQueueObject, 151
  - OmniTekRequestQueueRemoveRequest, 154

- OmniTekRequestQueueSize, 154
- POmniTekRequestQueue, 151
- POmniTekRequestQueueObject, 151
- OmniTekRequestQueueAddRequest
  - OmniTekRequestQueue.c, 145
  - OmniTekRequestQueue.h, 151
- OmniTekRequestQueueContains
  - OmniTekRequestQueue.c, 146
  - OmniTekRequestQueue.h, 151
- OmniTekRequestQueueInit
  - OmniTekRequestQueue.c, 146
  - OmniTekRequestQueue.h, 151
- OmniTekRequestQueueInitRequest
  - OmniTekRequestQueue.c, 146
  - OmniTekRequestQueue.h, 152
- OmniTekRequestQueueIsEmpty
  - OmniTekRequestQueue.c, 147
  - OmniTekRequestQueue.h, 152
- OmniTekRequestQueueMoveRequest
  - OmniTekRequestQueue.c, 147
  - OmniTekRequestQueue.h, 153
- OmniTekRequestQueueNext
  - OmniTekRequestQueue.c, 148
  - OmniTekRequestQueue.h, 153
- OmniTekRequestQueueObject
  - OmniTekRequestQueue.h, 151
- OmniTekRequestQueueRemoveRequest
  - OmniTekRequestQueue.c, 148
  - OmniTekRequestQueue.h, 154
- OmniTekRequestQueueSize
  - OmniTekRequestQueue.c, 149
  - OmniTekRequestQueue.h, 154
- OmniTekResources\_linux.c
  - AddResource, 155
  - DmaChannelFind, 156
  - ReleaseResource, 156
  - RemoveResource, 157
  - ResourceCheck, 157
  - ResourceFind, 158
  - ResourceItemRelease, 158
- OmniTekResources\_linux.h
  - AddResource, 161
  - DmaChannel, 160
  - DmaChannelFind, 161
  - DmaCtrl, 160
  - DmaResourceInit, 162
  - DmaSglBuffer, 160
  - FPGACtrl, 160
  - GeneralCtrl, 160
  - OmnitekDmaInterruptComplete, 160
  - PDmaChannel, 160
  - PDmaCtrl, 161
  - PResource, 161
  - ReleaseResource, 162
  - RemoveResource, 162
  - Resource, 161
  - ResourceCheck, 163
  - ResourceControl, 163
  - ResourceRegisterWatchdog, 164
  - ResourceVersion, 161
- OmniTekScanHw
  - OmniTek\_linux.c, 53
  - OmniTek\_linux.h, 66
- OmniTekUnRegisterIRQ
  - OmniTekDriver\_linux.c, 124
- OmniTekUserRequest, 46
  - OmniTekRequest\_linux.h, 144
- outBufferSize
  - \_OmniTekKernelRequest, 34
  - \_OmniTekUserRequest, 38
- pages
  - \_OmniTekDmaTransactionContext::\_-  
DMACoreInfo, 16
- pChannel
  - \_OmniTekDmaTransactionContext, 31
- PCI\_BAR\_INFO
  - OmniTek\_linux.h, 60
- pci\_driver
  - \_OmniTekDriver, 32
- PCI\_NUM\_BARS
  - OmniTek\_linux.h, 60
- pCurrentQueue
  - \_OmniTekRequestQueueObject, 36
- PDmaChannel
  - OmniTekResources\_linux.h, 160
- PDmaCtrl
  - OmniTekResources\_linux.h, 161
- pDmaCtrl
  - \_OMNITEK\_INTERFACE\_EXTENSION, 28
- pDriver
  - \_OMNITEK\_INTERFACE\_EXTENSION, 29
- Pending
  - \_DmaChannel, 15
- pExt
  - \_InterruptData, 23
  - \_OmniTekKernelRequest, 34
  - \_OmniTek\_dev, 25
  - \_Resource, 41
- pFlashProgrammer
  - \_OMNITEK\_INTERFACE\_EXTENSION, 29
- pFPGA
  - \_OMNITEK\_INTERFACE\_EXTENSION, 29
- Physical
  - \_PCI\_BAR\_INFO, 39
- pid
  - \_OmniTekUserRequest, 38
- pInBuffer

- [\\_OmniTekKernelRequest](#), 34
- [\\_OmniTekUserRequest](#), 38
- [POMNITEK\\_INTERFACE\\_EXTENSION](#)
  - [OmniTek\\_linux.h](#), 60
- [POmniTekDmaTransactionContext](#)
  - [OmniTekDma.h](#), 82
- [POmniTekKernelRequest](#)
  - [OmniTekRequest\\_linux.h](#), 144
- [POmniTekRequestQueue](#)
  - [OmniTekRequestQueue.h](#), 151
- [POmniTekRequestQueueObject](#)
  - [OmniTekRequestQueue.h](#), 151
- [POmniTekUserRequest](#)
  - [OmniTekRequest\\_linux.h](#), 144
- [pOutBuffer](#)
  - [\\_OmniTekKernelRequest](#), 34
  - [\\_OmniTekUserRequest](#), 38
- [PResource](#)
  - [OmniTekResources\\_linux.h](#), 161
- [pVa](#)
  - [\\_PCI\\_BAR\\_INFO](#), 39
- [QueueObject](#)
  - [\\_OmniTekDmaTransactionContext](#), 31
- [ReadHWValue](#)
  - [OmniTek\\_linux.h](#), 60
- [ReadHWValueByte](#)
  - [OmniTek\\_linux.h](#), 60
- [ReadRegValue](#)
  - [OmniTek\\_linux.c](#), 53
  - [OmniTek\\_linux.h](#), 66
  - [OmniTekDmaIsr\\_linux.c](#), 99
- [ReferenceCount](#)
  - [\\_Resource](#), 41
- [RegisterOffset](#)
  - [\\_GeneralCtrl](#), 21
- [regs](#)
  - [\\_OMNITEK\\_INTERFACE\\_EXTENSION::\\_-](#)  
[bar\\_registers](#), 10
- [RegValue](#)
  - [\\_OMNITEK\\_INTERFACE\\_EXTENSION](#), 29
- [ReleaseResource](#)
  - [OmniTekResources\\_linux.c](#), 156
  - [OmniTekResources\\_linux.h](#), 162
- [RemoveResource](#)
  - [OmniTekResources\\_linux.c](#), 157
  - [OmniTekResources\\_linux.h](#), 162
- [request](#)
  - [\\_OmniTekDmaTransactionContext](#), 31
- [REQUEST\\_QUEUE](#)
  - [OmniTek\\_debug.h](#), 48
- [RequestStatus](#)
  - [OmniTekRequest\\_linux.h](#), 144
- [Resource](#)
  - [OmniTekResources\\_linux.h](#), 161
- [ResourceCheck](#)
  - [OmniTekResources\\_linux.c](#), 157
  - [OmniTekResources\\_linux.h](#), 163
- [ResourceControl](#)
  - [OmniTekResources\\_linux.h](#), 163
- [ResourceFind](#)
  - [OmniTekResources\\_linux.c](#), 158
- [ResourceItemRelease](#)
  - [OmniTekResources\\_linux.c](#), 158
- [ResourcePool](#)
  - [\\_OmniTekDriver](#), 32
- [ResourceRegisterWatchdog](#)
  - [OmniTekResources\\_linux.h](#), 164
- [RESOURCES](#)
  - [OmniTek\\_debug.h](#), 48
- [Resources](#)
  - [\\_OMNITEK\\_INTERFACE\\_EXTENSION](#), 29
- [ResourceVersion](#)
  - [OmniTekResources\\_linux.h](#), 161
- [Running](#)
  - [\\_DmaChannel](#), 15
- [sem](#)
  - [\\_dma\\_interrupt\\_info](#), 12
- [Sgl](#)
  - [\\_OmniTekDmaTransactionContext](#), 31
- [SGL\\_ITEM\\_SIZE](#)
  - [OmniTekDma.h](#), 82
- [SglBuffer](#)
  - [\\_DmaChannel](#), 15
- [sgt](#)
  - [\\_OmniTekDmaTransactionContext::\\_-](#)  
[DMACoreInfo](#), 16
- [Size](#)
  - [\\_DmaSglBuffer::\\_Allocated](#), 9
  - [\\_OmniTekDmaTransactionContext::\\_-](#)  
[XferInfo](#), 45
  - [\\_PCI\\_BAR\\_INFO](#), 39
- [SpinLock](#)
  - [\\_DmaChannel](#), 15
  - [\\_DmaSglBuffer](#), 19
  - [\\_OMNITEK\\_INTERFACE\\_EXTENSION](#), 29
  - [\\_OmniTekRequestQueue](#), 35
  - [\\_Resource](#), 41
- [state](#)
  - [\\_OmniTekDmaTransactionContext](#), 31
- [STATUS\\_INVALID\\_PARAMETER\\_1](#)
  - [OmniTek\\_linux.h](#), 60
- [STATUS\\_INVALID\\_PARAMETER\\_2](#)
  - [OmniTek\\_linux.h](#), 60
- [STATUS\\_INVALID\\_PARAMETER\\_3](#)
  - [OmniTek\\_linux.h](#), 60

STATUS\_OMNITEK\_ILLEGAL\_SESSION\_ID  
    OmniTek\_linux.h, [60](#)

STATUS\_OMNITEK\_MEMORY\_ERROR  
    OmniTek\_linux.h, [60](#)

STATUS\_OMNITEK\_RESOURCE\_-  
    COMMAND\_ERROR  
    OmniTek\_linux.h, [60](#)

STATUS\_OMNITEK\_RESOURCE\_INVALID  
    OmniTek\_linux.h, [60](#)

STATUS\_OMNITEK\_RESOURCE\_LOCKED  
    OmniTek\_linux.h, [60](#)

TRANSACTION\_WAIT\_MSECS  
    OmniTekDma.c, [69](#)

Type  
    \_OmniTek\_dev, [25](#)  
    \_Resource, [41](#)

type  
    \_OmniTekDmaTransactionContext, [31](#)

typeInfo  
    \_OmniTekDmaTransactionContext, [32](#)

u  
    \_Resource, [41](#)

USE\_IRQ\_THREAD  
    OmniTekDriver\_linux.h, [126](#)

userRequest  
    \_OmniTekKernelRequest, [34](#)

Version  
    \_FPGACtrl, [20](#)

Write  
    \_OmniTekDmaTransactionContext::\_-  
        XferInfo, [45](#)

WriteHWValue  
    OmniTek\_linux.h, [60](#)

WriteHWValueByte  
    OmniTek\_linux.h, [60](#)

WriteRegValue  
    OmniTek\_linux.c, [54](#)  
    OmniTek\_linux.h, [67](#)

Xfer  
    \_OmniTekDmaTransactionContext, [32](#)